Universidade do Minho
Escola de Engenharia

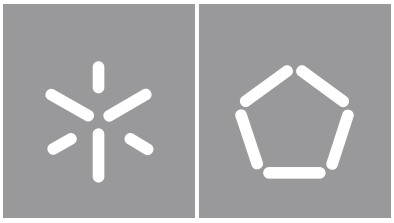
João André Correia Queiroga Pereira

Human-like Motion Generation Through Waypoints for Collaborative Robots in Industry 5.0

Outubro de 2021

João André Correia Queiroga Pereira
Human-like Motion Generation Through Waypoints for Collaborative Robots in Industry 5.0

UMinho | 2021



Universidade do Minho
Escola de Engenharia

João André Correia Queiroga Pereira

Human-like Motion Generation Through Waypoints for Collaborative Robots in Industry 5.0

Dissertação de Mestrado
Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Controlo, Automação e Robótica

Trabalho efetuado sob a orientação da
Professora Doutora Estela Guerreiro G. S. Bicho

Outubro de 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Acknowledgments

This dissertation represents many hours of effort and dedication, and this so important achievement would not be possible without the contribution of several people who accompanied and encouraged me during this journey.

First of all, I would like to thank my supervisor, Professor Estela Bicho, for all the support and availability provided throughout this dissertation, as well as all the trust placed in me. I also thank the opportunity to work and be part of the Mobile and Anthropomorphic Robotics Laboratory of the University of Minho, where I learned so much.

I would like to express a very special thanks to Gianpaolo Gulleta, who was crucial to accomplish the objectives of this dissertation. I learned a lot from you, thank you for guiding me and for always being available. I really admire your work. Grazie mille, Gianpaolo!

I would also like to thank my friends and colleagues with whom I shared my struggles to write such a demanding and challenging dissertation. Thank you for all your support and motivation.

Finally, I thank my parents and brothers for giving me unconditional support, encouragement and inspiration to make this journey possible.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Resumo

Geração de Movimentos Humanos através de Waypoints em Robôs Colaborativos na Indústria 5.0

A Indústria 4.0 tem motivado a comunidade científica a inovar soluções para garantir que as empresas mantenham os níveis de competitividade e satisfaçam as exigências dos clientes, cada vez mais em relevo devido à customização em massa [Villani et al. (2018)]. Os robôs precisam de mais flexibilidade, métodos de programação intuitivos e de fácil utilização, de modo a que estes sejam facilmente reprogramados para novas tarefas. Assim, surgiram os robôs colaborativos, que são mais pequenos, seguros e, acima de tudo, são capazes de partilhar o espaço de trabalho com operadores humanos [Villani et al. (2018)]. Além disso, já existem previsões relativamente à Indústria 5.0, onde humanos e robôs irão coexistir nas suas tarefas diárias [Schaal (2007)].

Esta dissertação propõe um planeamento de trajectórias que respondem às necessidades acima mencionadas. Este método permite aos operadores programar facilmente o robô para uma nova tarefa, definindo posições obrigatórias da trajectória. Os waypoints podem ser definidos através da manipulação física do robô ou através da utilização do joystick incorporado no painel de controlo do robô. Acrescenta-se que a trajectória gerada é baseada no modelo minimum-jerk de Flash and Hogan (1985), garantindo características humanas quantitativas e qualitativas. Tais propriedades tem impacto muito positivo no bem-estar e produtividade dos operadores [Koppenborg et al. (2017),El Zaatari et al. (2019)].

O método proposto é validado num cenário de inspecção de qualidade no contexto da indústria. Especificamente, o utilizador define pontos de passagem, que correspondem à melhor perspectiva para inspeccionar as placas, e posteriormente o robô manipula-as através dos pontos obrigatórios de uma forma humana. O planeador permite a realização de movimentos suaves, fluentes e intuitivos através dos pontos de passagem. Apesar dos movimentos resultantes possuírem características humanas, não podemos afirmar absolutamente que são movimentos humanos, uma vez que não há experiências em humanos com waypoints.

Palavras-chave: waypoints; human-like; geração de trajetórias; controlo ótimo; restriçoes de pontos interiores; robôs colaborativos; UR10; indústria 4.0; indústria 5.0

Abstract

Human-Like Motion Generation Through Waypoints for Collaborative Robots in Industry 5.0

Industry 4.0 has motivated the scientific community to innovate solutions to ensure that companies maintain levels of competitiveness and meet customer demands, which are increasingly higher due to mass customization [Villani et al. (2018)]. Robots need more flexibility, intuitive and user-friendly programming methods so that they can be easily reprogrammed for new tasks. Thus, collaborative robots have emerged, which are smaller, safer, and most importantly, are able to share the workspace with human operators [Villani et al. (2018)]. Moreover, there are already predictions regarding Industry 5.0, where humans and robots will coexist in their daily routines [Schaal (2007)].

This dissertation proposes a trajectory planning method that addresses the above needs. This method allows operators to easily program the robot for a new task by defining mandatory positions -waypoints- of the trajectory. Waypoints can be defined by physically manipulating the robot or by using the joystick built into the teach pendant robot. The generated trajectory is based on the minimum-jerk model introduced by Flash and Hogan (1985), which guarantees both quantitative and qualitative human characteristics. Such properties have a very positive impact on operators' well-being and productivity [Koppenborg et al. (2017),El Zaatari et al. (2019)].

The proposed method is validated in a quality inspection scenario in an industry context. Specifically, the user defines waypoints, which correspond to the position of the eye angle to inspect the plates, and subsequently the robot manipulates them through the mandatory points in a human-like manner. The planner allows smooth, fluent, and intuitive movements through the waypoints. Although the resulting movements have human characteristics, we cannot absolutely claim that they are human movements, since there are no experiments on humans with waypoints.

Keywords: waypoints; human-like; trajectory generation; optimal control; interior-point constraints; collaborative robots; UR10; industry 4.0; industry 5.0

Contents

I Dissertation Structure **1**

1 Introduction **2**

1.1 Evolution of Robotics 2

1.1.1 Collaborative Robots 4

1.2 Motivation and Objectives 7

1.3 Structure of the Dissertation 8

II State of the Art **9**

2 Programming Methods **10**

2.1 Introduction 10

2.2 Kinesthetic Teaching 11

2.3 Teach Pendant programming 12

3 Trajectory Generation **14**

3.1 Overview of Trajectory and Path 14

3.1.1 Path Planning Methods 17

3.1.2 Optimal Trajectory Planning 22

3.2 Trajectory Generation through waypoints in Robotics 24

3.3 Human-like Arm Motion Generation 30

3.3.1 Human-like Arm Motion Characteristics 30

3.3.2 Human-like Arm Motion Computational Models 33

3.4 Human-like Trajectory Generation through waypoints 35

3.5 Discussion 40

III Materials and Methods **42**

4 Collaborative Robot: UR10e **43**

4.1 Introduction . . . 43

4.2 Specification . . . 44

4.2.1 Singularities . . . 46

4.3 Kinematic Model . . . 47

4.3.1 Forward Kinematics . . . 49

4.3.2 Vacuum End-effector . . . 53

4.3.3 Inverse Kinematics . . . 54

5 Optimal Control **56**

5.1 Introduction to Optimal Control . . . 56

5.1.1 Lagrange multiplier . . . 58

5.2 Optimal Control Problems . . . 59

5.2.1 Equality Interior-point Constraints in State Variables . . . 61

IV Design and Implementation **64**

6 Trajectory Planning **65**

6.1 Problem statement . . . 65

6.1.1 Problem solving . . . 67

6.1.2 Final trajectory equation $t_0 \leq t \leq T$ . . . 72

6.1.3 Lagrange multiplier . . . 77

6.1.4 Waypoints time . . . 82

6.2 Time parametrization . . . 84

6.2.1 Total time . . . 84

6.2.2 Number of steps . . . 85

6.2.3 Time step . . . 85

V Validation of the Trajectory Planning **87**

7 Human-like Trajectory Planning with waypoints in a Human-Robot Collaboration Scene 88

7.1 Validation Architecture . . . 88

7.2 Human-likeness Evaluation . . . 90
7.3 Task: Quality inspection . . . 91
7.3.1 Task Description . . . 93
7.3.2 Waypoints definition . . . 94
7.4 Movements and Results Achieved . . . 96
7.4.1 Pick Movement . . . 97
7.4.2 Show Movement . . . 101
7.4.3 Place Movement . . . 107
7.4.4 Task Results . . . 115
7.5 Discussion . . . 116

VI Conclusion **119**

8 Conclusion and Future Work **120**
8.1 Future Work . . . 122

Appendices **131**

A Revisiting Universal Robot's Kinematics **132**

List of Abbreviations

C-free Free Configuration Space.

C-space Configuration Space.

CFRP Carbon Fiber Reinforced Polymer.

CNS Central Nervous System.

Cobot Colaborative Robot.

CPU Center Processing Unit.

DOF Degrees of Freedom.

EA Evolutionary Algorithm.

FDM Forward Dynamics Model.

GMP Gradient Projection Method.

GUI Graphical User Interface.

HRC Human-Robot Collaboration.

HRI Human-Robot Interaction.

HUMP Human-like Upper-limb Motion Planner.

IDM Inverse Dynamics Model.

IFR International Federation of Robotics.

IPOPT Interior-Point Optimizer.

NJS Normalised Jerk Score.

NMU Number of Movement Units.

PRM Probabilistic Roadmap Method.

RRT Rapidly-exploring Random Trees.

UI User Interfaces.

UR Universal Robots.

List of Figures

1.1 Evolution of robotics from the first to the fifth generation 4

1.2 On the left, cobot UR3 in a collaborative gluing application [Bloss (2016)]; On the right conventional industrial robot in a non-collaborative workspace [Tsarouchi et al. (2016)]. . 5

1.3 Collaborative robots – MRK-Systeme KR SI, Fanuc CR-35iA, ABB YuMi, UR5, KUKA LBR iiwa [Vysocky and Novak (2016)] 6

2.1 User manipulating a Baxter robot during the teaching process[Carfi et al. (2019)] 11

2.2 Robot teach pendants: a) the MOTOMAN NX100 teach pendant; b) the ABB IRC5 teach pendant that incorporates a joystick; c) teach pendant from KUKA Robotics incorporating a 6D mouse; d) wireless teach pendant from COMAU Robotics; e) FANUC robot tech pendant[Neto et al. (2010)] 12

3.1 C-space, C-free and C-obs for an articulated robot with two joints [Gasparetta et al. (2015)] 15

3.2 Brief framework of general trajectory planning [Lu et al. (2020)] 16

3.3 PRM Learning phase [Short et al. (2016)] 18

3.4 PRM Query phase [Short et al. (2016)] 19

3.5 Procedure in RRT algorithms [Elbanhawi and Simic (2014)] 20

3.6 RRT approach for searching the free space and avoiding obstacles (left), after 500 iterations. The root of the trees is represented as a green circle. Image taken from [Elbanhawi and Simic (2014)] 20

3.7 Typical trajectories obtained from different movement laws: a) constant acceleration; b) minimum-jerk c)limited jerk; d) harmonic jerk; [Aggogeri et al. (2020)] 23

3.8 Trajectory with a set of waypoints [Lynch and Park (2017)] 24

3.9 Spline trajectory with multiple polynomial segments of degree p [Kucuk (2017)] 25

3.10 Profile of a cubic spline trajectory with zero velocity in the waypoints in $t = 2$ and $t = 4$. In the figure, one can see the evolution of the joint angle position, velocity and acceleration, respectively [Spong et al. (2006)] 26
3.11 The comparison of seven-order B-spline with quintic B-spline and cubic spline method [Lan et al. (2020)] 27
3.12 Trajectory that interpolates waypoints based on parabolic blends [Kunz and Stilman (2011)]. 28
3.13 Robot path without stopping at the pre-defined waypoints [Pilz and KG (2019)]. 29
3.14 Trajectory profile of the robot position, velocity and acceleration [Pilz and KG (2019)]. . . 29
3.15 The kinematic model of human arm [Zanchettin et al. (2011)]. 31
3.16 In solid line is represented the predicted movement from the mathematical model of a typical point-to-point movement. In dashed line is illustrated the evolution of the real movement. The velocity, acceleration and jerk are similarly illustrated in subfigures b), c) and d), respectively [An adaptation from Flash and Hogan (1985)]. 32
3.17 Simulated reach-and-grasp movement [Rosenbaum et al. (2001)]. 33
3.18 Minimum-jerk model motion experiments with an animate human [Abdel-Malek et al. (2006)]. 36
3.19 Smooth transition between two motions using the minimum-jerk model and its adjustment to respect the constraints [Sung et al. (2015)] 36
3.20 Algorithm for extracting waypoints based on the minimisation of the square of the error between the given trajectory and generated trajectory [Wada and Kawato (1995)]. 37
3.21 Algorithm for producing the trajectory through waypoints by the FIRM model [Wada and Kawato (1995)]. 38
3.22 Experiment of handwritten task. On the left show the trajectory for 'abc' and on the right for 'def' letters [Wada and Kawato (1995)]. 39
3.23 Human arm reaching movement with waypoints. In Task 1 and Task 2 was performed a reaching movement with one waypoint and in Task 3 with 2 waypoints [Saito et al. (2006)]. 39
3.24 Waypoints estimated by Wada & Kawato's method and local minimum of tangential velocity [Saito et al. (2006)]. 40

4.1 Collaborative robots models from Universal Robots: UR3, UR5, UR16 and UR10 respectively. 44
4.2 Shoulder singularity, wrist singularity and elbow singularity of a Robot from Universal Robot [Adapted from FarzanehKaloorazi and Bonev (2018)]. 46

4.3 Relation between the forward kinematics and inverse kinematics. 47
4.4 Joints designation of robots from Universal Robot [Universal Robots (2015)]. 48
4.5 UR10e mechanical structure 49
4.6 Reference frames along the UR10e structure. 50
4.7 Robotic system mechanical structure and reference frames (UR10e with the UniGripper) 53
4.8 8 possible joint configurations for a desired end-effector pose [Oosterwyck (2018)]. . . . 54

5.1 Original Euler's representation of calculus of variations [Hanc (2017)]. 57
5.2 Illustration of the necessary conditions to minimise the objective function $f(x)$, subject to equality constraints $h(x) = 0$ [Chachuat (2016)]. 58

6.1 Joint trajectory from an initial joint position θ_{init} to a final joint position θ_{tar} that passes through N waypoints in unspecified time $[t1, t2, ..., tN]$ 66
6.2 Joint trajectory from the initial joint position θ_{init} to the first waypoint θ_{wp1}, time interval $t0 < t < t1$. 69
6.3 Joint trajectory from the first waypoint θ_{wp1} to the second θ_{wp2} (time interval $t1 < t < t2$). 70
6.4 Joint trajectory from a waypoint n, θ_{wpn}, to the next waypoint θ_{wpn+1} (or final joint position), time interval $t_n < t < t_{n+1}$ or $t_N < t < t_f$. 71
6.5 Denominator equations of the Lagrange multiplier π_1 in case of 1,2 and 3 waypoints, respectively. 76
6.6 Numerator equations of the Lagrange multiplier π_1 in case of 1,2 and 3 waypoints, respectively. 76

7.1 The validation of the generated trajectory is achieved by using the following modules: simulator CoppeliaSim; Polyscope; Motion Manager and Motion Planner. 89
7.2 Quality inspection scene 91
7.3 Quality inspection scene with human-robot collaboration 92
7.4 Universal Robots Graphical Programming Environment 95
7.5 Motion Manager Environment for defining waypoints 96
7.6 Posture where the robot starts and returns in each task cycle 97
7.7 Waypoints sequence of the pick movement 98
7.8 Position and velocity of the hand during the pick movement. The waypoint position is marked by the dashed line. 99

7.9 Joint position(black line), velocity(red line) and acceleration(blue line) profile during the pick movement. The waypoint is marked with a dashed line. 100
7.10 Waypoints sequence of the show movement 103
7.11 Position and velocity of the hand during the show movement. 104
7.12 Joint position(black line), velocity(red line) and acceleration(blue line) profile during the show movement. The waypoints are marked by dashed lines. 105
7.13 Waypoints sequence of the place approved movement 108
7.14 Position and velocity of the hand during the *Place Approved* movement. 109
7.15 Joint position (black line), velocity (red line) and acceleration (blue line) profile during the *Place Approved* movement. The waypoint θ_{wp2} is marked by a dashed line. 110
7.16 Waypoints sequence of the place approved movement 112
7.17 Position and velocity of the hand during the *Place Faulty* movement. 113
7.18 Joint position (black line), velocity (red line) and acceleration (blue line) profile during the *Place Faulty* movement. The waypoint θ_{wp2} is marked by a dashed line. 114
7.19 Position and velocity of the hand during the task 115
7.20 Position and velocity of the hand during the task 116

List of Tables

4.1 UR10 robot technical details 45
4.2 UR10e link distances and joint limits 49
4.3 UR10e Denavit-Hartenberg transformations between reference frames. 51
4.4 Robotic system Denavit-Hartenberg parameters (d_7 = 101 mm). 53

7.1 Home posture of the UR10 robot for the task 7.3.1 96
7.2 Waypoints to accomplish the show movement in the task 7.3.1 97
7.3 Results of the movement pick planning 99
7.4 Comparison between the expected and the calculated robot hand pose at the waypoints . 101
7.5 Comparison between the expected and the calculated robot joints values at the waypoints 101
7.6 Waypoints to accomplish the show movement in the task 7.3.1 102
7.7 Results of the movement pick planning 102
7.8 Comparison between the expected and the calculated robot hand pose at the waypoints . 106
7.9 Comparison between the expected and the calculated robot joints values at the waypoints. 107
7.10 Waypoints to accomplish the *Place Approved* movement in the task 7.3.1 108
7.11 Planning results of the movement place 109
7.12 Comparison between the expected and the calculated robot hand pose at waypoint θ_{wp2} of the *Place Approved* movement. 111
7.13 Comparison between the expected and the calculated robot joints values at waypoint θ_{wp2} of the *Place Approved* movement. 111
7.14 Waypoints to accomplish the *Place Faulty* movement in the task 7.3.1 111
7.15 Planning results of the movement *Place Faulty* 113
7.16 Comparison between the expected and the calculated robot hand pose at waypoint θ_{wp2} of the *Place Faulty* movement. 113
7.17 Comparison between the expected and the calculated robot joints values at waypoint θ_{wp2} of the *Place Faulty* movement. 115

Part I

Dissertation Structure

Chapter 1

Introduction

The industry has evolved tremendously over the past few years, and it promises not stopping here. The new technologies developed are increasingly human-oriented, and in this way the industry is implementing collaborative robots and promoting collaborative workspaces. This chapter briefly describes the evolution of robotics from its inception to collaborative robots and, consequently, to human-robot collaboration. However, it also presents some factors that affect the efficiency of human-robot interaction. In particular, the performance of a trajectory in a human-like manner. At the end of this chapter, the structure of this dissertation is presented.

1.1 Evolution of Robotics

The designation "Robot" and "Robotics" appeared for the first time, respectively, in 1921 and 1942. The former comes from the Czech word "Robota"[1], that means "forced labour". The latter [2] was introduced by Asimov, where he defined rules about robots' behaviour and its interaction with humans, to prioritise the well-being of humanity. In particular, the main rule states: *A robot may not injure humanity, or, through inaction, allow humanity to come to harm*. Since then, Asimov's laws of robotics have impacted on the ethics of robotics and artificial intelligence.

Over the last decades, robotics research has focused on the improvement of the human's quality of life, relieving humans from carrying out repetitive and heavy tasks, thus making the concept of Capek's a reality. Robots are used to perform mechanical labour in factories: removing workers from hazardous environments and increasing productivity and efficiency [Hockstein et al. (2007), Zamalloa et al. (2017)]; however, in recent applications, robots can also perform collaborative tasks with human operators in the

[1] Established by Karel Capek in his novel "Rossum's Universal Robots" [Karel (1920)]

[2] Introduced by Isaac Asimov (1920-1992) in his story "Runaround"[Clarke (1994)]

same environment: improving the ergonomics of the worker (e.g. reduction of diseases of the limbs from long-term) [Tlach et al. (2019),Vysocky and Novak (2016)]. Furthermore, the authors in [Gladden (2019),Zamalloa et al. (2017), Schaal (2007)] predicted a fifth generation of the robotics - the coexistence of robots and humans- where the robot also takes an active part on the daily routine of the human, such as: medical assisting robots, playmate robots in child education, personal robots for elderly, domestic robots, surveillance and protection. The definition of the word "robot" will change, transforming it in a concept of an ideal human companion [Nahavandi (2019)]. Robots will be capable of understanding and feeling the goals and expectations of a human collaborator. Nahavandi (2019) even anticipates that industry 5.0 will be suitably placed to make a positive impact on the environment and increasing the sustainability of human civilisation by reducing pollution and waste generation. However, since robotics inception, there are still four generations.

The first robotic generation had endured from 1950 to 1967. The earliest robot [3] was designed to enhance the productivity, and automate industrial processes, which encourage the research community to dedicate their work on robotics field [Zamalloa et al. (2017)]. In the following generation, robots handled more complex tasks, due to the introduction of many sensory systems, enabling the assembly line that had led to a mass production era. These first robots were characterised by an absence of information regarding the surrounding environment and elementary control algorithms, being considerably complicated to be reprogrammed. The robots of the third generation (1978-1999) had dedicated controllers and were reprogrammable - although, very difficult and expensive - for new tasks, as a result of the improvements on the hardware and software [Zamalloa et al. (2017),Gasparetto and Scalera (2019)]. The main characteristics of these robots are efficiency, repetitive accuracy and high-speed operations. However, these robots are enormous and dangerous for humans that they are mandatory to work inside of cages and no direct contact with humans is allowed [Boesl and Liepert (2016), Villani et al. (2018)].

During the mentioned robotics generations, the focus was on enhancing the production lines, with the highest speed and maximum of accuracy. However, these features hindered the robots from being flexible and versatile tools [Boesl and Liepert (2016)].

Nowadays, we are living the fourth industrial revolution, which has already resulted in a decrease of $10 - 30\%$ production and logistics costs [Nahavandi (2019)]. One of the most significant contributions for the emergence of industry 4.0 is the rise of mass customisation and the need to meet the consumer demands [Kurt (2019), El Zaatari et al. (2019)]. According to [Deloitte (2005)], more than 50% of costumers expressed interest in affording customised products and 1 in 4 consumers are willing to pay more for a

[3] George Devol and Joe Engleberger introduced the first industrial robot named UNIMATE

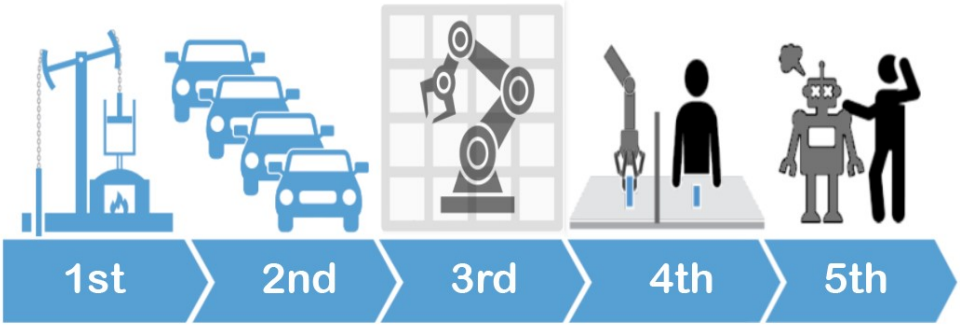


Figure 1.1: Evolution of robotics from the first to the fifth generation

personalised item. Therefore, the industries need more autonomy and flexibility, since re-programming a traditional industrial robot requires a specialised engineer considering the difficulty and time-consuming of the task. Hence, to maintain their competitive edge, the manufacturers need novel solutions and systematically improve production efficiency. To accomplish these demands, the research community has proposed the development of robotic platforms, named collaborative robots, capable of collaborating and sharing the workspace with human operators, enhancing the Human-Robot Collaboration (HRC) and Human-Robot Interaction (HRI) [(El Zaatari et al., 2019; Tsarouchi et al., 2016)].

1.1.1 Collaborative Robots

As described in the above section, the scientific research, during the past decades, has culminated on a robotic platform - collaborative robot - that aims to improve the quality of life of human operators.

Collaborative robots are designed to collaborate and interact with humans in the same workspace. These robots gather essential capabilities from humans and industrial robots to ensure more flexibility in many applications. The traditional industrial robots are ordinarily large and perform accurate, precise and high-speed motions, being capable of maintaining high efficiency, repeatability and accuracy for mass production [Vysocky and Novak (2016),El Zaatari et al. (2019)]. However, they are limited by their programming and are dangerous, since they are not aware of their surroundings. To prevent any accident, they are obligated to work inside of cages, where there is no interaction with operators [Boesl and Liepert (2016), Villani et al. (2018), Vysocky and Novak (2016)]. On the other hand, humans are capable of dealing with uncertainties, flexibility and variability in the workspaces due to their dexterity and cognitive skills [Villani et al. (2018)].

Figure 1.2: On the left, cobot UR3 in a collaborative gluing application [Bloss (2016)]; On the right conventional industrial robot in a non-collaborative workspace [Tsarouchi et al. (2016)].

The Human-Robot Collaboration (HRC) combines the above mentioned advantages of industrial automation and human capabilities, allowing the rise of a safe environment for a collaboration between humans and robots, where the robot performs the non-ergonomic, repetitive and dangerous tasks; and the humans execute the cognitive ones [(El Zaatari et al., 2019; Vysocky and Novak, 2016)]. Hence, an efficient human-robot interaction is crucial for the productivity and flexibility of production lines [Tsarouchi et al. (2016)].

Since the humans work side by side with robots, without any barriers among them, the human's safety and physical integrity must be ensured as a priority [Villani et al. (2018)]. Therefore, there are imperative safety standards to be followed, such as the requirements for integration of industrial robots (EN ISO 10218-2) and collaborative robots and work environments (ISO/TS 15066) [Tlach et al. (2019), Villani et al. (2018)]. Among these requisites stands out the force and speed limitations, no sharp edges and a system to avoid collisions and minimise the damage in case of an impact.

The collaborative robots bring countless advantages for humans that address their health and working satisfaction: the risk of injuries and the long term diseases are significantly decreased since the humans are removed from the uncomfortable, repetitive and tedious tasks [Vysocky and Novak (2016)]. Furthermore, in Bloss (2016) is mentioned that a collaborative task provides many benefits in robotic applications, which are maximised when a human and a robot work together at the same task.

Moreover, to fully exploit the human's skills, the robots must own user interfaces based on human-centred design. An intuitive user interface is the key for an easy-to-use and easy-programming methods with the primary goal to simplify the way the user interacts with the robot. As a result, production speed and product quality are increased, and costs are reduced that even small companies increase their competitiveness [Vysocky and Novak (2016)].

Figure 1.3: Collaborative robots – MRK-Systeme KR SI, Fanuc CR-35iA, ABB YuMi, UR5, KUKA LBR iiwa [Vysocky and Novak (2016)]

All these characteristics enhance the adoption of collaborative solutions by small and recognised companies that address a wide range of industrial applications [4] [Villani et al. (2018), El Zaatari et al. (2019)].

In addition to these advantages, the adoption of shared workspaces has a positive effect on labour demand, which contributes to an increase in job opportunities instead of the replacement of workers [Villani et al. (2018)].

In Figure 1.3 are illustrated some collaborative robots already implemented in factory floors. The first and second robot are conventional robots equipped with passive and active safety elements. The last three robots have safety functionalities incorporated on the robot itself. For instance, BMW Group has already introduced cobots for doors assembly process improving the operators' ergonomic [Giles and Hatzel (2013)] and Audi has added a UR3 cobot in the assembly line to install Carbon Fiber Reinforced Polymer (CFRP) on cars [5]. The operator first implements a CFRP roof to a rotary table and tilts the table. Then the cobot precisely applies the adhesive over the roof and then the operator with the aid of a handling device installs it in the car. During this process, the safety of the human is taken as an absolute priority. The operator is always in control of the application and can halt the process at any time. Moreover, the cobot does not requires a protective fence, due to the intelligent sensory programming, which saves space at assembly line.

According to the *International Federation of Robotics*[6] estimation, by 2020 there were already 2.7 million industrial robots operating in factories around the world. Similarly, a study conducted by *Mordor Intelligence* projects an increase of collaborative robots installations by 23% per year from 2020 to 2025 [Intelligence (2020)].

[4] https://www.universal-robots.com/applications/

[5] https://www.springerprofessional.de/en/manufacturing/production—production-technology/human-robot-cooperation-at-audi/14221870

[6] https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe

1.2 Motivation and Objectives

A few studies have shown that most of the human operators' cognitive interaction effort is related to programming a task [Villani et al. (2018)]. Therefore, intuitive programming and easy-to-use interfaces facilitate users' interaction with the robot that even a non-expert operator is capable of programming it. Thus, novel programming approaches, such as kinesthetic teaching or learning by demonstration, and interaction modes - gestures or speech, and augmented reality - play an essential role in the human-robot interaction [Villani et al. (2018), El Zaatari et al. (2019)]. Besides that, these approaches enable a situation awareness needed by humans to comprehend the current system behavior and facilitate intervention in unforeseen situations.

In [Fischer et al. (2016)] was concluded that the Programming by Demonstration method that maximizes the efficiency and performance of the robot control is a manual guidance approach. In particular, walk-through programming allows the user to physically move the robots' end-effector through the desired positions, and the robot controller records these points for later interpolation. Thus, with this approach, no explicit physical correspondence is needed, since the user demonstrates the skill with the robot's own body. Despite this subject being already advanced in the research area, the current collaborative robots are still not autonomous enough to enable a complete human-robot interaction [Tsarouchi et al. (2016)]. The main challenging research topics are the perception of human intent, learning by demonstration and observation, and optimization of the robots' behavior to improve human comfort and trust [El Zaatari et al. (2019)]. In this regard, Bortot et al. (2013) and Koppenborg et al. (2017) mentioned some robot's features that affect the efficiency and performance of Human-Robot Interaction (HRI), such as the design of the robot and the speed of a robotic movement. These authors concluded that co-workers' well-being and confidence increase as long as they can predict the following motion behaviour. Similarly, high-speed movements influence an operator's visual attention, since it increases their anxiety and instability, leading to lower performance or an elevated mental workload. Hence, the performance of human-like movements by the robot plays a crucial part in an efficient, safe, and productive HRI. During the cooperation, it is expected that the robot performs smooth, fluent, and collision-free movements in a human-like manner, making these easily understandable and foreseeable by human operators [Garcia et al. (2019), Koppenborg et al. (2017)].

In order to meet the requirements of industries 4.0 and 5.0, particularly mass customisation, full coexistence and interaction between robots and humans, both in human routine and on the factory floor, this project presents an easy-to-use and intuitive robot programming method that enables the performance

of an understandable and predictable trajectory. More specifically, a planning method is presented that allows a collaborative robot to execute a human-like trajectory through multiple waypoints. Firstly, the waypoints can be set manually by physically manipulating the robot or set using a joystick embedded in the user interface panel. This method takes inspiration from easy-to-use programming methods, such as kinesthetic teaching and walk-through programming. Thus, a flexible, intuitive, and easy-to-use approach is addressed to the planning method to allow ordinary operators to easily re-program a new task. Secondly, the human-like trajectory takes inspiration from kinematic features observed in human arm movements, particularly the minimum jerk model [Flash and Hogan (1985)], which guarantees the production of the smoothest possible movement of the arm. Thus, human satisfaction, well-being, and productivity are enhanced by the performance of human-like movements during the task.

The proposed method will be validated in the context of a quality inspection task with human-robot interaction, handled by a collaborative robot UR10 and an operator. The main goal is to analyse the human-likeness of the trajectory executed through the waypoints selected. Additionally, the advantages of the usability of waypoints are discussed.

1.3 Structure of the Dissertation

This dissertation is divided into fifth parts and eight chapters, each one organized to give a coordinate thought from problem presentation to its solution, implementation and validation.

In **Part I** is described the evolution of robotics and its future, specifically the main challenges that from there up-come, which are the main motivation of this project. Afterwards, is presented the state of art in, **Part II**, referred to: programming methods (**Chapter 2**) used in collaborative robots; and trajectory generation (**Chapter 3**), focused in two points: Human-like characteristics and waypoints.

In **Part III** are analysed the materials and methods used along the development of this dissertation. Specifically the UR10 robotic platform from Universal Robots in **Chapter 4**, and some fundamental concepts of optimal control theory in **Chapter 5**.

In **Part IV**, composed by **Chapter 6**, the design of the proposed trajectory generation through waypoints method and its implementation in a collaborative robot are thoroughly described.

Part V, composed by **Chapter 7**, specifies a collaborative task used to validate the proposed trajectory generation method. In this chapter is presented a human-robot collaboration scene, where the main task is the quality inspection of boards from different eye angles. Lastly, **Part VI**, and **Chapter 8**, discusses the main achievements and key conclusions of the dissertation, and also proposes future work.

Part II

State of the Art

Chapter 2

Programming Methods

2.1 Introduction

The rise of customers demands have led to higher product variant and shorter product life cycles. To tackle these requirements, smart factories have adapted the production process by giving more flexibility, autonomy and versatility to robots. For instance, to reprogram a robot for a new task, more easy-to-use and user-friendly programming approaches are mandatory [Vysocky and Novak (2016),El Zaatari et al. (2019)].

There are two main approaches considered in robot programming: online programming and offline programming. When using online programming, the robot is used during the teaching process; while using offline programming, the robot is programmed remotely using a computer by simulating the complete robot task in the 3D model. Additionally, the modelling functions allow graphical representation and give immediate feedback to the user. However, each robot manufacturer has its own software, which is normally very expensive and requires high programming effort. Also, the time required to program a robot task is remarkably long [Villani et al. (2018)].

The online programming method provides a direct interaction between human and robot. Thus, the teaching process is more natural since the operator has close perception and constantly observes the robot action [Mansour and Waldemar (2004)]. Moreover, teaching a robot new behaviours with this approach is time effective, easy and intuitive since visual perception and low-level computer programming skills are needed [Fischer et al. (2016)].

In Fischer et al. (2016), an interesting study about different online programming methods was conducted. The first was *Manual Guidance*, or *Kinesthetic Teaching*, where the robot's joints are manually moved into the goal posture. Secondly, the operator can manipulate each joint by itself, which is achieved through a graphical user interface (GUI), normally called *teach pendant*. Lastly, the operator

demonstrates the movements remotely by teleoperation. The experiments were carried out in a UR5 robot by Universal Robots where 51 participants explored three different methods in terms of efficiency, effectiveness, accuracy, and usability. The results show that the *kinesthetic teaching* method is superior to the other two methods in terms of success rate and speed. However, it is less accurate than the *teach pendant* method. Although *kinesthetic teaching* is the fastest method of teaching a robot, the large size of the robot arm and its temperature may make it less comfortable to manipulate. Regarding the *teach pendant* method, the participants considered it easy to use and felt precise control in the robot's movements, which resulted in highly accurate results. On the other hand, this method was considered too slow, tired, and exhaustive when programming an entire task.

In this regard, to give the best experience to the operator when programming a task, this project allows the definition of waypoints using *Kinesthetic Teaching* and *Teach Pendant* approaches.

2.2 Kinesthetic Teaching

Kinesthetic Teaching is one of the easiest and intuitive teaching methods. As illustrated in Figure 2.1, the operator sets the robot in a free movement mode and controls it manually through the desired path, while the robot records the joints coordinates into controlled memory for further playback [(Gupta et al., 2015; Carfi et al., 2019)]. Thus, the robot can be intuitively programmed, and no experience in programming languages is required. However, the lack of knowledge of the workplace and robot kinematics may lead to a low-quality teaching process [Carfi et al. (2019)]. Nevertheless, since the human physically manipulates the robot, the correspondence problem is avoided and the demonstrations are restricted to the kinematics limits of the robot [Akgun et al. (2012)].

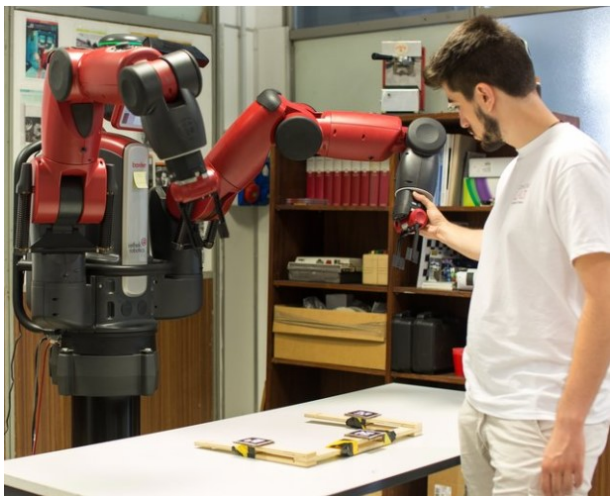


Figure 2.1: User manipulating a Baxter robot during the teaching process[Carfi et al. (2019)]

Obviously, in this scenario, safety issues related to physical human-robot interaction are important. The operator is more exposed in the workplace due to the physical control of the robot movements, and, therefore, appropriate strategies to improve operator safety are mandatory [Villani et al. (2018)]. For instance, a typical approach to accommodate the forces applied by the operator is to mount a force/torque sensor on the robot wrist.

According to Carfi et al. (2019), the best practices for using the kinesthetic teaching method require following a sequence of operations. First, the robot must be notified when the teaching procedure starts. Secondly, the user manipulates the robot through the desired path or even through a series of waypoints and presses the proper button to be recorded. Then, if necessary, the end-effector should be manually activated or deactivated. Lastly, the robots must be notified when the teaching process ends.

2.3 Teach Pendant programming

Many characteristics distinguish a collaborative robot from a traditional robot, for instance, its weight and size, velocity, and incorporated safety sensors. However, the most differentiating factor is probably the User Interface (UI). The UI's of most collaborative robots bring many additional features as real-time joints and sensor state control panel, and most importantly, allow modular symbolic programming. All these features turn the robots into intuitive and user-friendly platforms [El Zaatari et al. (2019)]. A teach pendant, Figure 4.1, is a hand-held control device with a UI incorporated for robot programming by remotely moving the robotic arm to the desired positions for later playback. The teach pendant also allows constant monitoring operations.

Figure 2.2: Robot teach pendants: a) the MOTOMAN NX100 teach pendant; b) the ABB IRC5 teach pendant that incorporates a joystick; c) teach pendant from KUKA Robotics incorporating a 6D mouse; d) wireless teach pendant from COMAU Robotics; e) FANUC robot tech pendant[Neto et al. (2010)]

An essential function of the teach pendant is to program a task by recording waypoints along the path,

when a specific button in the UI is pressed. Then, a trajectory that interpolates the defined waypoints is generated by the methods presented in the section 3.2. Since the user controls the robot within the working area, for safety reasons, the teach pendant has a palm-activated switch, which immediately stops the robot when pressed. Furthermore, during this teaching process, the robot joints' velocities are constrained.

The teaching pendant is simple, user-friendly and easy to program, which makes it accessible to non-experienced operators. However, teaching trajectories to the robot may be a tedious and time-consuming task [Fischer et al. (2016),Gupta et al. (2015),Villani et al. (2018)]. Additionally, this method may be unattainable for small production batches due to the time-consuming and the need to stop the production line while programming. Therefore, the demanded time consuming and the task complexity must be considered when defining the programming method. Hence, some companies as Universal Robots have the teach pendant UI available for offline programming and simulation, which allows to first program the task with the teach pendant logic and then just run it in the robot.

Chapter 3

Trajectory Generation

This chapter presents the state of the art of trajectory generation in robotic manipulators with a particular interest in generating human-like trajectories. In section 3.1, an extensive overview of the trajectory generation concept is presented. Section 3.2 describes the current trajectory generation methods through waypoints in robotics. Subsequently, section 3.3 presents a full review concerning human-like arm motion generation in robotics. Section 3.4 analyses the current methods of human-like trajectory generation through waypoints. Finally, this chapter ends with a discussion regarding the limitations of the current methods and how the research in this project improves them.

3.1 Overview of Trajectory and Path

One requirement in the emerging industry is the easiness of reprogramming a robot for different tasks. Different tasks require different motions, i.e., the capability to move from an initial configuration, q_i, to a final configuration, q_f, in a collision-free path. Moreover, the trajectory must be smooth and carefully planned. For instance, high operating speed may hinder the accuracy and repeatability of a robot's motion. This section highlights the path and trajectory planning algorithms that assume massive importance in robotics.

To perform a trajectory from an initial to a final configuration, the robot geometry or configuration, $\boldsymbol{q}$, must be provided. This problem is typically solved in the configuration space, *C*-space, which is the space of all possible robot configurations. The C-space is defined by the space that represents the obstacles (*C*-obs) and the free C-space (C_{free}), which is a set of robot configurations that do not intersect any obstacles (Figure 3.1). In Lozano-Perez (1990) was concluded that C-space is a useful way to abstract planning problems since a complex robot geometry can be represented with a single point in the C-space

as a vector of *n* joint positions.

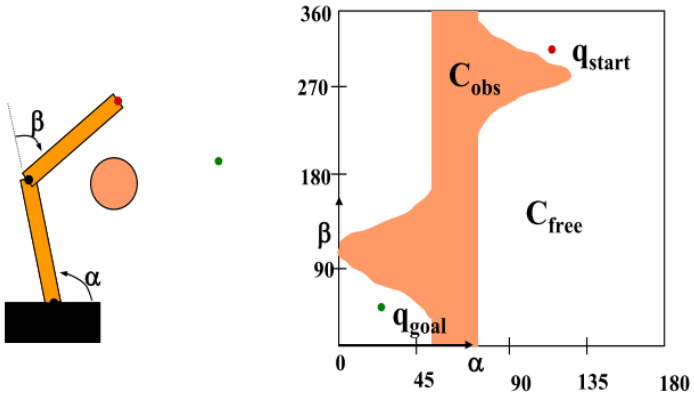


Figure 3.1: C-space, C-free and C-obs for an articulated robot with two joints [Gasparetta et al. (2015)]

The problem of finding a collision-free path $\boldsymbol{\theta}(s) : [0, 1]$ from an initial configuration, q_i, to a final configuration, q_f, without concerning the dynamics, duration of motion or its constraints is named *path planning*. Hence, a *path* $\boldsymbol{\theta}(s)$ maps a parameter $s \in [0, 1]$ to a point in the robot's configuration space, $\boldsymbol{\Theta}$ (equation3.1), and as *s* increases to 1, the robot moves along the path [Lynch and Park (2017)].

$$\boldsymbol{\theta(s)} : [0, 1] \rightarrow \boldsymbol{\Theta} \tag{3.1}$$

Besides this configuration, the *path*, $\boldsymbol{\theta}(s)$ can be scaled with respect to time *t*, where a s value is assigned to each time $t \in [0, T]$, equation 3.2.

$$s(t) : [0, T] \rightarrow [0, 1] \tag{3.2}$$

Thus, a time-scaled *path* connecting two points in the robot's configuration space (*C*-space) is known as a *trajectory*, equation 3.3.

$$\boldsymbol{\theta}(s(t)) : [0, T] \rightarrow [0, 1] \tag{3.3}$$

As the Figure 3.2 illustrates, a trajectory planner receives as an input the path description, the path constraints, and the kinematic and dynamics constraints. In contrast, the outputs are the end-effector or joint trajectories in terms of a time sequence of the values attained by position, velocity and acceleration [Lu et al. (2020)].

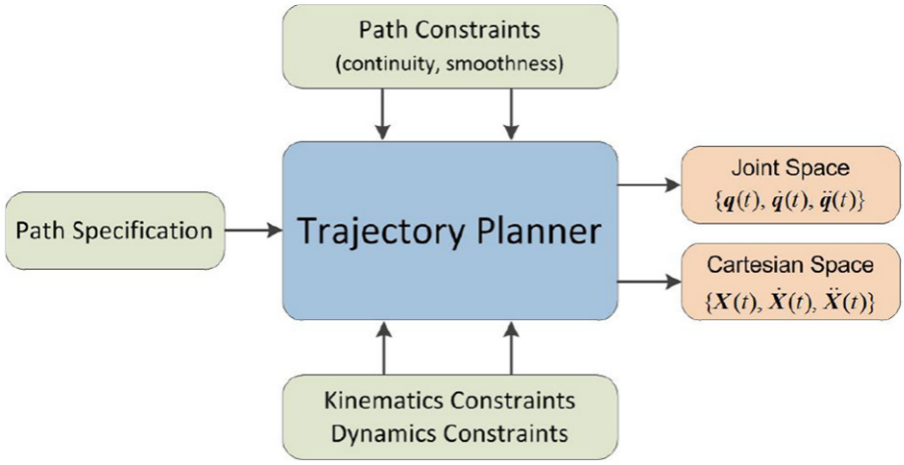


Figure 3.2: Brief framework of general trajectory planning [Lu et al. (2020)]

Cartesian space trajectories are directly related to task properties, allowing direct visualization of the generated path. For a simple motion like a hand over an object, a trajectory planned in Cartesian space can produce a straight-line movement, which is not easy to guarantee while planning in joint space.

The trajectory planning can be carried out in the operational space or in the joint space. In the former, although being directly related to the properties of the task, which allows direct visualization of the generated path, this would easily lead to complex problems such as kinematic singularities and manipulative redundancy. Conversely, the latter, after a kinematic inversion of the given geometric path, allows the avoidance of such problems and involves less computational costs. However, the motion performed is not easily foreseeable due to the non-linearities introduced by the computation of the forward kinematics to obtain the position of the end-effector [Huang et al. (2018)].

A trajectory is classified as a *point-to-point trajectory* or *multipoint trajectory*. In the former, the robot generates a motion between two points; the trajectory is commonly defined by a polynomial function, where only the initial and final boundary conditions are considered. A complex motion is obtained by joining several point-to-point trajectories. The latter considers multiple points where the robot must pass through them. Not only the initial and final boundary conditions are considered, but also the specification of intermediate points, called waypoints. These intermediate points may be then interpolated through piecewise polynomials [Lynch and Park (2017)].

Furthermore, the trajectory generated must be executed in a coordinated movement by all robot joints. To accomplish such coordination, all joints are normalized by a factor that influences their movement speed. Thus, every joint starts and stops its motion simultaneously [Niku (2020)].

As already explained, path planning refers to the design of kinematic specifications of the robot's position and orientation, whereas trajectory planning also includes the design of the linear and angular

velocities. Thus, path planning is a subject of trajectory planning. Therefore, in the subsection 3.1.1, some of the most popular path planning algorithms are briefly discussed. Then, the most significant approaches for trajectory planning, considering no obstacles in the workspace, are thoroughly analyzed.

3.1.1 Path Planning Methods

Path planning, one of the most studied subjects in robotics, is known to be a hard problem to solve due to its computational cost. The emergence of practical planners came around with the cell decomposition [Brooks and Lozano-Perez (1985)] and potential field [Khatib (1986)]. These planners demonstrated high efficiency in some applications, although they are not so suitable for complex problems with several dimensions [Karaman and Frazzoli (2011)]. Therefore, sampling-based algorithms capable of dealing with complex problems, such as high dimensions, were proposed.

The main idea of the sampling-based planners is to avoid explicit construction of *C*-obs, searching in the C-space by a sampling scheme [LaValle (2006)]. The most common sampling-based algorithms are the Probabilistic Roadmap Method (PRM) (section 3.1.1) and Rapidly-exploring Random Trees (RRT) (section 3.1.1). These methods have different methods for connecting the sampled points. The PRM is a multiple-query method that first executes a learning phase, where the roadmap is built, and then a query phase that connects the samples in order to define a free path. By contrast, the RRT method is a single-query method, which do not build a road map and avoid the connection of thousands of configurations. Both approaches are probabilistically complete, which means that they cover all possible configurations as the number of iterations tends to infinity. In other words, if a solution exists, the planner finds it given the sufficient runtime; however, it could be in an infinite runtime [Elbanhawi and Simic (2014)].

These approaches have been applied in many robotic applications, which most require an optimal path. Therefore, more recently, aiming to reduce the costs of the planned path, the algorithms PRM and RRT were modified by Karaman and Frazzoli (2011), whereby the methods $PRM^*$and RRT^* were presented, which guarantee asymptotically optimal (the cost of the returned solution converges to the optimum)[Karaman and Frazzoli (2011)].

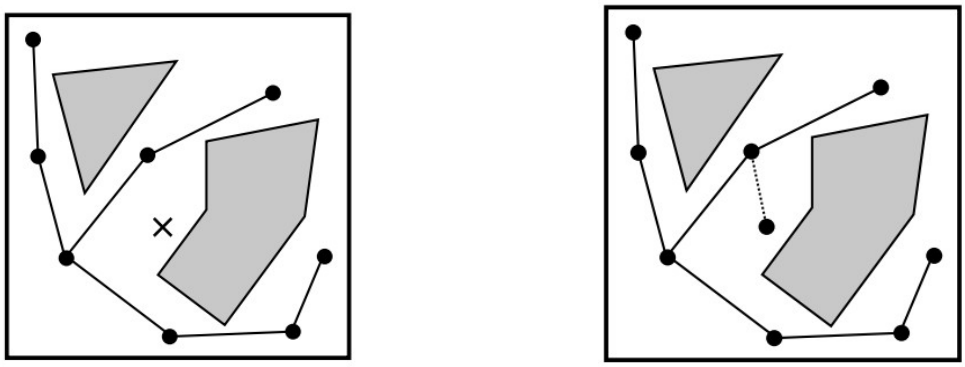
As mentioned above, the most common path planning methods are based on sampling-based algorithms, mainly PRM (section 3.1.1) and RRT (section 3.1.1). Hence, these two methods are clearly explained in the following sections.

Probabilistic Roadmap Method (PRM)

The Probabilistic Roadmap Method (PRM), presented by Kavraki et al. (1996), consists of finding a finite set of collision-free configurations in the *C*-space, from an initial q_{init} position to a final q_{goal} position, which are applied to build a roadmap.

Probabilistic planners represent a class of methods with remarkable speed and efficiency to solve complex problems, specifically ones involving high-dimensional *C*-space. The dimension of the *C*-space is defined by the number of joints of the robot [Geraerts and Overmars (2004)]. Furthermore, it also handles problems with many different constraints, such as kinematic and dynamic constraints [Hsu et al. (2002)] and closed-loop kinematics [Han and Amato (2001)].

The PRM proceeds according to two phases: a learning phase and a query phase. In the learning phase (Figure 3.3), a probabilistic roadmap is constructed by repeatedly generating random configurations free of collisions in the *C*-space (Figure 3.3a). After every new sample, a local planner is used to connect it to the nearest one (Figure 3.3b)[Short et al. (2016)].



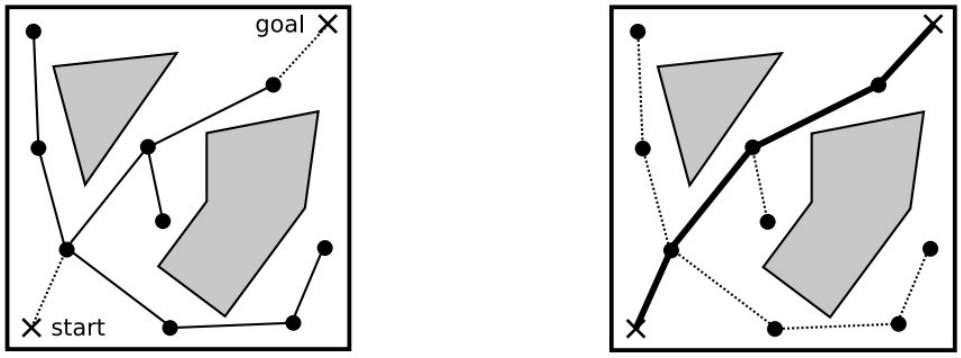
(a) The learning phase: a random sample, denoted by X is generated.

(b) A local planner is used to connect the new sample to nearby roadmap vertices.

Figure 3.3: PRM Learning phase [Short et al. (2016)]

To validate a sampled robot configuration, a collision detection algorithm is executed and responds to whether the objects collide or not. When the configuration is free of collisions, that sample is added to the roadmap.

In the query phase, the given start and goal configuration, are implemented into the roadmap (Figure 3.4a). Thereafter, as represented in Figure 3.4b, a graph search algorithm is used to connect the start and goal through the roadmap by the shortest path [Short et al. (2016)].



(a) The query phase: the start and goal configurations are added to the roadmap.

(b) A graph search algorithm is used to connect the start and goal through the roadmap.

Figure 3.4: PRM Query phase [Short et al. (2016)]

The PRM assumes a fully known environment, so with the evolution of robotics systems, the necessity of expanding this method for dynamic environments has emerged. Moreover, the research presented in Hsu and Sun (2004) stated that more than 90% of planning time is spent in the collision checking processing. In Short et al. (2016) and Elbanhawi and Simic (2014) are it presented some variations that aim to improve the traditional sampling-based methods. For instance, Bohlin and Kavraki (2000) presented a Lazy PRM that minimizes collision checking.

Although the PRM approach is being costly because it connects thousands of random configurations by a local planner's action, this process may be suitable for multi-query planning and point-to-point motions [LaValle and Kuffner Jr (2000)]. However, this method is not convenient for problems with differential constraints. To address this drawback, LaValle (1998) developed the technique Exploring Random Tree (RRT).

Rapidly-Exploring Random Tree (RRT)

The Rapidly-Exploring Random Tree approach incrementally constructs a search tree that randomly improves the resolution towards unexplored space. The tree grows from the initial configuration to the goal configuration (Figure 3.5). As one can see in Figure 3.5(a), the first step of this process is the generation of a random configuration, q_{rand}, by a uniform probability distribution in the *C*-free. Then, Figure 3.5(b), the planner searches for the nearest node, q_{near}, which is attempted to connect to q_{rand} by the local planner. A new configuration q_{new} may be returned whether q_{rand} is not reachable, which in these cases is discarded, Figure 3.5(c). A collision checking is then performed to ensure the collision-free path between q_{near} and q_{new}. Finally, if succeeded, q_{new} is added to the tree as shown in, as show the Figure 3.5(d).

The search ends when the $q_{new} = q_{goal}$, if the number of iterations or the specified time period is exceeded [Elbanhawi and Simic (2014)].

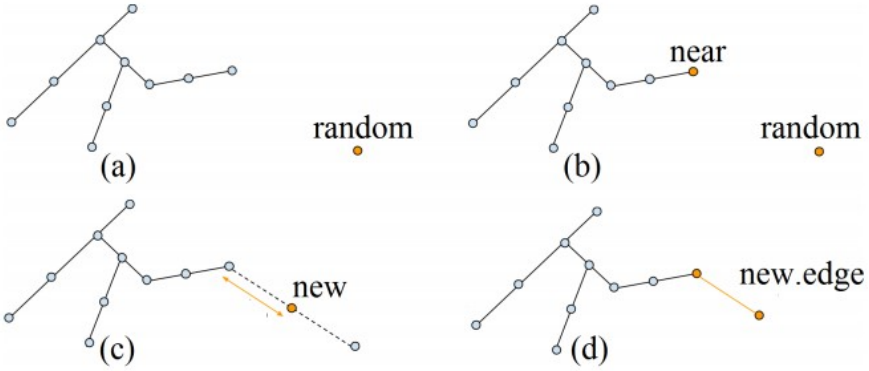


Figure 3.5: Procedure in RRT algorithms [Elbanhawi and Simic (2014)]

The Figure 3.6 illustrates the result of RRT approach after 500 iterations in a environment totally free and another with an obstacle to avoid. Furthermore, one can see that the tree uniformly covers the four corners of the square. As the number of iterations increases, the planner reaches arbitrarily all points in the space [LaValle (2006)].

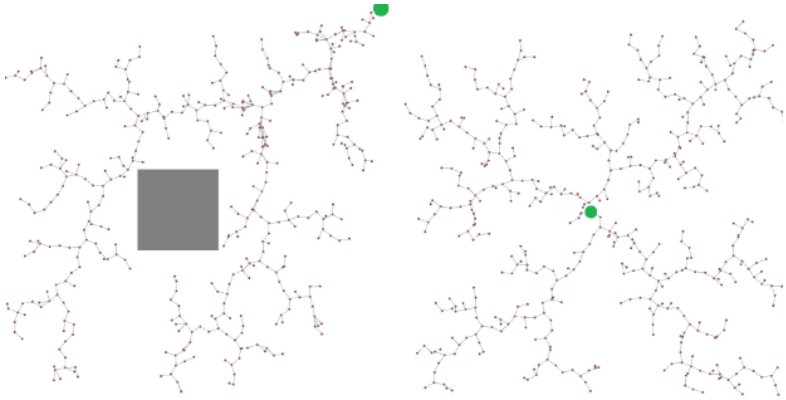


Figure 3.6: RRT approach for searching the free space and avoiding obstacles (left), after 500 iterations. The root of the trees is represented as a green circle. Image taken from [Elbanhawi and Simic (2014)]

Compared to PRM, the RRT based planners are faster for single query problems, since there is no learning phase to build a roadmap, which is computationally challenging [LaValle (1998)]. Moreover, a key advantage of RRT over PRM is the integration of motion parameters as system dynamics [Short et al. (2016)].

During the past years, many variations of RRT based planners have been proposed to reduce the solution cost or planning time [Elbanhawi and Simic (2014)]. Therefore, Kuffner and LaValle (2000)

presented the RRT-Connect method that combines the RRT with a simple greedy heuristic. This method uses a second tree to perform a bidirectional search - one starting from q_{init} and another from q_{goal} - which grows towards each other, ending when both are connected. Thus, providing significant improvements in search efficiency. Moreover, similarly to RRT and PRM, RRT-connect is probabilistically complete, which means that if there is a solution, the probability to finds it approaches one, as the number of iterations increases [Kuffner and LaValle (2000)].

3.1.2 Optimal Trajectory Planning

The trajectory planner returns a trajectory defined by a time sequence of the joint values. The interpolation of joint configurations is generally determined by polynomials, splines, or trigonometric functions, which meet the imposed kinematic constraints. Cubic splines are commonly used due to the assurance of continuous velocity and acceleration and prevent large oscillations of the trajectory resulting from high-order polynomials. However, it does not allow the continuity of the jerk (the derivative of the acceleration) [Perumaal and Jawahar (2013)].

Almost every technique found in the scientific literature on the trajectory planning problem is based on optimizing some objective function. For instance, the duration of a task or movement to increase productivity [Gasparetto and Zanotto (2008)]; the energy consumption to reduce costs, where robots have a high impact since 8% of the electrical energy in production processes is consumed by them [Liu et al. (2018)]; and movements vibration to increase the life span of the robots and increase the naturalness of movements and reduce operator's psychological stress [Rojas et al. (2019)]. Thus, the most common optimization approaches are the minimum-jerk, minimum-torque, minimum energy, and minimum-time models.

In Bobrow et al. (1985) was presented a method that minimizes the execution time, maximizing the velocity of the manipulator along a given path using a switching curve in the phase plane. However, it generates discontinuities in the acceleration profile, leading to undesired effects such as decreasing the accuracy and creating high-frequency vibrations that can damage the robot structure [Gasparetto and Zanotto (2008)]. To overcome these issues, Constantinescu and Croft (2000) introduced limits on the actuator jerks, with the drawback that the trajectory returned is not precisely time optimal. In Liu et al. (2018) was presented a multi-objective optimization approach where the energy is measured by the sum of the square joints' accelerations, resulting in a reduction of 6.1% of energy consumption. Lately, Liu introduced an approach to minimize the energy along with the execution time [Liu et al. (2020)].

In [Gasparetto and Zanotto (2008)] is presented an approach that minimizes the jerk and execution time, taking into account kinematic constraints as velocity, acceleration, and jerk. However, there is a trade-off between both terms in the objective function; the higher weight of the jerk term leads to slower trajectories, and the higher weight of the time term leads to faster but not so smooth trajectories. Furthermore, it has been established that minimizing the jerk, defined as the time derivative of the acceleration, reduces the manipulator wear and improves accuracy and speed [Kucuk (2017)]. Kyriakopoulos and Saridis (1988) confirmed that the joint position error is decreased as the jerk decreases. It also reduces the resonant frequencies of the robot, which could cause severe damage to the robot structure. Thus,

increasing the lifetime of the manipulator [Gasparetto and Zanotto (2008)]. An exciting characteristic of the jerk minimization model introduced by Flash and Hogan (1985) is the description of voluntary human arm movements. Thus, the minimum-jerk model has been used to obtain a natural and coordinated robot motion. Bearee et al. (2005) compared the minimum-jerk approach with other jerk-laws (constant acceleration; limited jerk and harmonic jerk). Figure 3.7 compares the typical trajectories obtained from these models, and the results show that the minimum-jerk model performs the smoothest movements. This model is formulated as the square time integral of the jerk (the third derivative of the joint position), which results in a fifth-order polynomial to describe the joints' evolution in time (equation 3.4).

$$\theta(t) = a_0 t + a_1 t^2 + a_2 t^3 + a_4 t^4 + a_5 t^5 \tag{3.4}$$

The coefficients $(a_0, ..., a_5)$ are normally determined given the boundary conditions of the position, velocity, and acceleration. Generally, the velocity and acceleration of the initial and final position are zero since the movement is stationary in those points. After successive derivatives of joint evolution over time, one can achieve the equations for velocity, acceleration, and jerk (Figure 3.7(b)).

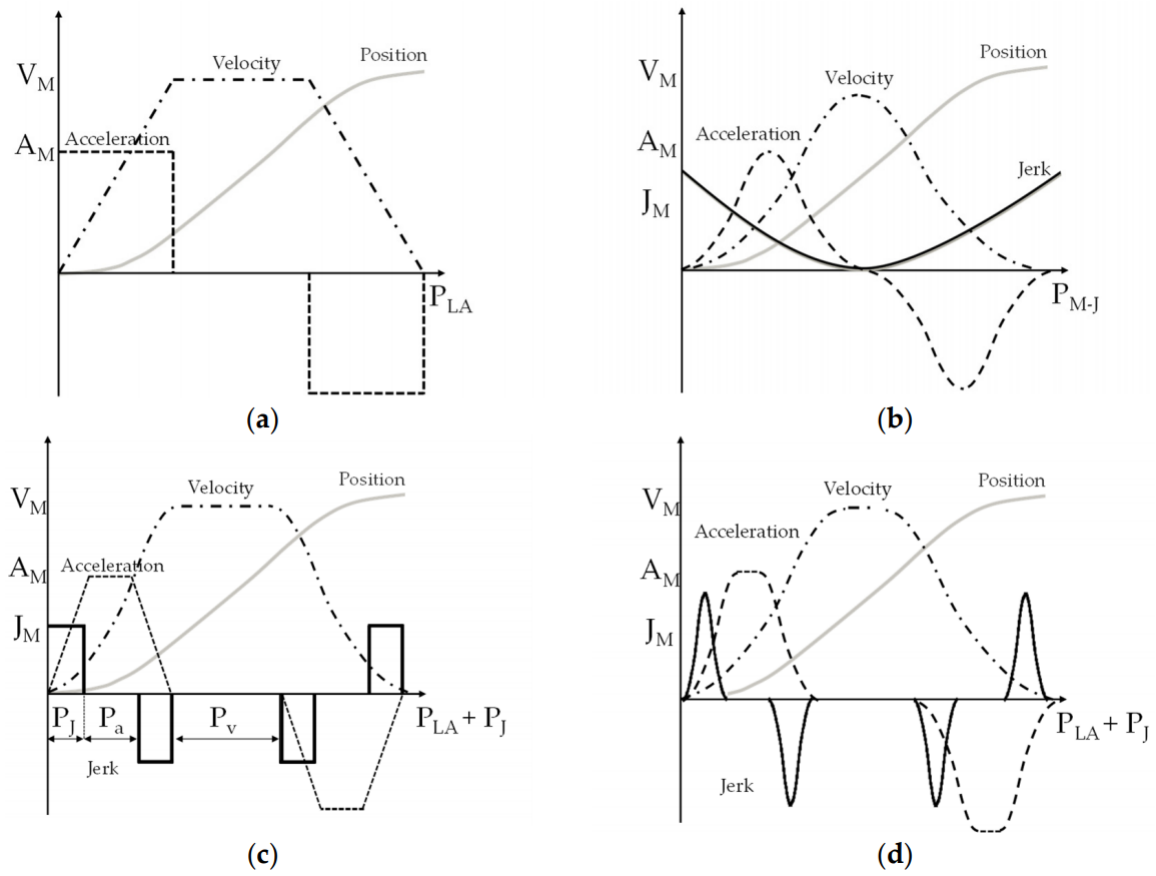


Figure 3.7: Typical trajectories obtained from different movement laws: a) constant acceleration; b) minimum-jerk c)limited jerk; d) harmonic jerk; [Aggogeri et al. (2020)]

3.2 Trajectory Generation through waypoints in Robotics

A trajectory that passes through waypoints is defined as *multipoint trajectory*, or trajectory with path constraints. In these trajectories, Figure 3.8, not only the initial and final boundary conditions need to be specified, but also the intermediate points where the robot must pass [Biagiotti and Melchiorri (2008)].

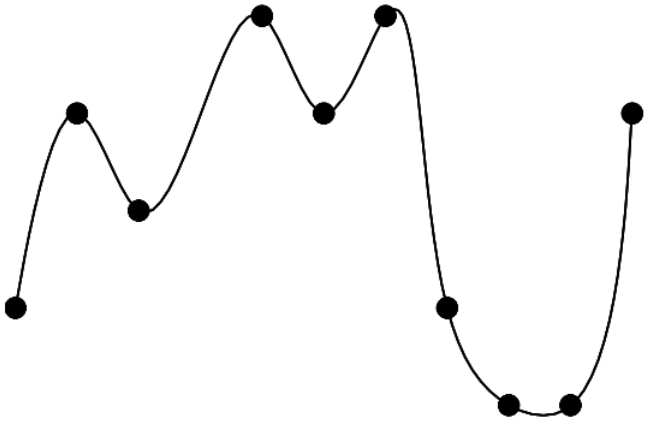


Figure 3.8: Trajectory with a set of waypoints [Lynch and Park (2017)]

The problem of computing a trajectory passing through N waypoints can be solved by a single polynomial function. For instance, let us consider an example of a trajectory that passes through a single waypoint. In addition to the constraint of the waypoint position, to perform a smooth trajectory, constraints in the position, velocity and acceleration at the initial and final point should be imposed. Thus, there are a total of 7 constraints that can be satisfied with a sixth-order polynomial (equation 3.5).

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 \tag{3.5}$$

As one can conclude, although this approach is easily described and attend the constraints imposed in the waypoint, the degree of the polynomial depends on the number of constraints, which turns out many disadvantages: the computation costs, vibrations and numerical error increases as the polynomial order increases [Biagiotti and Melchiorri (2008)].

An alternative to a single polynomial is the spline function, which is a function subdivided in multiple segments of low order polynomials between the waypoints. In general, as one can observe in Figure 3.9, for the interpolation of $n+1$ given waypoints $(\theta_{wp1}, \theta_{wp2}, ..., \theta_{wpn+1})$, n polynomials $(x_1(t), x_2(t), ..., x_n(t))$ of degree p are considered.

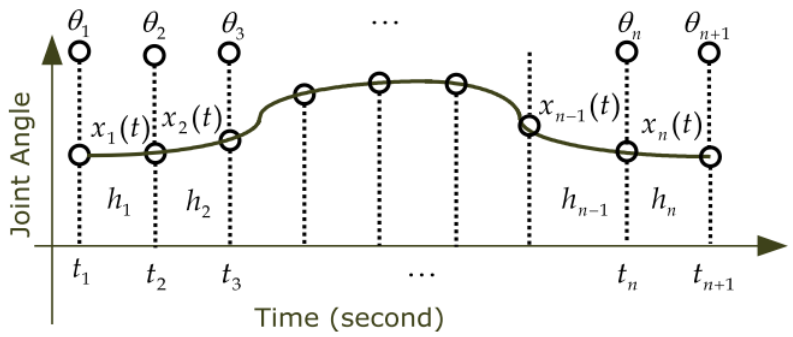


Figure 3.9: Spline trajectory with multiple polynomial segments of degree p [Kucuk (2017)]

Among the polynomial splines, the cubic splines defined in equation 3.6 are the preferred since they provide continuous velocity and acceleration with the lowest degree and the lowest possible jerk peak [Kucuk (2017)].

$$\begin{aligned} x(t) &= \{x(t), t \in [t_k, t_{k+1}], k = 1, ..., n\}, \\ x_k(t) &= a_{k0} + a_{k1}(t - t_k) + a_{k2}(t - t_k)^2 + a_{k3}(t - t_k)^3 \end{aligned} \tag{3.6}$$

Note that the time when the robot must pass through the waypoints may or not be defined. If the time is not specified, a common technique to calculate it is defined by the equation 3.7 [Biagiotti and Melchiorri (2008)].

$$t_k = t_{k-1} + \frac{d_k}{d} \qquad with \qquad d = \sum_{k=1}^{n+1} d_k \tag{3.7}$$

There are some possible ways to define d_k, according to the result that is expected. One can define $d_k = \frac{1}{n-1}$, to achieve highest speed; or $d_k = |\theta_{k+1} - \theta_k|^{\frac{1}{2}}$, known as centripetal distribution, normally used to reduce the accelerations [Biagiotti and Melchiorri (2008)].

In the research community, many approaches have adopted the cubic splines. Piazzi and Visioli (2000) presented an interpolation of waypoints based on a sequence of cubic splines, despite the high computational demands. Then, in Craig (1986) two third-order polynomials were used to move the robot from an initial to a final configuration passing through a via point. The cubic spline function presents continuous velocity and acceleration; however, it is not possible to define the initial and final acceleration. Therefore, the velocity and acceleration profile is characterised by discontinuities (Figure 3.10). Although commonly used, according to Williams (2013), these methods must be avoided due to the infinite jerk spikes created by the discontinuities of the acceleration, which results in many problems for the robot structure such as

unacceptable noise, wear, reduced life and bad dynamics [Barre et al. (2005)].

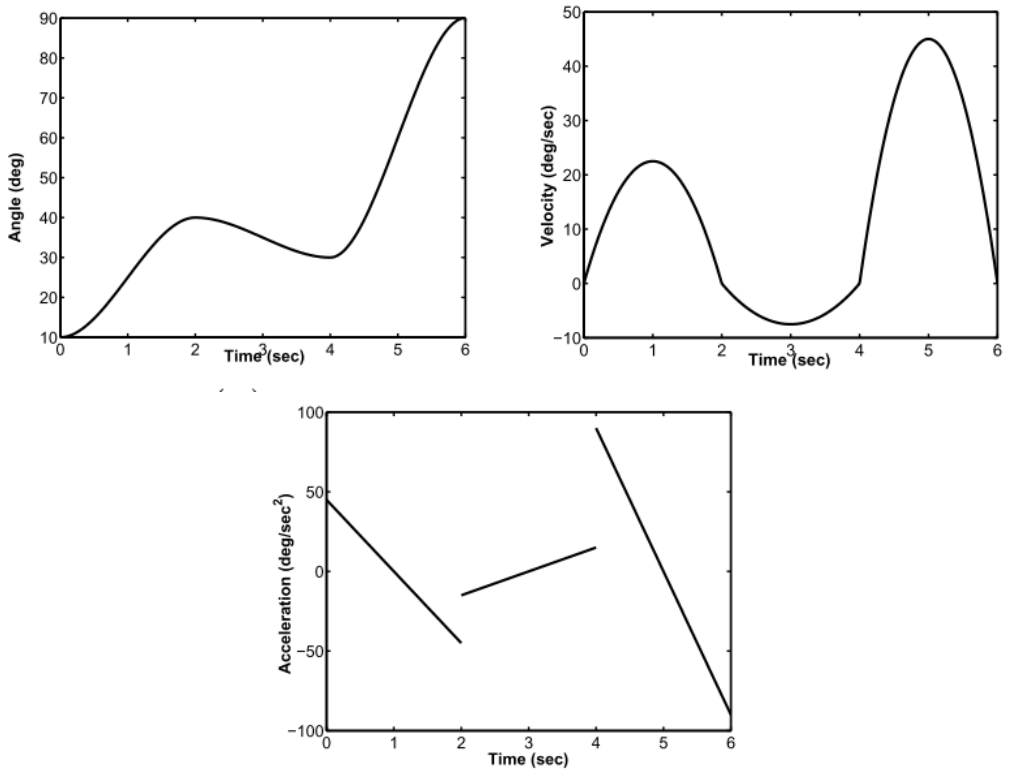


Figure 3.10: Profile of a cubic spline trajectory with zero velocity in the waypoints in $t = 2$ and $t = 4$. In the figure, one can see the evolution of the joint angle position, velocity and acceleration, respectively [Spong et al. (2006)]

With the aim to address the above-mentioned problems, different algorithms have been proposed. For instance, Kucuk (2017) developed an approach that avoids the acceleration ripples of a cubic spline in the first and last waypoints by interpolating the cubic splines with a seven-order polynomial. Thus, the first three derivatives of joint positions at the boundary points can be constrained and zero jerk at the beginning and endpoints were guaranteed. Comparing to 5th and 9th-degree polynomials, the 7th-degree polynomial presents a lower linear and angular jerk.

In the scientific literature, another common approach is based on B-spline functions. B-spline function, or *basic-spline*, is a efficient technique for the computation of splines. The reason of the name B-spline is that a generic spline can be obtained through linear combination of a set of basis functions, the B-splines ($B_j^p(u)$)

$$s(u) = \sum_{j=0}^{m} p_j B_j^p(u) \quad u_{min} \leq u \leq u_{max} \tag{3.8}$$

where the coefficient $p_j, j = 0, ..., m$, called control points, define the curve of the spline. Lan et al. (2020) used seven-order B-spline functions to construct join trajectory with a continuous position, velocity, acceleration and jerk of each joint. The experiments were conducted in a collaborative robot AUBO-I5, a robot with six degrees of freedom. Firstly, 8 waypoints from the operational space were obtained arbitrarily and then converted into joint space by the inverse kinematics algorithm. Then, the trajectory to be performed by the robot is determined by a seven-order B-spline. The authors also compared the smoothness of seven-order B-spline with cubic spline and quintic B-spline. The results, in Figure 3.11, show that the jerk curve of the cubic spline is not continuous, and the quintic B-spline present non-smooth and jerk pikes at the beginning and end of the movements. Conversely, the seven-order B-spline ensures smoothness, however, with more high values of jerk. Intending to limit the initial and final value of the jerk at the boundaries of the trajectory with a quintic B-spline function, Huang et al. (2018) introduced two virtual points at the second and the second-last position of the waypoints. The experiments proved the smoothness and continuity of the jerk along the entire trajectory.

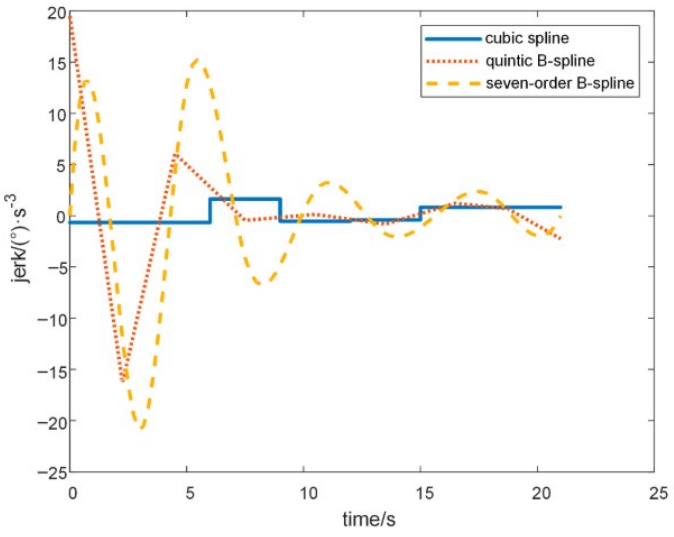


Figure 3.11: The comparison of seven-order B-spline with quintic B-spline and cubic spline method [Lan et al. (2020)]

In addition to the above methods, another approach for the interpolation of a set of waypoints is based in trigonometric splines, which are convenient to use for trajectories that represents a periodic motion $\theta(t+T) = \theta(t)$. The trigonometric spline performs smooth trajectories due to the continuous derivative of sin and cos expressions, which thus guarantees steady velocity, acceleration, and jerk, even in the first and last waypoint [Biagiotti and Melchiorri (2008)]. Simon et al. (1991) demonstrated that this approach demands less computational cost and handles more constraints in the trajectory than algebraic splines.

However, practical applications with trigonometric splines have shown more pronounced oscillations with higher values of acceleration and jerk than algebraic splines [Biagiotti and Melchiorri (2008)].

An interesting approach was presented by Sung et al. (2013), where the author used a waypoint representation to specify conditions for the achievement of a kicking motion. The generated trajectories follow the minimum-jerk model, which is represented by a fifth-order polynomial. This polynomial only takes into account the position, velocity and acceleration in the initial and final points. Therefore, to deal with the constraints imposed, the problem was solved as a nonlinear optimisation with equality constraints, which are defined by the waypoints position and velocity. However, the time where the trajectory passes through the waypoints must be given.

Kunz and Stilman (2011) used parabolic blends to generate a trajectory that follows a path with constraints on position, velocity and acceleration. The waypoints correspond to path constraints and are connected through straight line segments. The velocity and acceleration at the edge of the trajectory and at the waypoints are established as zero and arbitrary, respectively. The trajectory generation consists in a linear phase followed by parabolic blend phase. During the linear phase, the velocity is constant, acceleration is zero and the position is linear in time. During the parabolic blend phase, the acceleration is constant, velocity is linear, and the position is quadratic, and therefore, parabolic in time. The results show that the method follows the waypoints satisfying the velocity and acceleration constraints. However, it can only approximate and not interpolate the waypoints. Additionally, the authors could not guarantee that the resulting trajectory is optimal.

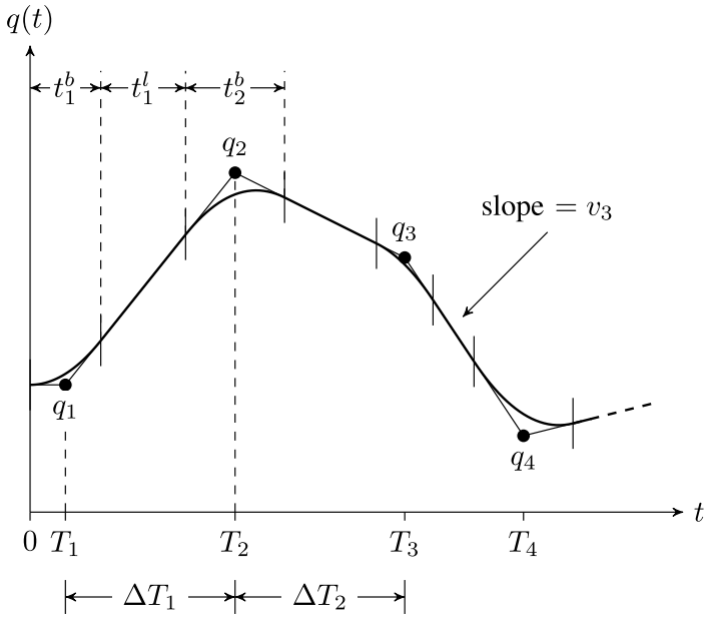


Figure 3.12: Trajectory that interpolates waypoints based on parabolic blends [Kunz and Stilman (2011)].

A similar approach to generate blending trajectories is used in the *MoveIt library* [1] to concatenate multiple linear trajectories and plan a trajectory at once. As illustrated in Figure 3.13, the robot has to pass by a sequence of waypoints, denominated as *P0* to *P4*. Considering that direct trajectories between every two waypoints are already defined, a *blend_radius* sphere is established for each waypoint, and thus, the robot is able to perform a single trajectory without stopping at the waypoints [Pilz and KG (2019)].

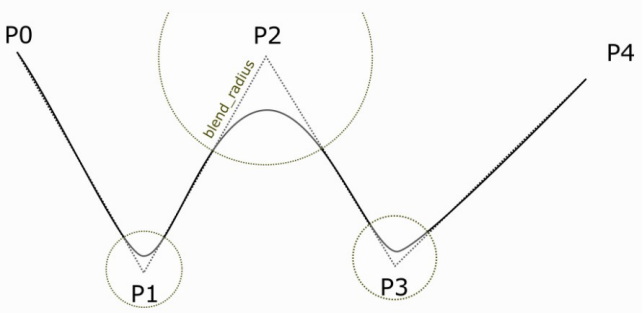


Figure 3.13: Robot path without stopping at the pre-defined waypoints [Pilz and KG (2019)].

The *blend_radius* enables the transition between the trajectories, and its magnitude defines the distance to the waypoint where the robot initialises the blend motion. Lastly, when the trajectory is outside of the *blend_radius* sphere around the waypoint, the robot returns to the trajectory that it would have taken without blending.

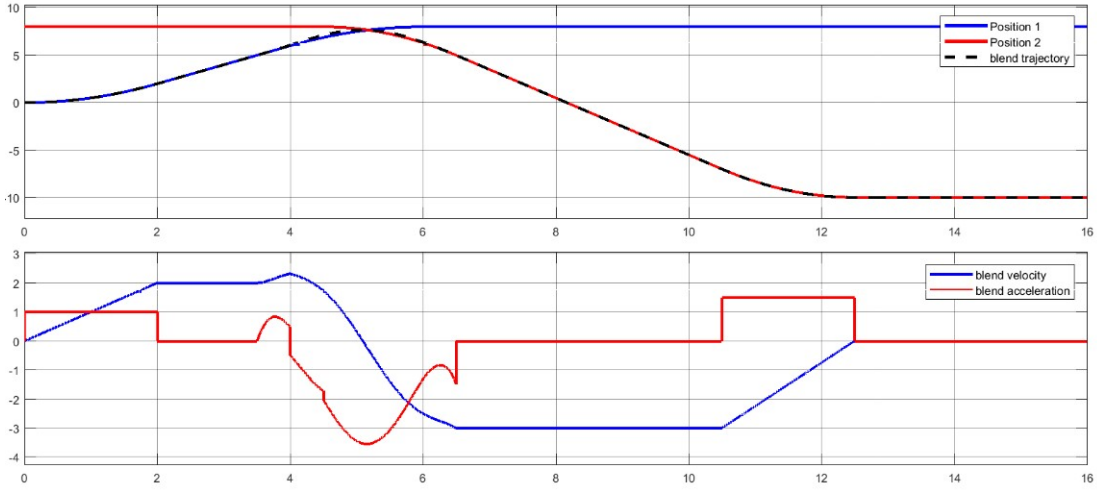


Figure 3.14: Trajectory profile of the robot position, velocity and acceleration [Pilz and KG (2019)].

The results, Figure 3.14, show that the trajectory transition at the waypoints, for instance, t$\in [4, 6]$, is smooth and the robot can perform a single trajectory without stopping at the waypoints. However, the trajectory only approximates to these positions and the acceleration has multiple discontinuities that cause jerk pikes and, thus, may hinder the robot.

[1] https://ros-planning.github.io/moveit_tutorials/doc/pilz_industrial_motion_planner/pilz_industrial_motion_planner.html

3.3 Human-like Arm Motion Generation

3.3.1 Human-like Arm Motion Characteristics

In the industry 5.0 is expected to have humans and robots coexistence in the same environment and work together to improve the human's quality of life [Gulletta et al. (2020)]. Recognised by the scientific community, a human-like motion has a substantial positive impact on human-robot collaboration. Such a motion ensures an easily predictable and understandable movement from a human's point of view since humans are familiar with those movements and, therefore, they may adjust their behaviour to avoid possible collisions or enhance the collaboration [Garcia et al. (2019)]. To perform movements with these features, the knowledge about humans' upper-limb motions must be passed to robotics.

In Figure 3.15 is illustrated a human arm, which has a total of seven revolute joints. The first three joints (q_1, q_2, q_3) represent the shoulder; the elbow is the fourth joint, q_4, and the last three joints (q_5, q_6, q_7) symbolise a spherical wrist. However, only 6 degrees of freedom of the musculoskeletal system are required to move in free space. This property is called redundancy, i.e., there are more DOFs than those necessary to perform a movement, which allows the selection of an ideal posture for a particular task [Burdet et al. (2013)].

Although there are infinitely joint configurations to reach and grasp an object in a workspace, humans can perform a movement in a completely natural and effortless way [Burdet et al. (2013),Wolpert and Ghahramani (2012)]. Nevertheless, the ability to perform complex actions without stereotyped effort is still not well understood [Meirovitch et al. (2016),Tsuzuki and Ogihara (2018)]. Thus, understanding how humans resolve such problems is a challenge addressed by many researchers. Based on behavioural observations, Fitts (1954) stated that movements are generated in a stereotypical way, which enhanced the human motion study and the development of computational approaches.

In order to analyse human-like movements, it is important to have a insight in the most relevant principles and metrics of human-like motion in robotics. According to Gulletta et al. (2020) study, the vast majority of studies regarding human-like movements consider the principles: biomimetics, uni-modal bell-shaped hand velocity profile, quasi-straight hand path, repeatability, trajectory smoothness. The most common principle is based on biomimetics, which consists on the extraction of bio-markers and of physical regularities from recorder human movements in order to then mimic arm trajectories. Although this method is intuitive and preferred, it requires sophisticate and expensive motion capture systems. Moreover, qualitative assessment is relatively common in human-like metrics, specifically the notions of legibility

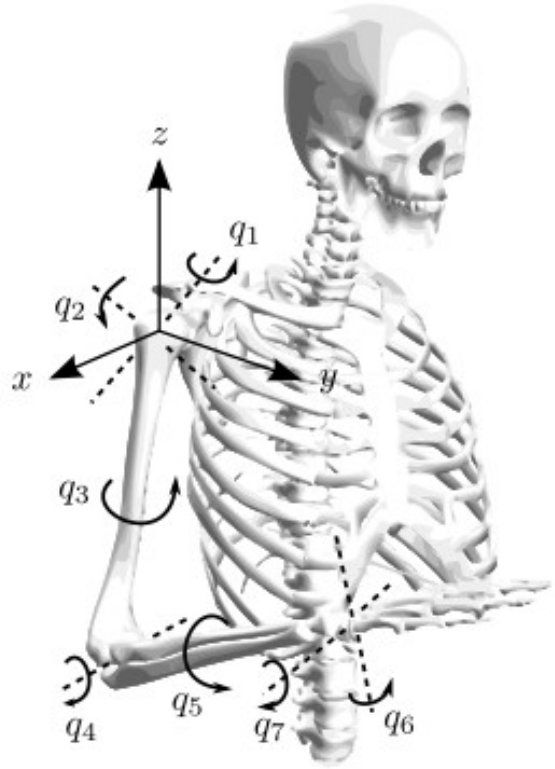


Figure 3.15: The kinematic model of human arm [Zanchettin et al. (2011)].

and predictability. For instance, a movement is categorized as predictable when the target of the robot matches human expectations, and legible when a human observer can easily infer the correct goal [Gulletta et al. (2020)].

The first experiments by Flash and Hogan (1985) demonstrated that upper-limb movements follow the minimum jerk principle. Particularly, it was discovered that in point-to-point trajectories, or reaching trajectories, the arm executes a movement in a straight line, however, slightly curved, where the velocity match a bell-shaped profile. Therefore, the speed of the arm increases gradually and reaches its peak around the middle of the movement. Furthermore, there is an inverse relationship between speed and curvature in extemporaneous drawing movements, a property known as 2/3 power law [Todorov and Jordan (1998)]. In Figure 3.16, one can see a comparison of the position, velocity, acceleration and jerk between a human point-to-point movement and the prediction from the minimum-jerk model.

Some researchers stated that the human Central Nervous System (CNS) decomposes a movement into several sub-movements [Dehghani and Bahrami (2020)]. For instance, Kang and Ikeuchi (1994) defined temporal segmentation of grasping tasks sequences based on human motion. Specifically, the author specified three phases in a task: *pre-grasp*, *grasp* and *manipulation*. For instance, in a pick-and-place task where a human needs to pick an object from a position and place it into another: the *pre-grasp*

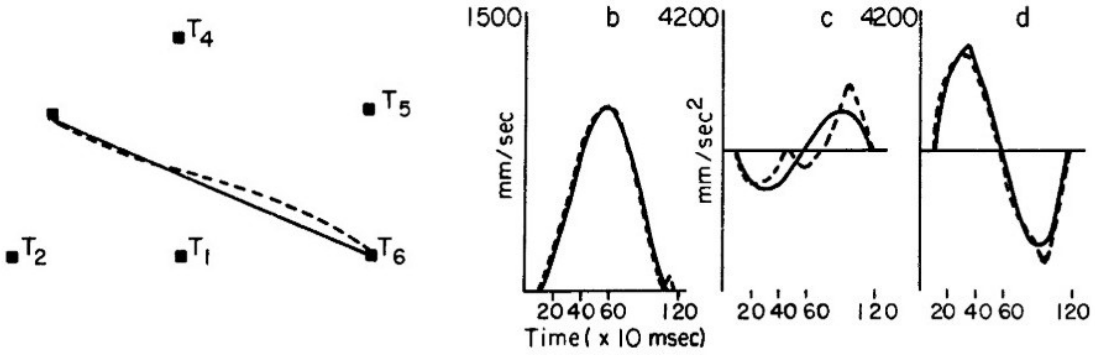


Figure 3.16: In solid line is represented the predicted movement from the mathematical model of a typical point-to-point movement. In dashed line is illustrated the evolution of the real movement. The velocity, acceleration and jerk are similarly illustrated in subfigures b), c) and d), respectively [An adaptation from Flash and Hogan (1985)].

phase represents the planning of the trajectory and its execution so that the object is successfully grasped; the following phase, *grasp*, is a transition from the *pre-grasp* to *manipulation* phase, where the object is picked and is ready to be moved. Next, the *manipulation* phase deals with the transportation of the object from the previous position to the goal position and, in the end, the object is released (For more details, see Martins de Sá (2018)).

Similarly, a common characteristic of human movements is that the hand preshape is affected by the object size and orientation, and object distance only influences hand transportation. This is known as the isochrony principle, which is a phenomenon of motor control, whereby the speed of the hand increases as its trajectory distance increases [Yokoyama et al. (2018)]. According to Martins de Sá (2018) and Gulletta et al. (2021), one can assume that every movement can be classified as *pick*, *place* or *move*, and is composed by the phases above mentioned.

Another research of interest was handled by Rosenbaum et al. (1995) that describes human's reaching movements planning. The author claimed that it is possible to specify the movement duration endogenously: the goal postures are selected considering the stored postures and are planned before the movement execution. Afterwards, in Rosenbaum et al. (2001), the same author improved the previous model to deal with grasping tasks. Analysing Figure 3.17, is interesting to note that the hand aperture increases gradually and it is adjusted when the hand velocity is low; the arm takes less time for speeding up than for slowing down; the angular velocity profiles of the elbow and shoulder are bell-shaped. In case of presence of obstacles a *back-and-forth* movement is summed to the *direct* movement, and a free-collision trajectory is achieved.

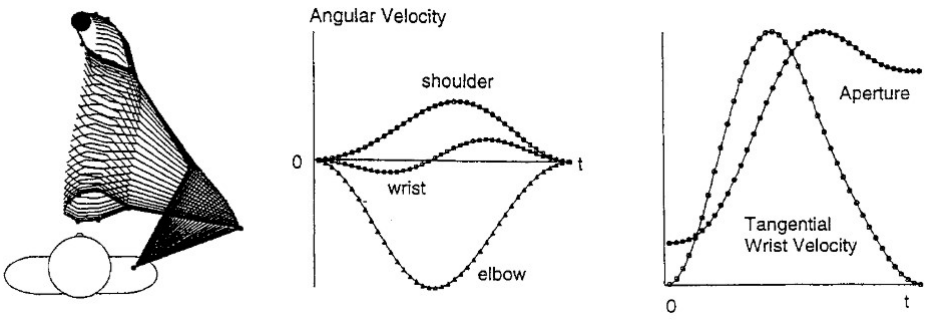


Figure 3.17: Simulated reach-and-grasp movement [Rosenbaum et al. (2001)].

3.3.2 Human-like Arm Motion Computational Models

According to Nguiadem et al. (2020), the human's movements characteristics already described can be achieved by data-based models and optimisation-based models. In the former, also called Learning by Demonstration, consists of two stages: (i) empirical behavioural data is gathered to construct a database for a specific motion; (ii) reproduction of similar motion using appropriate control models. However, the efficiency of this approach relies on the learning phase [Emadi Andani and Bahrami (2012)]. For instance, Park and Kim (2010) constructed an optimal database of human-like arm motion using an imitation learning algorithm to learn the arm motion primitives for real-time human-like trajectory planning. Specifically, imitation learning constructs a motion database that contains human-likeness derived from human arm motions, which is optimised by an Evolutionary Algorithm (EA) that selects minimal joint torques trajectories. The human's arm motion data are clustered according to the task-related conditions. The results proved that the proposed method is capable of learning minimal torque arm trajectories and generating human-like arm motion in real-time. However, the performance decreases as the size of the clusters increases.

More recently, Garcia et al. (2019) conducted an approach to endow an anthropomorphic dual-robot system, composed of two UR5 robots with 6 DOF's from Universal Robots, with natural human-like movements in a manipulation task. Specifically, the movements of a human operator solving a manipulation task are captured by using magnetic trackers and gloves with sensors. The demonstrated trajectories are converted into attractive potential fields over the $\mathbf{C}$-space. Then, the motion planning problem is solved using an RRT*-based algorithm with a stochastic gradient descent method to minimise the path-cost function. The human-likeness is achieved by the algorithm that navigates through the potential field and biases the three growths towards the human-like movements.

Regarding the optimisation-based models, a cost function representing a particular task is minimised to

get an optimal solution for that goal. This technique is commonly used to describe and reproduce voluntary human movements, e.g. hand jerk, which is the third time-derivative of hand position [Flash and Hogan (1985)], joint jerk [Piazzi and Visioli (2000)] or even a combination of several cost functions [Albrecht et al. (2011)]. Flash and Hogan (1985) developed a computational model, known as minimum-jerk model, that minimises the variation of the hand acceleration and aims to find the smoothest possible movement. The results showed that the model predicts qualitative features and quantitative details observed experimentally in human movements. Albrecht et al. (2011) used video sequences of humans performing everyday manipulation tasks to extract the posture of the human in each frame. This data was then clustered regarding the characteristics of human sub-movements, and a cost function that describes the human arm movement best was obtained. To select the cost function out of the hand jerk [Flash and Hogan (1985)], joint jerk[Piazzi and Visioli (2000)], and torque change[Uno et al. (1989)], a bi-level optimisation algorithm analyses the clustered data and determines a weight that corresponds to the cost function that most accurately represents the human movement. The optimal trajectories transferred to an iCub robot, a 53 degrees of freedom humanoid robot, allowed the performance of human-like motions.

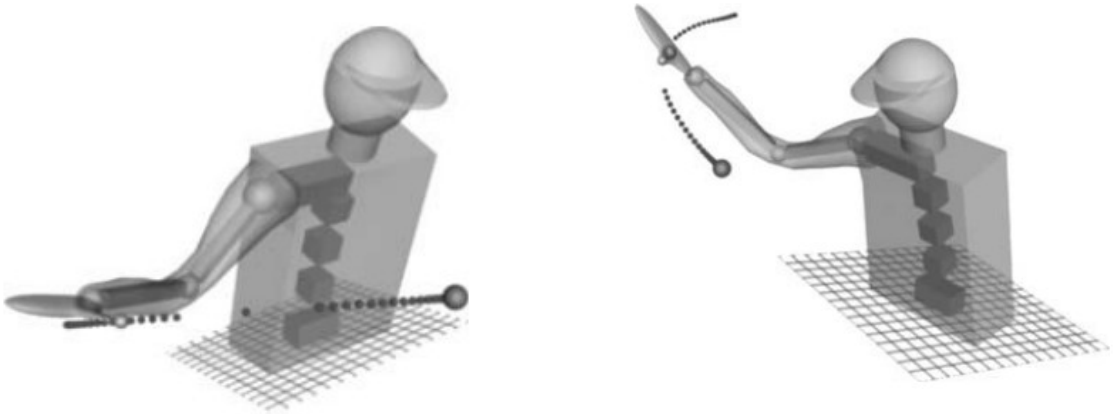
Zhao et al. (2014) studied methods to plan human-like reaching and grasping movements for robotic arms. More specifically, the minimum-jerk model is used to obtain a reaching human-like end-effector trajectory. Then, employing a Gradient Projection Method (GPM), the total potential energy is optimised, and the human-like joint trajectory is generated. In grasping movements, besides the position, the orientation is also considered, however, its trajectory is unknown. Thus, the author proposed a novel human-like based planner that combines the RRT method with the Gradient Projection Method (GPM-RRT). The RRT planner defines a trajectory for the end-effector orientation and biases the construction of a tree with optimal configurations provided by the GPM. Specifically, the end-effector trajectory is obtained by the goal biasing algorithm, and the human-like joint trajectory is generated by the GPM-RRT planner that minimises the synthesis of total potential energy and wrist discomfort.

Moreover, Wang et al. (2019) investigated patterns in human reaching and reach-to-grasp movements to solve the redundancy problem of an exoskeleton robot. The observations were conducted through a six-camera optical motion tracking system that captured the participant's arm motion. Then, statistical analysis revealed that the swivel angle could be constrained to a mean value in resolving the redundancy problem. Lastly, the minimum-jerk model integrated with the swivel angle generated well-accepted reference trajectories of all joints, with low error compared to the real movements.

3.4 Human-like Trajectory Generation through waypoints

The minimum-jerk model has demonstrated to be capable of generating a straight path between two points with null boundary conditions. However, this is considered a limitation in human-like motion, specifically, in presence of obstacles. Thus, some approaches have been proposed to tackle this limitation. Specifically, Flash and Hogan (1985) introduced a waypoint to replicate curve human movements and to deal with obstacle avoidance. To generate curve motion, it is assumed that the hand is required to pass through a specified point between the initial and final points. The time to pass through this waypoint is not known, however, is derived from an optimisation model. Such a problem is known as a dynamic optimisation problem with interior point equality constraints. To guarantee smoothness in the entire movement the velocity and acceleration at the waypoint must be continuous. Then, Abdel-Malek et al. (2006) extended this approach and introduced a methodology for predicting the path generated by humans in a natural motion of the torso and upper extremity (Figure 3.18). First, the path of the human hand in Cartesian space is determined through an extended minimum-jerk model. Then the correspondent joint values are calculated using B-splines with an optimisation based algorithm that aims to minimise the joint displacement, non-consistency, non-smoothness and non-continuity. Considering curved and obstacle-avoidance movements, Figure 3.18b, an artificial intelligence algorithm provides a waypoint that meets the kinematic constraints and, thus, the hand passes through that position and avoids collisions. The results show the prediction of smooth and pleasant movements of the upper body for point-to-point and curve paths. Furthermore, the hand moves slower at the beginning and the end than in the middle of the movement. This is known as the bell-shape velocity profile, which is a property of a smooth and natural human arm motion [Abdel-Malek et al. (2006)].

Besides, Sung et al. (2015) used the minimum-jerk with a waypoint approach to generate a humanoid smooth transition between motions, Figure 3.19a. The purpose of the waypoint is to establish the limits of some constraints as, for instance, the joint angle range, joint torque limits, and collision avoidance. The joint trajectory in the transition motion is achieved by the minimum-jerk model. However, the initial, final and waypoint time must be first calculated. More specifically, the initial and final transition times are determined by an optimisation problem that minimises the maximum value of the sum of squared torques. Then, the time of the waypoint is determined according to the calculation of the optimisation problem, which defines the time when the value of the square torque is a maximum. Once the time and trajectory are defined, if the constraints imposed are violated, the waypoint position is optimised using Semi Infinite Programming (SIP) [Hettich et al. (2009)]. As the Figure 3.19b illustrates, the calculated position was

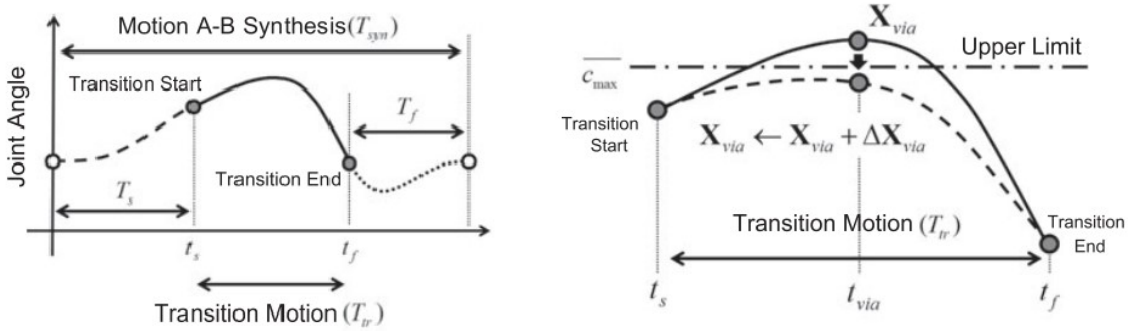


(a) Point-to-point straight trajectory.

(b) Curve motion generation for obstacle avoidance through a waypoint.

Figure 3.18: Minimum-jerk model motion experiments with an animate human [Abdel-Malek et al. (2006)].

above the joint upper limit, therefore the waypoint position must be optimised so that the trajectory meets the constraints. The experiments confirmed a smooth transition between walking and kicking motions of a humanoid robot. Unfortunately, the optimisation of the waypoint position requires a large computation time, which is a relevant limitation for motion planning problems in human environments.



(a) Smooth transition between two motions.

(b) Optimisation of the waypoint position so the imposed constraints are satisfied.

Figure 3.19: Smooth transition between two motions using the minimum-jerk model and its adjustment to respect the constraints [Sung et al. (2015)]

The research community has also used waypoints to mimic complex movements, such as writing and drawing. For instance, Wada and Kawato (1995) extracted waypoints from human movements and were able to reproduce handwritten characters. However, the problem of extracting optimal waypoints from a given trajectory is complex, since there is an infinite number of possible positions. Thus, the authors proposed an approach called waypoint estimation algorithm to estimate the waypoints position and time.

This model is formulated to find the minimum number of waypoints necessary to reproduce the given trajectory within an error bound. Given the waypoints, the model can generate a complete trajectory using the FIRM model, which is based on an approximate minimum torque-change model. First, the torque is calculated from the joint angle trajectory, satisfying the terminal conditions, using an Inverse Dynamics Model (IDM). The terminal condition errors are founded and the torque is smoothed by a Forward Dynamics Model (FDM). Then, a compensatory trajectory which cancels the terminal condition errors is determined. Computing this algorithm cyclically, a smooth trajectory that passes through waypoints and attends the robots dynamics is generated.

Figure 3.20 explains the process to estimate a minimal number of waypoints from a given trajectory. Firstly, a direct trajectory from the starting point and the final point is created by using the minimum-jerk model. Secondly, the point on the given trajectory with the maximum error value, lower than an established threshold, between the given trajectory and the generated trajectory is selected as a waypoint candidate. Then, a trajectory based on the minimum-jerk model from the start point to the end point passing through a waypoint is generated and added to the previous trajectory. By repeating the previous steps a set of waypoints is found.

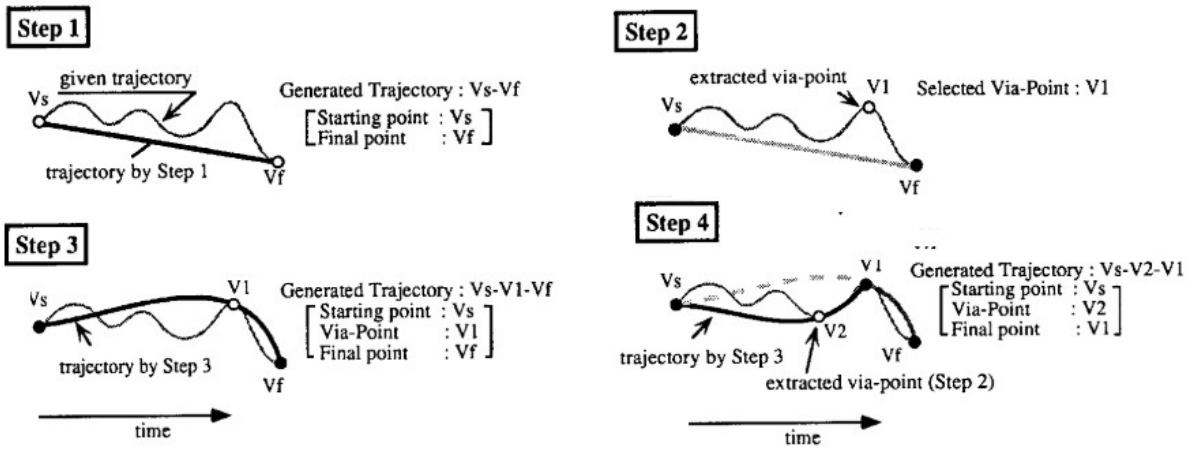


Figure 3.20: Algorithm for extracting waypoints based on the minimisation of the square of the error between the given trajectory and generated trajectory [Wada and Kawato (1995)].

After having the estimation of the position of the waypoints, the compensatory trajectory through the waypoints is determined by the FIRM method, which is based on the minimum commanded torque change model. However, when the arm dynamics are linearly approximated to equation 3.9, the trajectory generation is equivalent to the minimum-jerk model.

$$\tau^j = I^j \ddot{\theta}^j \qquad (j = 1, ..., M) \tag{3.9}$$

where τ^j is the torque generated by the jth actuator. I^j and $\ddot{\theta}^j$ are the link's inertia and the jth joint's angle acceleration, respectively.

Illustrated in Figure 3.21, the compensatory trajectory generation through waypoints consists in 4 steps. The first, similarly to the waypoints extraction algorithm, is the generation of a direct trajectory from the initial point to the end point using the minimum-jerk model. Then, from the waypoints previously extracted, is selected the one which minimises a smooth criteria, and a trajectory passing through the waypoint is generated also using the minimum-jerk model and added to the previous generated trajectory. Specifically, in the third step, is generated a trajectory, which starts from the previous waypoint $V4$ and ends in final point Vf passing through the selected waypoint $V5$, and it is added to the trajectory obtained in step 2. The same process is continuously repeated and a trajectory that interpolates the minimum number of waypoints that guarantees smoothness is obtained. The results show that the experiments were successfully reproduced to estimate the minimum number of waypoints necessary for the task, Figure 3.22.

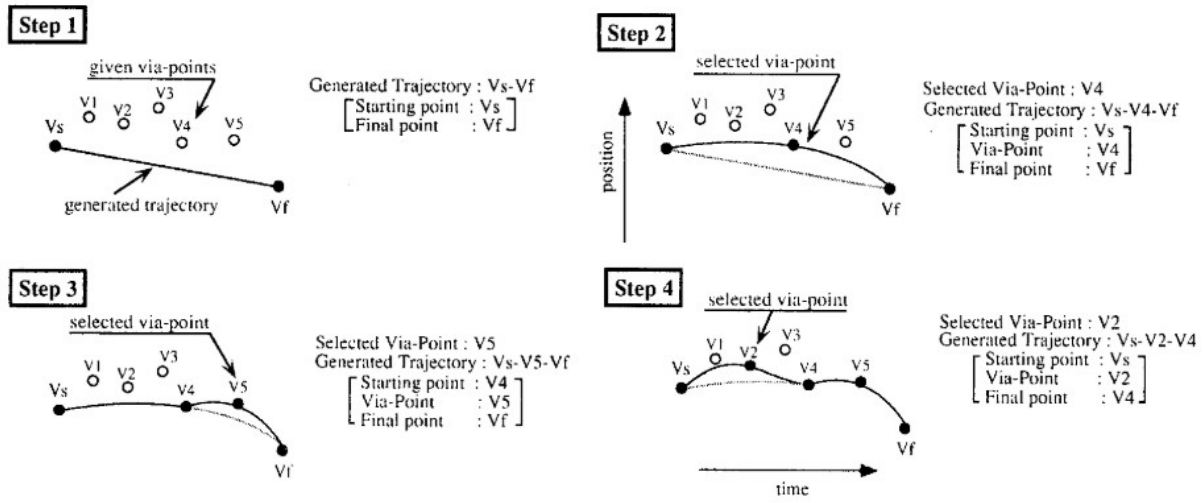


Figure 3.21: Algorithm for producing the trajectory through waypoints by the FIRM model [Wada and Kawato (1995)].

Since in Wada and Kawato (1995) method, when extracting waypoints both position and time are obtained, the generation of slow and fast movements for the same motion is not possible. Thus, in Wada and Kawato (2004), the method FIRM was extended to do not need prior temporal information about via-points. More specifically, a cost function is optimised on the condition that waypoint time average of the integration of the square of the smoothness of the motor command between each via-point is equal. Therefore, the time optimisation model decides the waypoint time that minimises the commanded torque change, equation 3.10.

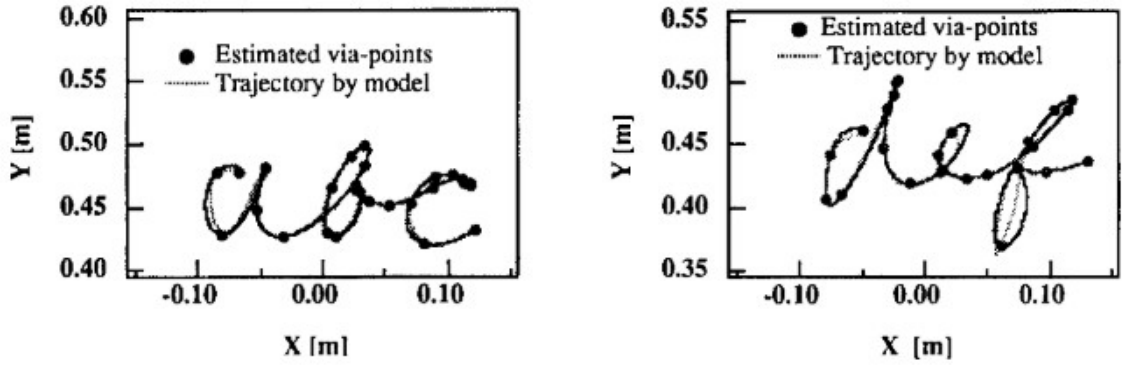


Figure 3.22: Experiment of handwritten task. On the left show the trajectory for 'abc' and on the right for 'def' letters [Wada and Kawato (1995)].

$$C_i^{via} = \frac{1}{t_i - t_{i-1}} \int_{t_{i-1}}^{t_i} \sum_{k=1}^{K} \left(\frac{d\tau_i^k}{dt} \right)^2 dt \tag{3.10}$$

where i is the indices of the waypoints $(i = 1, 2, ..., n)$; t_i is the time in the waypoint i; τ^k is the commanded torque of joint k and K is the number of joints.

Afterwards, Saito et al. (2006) evaluated the same method in human arm reaching movements with waypoints. More specifically, the author investigated whether the human's Central Nervous System (CNS) generates a complex trajectory in order to equalise the average duration of the commanded torque changes between each waypoint interval. Therefore, to validate this concept, 30-40 trials composed of two reaching tasks with one waypoint and one reaching task with two waypoints, Figure 3.23, were evaluated.

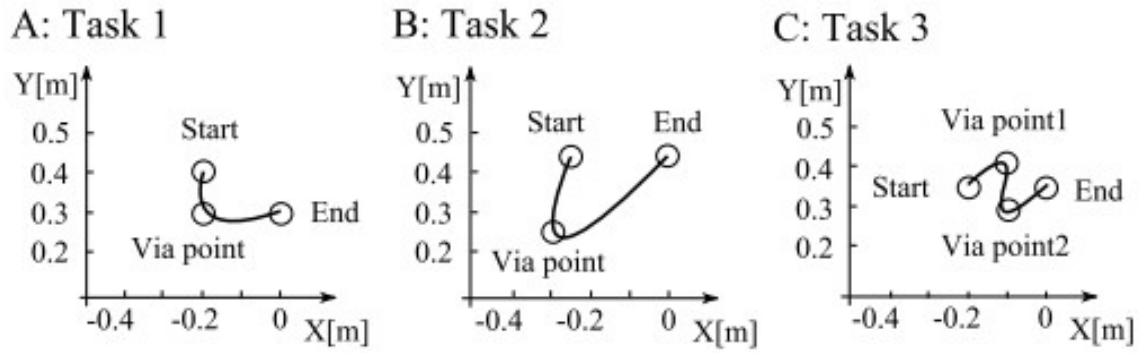


Figure 3.23: Human arm reaching movement with waypoints. In Task 1 and Task 2 was performed a reaching movement with one waypoint and in Task 3 with 2 waypoints [Saito et al. (2006)].

To calculate the value waypoint time average, C_i^{via}, equation 3.10, the waypoints time of the measured trajectories are estimated by the algorithm developed in Wada and Kawato (1995), and the waypoint

locations are estimated by the locations of the local minimum of tangential velocity [Saito et al. (2006)]. Furthermore, the trajectories which include some correction movements, a local minimum velocity lower than 5% of maximum velocity and did not have a local minimum in the velocity profile were rejected. The results of the waypoints estimation are shown in Figure 3.24.

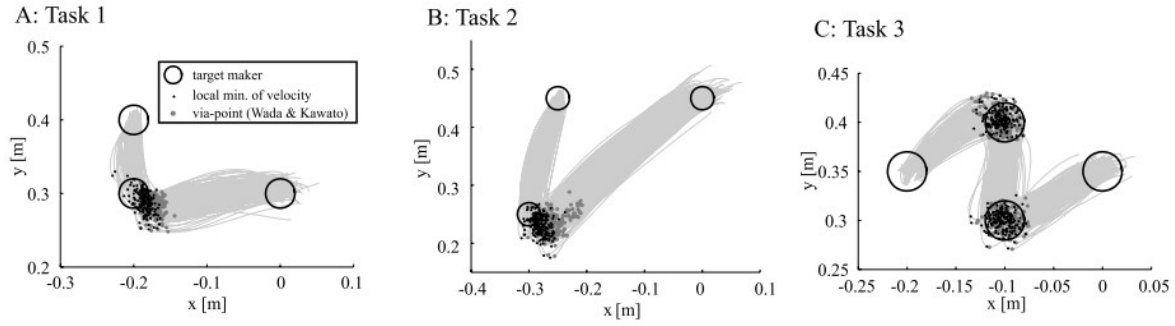


Figure 3.24: Waypoints estimated by Wada & Kawato's method and local minimum of tangential velocity [Saito et al. (2006)].

3.5 Discussion

A systematic literature review on the most used techniques of human-like arm motion generation and trajectory generation through waypoints has been presented.

Currently, the most used techniques to generate trajectories through waypoints are based on polynomials, splines and blends functions. The polynomial method is easily described to attend the waypoints constraints, however, more constraints demands higher polynomial degree, which results in high computation costs, vibrations and numerical error [Biagiotti and Melchiorri (2008)]. An effective alternative is the usage of splines. Cubic splines are commonly used due to the assurance of continuous velocity and acceleration and prevent large oscillations of the polynomial trajectories, but also contains jerk (derivative of the acceleration) spikes. Many researchers consider that such methods that do not avoid the jerk spikes, created by the discontinuities of the acceleration, should be avoided to prevent severe problems of the robot structure such as unacceptable noise, wear and bad dynamics [Williams (2013)].

To address the discontinuities in the acceleration were presented B-splines methods, which are capable to produce continuity in the position, velocity, acceleration and jerk. Despite the smoothness, it presents high values of jerk. Another approach commonly used in industry is based on blends. For instance, the MoveIt library provides the possibility to generate trajectories through waypoints using the blend approach. However, this method is not accurate since it only approximates and not interpolates the waypoints.

A very interesting approach considering waypoints was presented by Flash and Hogan (1985). The authors studied the coordination of voluntary arm movements and formulated a mathematical model capable of predicting qualitative and quantitative features observed in human movements. To reproduce human curve movements and to avoid obstacles, the authors used the minimum-jerk model with constraints in position, a waypoint. Sung et al. (2013) used the same approach to perform a kicking motion in a humanoid robot, specifying position and velocity constraints by using waypoints.

The most used techniques to generate trajectories are focused on the optimization of some criteria, especially to improve productivity, efficiency and reduce costs. For instance, the minimization of the duration of a task [Gasparetto and Zanotto (2008)], and the minimization of the energy consumption [Liu et al. (2018)]. However, due to the emergence of industry 4.0 and 5.0, and consequent integration of collaborative robots in workplace shared with human operators, the robotics research community has changed the attention to methods capable of executing movements similar to those executed by humans, which have positive effect in humans productivity and well-being [Gulletta et al. (2020), Bortot et al. (2013), Koppenborg et al. (2017)].

From this literature review we conclude that there are no methods capable of completely attending the needs of industry 4.0, and even less the upcoming industry 5.0. Currently, the methods in the industry are focused on production and company goals, and take no attention to the humans needs, such as removing them from tired, tedious and repetitive tasks, making them psychology and physically exhaustive, which contributes for medium term diseases. Thus, to meet human needs is fundamental a coexistence and interaction between humans and robots, accompanied by predictable, fluent and understandable robot movements. This increases the safety and humans' trust, which results in better and fluent human-robot interaction, and therefore, production speed and quality increases.

In addition to this, Industry 4.0 also requires more autonomy and flexibility from the industry due to the rise of mass customisation and the need to meet the consumer demands [Kurt (2019), El Zaatari et al. (2019)]. Therefore, easy programming methods with intuitive and easy-to-use UI are absolutely important for the upcoming years, so that all companies can maintain the competitive edge.

The existing methods that bring flexibility and allow companies to easily reprogram robots are not capable of performing human-like movements. Thus, such methods do not accomplish the best possible outcome.

For all the above mentioned reasons, a combination of easy programming methods, such as programming through waypoints, with human-like movements is, from our perspective, a great solution that attends, at the same time, humans and industry needs.

Part III

Materials and Methods

Chapter 4

Collaborative Robot: UR10e

This chapter analyses the UR10 robotics platform from Universal Robots and Uniggriper Co/Light Regular, which are used in the implementation and validation of this project. First, a brief introduction to the description of the robot and gripper is made. Then, some theoretical concepts related to forward and inverse kinematics will be addressed. Finally, the kinetic model of the robot is thoroughly explored.

4.1 Introduction

The UR10 robot is one of the 4 available models of collaborative robots developed by Universal Robots, which started the introduction of cobots in 2009 with the aim to automate and simplify repetitive and monotonous processes in a more flexible, safe and easy-to-use way. The other models are UR3, UR5, and UR16, where the number indicates the maximum payload in kg that the robot supports.

Universal Robots' robots have a control box and a teach pendant with embedded software (*Polyscope*), which is a user-friendly graphical user interface that makes programming new tasks in an intuitive and easy to use management that makes them accessible for every operator, including those with less experience.

According to Universal Robots [1], their robots are suitable for many applications - assembly, dispensing, finishing, machine tending, welding and quality inspection - in a wide variety of industries such as automotive, electronics, food and beverage, medical and cosmetics.

[1] https://www.universal-robots.com/pt/aplicaciones/

Figure 4.1: Collaborative robots models from Universal Robots: UR3, UR5, UR16 and UR10 respectively.

4.2 Specification

The UR10 robot is a 6-DOF (Degrees of Freedom) serial robotic arm with six active rotational joints, each with a 640° rotation range and a 120°/s - 180°/s speed limit. It can lift up a maximum payload of 10 kg, however, it decreases as long as the distance between the centre of the tool output flange and the centre of gravity increases. Furthermore, it reaches an area of 1300mm from the base joint, however, the positions in the cylindrical space directly above and under the base should be avoided due to singularities.

The robot can be controlled manually or autonomously. When controlled manually, the user can move each joint separately or move all joints synchronously by setting a goal position for the end-effector of the arm. The robot also has a *Freedrive mode* on the user interface *Polyscope* where it is the user who controls physically the robot. In the automatic mode, the robot can only perform pre-defined tasks, therefore the *MoveTab* and *Freedrive Mode* are unavailable in the teach pendant user interface. Independent of the operation mode, in case of collision an emergency stop is automatically activated due to force sensors embedded in all joints of the robot. A more detailed specification about the robot features and control is presented in the table 4.1

Table 4.1: UR10 robot technical details

Specifications	
Maximum Payload	10 kg
Reach	1300 mm
Degrees of Freedom	6
Footprint	190 mm
Weight	33.5 kg
Operate temperature	0-50°C
Performance	
Safety	17 safety functions
Certifications	EN ISO 13849-1, PLd Category 3, and EN ISO 10218-1
Force sensing - Range	100 N
Force sensing - Precision	5.0 N
Force sensing - Accuracy	5.5 N
TCP speed	1 m/s
Repeatability	0.05mm
Joint Range	360°
Control box	
Weight	12 kg
Operating Temperature	0-50C
Digital input ports	16
Digital output ports	16
Analog input ports	2
Analog output ports	2
I/O power supply	24V 2A
Communication	500 Hz Control frequency; Modbus TCP; PROFINET; Ethernet/IP

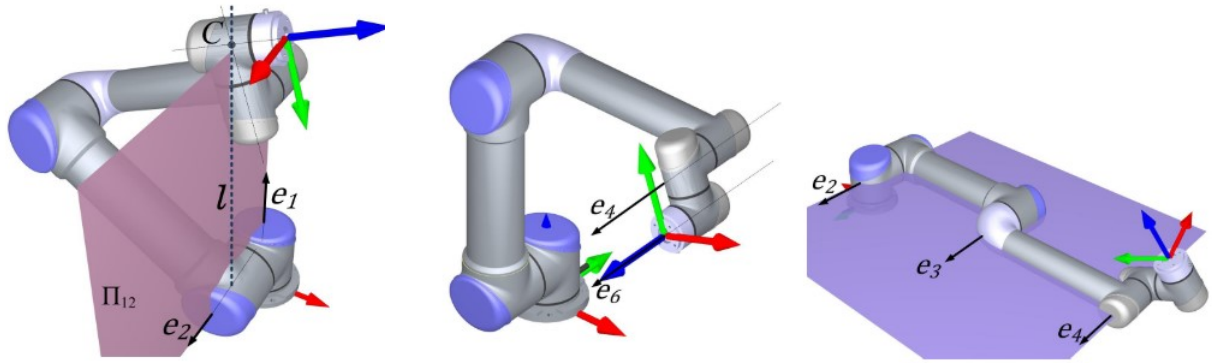
4.2.1 Singularities

Although the robot has a workspace of 1300 mm, there are some areas that are not advisable to reach, and when that happens it is called that a *singularity* occurred. According to ISO-120218, a singularity is defined as a *" condition caused by the collinear alignment of two or more robot axes resulting in unpredictable robot motion and velocities"* [FarzanehKaloorazi and Bonev (2018)]. Thus, the robot has safety controllers that automatically check for singularities to prevent any damage. There are three types of singularities: wrist, shoulder, and elbow singularities, Figure 4.2.

Wrist singularities: Happen when the axes of joints 4 and 6 are parallel, which corresponds to a spin of 180 degrees instantaneously. This can be seen in the Universal Robots when $\theta_5 = 0$, $\theta_5 = \pm 180$ or $\theta_5 = \pm 360$;

Elbow singularities: Happen when the exes of joints 2,3 and 4 are co-planar, inhibiting the elbow joint to move. This can be seen in the Universal Robots when $\theta_3 = 0$;

Shoulder singularities: Happen when the intersection point of joints 5 and 6 aligns with the plane passing through the axes of joints 1 and 2, causing joints 1 and 4 to spin 180 degrees instantaneously $\theta_3 = 0$;



(a) Shoulder singularity (b) Wrist singularity (c) Elbow singularity

Figure 4.2: Shoulder singularity, wrist singularity and elbow singularity of a Robot from Universal Robot [Adapted from FarzanehKaloorazi and Bonev (2018)].

4.3 Kinematic Model

The space in which the end-effector position and orientation is represented is of great relevance. In this project, the user defines waypoints where the robot must pass during the movement, and they can be provided in the joint space or in the operational space. The process to obtain the position of the joints of the manipulator through the position of the end-effector is called Inverse Kinematics. On the contrary, the process to obtain the position of the end-effector through the value of the joints is called Forward Kinematics.

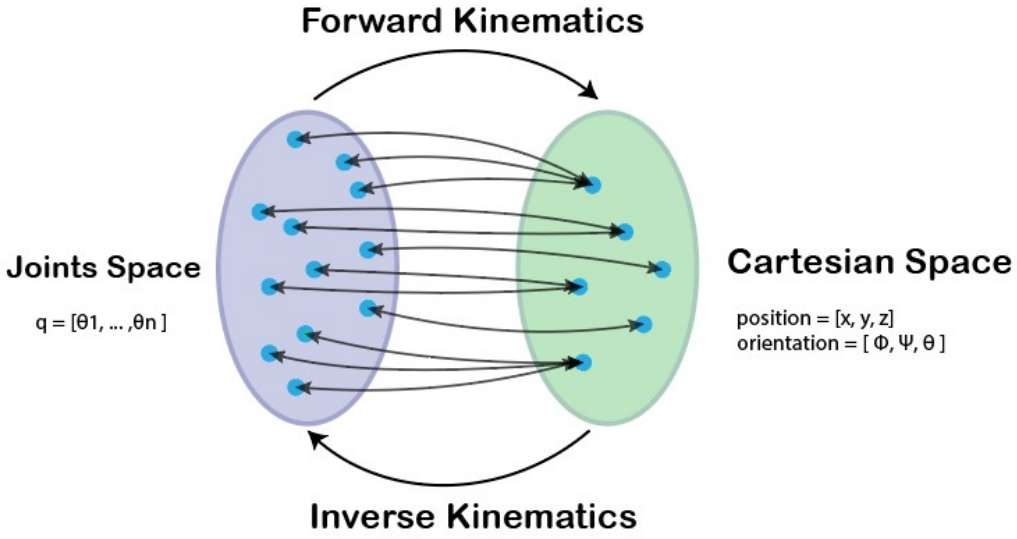


Figure 4.3: Relation between the forward kinematics and inverse kinematics.

The kinematic analysis is most commonly achieved through the Denavit-Hartenberg convention[2], where the robot is described kinematically by defining four parameters for each link. In essence, the relation between two consecutive coordinate axis $link_i$ and $link_{i+1}$ can be obtained through the application of the four following transformations [Jazar (2008)]:

- the translation along the x_i axis of a distance - a_{i-1};
- the rotation around the x_i axis of an angle - α_{i-1};
- the translation along the z_i axis of a distance - d_i;
- the rotation around the z_i axis of an angle θ_i.

[2] A convention introduced by Jacques Denavit and Richard Hartenberg in 1955.

The transformations between the reference frames along the robot's links results in the robot's kinematic equations. However, before determining kinematics, it is indispensable to first analyse the robotic manipulator's mechanical structure. A manipulator can be schematically represented from a mechanical viewpoint as a kinematic chain of rigid bodies (**links**) connected by means of rotational or prismatic **joints** [Jazar (2008)].

The UR10e has 6 rotational joints that can rotate $\pm 360°$and ten displacements, refer to Figure 4.5 and table 4.2. This mechanical analysis was conducted using the documentation available in the Universal Robots' website [3]. Across their documentation, joint 1 is referred as "base", joint 2 "shoulder", joint 3 "elbow", joint 4 "wrist 1", joint 5 "wrist 2", and joint 6 "wrist 3".

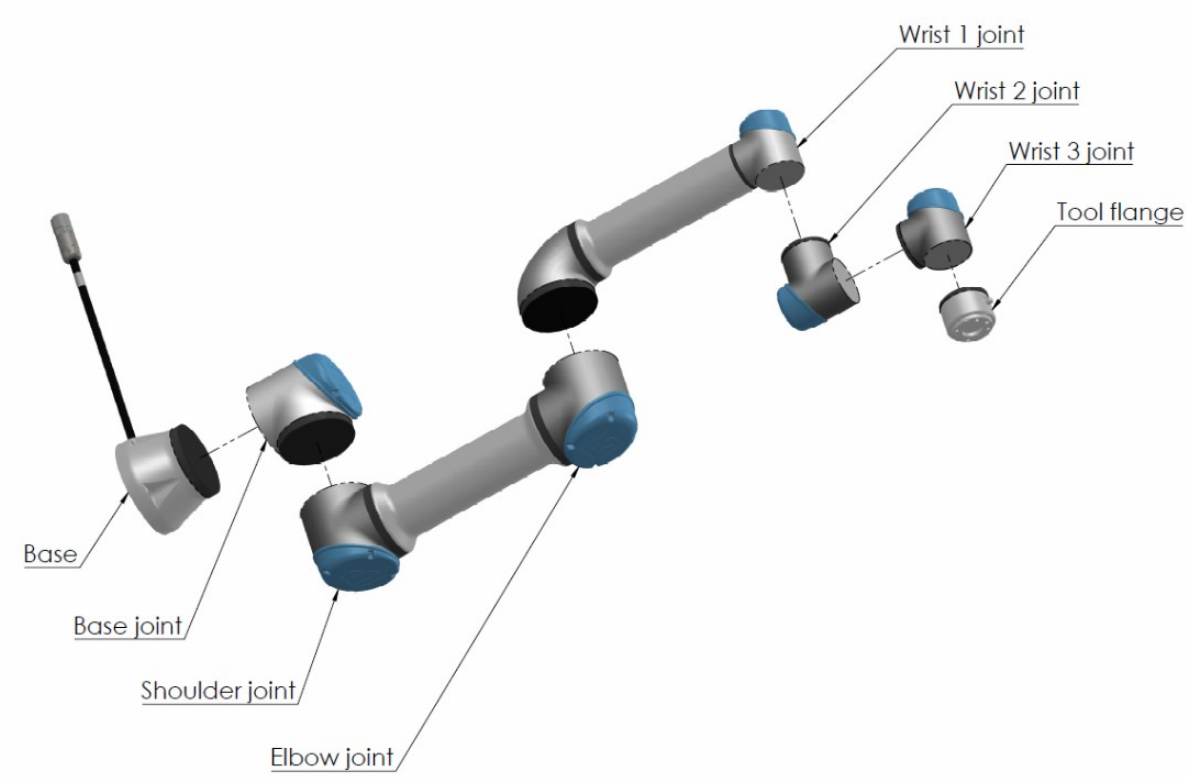


Figure 4.4: Joints designation of robots from Universal Robot [Universal Robots (2015)].

[3] https://www.universal-robots.com/pt/

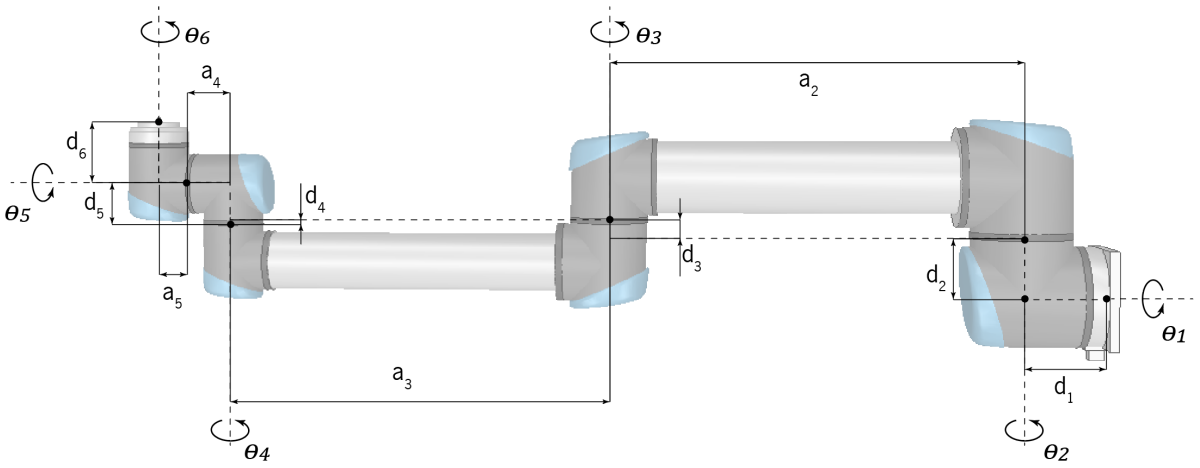


Figure 4.5: UR10e mechanical structure

Table 4.2: UR10e link distances and joint limits

Joints	Value (degrees)	***d***	Value (meters)	***a***	Value (meters)
θ_1	$[-360; 360]$	d_1	0.181	-	-
θ_2	$[-360; 360]$	d_2	0.088	-	-
θ_3	$[-360; 360]$	d_3	0.020	a_3	0.613
θ_4	$[-360; 360]$	d_4	0.001	a_4	0.571
θ_5	$[-360; 360]$	d_5	0.060	a_5	0.060
θ_6	$[-360; 360]$	d_6	0.117	a_6	0.067

4.3.1 Forward Kinematics

To derive the forward kinematics, one must first position the manipulator in its home pose, and secondly attribute a reference frame $\{x_i, y_i, z_i\}$ for every joint. In this analyse, the manipulator's home pose is defined as all joint values equal to zero, which determine an upright pose.

The reference frames {0} to {6} are represented in Figure 4.6 (where the x, y and z components are represented with red, green, and blue arrows respectively), as well as the robot's home pose. To attribute these reference frames, three basic rules were followed: a) the right-hand rule; b) the location of the frame

is the centre of the corresponding joint, *e.g.*, reference frame {2} is located at the centre of joint 2; and c) the z-axis must indicate the rotational axis of the joint [Ben-Ari et al. (2018)].

Reference frame {0} is the origin of the robot, and is located in the centre of its base. From henceforth, reference frames [{1}, {2}, {3}, {4}, {5}, {6}] all follow the above stated rules. However, due to the order of multiplication of the transformations in the modified Denavit-Hartenberg convention, it was necessary to add two auxiliary reference frames: {4'}, and {5'} located at the centre of joint 5.

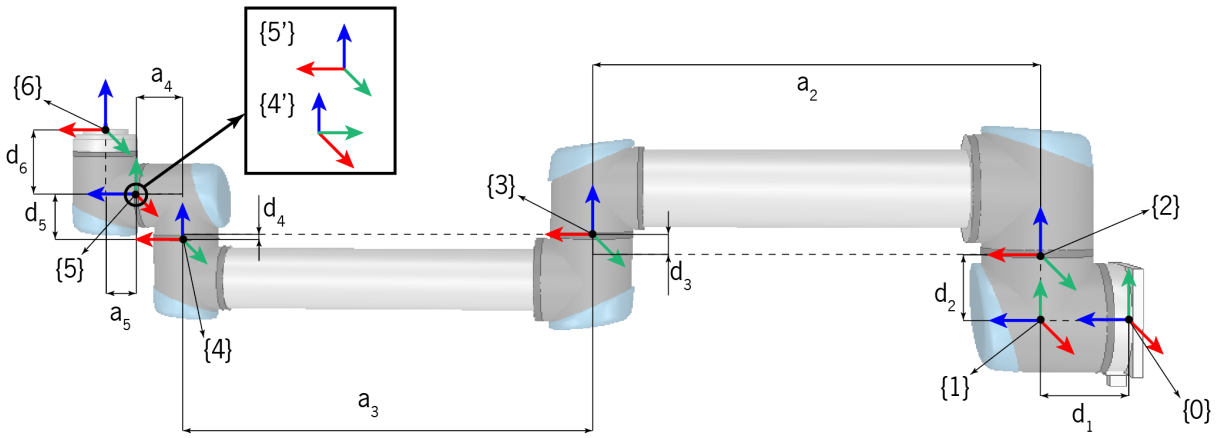


Figure 4.6: Reference frames along the UR10e structure.

After attributing the reference frames, it is possible to compute the transformations that occur between them, using the Denavit-Hartenberg parameters mentioned previously, table 4.3.

Table 4.3: UR10e Denavit-Hartenberg transformations between reference frames.

$^{i-1}\boldsymbol{T_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
$\{0\} \rightarrow \{1\}$	0	0	d_1	θ_1
$\{1\} \rightarrow \{2\}$	$-\pi/2$	0	d_2	$\theta_2 - \pi/2$
$\{2\} \rightarrow \{3\}$	0	a_2	d_3	θ_3
$\{3\} \rightarrow \{4\}$	0	a_3	d_4	θ_4
$\{4\} \rightarrow \{4'\}$	0	a_4	d_5	$\pi/2$
$\{4'\} \rightarrow \{5\}$	$\pi/2$	0	0	θ_5
$\{5\} \rightarrow \{5'\}$	$-\pi/2$	0	0	$-\pi/2$
$\{5'\} \rightarrow \{6\}$	0	a_5	d_6	θ_6

Using a set of multiplications of Denavit-Hartenberg matrices and a table of Denavit-Hartenberg parameters, the final result is a transformation matrix of a final system of coordinates in relation to the initial one [Jazar (2008)].

The modified Denavit-Hartenberg homogeneous transformation $^{i-1}T_i$, from frame $i-1$ to i is defined in equations 4.1, 4.2, and 4.3.

$$^{i-1}T_i = Rot_{x(\alpha_{i-1})} * Trans_{x(a_{i-1})} * Trans_{z(d_i)} * Rot_{z(\theta_i)} \tag{4.1}$$

$$^{i-1}T_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

$$^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & \alpha_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

This matrix is composed by the rotation and translation matrix that generate the required motion from frame $i-1$ to frame i:

$$^{i-1}T_i = \left[\begin{array}{c|c} ^{i-1}\boldsymbol{R}_{i(3x3)} & ^{i-1}\boldsymbol{P}_{i(3x1)} \\ \hline 0 & 1 \end{array}\right] \tag{4.4}$$

In equation 4.4, $^{i-1}P_{i(3x1)}$ represents the translation from reference frame $i-1$ to reference frame i, and matrix $^{i-1}R_{i(3x3)}$ represents the rotation matrix in relation to the same frame. For instance, knowing the matrix $^0T_6$, which is the transformation matrix of the end-effector in relation to the base, it is possible to determine the end-effector orientation: Roll(ϕ_e), Pitch (ψ_e) and Yaw (θ_e); through the sub-matrix $^0Rot_6$, and the end-effector position (x_e, y_e, z_e) through the matrix $^0P_6$.

Considering the first line of the table 4.3, the transformation between the origin frame and the first joint is defined by the matrix 4.5.

$$^0T_1 = \begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0 & 0 \\ sin(\theta_1) & cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Having the homogeneous transformation matrices defined along the robotic arm, the transformation from the robot base to robot end-effector is given by:

$$^0T_6 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_{4'}\,{}^{4'}T_5\,{}^5T_{5'}\,{}^{5'}T_6 \tag{4.6}$$

Using the same process as in equation 4.6, it is also possible to determine the transformation matrices of the other joints in relation to the base (equations 4.7 - 4.12).

$$^0T_{Base} = {}^0T_1 \tag{4.7}$$

$$^0T_{Shoulder} = {}^0T_2 \tag{4.8}$$

$$^0T_{Elbow} = {}^0T_3 \tag{4.9}$$

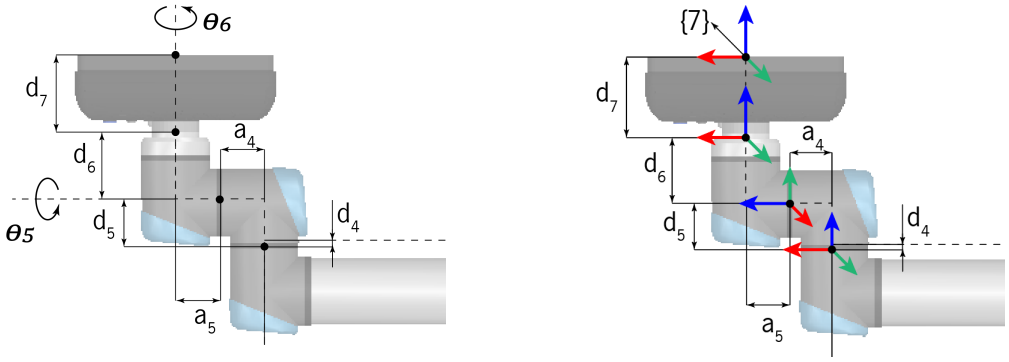
$$^0T_{Wrist1} = {}^0T_4 \tag{4.10}$$

$$^0T_{Wrist2} = {}^0T_5 \tag{4.11}$$

$$^0T_{Wrist3} = {}^0T_6 \tag{4.12}$$

4.3.2 Vacuum End-effector

An important aspect to be mentioned is the end-effector that is used to validate the proposed method. The considered end-effector is the UniGripper Co/Light Regular [4], which is a collaborative vacuum gripper. The gripper has a height of 101 mm, which means the robot's tip is dislocated an additional 101 mm (Figure 4.7a). Kinematically it is necessary to add a new reference frame, Figure 4.7b, and alter the Denavit-Hartenberg parameters (4.3) by adding a new line that corresponds to a translation on the z-axis, table 4.4.



(a) Mechanical structure

(b) Reference frames

Figure 4.7: Robotic system mechanical structure and reference frames (UR10e with the UniGripper)

Table 4.4: Robotic system Denavit-Hartenberg parameters (d_7 = 101 mm).

${}^{i-1}\boldsymbol{T_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
$\{6\} \rightarrow \{7\}$	0	0	d_7	0

In terms of forward kinematics this means that there will be an additional transformation, ${}^{6}T_7$:

$$ {}^{6}T_7 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13} $$

[4] https://www.unigripper.com/en/colight.html

Additionally, it is necessary to compute a new complete transformation matrix ${}^{0}T_{7}$, that is given by:

$$ {}^{0}T_{7} = {}^{0}T_{6} \times {}^{6}T_{7} \tag{4.14} $$

4.3.3 Inverse Kinematics

In this section, a brief and concise inverse kinematics of the UR10 robot is introduced. The literature found on the UR robot's kinematics was not very clear and detailed [Kebria et al. (2017); Andersen (2018); Hawkins (2013); Sun et al. (2017)]. This lack of research was the motivation to present a paper with an extension and more detailed approach of Universal Robots kinematics, which can be found in the Appendix A.

There are two mains strategies for solving the inverse kinematics of a robotic manipulator: *closed-form solutions*, which use an analytical approach and can be based on geometrical properties; and *numerical solutions*, which use iterative techniques to determine the joint angles, however, it demands more computational time. The strategy chosen for this project is the closed-form approach.

As demonstrated in Appendix A and illustrated in Figure 4.8, the Inverse Kinematics of the UR10 robot present 8 possible joint configurations for a given robot end-effector pose.

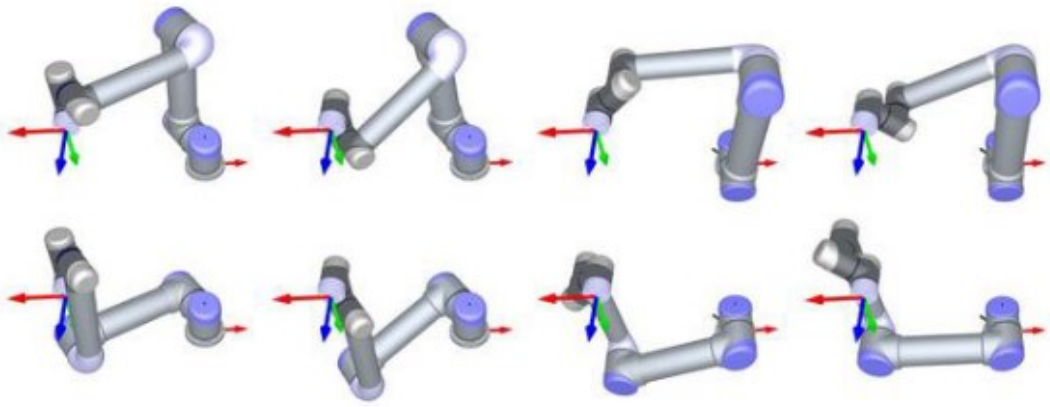


Figure 4.8: 8 possible joint configurations for a desired end-effector pose [Oosterwyck (2018)].

Hence, the space in which the waypoints are defined is essential to accomplish the task and demands the most attention from the user when selecting them. Specifically, if the user defines a waypoint in the cartesian space, so there would exist 8 possibilities of joints configurations to achieve that pose (refer to figure 4.8). In case the robot joints configuration is not important for the task, and only the end-effector pose, the planner could automatically select the best possibility to accomplish that pose. However, if the

joints configuration defined by the user is mandatory, the user can not define the waypoints in the cartesian space and must define in the joint space.

Therefore, in this dissertation the waypoints are obtained only in the joints space, and thus inverse kinematics are avoided. In this way, we can guarantee that the configurations selected by the operator are successfully accomplished.

Chapter 5

Optimal Control

In this chapter, some concepts of variational calculus and optimal control theory [Bryson and Ho (1975); Pontryagin (1986)] are applied to find the trajectory that minimizes the criterion function, subject to dynamic constraints imposed by the system differential equations and algebraic constraints. These concepts are fundamental to understand chapter 6. Firstly, the emergence of optimal control from variations in the calculation is briefly introduced. Then, the simplest optimal control problem with no path or end-points constraints is described by introducing the Hamiltonian function. The last section analyses optimal control problems with equality state variable constraints in unspecified time $t_1 \in [t_0, t_f]$. A deeper analysis of constrained optimal control is found in books Bryson and Ho (1975) and Pontryagin (1986).

5.1 Introduction to Optimal Control

The optimal control theory has emerged from a generalization of the calculus of variations over more than three centuries [Liberzon (2011)]. The calculus of variations was introduced by Johann Bernoulli who challenged other mathematicians to solve the *brachistochrone* problem: *"if a particle moves, under the influence of gravity, which path between two fixed points enables the trip of shortest time?"* [Chachuat (2016)].

Considering a function $L = L(t, x, \dot{x})$, where t is an independent coordinate, x an dependent coordinate and $\dot{x}$ its derivative with respect to the independent coordinate. Euler formulated the problem by determining the curve $x = x(t)$ where $a \leq t \leq z$, which make the definite integral $\int_a^z L(t, x(t), \dot{x}(t))dt$ extremal. Euler's solution included the division of the interval between $x = a$ and $x = z$ into multiple small subintervals (Figure 5.1), each one with a width of Δx; and the replacement of the given integral by a sum $\sum L(t, x, \dot{x})\Delta x$. Euler demonstrated that by doing a variation in an arbitrary point (point *n*

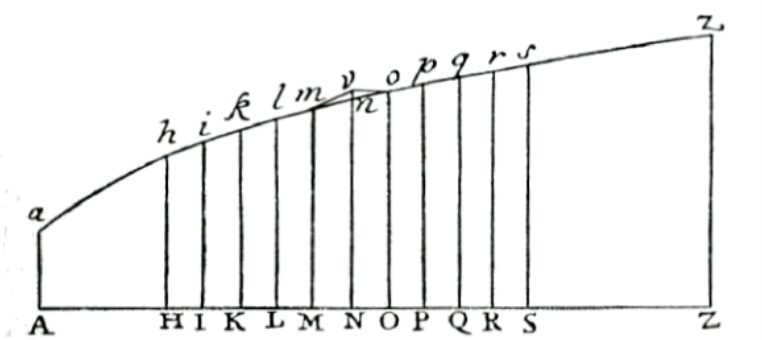


Figure 5.1: Original Euler's representation of calculus of variations [Hanc (2017)].

converted in point *v*), there are obvious changes in that point, but also in neighboring segments (segment *mn* and *no* converted in *mv* and *vo*, respectively). All other points remain unchanged. This means that only two terms in the integral sum $\sum L(t, x, \dot{x})\Delta x$ are affected.

At this point, Euler's solution was mainly geometric, however, Lagrange proposed a mathematical approach that extends Euler's method. The Lagrange method is based on the condition that, for the sum to be stationary, there is a correspondent zero value of its derivative in that point, equation 5.1. The result is called the *Euler-Lagrange* equation, and it expresses the first necessary condition for the existence of an extremal value in the integral.

$$0 = \frac{\partial L}{\partial x} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) \qquad a \leq t \leq z \tag{5.1}$$

After the development of more approaches as the second-order necessary condition of optimality by Legendre [Chachuat (2016)] and the reformulation of the Hamiltonian's "principle of least action" in mechanics as a variational principle, extended later by Jacobi leading to the Hamilton-Jacobi equations [LaValle (2006)], it followed the consideration of restrictions of admissible functions $\dot{x}(.)$ to a set of equations

$$g(t, x(t), \dot{x}(t)) = 0 \qquad t \in [a, z], \tag{5.2}$$

Which are recognized as a set of differential equations. The resulting problem is known as the Lagrange problem, and is solved using Lagrange multipliers. This enhanced the *optimal control problem* where the control variable $u(t)$ emerges in place of $\dot{x}(t)$ in the calculus of variations problems. Then, a decisive breakthrough of the extension of the calculus of variations was achieved by Lev. Pontryagin with the formulation of the Pontryagin Maximum Principle [Pontryagin (1986)]. Since then, optimal control has been applied in many scientific fields, ranging from mathematics and engineering to biomedical sciences [Chachuat (2016)].

5.1.1 Lagrange multiplier

The Lagrange multiplier approach aims to find the local extremum (minimum or maximum) of a function subject to constraints, by transforming the constrained problem into an unconstrained problem. The main concept of this idea is that in any extremum point of the function evaluated, the function's gradient can be defined as a linear combination of the gradients of the constraints, with the Lagrange multipliers λ acting as coefficients (Figure 5.2).

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) = 0 \tag{5.3}$$

Considering that the objective function and the constraint function are continuously differentiable, and x^* is a local minimum, the objective function's gradient is orthogonal to a tangent line in the local minimum point x^*.

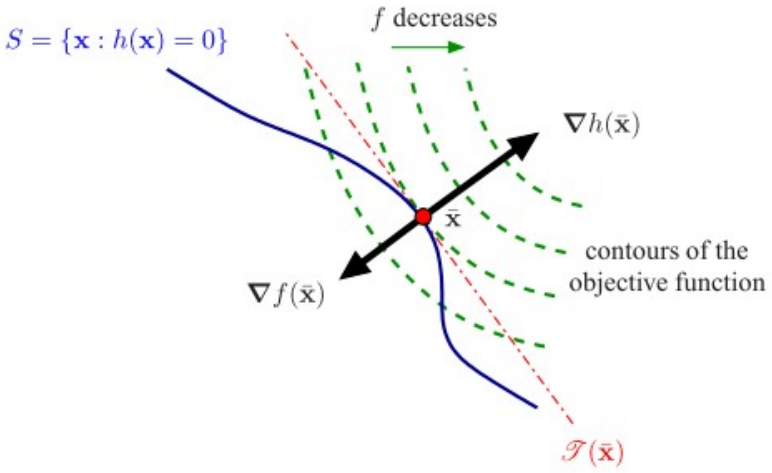


Figure 5.2: Illustration of the necessary conditions to minimise the objective function $f(x)$, subject to equality constraints $h(x) = 0$ [Chachuat (2016)].

Summarising, optimal control problems subject to equality or inequality constraints are dealt with by introducing Lagrange multipliers, which allow to adjoin the imposed constraints to the objective function and, thus, transform a constrained optimization problem into an unconstrained problem [Chachuat (2016)]. Additionally, Lagrange multipliers, λ, may be interpreted as the sensitivity of the objective function with respect to the constraints imposed and, thus, they can be seen as the rate that the optimal cost function would change, if the constraints were "perturbed" [Chachuat (2016)].

5.2 Optimal Control Problems

In optimal control problems, the evolution of *state* variables $x = (x_1, ..., x_n)$ is dictated by *control* variables $u = (u_1, ..., u_n)$, by means of a set of differential equations, equation 5.4.

To formulate an Optimal Control problem, it is necessary to define: the admissible control; describe mathematically the system to be controlled; specify a performance criterion, and the physical constraints that should be satisfied. Thus, the problem of optimal control can be stated as: "*Determine the control signals that will cause a system to satisfy the physical constraints and, at the same time, minimize (or maximize) some performance criterion.* On the contrary, the calculus of variation aims to find the optimal path for a *state* variable [Liberzon (2011)]. Clearly, once the optimal control path, $u^*(t)$ is found, the optimal state path, x^*, that corresponds to it, is also found.

The most important constraints in an optimal control problem are the ordinary differential equations (ODE) in the state-space form, which link the state and control variables. The differential equations take the form,

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{5.4}$$

Where $t \in \mathcal{R}$ stands for the independent variable called time; the vector $\boldsymbol{u}(t)$ represents the *control* variables at time instant *t* ; the vector $\boldsymbol{x}(t)$ represents the *state* variables, which characterize the behavior of the system at any time instant *t*.

The performance criterion (also called cost function or performance index) must be specified for evaluating the system's performance quantitatively. For instance, considering that we want to transfer a system from point A to point B in the minimum time possible, it clearly indicates that the total time is the performance measure to be minimized. The performance of a system can be defined in the so-called *Lagrange*, *Mayer* or *Bolza* form. However, interestingly, all formulations are theoretically equivalent (for more details see Chachuat (2016) in section 4.2.3). Considering the *Bolza* form, defined by

$$J(\boldsymbol{u}) = \psi(x_{t_f}, t_f) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{5.5}$$

where t_0 and t_f are the initial and final time; J is the objective function to be minimized; L and ψ are given functions, the performance index, and terminal cost, respectively. If there were no terminal cost ($\psi = 0$), we would have the Lagrange formulation, and if there were no running cost (L = 0), we would have the Mayer formulation.

To transform the problem of the objective function 5.5, subject to differential constraints (5.4), in an

unconstrained problem, *adjoint* functions, which can be interpreted as Lagrange multipliers, $\lambda(t)$, are added to the integrator (equation 5.6). The adjoint variables, λ, are functions of t because each constraint in 5.4 must be satisfied at every point of time in the interval $[t_0, t_f]$. Thus, the equation 5.5 becomes the following

$$J = \psi(x(t_f), tf) + \int_{t_0}^{t_f} [L(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) + \lambda^T(t)\{f(t, \boldsymbol{x}(t), \boldsymbol{u}(t) - \dot{x}\}]dt \tag{5.6}$$

After the definition of the performance criterion, the set of physical constraints to be satisfied, and the set of admissible controls, one can then state the optimal control problem as follows: "Find an admissible control $\boldsymbol{u}^* \in u[t_0, t_f]$ which satisfies the physical constraints in such a manner that the cost functional $J(\boldsymbol{u}^*)$ has a minimum value". Therefore, one shall say that the performance criterion assumes its minimum value at $\boldsymbol{u}^*$, provided that

$$J(\boldsymbol{u}^*) \leq J(\boldsymbol{u}), \tag{5.7}$$

To find the optimal value of control variables, necessary conditions were defined by the mathematician Pontryagin in the Pontryagin Minimum Principle (PMP), which involves the concepts of the Hamiltonian function and co-state variable. This principle overcomes the limitations of the variational approach, specifically, constraints on inputs and differentiability of the Lagrangian [Kirk (1970)].

Hence, when considering a cost function as in equation 5.6, it is convenient to introduce a scalar function *Hamiltonian*, H, as follows:

$$H(t, \boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = L(t, \boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\lambda}^T \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}). \tag{5.8}$$

Then, integrating the last term on the right side of equation 5.6 by parts, yields

$$J = \psi(x(t_f), tf) - \lambda^T(t_f)x(t_f) + \lambda^T(t_0)x(t_0) + \int_{t_0}^{t_f} \{H(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) + \dot{\lambda}^T(t)x(t)\}dt \tag{5.9}$$

Now considering the variation in J (equation 5.9) due to variations in the control vector $u(t)$ for fixed times t_0 and t_f

$$\delta J = \left[\left(\frac{\partial \psi}{\partial x} - \lambda^T\right)\right]_{t=t_f} + [\lambda^T \delta x]_{t=t_0} + \int_{t_0}^{t_f} \left[\left(\frac{\partial H}{\partial x} + \dot{\lambda}^T\right)\delta x + \frac{\partial H}{\partial u}\delta u\right] dt \tag{5.10}$$

It would be tedious to determine the variations $\delta x(t)$ produced by a given $\delta u(t)$, so a co-state function $\lambda(t)$ is defined to vanish the coefficients of δx in equation 5.10:

$$\dot{\lambda}^T = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f}{\partial x} \tag{5.11}$$

with boundary conditions

$$\lambda^T = \frac{\partial \psi}{\partial x(t_f)} \tag{5.12}$$

Therefore, the equation 5.10 becomes:

$$\delta J = \lambda^T(t_0)\delta x(t_0) + \int_{t_0}^{t_f} \frac{\partial H}{\partial u}\delta u dt \tag{5.13}$$

where $\lambda^T(t_0)$ is the gradient of J with respect to variation in the initial conditions while holding $u(t)$ constant and satisfying equation 5.4. The adjoint functions $\boldsymbol{\lambda}(t)$ are the marginal valuation in the optimal control problem of the state variable at time t. For instance, if there were a perturbation on the state variable at time t and subsequently the control variable was modified optimally, the optimal cost value would change at the rate $\boldsymbol{\lambda}(t)$.

Note also that the Hamiltonian function can be understood as an instantaneous increment of the Lagrangian expression of the problem that is to be optimized over a certain period. Thus, the function $\frac{\partial H}{\partial u}$ is called *impulse response* since each component of $\frac{\partial H}{\partial u}$ represents the variation in J, due to an impulse in the corresponding component of δu, while holding $x(t_0)$ constant and satisfying equation 5.4.

To find an extreme (minimum or maximum), δJ must be zero for arbitrary $\delta u(t)$. Therefore, this implies that

$$\frac{\partial H}{\partial u} = 0, \qquad t_0 \leq t \leq t_f \tag{5.14}$$

The necessary conditions to find the optimal control $\boldsymbol{u^*}$, and consequently the triple $(\boldsymbol{u^*}, \boldsymbol{x^*}, \boldsymbol{\lambda^*})$ are achieved by equations 5.14, 5.10 and 5.4. The local minimum is such that the following expression is verified for every time $t \in [t_0, t_f]$.

$$H(t, \boldsymbol{x}^*(t), \boldsymbol{u^*}(t), \boldsymbol{\lambda^*}(t)) \leq H(t, \boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda^*}(t)) \tag{5.15}$$

Considering that the problem is *autonomous*, i.e the functions L or $\boldsymbol{f}$ does not depend explicitly on time, t, the optimal Hamiltonian, $H(t, \boldsymbol{x}^*(t), \boldsymbol{u^*}(t), \boldsymbol{\lambda^*}(t))$, the Hamiltonian evaluated along the optimal path of x^*, u^*, and λ^*, have a constant value over time.

5.2.1 Equality Interior-point Constraints in State Variables

Consider the performance index of the form

$$\psi[x(t_f), tf] + \int_0^{t_f} L(t, \boldsymbol{x}(t), \boldsymbol{u}(t))dt, \tag{5.16}$$

that is described by the following nonlinear differential equations $\dot{x}(t)$

$$\dot{x}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{5.17}$$

$$and: \quad boundary \quad conditions \tag{5.18}$$

Note that this problem is similar to the one in the section 5.2, however, here the following constraints in an intermediate time $t_1 \in [0, t_f]$ are applied to the system.

$$N[x(t_1), t_1] = 0 \tag{5.19}$$

Thus, we now have a three-point boundary-problem instead of a two-point boundary-value problem. The equation 5.19 represents a set of terminal constraints for the part of the path from the initial point $t = 0$ to the intermediate point $t = t_1$. Let t_1^- signify the time just before t_1 and t_1^+ the time just after t_1.

For the interior-point constraints be satisfied, they are adjoined to the performance index by a set of Lagrange multipliers, π, which becomes the following:

$$J = \psi(x(t_f), tf) + \pi N + \int_0^{t_f} (H - \lambda \dot{x}) dt \tag{5.20}$$

Thus, the first variation of the augmented performance index is, then,

$$\delta J = \delta[\psi(x(t_f), tf) + \pi N] + \delta \int_0^{t_f} (H - \lambda \dot{x}) dt \tag{5.21}$$

Dividing the integral into $\int_{t_0}^{t_1-} + \int_{t_1+}^{t_f}$ and integrating by parts (allowing for possible discontinuities in λ at $t = t1$), yields:

$$\begin{aligned} \delta J = & \left. \frac{\partial \psi}{\partial x} \delta x \right|_{t=t_f} + \pi \frac{\partial N}{\partial t_1} dt_1 + \pi \frac{\partial N}{\partial x(t_1)} dx(t_1) \\ & - \lambda \delta x \Big|_{t_1+}^{t_f} - \lambda \delta x \Big|_{t_0}^{t_1-} + (H - \lambda \dot{x}) \Big|_{t=t_1-} dt_1 \\ & - (H - \lambda \dot{x}) \Big|_{t=t_1+} dt_1 + \int_{t_0}^{t_f} \left[\left(\dot{\lambda} + \frac{\partial H}{\partial x} \right) \delta x + \frac{\partial H}{\partial u} \delta u \right] dt \end{aligned} \tag{5.22}$$

Next, using the relations

$$dx(t_1) = \begin{cases} \delta x(t_1-) + \dot{x}(t_1-) dt_1, \\ \delta x(t_1+) + \dot{x}(t_1+) dt_1, \end{cases} \tag{5.23}$$

the terms $\delta x(t_1-)$ and $\delta x(t_1+)$ in equation 5.22 are eliminated, thus, we get:

$$\begin{aligned}\delta J = \left(\frac{\partial \psi}{\partial x} - \lambda^T\right)\delta x\Big|_{t=t_f} + \lambda^T(t_1^+) - \lambda^T(t_1^-) + \pi^T\frac{\partial N}{\partial x(t_1)}dx(t_1) \\ +H(t_1^-) - H(t_1^+) + \pi^T\frac{\partial N}{\partial t_1}dt_1 + \lambda^T\delta x\Big|_{t=t_0} + \int_{t_0}^{t_f}\left[\left(\dot{\lambda} + \frac{\partial H}{\partial x}\right)\delta x + \frac{\partial H}{\partial u}\delta u\right]dt\end{aligned} \tag{5.24}$$

Finally, let us choose $\lambda(t_1-)$ and $H(t_1-)$ to cause the coefficients of $dx(t_1)$ and dt_1 to vanish, yielding

$$\lambda^T(t_1^-) = \lambda^T(t_1^+) + \pi^T\frac{\partial N}{\partial x(t_1)} \tag{5.25}$$

$$H(t_1^-) = H(t_1^+) - \pi^T\frac{\partial N}{\partial t_1} \tag{5.26}$$

Note that the Lagrange multiplier, π, only influences the constraints (5.19) indirectly by propagating through the co-state equations via equation 5.25.

In addition, the time of the interior point, t_1, is determined by the equation 5.26. The function N represents the constraints in the *state* variables (equation 5.19), and its value is a constant that does not explicitly depends on time t_1. Thus, the second term of equation 5.26, $\frac{\partial N}{\partial t_1}$ is zero. This means that Hamiltonian is continuous at time t_1, and, therefore, the equation 5.26 results in:

$$H(t_1^+) = H(t_1^-) \tag{5.27}$$

Part IV

Design and Implementation

Chapter 6

Trajectory Planning

The current chapter describes the design and implementation of the proposed trajectory planning method that allows a collaborative robot to execute a human-like trajectory passing through multiple waypoints. This method is resultant from an extension of optimal control theory with interior point constraints at state variables (chapter 5, section 5.2.1).

6.1 Problem statement

As this dissertation's objective is to generate human-like trajectories in collaborative robots, the criterion function used is based on the minimum-jerk model, i.e., the minimisation of the variation of joints' acceleration. Thus, considering a robot with K joints, the cost function takes the form:

$$C = \frac{1}{2} \int_0^T \sum_{i=1}^{K} \left(\frac{d^3\theta_i}{dt^3} \right)^2 dt \tag{6.1}$$

where T is the total duration of the movement and θ_i is value of the joint i.

The system we need to solve is characterised by a set of *state* variables s and a *control* variable u.

$$s^T(t) = [\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a}], \qquad u^T(t) = [\boldsymbol{z}] \tag{6.2}$$

Where $\boldsymbol{x}$ is a vector of joints position; $\boldsymbol{v}$ is a vector of joints velocity, $\boldsymbol{a}$ is a vector of joints acceleration, and $\boldsymbol{z}$ is a vector of joints jerk, or by other words, the variation of the joints acceleration. These variables are vectors of dimension K, and defined by:

$$x = [\theta_1, \theta_2, ..., \theta_K] \tag{6.3}$$

$$v = \dot{x} = [\dot{\theta}_1, \dot{\theta}_2, ..., \dot{\theta}_K] \tag{6.4}$$

$$a = \dot{v} = \ddot{x} = [\ddot{\theta}_1, \ddot{\theta}_2, ..., \ddot{\theta}_K] \tag{6.5}$$

$$z = \dot{a} = \dddot{x} = [\dddot{\theta}_1, \dddot{\theta}_2, ..., \dddot{\theta}_K] \tag{6.6}$$

By means of the differential equations system (equation 6.7), the control variables u dictate the evolution of the state variables x.

$$\dot{s} = f[s(t), u(t), t] \tag{6.7}$$

where $s(t)$ and $u(t)$ are *state* and *control* functions, respectively. Besides, as already mentioned, this problem aims to solve optimal control with interior point equality constraints in the state variables. These constraints are designed as waypoints, which are positions that the robot must pass through during the movement at an unspecified time (equation 6.8). In these points, the only restriction applied is in the joints' position, therefore, the velocity and acceleration must be computed.

$$\psi(s(t_1), s(t_2), ..., s(t_N); t_1, t_2, ..., t_N) = 0 \tag{6.8}$$

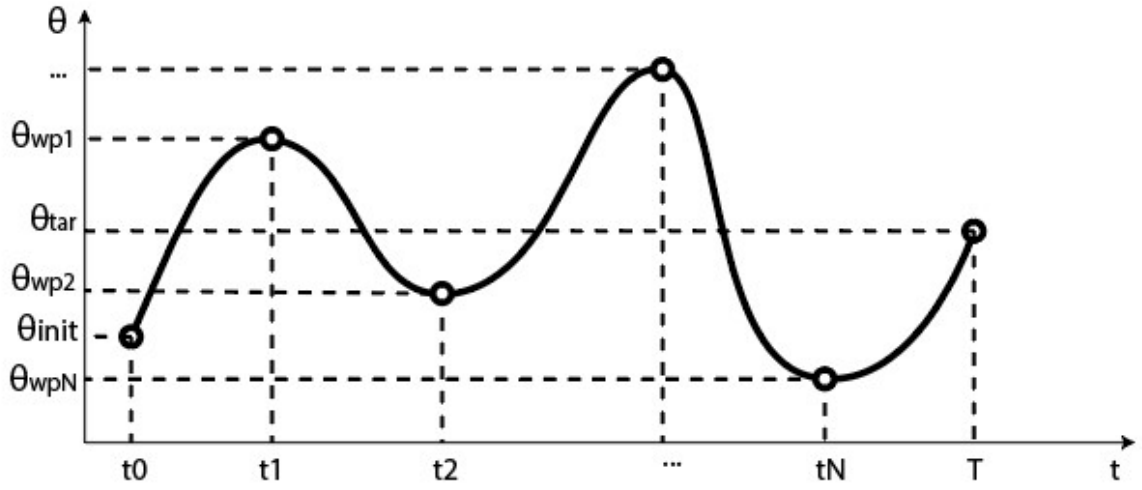


Figure 6.1: Joint trajectory from an initial joint position θ_{init} to a final joint position θ_{tar} that passes through N waypoints in unspecified time $[t1, t2, ..., tN]$

.

As mentioned in section 5.2, the differential equations constraints are introduced in the cost function by costate variables λ, which are functions over time, and are augmented to the cost function due to

the Hamiltonian function (equation 5.8). Furthermore, the interior constraints at an unspecified point are augmented to the cost function by a Lagrange multiplier $\boldsymbol{\pi}$. Thus, the cost functions take the form:

$$C = \pi^T \psi + \int_0^T (H - \lambda^T \dot{s}) \tag{6.9}$$

where π is a Lagrange multiplier vector of dimension N, correspondent to the number of interior-point constraints; λ is the co-state variable, which allows dealing with the nonlinear constraints and is introduced in the cost function by the Hamiltonian function, H (equation 5.8).

6.1.1 Problem solving

The problem presented in this dissertation is defined by:

$$\begin{aligned} min: \quad & C = \pi^T \psi + \int_0^T (H - \lambda^T \dot{s}) \\ s.t.: \quad & \dot{s} = f[s(t), u(t), t] \\ and \quad & \psi(s(t_1), s(t_2), ..., s(t_N); t_1, t_2, ..., t_N) = 0 \\ & boundary \quad conditions \end{aligned} \tag{6.10}$$

This problem is solved using the method in 5.2.1 that relies on the Pontryagin's minimum principle is crucial since addresses the necessary conditions for the existence of a minimum (equation 5.11 and 5.14). The solution for such problem is obtained by allowing discontinuities in the costate functions (refer to section 5.2.1). Therefore, we must analyse each trajectory segmentation individually and make a relation between them. Let us consider the costate variable λ^0 and Hamiltonian H^0 for all time before the first waypoint $t_0 \leq t \leq t_1$; and the costate variable λ^n and Hamiltonian H^n for the time after waypoint n and before waypoint $n+1$, $t_n \leq t \leq t_{n+1}$.

According to equation 5.25, which defines the expression of costate variables that enable the transformation of the constrained problem into an unconstrained problem, the general costate expression at the borders of a trajectory segmentation n is defined as:

$$\lambda^{n-1}(t_n) = \lambda^n(t_n) + \pi^T \frac{\partial N}{\partial s(t_n)} \tag{6.11}$$

Since the only constraints applied to the problem are in the joint's position, the term $\frac{\partial N}{\partial s(t_n)}$ is null for the state variables v and a, and control variable z, which corresponds to velocity, acceleration and jerk, respectively. Thus, from equation 6.11 we obtain the expression of the costate variables λ, for all segments and waypoints (refer to Eq. 6.12).

$$
\begin{aligned}
&\lambda_x^0(t_1) = \lambda_x^1(t_1) + \pi_1 \\
&\lambda_x^1(t_2) = \lambda_x^2(t_2) + \pi_2 \\
&\dots \\
&\lambda_x^{N-1}(t_N) = \lambda_x^N(t_N) + \pi_N
\end{aligned}
\tag{6.12}
$$

$$
\begin{aligned}
&\lambda_v^{n-1}(t_n) = \lambda_v^n(t_n) \\
&\lambda_a^{n-1}(t_n) = \lambda_a^n(t_n) \\
&\lambda_z^{n-1}(t_n) = \lambda_z^n(t_n)
\end{aligned}
$$

Now, using equations 5.8 and 6.12, the Hamiltonian for the first trajectory segmentation, $t_0 \leq t \leq t_1$, can be defined as

$$
H^0(t_1) = \sum_{i=1}^{K} \left[\lambda_{x_i}^0 v_i + \lambda_{v_i}^0 a_i + \lambda_{a_i}^0 z_i + \frac{1}{2} {z_i}^2 \right] \tag{6.13}
$$

The Hamiltonian H^n of the trajectory segmentation n, such that $t_n \leq t \leq t_{n+1}$ is

$$
H^n(t_n) = \sum_{i=1}^{K} \left[\lambda_{x_i}^n v_i + \lambda_{v_i}^n a_i + \lambda_{a_i}^n z_i + \frac{1}{2} {z_i}^2 \right] \tag{6.14}
$$

Note that hamiltonian and costate equations have the same structure for every joint since they have the same number of waypoints, i.e. the same constraints. Therefore, for simplicity sake, the problem is structured and analysed for a single joint i.

Time interval $t_0 \leq t \leq t_1$

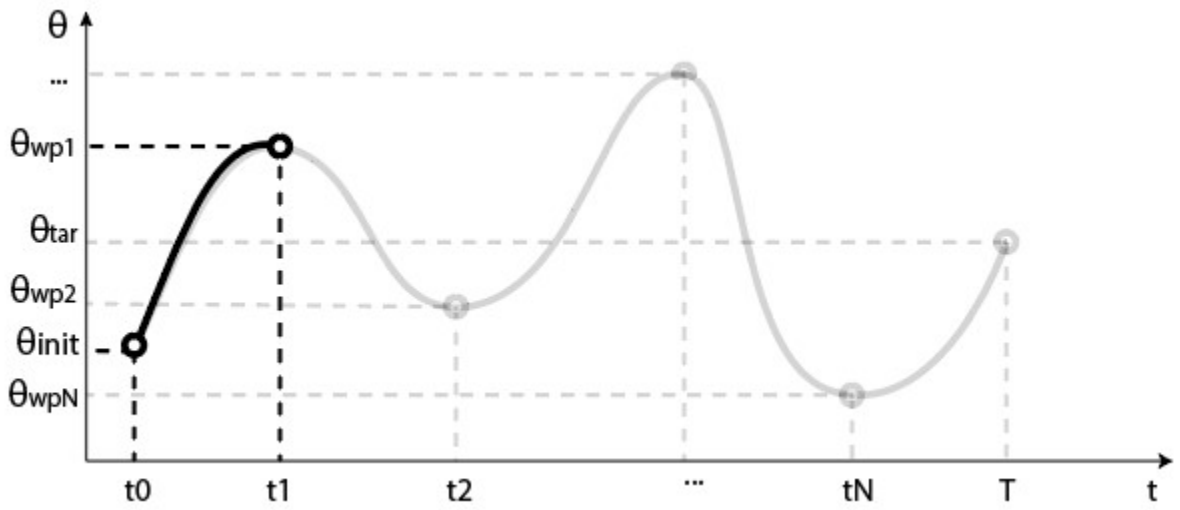


Figure 6.2: Joint trajectory from the initial joint position θ_{init} to the first waypoint θ_{wp1}, time interval $t0 < t < t1$.

From equation 5.11, which gives the necessary conditions for a minimum in the state variable, we obtain:

$$\begin{aligned}
\frac{\partial H^0}{\partial x_i} &= -\dot{\lambda}^0_{x_i} \Rightarrow \dot{\lambda}^0_{x_i} = 0 \Rightarrow \lambda^0_{x_i} = c_1 \\
\frac{\partial H^0}{\partial v_i} &= -\dot{\lambda}^0_{v_i} \Rightarrow \dot{\lambda}^0_{v_i} = -\lambda^0_{x_i} \Rightarrow \lambda^0_{v_i} = -\lambda^0_{x_i} t - c_2 \\
\frac{\partial H^0}{\partial a_i} &= -\dot{\lambda}^0_{a_i} \Rightarrow \dot{\lambda}^0_{a_i} = -\lambda^0_{v_i} \Rightarrow \lambda^0_{a_i} = +\lambda^0_{x_i} t^2 + c_2 t + c_3
\end{aligned} \tag{6.15}$$

Then, the necessary conditions for a minimum in the control variable are defined by equation 5.14:

$$\frac{\partial H^0}{\partial z_i} = 0 \Rightarrow \frac{\partial H^0}{\partial z_i} = z_i + \lambda^0_{a_i} = 0 \Rightarrow z_i = -\frac{\lambda^0_{x_i}}{2} t^2 - c_2 t - c_3 \tag{6.16}$$

These equations yield:

$$\begin{cases} z_i & = -\frac{\lambda^0_{x_i}}{2} t^2 - c_2 t - c_3 \\ \lambda^0_{x_i} & = c_1 (constant) \end{cases} \Rightarrow z_i = -\frac{c_1}{2} t^2 - c_2 t - c_3 \tag{6.17}$$

Since the jerk is the third derivative of the joint position, we can clearly obtain the following expression for the joint trajectory between the initial position and first waypoint:

$${}^0\theta_1(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{6.18}$$

where $(a_0, a_1, a_2, a_3, a_4, a_5) \in \Re$.

Time interval $t_1 \leq t \leq t_2$

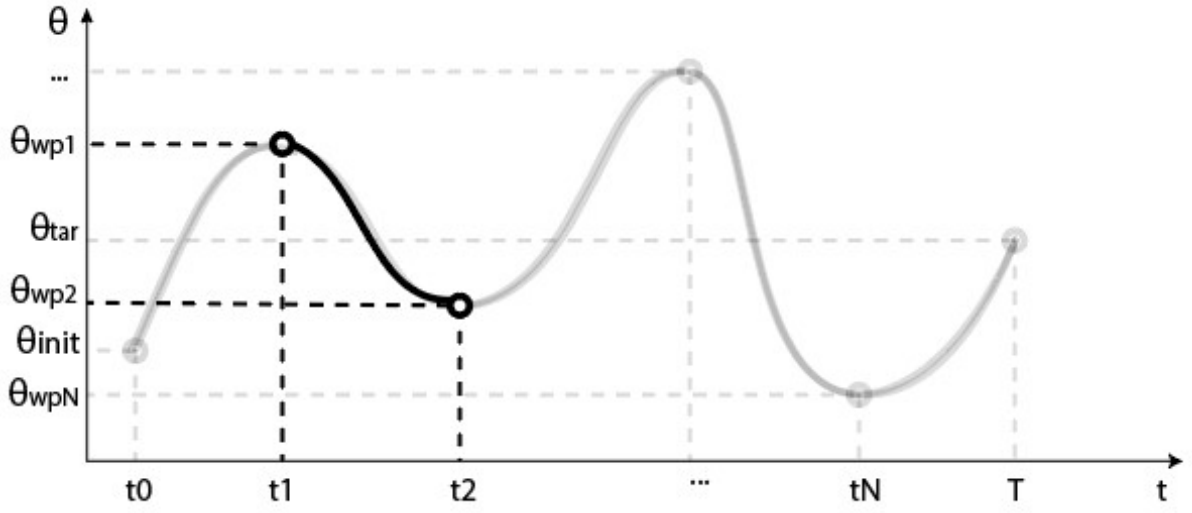


Figure 6.3: Joint trajectory from the first waypoint θ_{wp1} to the second θ_{wp2} (time interval $t1 < t < t2$).

From equation 5.11, which gives the necessary conditions for a minimum in the state variable, we obtain:

$$\begin{aligned}
\frac{\partial H^1}{\partial x_i} &= -\dot{\lambda}^1_{x_i} \Rightarrow \dot{\lambda}^1_{x_i} = 0 \Rightarrow \lambda^1_{x_i} = c_1 \\
\frac{\partial H^1}{\partial v_i} &= -\dot{\lambda}^1_{v_i} \Rightarrow \dot{\lambda}^1_{v_i} = -\lambda^1_{x_i} \Rightarrow \lambda^1_{v_i} = -\lambda^1_{x_i} t - c_2 \\
\frac{\partial H^1}{\partial a_i} &= -\dot{\lambda}^1_{a_i} \Rightarrow \dot{\lambda}^1_{a_i} = -\lambda^1_{v_i} \Rightarrow \lambda^1_{a_i} = +\lambda^1_{x_i} t^2 + c_2 t + c_3
\end{aligned} \tag{6.19}$$

Then, the necessary conditions for a minimum in the control variable are defined by equation 5.14:

$$\frac{\partial H^1}{\partial z} = 0 \Rightarrow \frac{\partial H^1}{\partial z} = z + \lambda_a = 0 \Rightarrow z = -\frac{\lambda^1_{x_i}}{2} t^2 - c_2 t - c_3 \tag{6.20}$$

Then, from the relation of the costate equations at the waypoints (6.12):

$$\begin{cases} z_i &= -\frac{\lambda^1_{x_i}}{2} t^2 - c_2 t - c_3 \\ \lambda^1_{x_i} &= \lambda^0_{x_i} - \pi_1 \end{cases} \Rightarrow z_i = -\frac{c_1}{2} t^2 + \frac{\pi_1}{2} t^2 - c_2 t - c_3 \tag{6.21}$$

Since the jerk, z, is the third derivative of the joint position, we can clearly obtain the following expression for the joint trajectory between the first and second waypoint:

$${}^1\theta_2(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \pi_1 \frac{(t - t_1)^5}{120} \tag{6.22}$$

where $(a_0, a_1, a_2, a_3, a_4, a_5) \in \Re$.

Time interval $t_n \leq t \leq t_{n+1}$

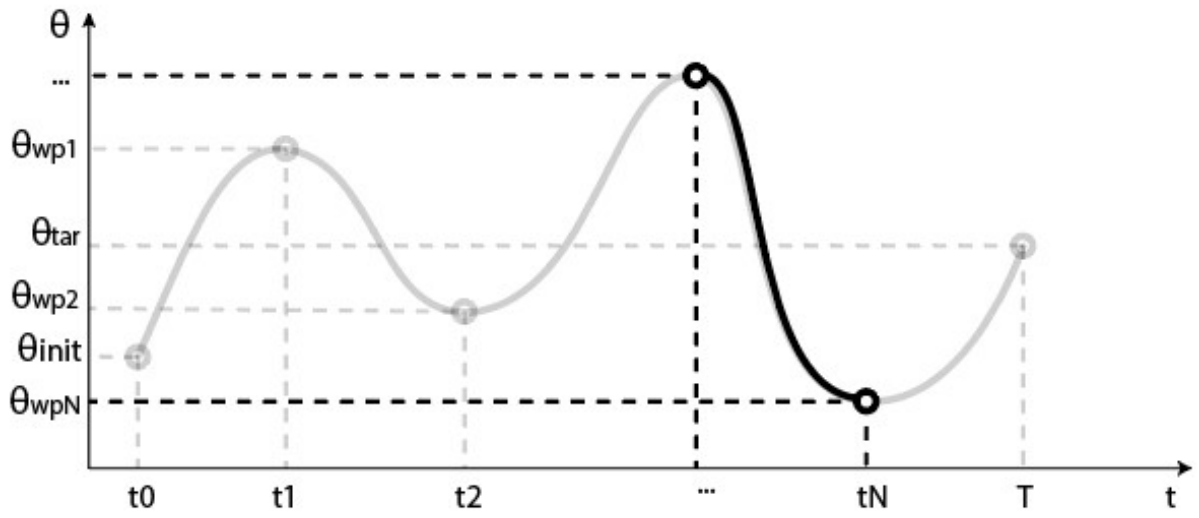


Figure 6.4: Joint trajectory from a waypoint n, θ_{wpn}, to the next waypoint θ_{wpn+1} (or final joint position), time interval $t_n < t < t_{n+1}$ or $t_N < t < t_f$.

From equation 5.11, which gives the necessary conditions for a minimum in the state variable, we obtain:

$$\begin{aligned}
\frac{\partial H^n}{\partial x_i} &= -\dot{\lambda}^n_{x_i} \Rightarrow \dot{\lambda}^n_{x_i} = 0 \Rightarrow \lambda^n_{x_i} = c_1 \\
\frac{\partial H^n}{\partial v_i} &= -\dot{\lambda}^n_{v_i} \Rightarrow \dot{\lambda}^n_{v_i} = \lambda^n_{x_i} \Rightarrow \lambda^n_{v_i} = -\lambda^n_{x_i} t - c_2 \\
\frac{\partial H^n}{\partial a_i} &= -\dot{\lambda}^n_{a_i} \Rightarrow \dot{\lambda}^n_{a_i} = \lambda^n_{v_i} \Rightarrow \lambda^n_{a_i} = \lambda^n_{x_i} t^2 + c_2 t + c_3
\end{aligned} \tag{6.23}$$

Then, the necessary conditions for a minimum in the control variable are defined by equation 5.14:

$$\frac{\partial H^n}{\partial z_n} = 0 \Rightarrow \frac{\partial H^n}{\partial z_n} = z_n + \lambda^n_{a_i} = 0 \Rightarrow z_n = -\frac{\lambda^n_{x_i}}{2} t^2 - c_2 - c_3 \tag{6.24}$$

Then, from the relation of the costate equations at the waypoints (6.12):

$$\begin{cases} z_n &= -\frac{\lambda^n_{x_i}}{2} t^2 - c_2 t - c_3 \\ \lambda^n_{x_i} &= \sum_{k=0}^{n-1} \lambda^k_{x_i} - \pi_n \end{cases} \Rightarrow z_n = -\frac{c_1}{2} t^2 + \sum_{i=1}^{n} \pi_n t^2 - c_2 t - c_3 \tag{6.25}$$

Since the jerk is the third derivative of the joint position, we can clearly obtain the expression for the joint trajectory between a waypoint n and waypoint $(n+1)$, or target position θ_{tar}.

$$^{n}\theta_{n+1}(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \sum_{i=1}^{n} \frac{\pi_i (t - t_i)^5}{120} \tag{6.26}$$

where $(a_0, a_1, a_2, a_3, a_4, a_5) \in \Re$.

6.1.2 Final trajectory equation $t_0 \leq t \leq T$

After the individual analysis of each trajectory segmentation, we can determine the full expression for the entire trajectory of a robot **joint i** that passes through N **waypoints** previously defined:

$$\theta_i(t) = \begin{cases} a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 & (t_0 \leq t \leq t_1) \\ a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \dfrac{\pi_1 (t - t_1)^5}{120} & (t_1 \leq t \leq t_2) \\ \dots & \\ a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + \sum\limits_{k=1}^{N} \dfrac{\pi_k (t - t_k)^5}{120} & (t_N \leq t \leq T) \end{cases} \tag{6.27}$$

where π_k is the Lagrange multiplier of the correspondent waypoint constraint.

Equation 6.27 has $\mathbf{6 + N}$ parameters ($(\boldsymbol{a_0, a_1, a_2, a_3, a_4, a_5}) + (\boldsymbol{\pi_1, ..., \pi_N})$). These parameters are determined by six boundary conditions and N conditions implied by the waypoints. The constraints imposed in the boundary points are the initial and final joint values and null velocity and acceleration at those points for a smooth trajectory.

$$initial\ point \begin{cases} \theta(0) &= \theta_{init} \\ \dot{\theta}(0) &= 0 \\ \ddot{\theta}(0) &= 0 \end{cases} \tag{6.28}$$

$$final\ point \begin{cases} \theta(T) &= \theta_{tar} \\ \dot{\theta}(T) &= 0 \\ \ddot{\theta}(T) &= 0 \end{cases} \tag{6.29}$$

The waypoints define the interior points constraints in the joint position. Thus, for N waypoints, one can describe the constraints as:

$$interior\ points \begin{cases} {}^{0}\theta_1(t_1) = {}^{1}\theta_2(t_1) = \theta_{wp1} \\ {}^{1}\theta_2(t_2) = {}^{2}\theta_3(t_2) = \theta_{wp2} \\ \dots \\ {}^{n-1}\theta_n(t_N) = {}^{n}\theta_T(t_N) = \theta_{wpN} \end{cases} \tag{6.30}$$

where ${}^{0}\theta_1$ represents the trajectory between the initial joint position and the first waypoint (equation 6.22); ${}^{1}\theta_2$ is the trajectory between the first waypoint and the second waypoint (equation 6.22); ${}^{n}\theta_T$ is the trajectory between the last waypoint, N, and the last joint position θ_{tar} (equation 6.26);

The conditions in 6.28, 6.29 and 6.30 result in the following system of equations:

$$boundary\ cond. \begin{cases} a_0 & = \theta_{init} \\ a_1 & = 0 \\ a_2 & = 0 \\ a_3T^3 + a_4T^4 + a_5T^5 + \sum_{i=1}^{N} \dfrac{\pi_i(T-t_i)^5}{120} & = \theta_{tar} - \theta_{init} \\ 3a_3T^2 + 4a_4T^3 + 5a_5T^4 + 5\sum_{i=1}^{N} \dfrac{\pi_i(T-t_i)^4}{120} & = 0 \\ 6a_3T + 12a_4T^2 + 20a_5T^3 + 20\sum_{i=1}^{N} \dfrac{\pi_i(T-t_i)^3}{120} & = 0 \end{cases} \tag{6.31}$$

$$interior\ cond. \begin{cases} a_3t_1^3 + a_4t_1^4 + a_5t_1^5 & = \theta_{wp1} - \theta_{init} \\ a_3t_2^3 + a_4t_2^4 + a_5t_2^5 + \dfrac{\pi_1(t_2-t_1)^5}{120} & = \theta_{wp2} - \theta_{init} \\ \dots \\ a_3t_N^3 + a_4t_N^4 + a_5t_N^5 + \sum_{i=1}^{N} \dfrac{\pi_i(t_N-t_i)^5}{120} & = \theta_{wpN} - \theta_{init} \end{cases} \tag{6.32}$$

From these systems of equations, we can determine the expressions of the parameters $(a_0, a_1, a_2, a_3, a_4, a_5)$,

resulting in the following:

$$a_0 = \theta_{init} \tag{6.33}$$

$$a_1 = 0 \tag{6.34}$$

$$a_2 = 0 \tag{6.35}$$

$$a_3 = \frac{10(\theta_{tar} - \theta_{init})}{T^3} + \sum_{i=1}^{N} \left(-\frac{10\pi_i (T - t_i)^5}{120T^3} + \frac{20\pi_i (T - t_i)^4}{120T^2} - \frac{10\pi_i (T - t_i)^3}{120T} \right) \tag{6.36}$$

$$a_4 = -\frac{15(\theta_{tar} - \theta_{init})}{T^4} + \sum_{i=1}^{N} \left(\frac{15\pi_i (T - t_i)^5}{120T^4} - \frac{35\pi_i (T - t_i)^4}{120T^3} + \frac{20\pi_i (T - t_i)^3}{120T^2} \right) \tag{6.37}$$

$$a_5 = \frac{6(\theta_{tar} - \theta_{init})}{T^5} + \sum_{i=1}^{N} \left(-\frac{6\pi_i (T - t_i)^5}{120T^5} + \frac{15\pi_i (T - t_i)^4}{120T^4} - \frac{10\pi_i (T - t_i)^3}{120T^3} \right) \tag{6.38}$$

Then, substituting the expressions of $\{a_0, a_1, ..., a_5\}$ into the system of equations 6.32 and 6.30, we obtain the general equation of the joint trajectory at a waypoint k, θ_{wpK}, of a set of waypoints N.

$$\begin{aligned} &(\theta_{tar} - \theta_{init})(10\tau_k^3 - 15\tau_k^4 + 6\tau_k^5) + \\ &(-10\tau_k^3 + 15\tau_k^4 - 6\tau_k^5) \sum_{i=1}^{N} \left(\frac{\pi_i T^5 (1 - \tau_i)^5}{120} \right) + \\ &(+20\tau_k^3 - 35\tau_k^4 + 15\tau_k^5) \sum_{i=1}^{N} \left(\frac{\pi_i T^5 (1 - \tau_i)^4}{120} \right) + \\ &(-10\tau_k^3 + 20\tau_k^4 - 10\tau_k^5) \sum_{i=1}^{N} \left(\frac{\pi_i T^5 (1 - \tau_i)^3}{120} \right) + \\ &\sum_{i=1}^{k-1} \frac{\pi_i T^5}{120} (\tau_k - \tau_i)^5 = \theta_{wpK} - \theta_{init} \end{aligned} \tag{6.39}$$

Note that $(T - t_i)^k = T^k (1 - \tau_i)^k$, where $\tau_i = \dfrac{t_i}{T}$.

Then, from the system 6.30 and consequently the general equation 6.39, the Lagrange multipliers can be determined. The Lagrange multipliers expressions are not constant and get more complex accordingly to the number of waypoints. Thus, we need to find a numerical approach capable to deal with N waypoints and, therefore, N Lagrange multipliers. To give a precise description of how this method is obtained, let us consider three examples. Firstly, considering a trajectory with only one waypoint, the respective Lagrange multiplier, π_1, takes the form:

$$\pi_1 = \frac{T^5}{120} \frac{(\theta_{tar} - \theta_{init})(10\tau_1^3 - 15\tau_1^4 + 6\tau_1^5) - (\theta_{wp1} - \theta_{init})}{(-10\tau_1^3 + 15\tau_1^4 - 6\tau_1^5)(1 - \tau_1)^5 + (+20\tau_1^3 - 35\tau_1^4 + 15\tau_1^5)(1 - \tau_1)^4 + (-10\tau_1^3 + 20\tau_1^4 - 10\tau_1^5)(1 - \tau_1)^3} \tag{6.40}$$

For trajectories with more waypoints, each Lagrange multiplier has a more complex expression and, therefore it is helpful to consider the following notation: $\alpha_{ik} = (-10\tau_i^3 + 15\tau_i^4 - 6\tau_i^5)(1-\tau_k)^5 + (+20\tau_i^3 - 35\tau_i^4 + 15\tau_i^5)(1-\tau_k)^4 + (-10\tau_i^3 + 20\tau_i^4 - 10\tau_i^5)(1-\tau_k)^3$, and $\beta_k = (\theta_{tar} - \theta_{init})(10\tau_k^3 - 15\tau_k^4 + 6\tau_k^5) - (\theta_{wpk} - \theta_{init})$. Hence, we can rewrite equation 6.40 as:

$$\pi_1 = \frac{T^5}{120}\frac{\beta_1}{\alpha_{11}} \tag{6.41}$$

Secondly, considering a trajectory with two waypoints, the Lagrange multipliers $[\pi_1, \pi_2]$ correspondent to the waypoints $[\theta_{wp1}, \theta_{wp2}]$ are formulated as:

$$\pi_1 = \frac{T^5}{120}\frac{-\alpha_{22}\beta_1 + \alpha_{21}\beta_2}{-\alpha_{22}\alpha_{11} + \alpha_{12}\alpha_{21}} \tag{6.42}$$

$$\pi_2 = \frac{T^5}{120}\frac{-\alpha_{11}\beta_2 + \alpha_{12}\beta_1}{-\alpha_{22}\alpha_{11} + \alpha_{12}\alpha_{21}} \tag{6.43}$$

Lastly, regarding a trajectory with three waypoints, three Lagrange multipliers $\pi = [\pi_1, \pi_2, \pi_3]$ correspondent to the waypoints $[\theta_{wp1}, \theta_{wp2}, \theta_{wp3}]$ are formulated as:

$$\pi_1 = \frac{T^5}{120}\frac{\alpha_{33}\Big(-\alpha_{22}\beta_1 + \alpha_{21}\beta_2\Big) - \alpha_{32}\Big(-\alpha_{23}\beta_1 + \alpha_{21}\beta_3\Big) + \alpha_{31}\Big(-\alpha_{23}\beta_2 + \alpha_{22}\beta_3\Big)}{\alpha_{33}\Big(-\alpha_{22}\alpha_{11} + \alpha_{12}\alpha_{21}\Big) - \alpha_{23}\Big(-\alpha_{32}\alpha_{11} + \alpha_{12}\alpha_{31}\Big) + \alpha_{13}\Big(-\alpha_{32}\alpha_{21} + \alpha_{22}\alpha_{31}\Big)} \tag{6.44}$$

$$\pi_2 = \frac{T^5}{120}\frac{\alpha_{33}\Big(-\alpha_{11}\beta_2 + \alpha_{12}\beta_1\Big) - \alpha_{31}\Big(-\alpha_{13}\beta_2 + \alpha_{12}\beta_3\Big) + \alpha_{32}\Big(-\alpha_{13}\beta_1 + \alpha_{11}\beta_3\Big)}{\alpha_{33}\Big(-\alpha_{22}\alpha_{11} + \alpha_{12}\alpha_{21}\Big) - \alpha_{23}\Big(-\alpha_{32}\alpha_{11} + \alpha_{12}\alpha_{31}\Big) + \alpha_{13}\Big(-\alpha_{32}\alpha_{21} + \alpha_{22}\alpha_{31}\Big)} \tag{6.45}$$

$$\pi_3 = \frac{T^5}{120}\frac{\alpha_{22}\Big(-\alpha_{11}\beta_3 + \alpha_{13}\beta_1\Big) - \alpha_{21}\Big(-\alpha_{12}\beta_3 + \alpha_{13}\beta_2\Big) + \alpha_{23}\Big(-\alpha_{12}\beta_1 + \alpha_{11}\beta_2\Big)}{\alpha_{33}\Big(-\alpha_{22}\alpha_{11} + \alpha_{12}\alpha_{21}\Big) - \alpha_{23}\Big(-\alpha_{32}\alpha_{11} + \alpha_{12}\alpha_{31}\Big) + \alpha_{13}\Big(-\alpha_{32}\alpha_{21} + \alpha_{22}\alpha_{31}\Big)} \tag{6.46}$$

Looking at equations 6.41, 6.42 6.44, a pattern can be observed as the number of waypoints increases. To demonstrate it more clearly, the analysis of the Lagrange multiplier is divided into the denominator and numerator of its expressions, refer to figures 6.5 and 6.6.

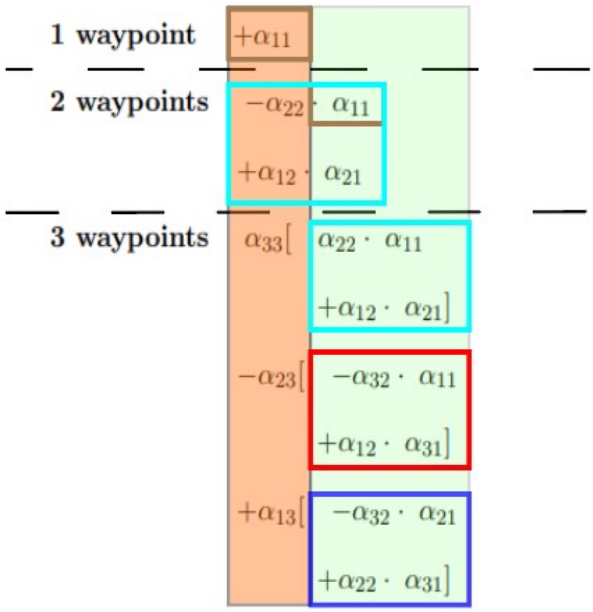


Figure 6.5: Denominator equations of the Lagrange multiplier π_1 in case of 1,2 and 3 waypoints, respectively.

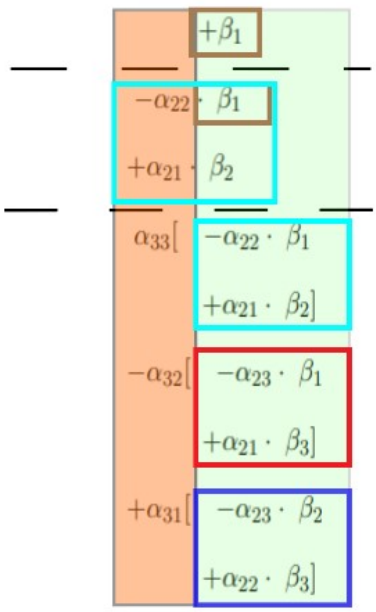


Figure 6.6: Numerator equations of the Lagrange multiplier π_1 in case of 1,2 and 3 waypoints, respectively.

As one can observe in figures 6.5 and 6.6, the pattern is divided in two main parts, represented by an orange and a yellow rectangle in the same figures. This division is essential to achieve a general equation of Lagrange multiplier since it is from the last part that the next is derived. Analysing the Lagrange multiplier π_1 of a trajectory with 3 waypoints, equation 6.44, and figures 6.5 and 6.6 :

Orange rectangle The orange rectangle is defined by as many equations as the number of waypoints, and changing the indexes α_{ik} between them. Specifically, considering the denominator algorithm (Figure 6.5), we have 3 equations ($\alpha_{33}, -\alpha_{23}$ and α_{13}), where the equations are defined by $\sum_{i=1}^{3} \alpha_{i3}$. In the numerator algorithm (Figure 6.6), the equations are slightly different, ($\alpha_{33}, -\alpha_{32}$ and α_{31}), where the equations are defined by $\sum_{i=k}^{3} \alpha_{3k}$. The signal of equation α_{ik} is defined whether i for the denominator and k for the numerator is even or odd.

Green rectangle The yellow rectangle is essentially achieved by a recursive process, where its equations (selected as light blue, red and dark blue rectangle) are achieved by computing the previous waypoints (waypoint 2 and waypoint 1). For instance, the blue rectangle is dictated by the complete equation of a trajectory with 2 waypoints. Then, the remaining part of the equation (red and green rectangles), is achieved by computing the same process, however, with the indexes of α_{ik} and β_i alternated. Regarding the denominator equation (Figure 6.5), the red rectangle is obtained by re-computing the blue rectangle with

α_{2k} changed to α_{3k}, and the green rectangle by re-computing the red rectangle with α_{1k} changed to α_{2k}. Similarly, considering the numerator algorithm, the red rectangle is obtained by re-computing the blue rectangle with α_{i2} and β_2 changed to α_{i3} and β_3, respectively, and the green rectangle by re-computing the red rectangle with α_{i1} and β_1 changed to α_{i2} and β_2, respectively.

Note that the recursiveness illustrated in figures 6.5 and 6.6 is implemented, in case of having N waypoints, by means of the algorithms in section 6.1.3.

After determining the Lagrange multipliers $\pi = [\pi_1, \pi_2, ..., \pi_N]$ and the time of each waypoint (section 6.1.4), all variables are known and, therefore, the equation of the joint trajectory $\theta(t)$ can be achieved. The expression of the joint trajectory in $t_0 \leq t \leq t_1$ is described by the equation 6.47.

$$\begin{aligned} {}^0\theta_1(\tau) = \theta_{init} + (\theta_{tar} - \theta_{init})(10\tau^3 - 15\tau^4 + 6\tau^5) + \\ (-10\tau^3 + 15\tau^4 - 6\tau^5) \sum_{k=1}^{N} \left(\frac{\pi_k T^5 (1-\tau_k)^5}{120} \right) + \\ (+20\tau^3 - 35\tau^4 + 15\tau^5) \sum_{k=1}^{N} \left(\frac{\pi_k T^5 (1-\tau_k)^4}{120} \right) + \\ (-10\tau^3 + 20\tau^4 - 10\tau^5) \sum_{k=1}^{N} \left(\frac{\pi_k T^5 (1-\tau_k)^3}{120} \right) \end{aligned} \tag{6.47}$$

where $\tau = \dfrac{t}{T}$ and $\tau_k = \dfrac{t_k}{T}$

Subsequently, knowing equation 6.47, one can redefine the system 6.27 that represents the trajectory segmentations for all time $t_0 \leq t \leq T$ (Equation 6.48):

$$\theta(t) = \begin{cases} {}^0\theta_1(\tau) & (t_0 \leq t \leq t_1) \\ {}^1\theta_2(\tau) = {}^0\theta_1(\tau) + \dfrac{\pi_1 T^5 (\tau - \tau_1)^5}{120} & (t_1 \leq t \leq t_2) \\ \dots & \\ {}^N\theta_T(\tau) = {}^{N-1}\theta_N(\tau) + \dfrac{\pi_N T^5 (\tau - \tau_N)^5}{120} & (t_N \leq t \leq T) \end{cases} \tag{6.48}$$

6.1.3 Lagrange multiplier

This subsection presents the algorithms responsible for determining the Lagrange multipliers $\pi = [\pi_1, \pi_2, ..., \pi_N]$ of a single joint trajectory that passes through N waypoints.

The process to obtain the expression of each Lagrange multiplier is divided into two main parts:

denominator and numerator. Furthermore, as the equations 6.44-6.46 show, the denominator of all Lagrange multiplier $[\pi_1, ..., \pi_N]$ is equal, whereas the numerator is different between each expression. Although both patterns (figures 6.5 and 6.6) are similar, they are determined separately by means of two recursive processes.

The main algorithm to get the Lagrange multipliers of N waypoints of a single joint is Algorithm 1, where it receives as input an array with the time normalised of each waypoint $\tau_{wp} = [\tau_1, ..., \tau_N]$, the waypoints $\theta_{wp} = [\theta_{wp1}, ..., \theta_{wpN}]$ and the initial and final joint position, θ_{init} and θ_{tar}, respectively. The denominator of π is determined using the Algorithm 2, and the numerator of each $[\pi_1, \pi_2, ..., \pi_N]$ using the Algorithm 3, which is essentially obtained by computing cyclically the Algorithm 4.

Figures 6.6 and 6.5 illustrate the denominator and numerator of the Lagrange multiplier for the first waypoint π_1, where $\tau_{wp} = [\tau_1, \tau_2, ..., \tau_N]$ and $\theta_{wp} = [\theta_{wp_1}, \theta_{wp_2}, ..., \theta_{wp_N}]$. As one can observe from equations 6.44 and 6.45, interestingly the numerator of π_2 has the same form of π_1, however, the indexes of α_{ik} and β_i that contain "2" or "1" are switched. For instance, α_{32} in π_1 is switched to α_{31} in π_2. Similarly, π_3 (equation 6.46) is obtained by switching the indexes "2" and "3" of π_2 (equation 6.45). Specifically, for determining π_1 we have $\tau_{wp} = [\tau_1, \tau_2, ..., \tau_N]$ and $\theta_{wp} = [\theta_{wp_1}, \theta_{wp_2}, ..., \theta_{wp_N}]$, for determining π_2 we have $\tau_{wp} = [\tau_2, \tau_1, ..., \tau_N]$ and $\theta_{wp} = [\theta_{wp_2}, \theta_{wp_1}, ..., \theta_{wp_N}]$, and in case of π_3 we have $\tau_{wp} = [\tau_3, \tau_1, \tau_2, ..., \tau_N]$ and $\theta_{wp} = [\theta_{wp_3}, \theta_{wp_1}, \theta_{wp_2}, ..., \theta_{wp_N}]$ (lines 3-8 of Algorithm 3). All these changes are handled by Algorithm 3 in lines 3-8.

As already mentioned, the denominator of all Lagrange multiplier is the same, however, the numerator is different. Thus, Algorithm 2 returns the denominator of all Lagrange multiplier $\pi = [\pi_1, ..., \pi_N]$ and Algorithm 4 returns the numerator of the Lagrange multiplier π_i. Both Algorithms are essentially recursive, dictated by the size of the arrays $\boldsymbol{\tau_{wp}}$ and $\boldsymbol{\tau_{or}}$, which are composed by the normalised time of the waypoints $\tau_{wp} = [\tau_1, \tau_2, ..., \tau_N]$. The process of both algorithms starts by determining the orange rectangle, $\boldsymbol{P_{or}}$, of Figure 6.5 and 6.6 by means of Algorithm 5. Then, the yellow rectangle is determined. As explained in Figure 6.5 and 6.6, the key to obtain the entire equation is, basically, switching the indexes of the equation α_{ik} and β_i, and that is possible by switching the indexes of the array $\boldsymbol{\tau_{wp}}$ and $\boldsymbol{\theta_{wp}}$ (lines 7-9 of Algorithm 2 and lines 7-12 of Algorithm 4.

Note that equation β_k is a simplification of the expression $\beta_k = (\theta_{tar} - \theta_{init})(10\tau_k^3 - 15\tau_k^4 + 6\tau_k^5) - (\theta_{wpk} - \theta_{init})$, which is computed in line 3 of Algorithm 4). Equation α_{ik} is obtained with Algorithm 6 where it receives as input the normalised time of two waypoints $\tau = [\tau_1, \tau_2, ..., \tau_N]$ (lines 4 and 6 of Algorithm 5). Algorithm 6 is the key of Algorithm 5, which essentially determines the orange rectangle of figures 6.6 and 6.5.

Algorithm 1 Lagrange multiplier equations for a single joint $\pi_i = [\pi_1, \pi_2, ..., \pi_N]$

Input: $\boldsymbol{\tau_{wp}}$ // array of $\tau_{wp} = [\tau_1, ..., \tau_N]$

θ_{tar} // final position of joint i

θ_{init} // initial position of joint i

$\boldsymbol{\theta_{wp}}$ // array of N waypoints of joint a single joint, $\theta_{wp} = [\theta_{wp1}, ..., \theta_{wpN}]$

Output: $\boldsymbol{\pi}$ // array of Lagrange multiplier for all waypoints $\pi_i = [\pi_1, ..., \pi_N]$

$D_{en} \leftarrow$ Algorithm 2$(\boldsymbol{\tau_{wp}}, \boldsymbol{\tau_{wp}})$ // Denominator of π

$N_{um} \leftarrow$ Algorithm 3$(\boldsymbol{\tau_{wp}}, \boldsymbol{\theta_{wp}}, \theta_{tar}, \theta_{init})$ // Numerator of π_i

for $i < N$ **do** // Get the Lagrange multiplier for N waypoints

  $\boldsymbol{\pi}.append\left(\frac{-120}{T^5} \times \frac{N_{um}[i]}{D_{en}}\right)$

Algorithm 2 Complete denominator equation of Lagrange multiplier π_i

Input: $\boldsymbol{\tau_{wp}}$ // array of $\tau = [\tau_1, ..., \tau_N]$ where the changes are computed

$\boldsymbol{\tau_{or}}$ // original array of $\tau = [\tau_1, ..., \tau_N]$

Output: $\boldsymbol{D_{en}}$ // final denominator equation

$n \leftarrow size(\boldsymbol{\tau_{wp}})$

if n=0 **then** // end of the recursiveness

  return 1

$\boldsymbol{P_{or}} \leftarrow$Algorithm 5$(\boldsymbol{\tau_{wp}}, \boldsymbol{\tau_{or}})$ // Orange rectangle, refer to Figure 6.5

for $i < size(\boldsymbol{\tau_{wp}})$ **do**

  if $n > 1$ **and** $i > 0$ **then** // Changes of the α_{ik} indexes (refer to Figure 6.5)

    $aux \leftarrow \boldsymbol{\tau_{wp}}[n-i-1]$

    $\boldsymbol{\tau_{wp}}[n-i-1] \leftarrow \boldsymbol{\tau_{wp}}[n-1]$

    $\boldsymbol{\tau_{wp}}[n-1] \leftarrow aux$

  $\boldsymbol{D_{en}} \leftarrow \boldsymbol{D_{en}} + \boldsymbol{P_{or}}[i] \times$Algorithm 2$(\boldsymbol{\tau_{wp}}[\boldsymbol{0:n-1}], \boldsymbol{\tau_{or}}[\boldsymbol{0:n-1}])$ // compute the same function for all waypoints and with array τ_{wp} changed)

Algorithm 3 Numerator part of Lagrange multiplier π_i for all waypoints of a single joint

Input: $\boldsymbol{\tau_{wp}}$ // array of $\tau = [\tau_1, ..., \tau_N]$ where the changes are computed

$\boldsymbol{\theta_{wp}}$ // array of waypoints $\theta_{wp} = [\theta_{wp1}, ..., \theta_{wpN}]$

θ_{init} // initial joint position

θ_{tar} // final joint position

Output: $\boldsymbol{\pi}$ // array with the numerator part of Lagrange multipliers for waypoints

for $k < size(\boldsymbol{\theta_{wp}})$ **do** // numerator of all Lagrange multipliers
  if $k > 0$ **then** // switch α_{ik} and β_i (refer to equations 6.44, 6.45, 6.46)
    $aux \leftarrow \boldsymbol{\tau_{wp}}[0]$
    $\boldsymbol{\tau_{wp}}[0] \leftarrow \boldsymbol{\tau_{wp}}[k]$
    $\boldsymbol{\tau_{wp}}[k] \leftarrow aux$
    $aux \leftarrow \boldsymbol{\theta_{wp}}[0]$
    $\boldsymbol{\theta_{wp}}[0] \leftarrow \boldsymbol{\theta_{wp}}[k]$
    $\boldsymbol{\theta_{wp}}[k] \leftarrow aux$
  $\boldsymbol{\pi}.append($Algorithm $4(\boldsymbol{\tau_{wp}}, \boldsymbol{\tau_{wp}}, \boldsymbol{\theta_{wp}}, \theta_{init}, \theta_{tar}))$ // numerator of π of the waypoint k

Algorithm 4 Numerator equation of Lagrange multiplier π_i

Input: $\boldsymbol{\tau_{or}}$ // Original array of $\tau = [\tau_1, ..., \tau_N]$

$\boldsymbol{\tau_{wp}}$ // array of $\tau = [\tau_1, ..., \tau_N]$ where the changes are computed

$\boldsymbol{\theta_{wp}}$ // array of waypoints $\theta_{wp} = [\theta_{wp1}, ..., \theta_{wpN}]$

θ_{init} // initial joint position

θ_{tar} // final joint position

Output: $\boldsymbol{N_{um}}$ // final numerator equation

$n \leftarrow size(\boldsymbol{\tau_{wp}})$

if n=0 **then** // equation β_k

    return $(\theta_{tar} - \theta_{init})(10\boldsymbol{\tau_{wp}}[\mathbf{0}]^3 - 15\boldsymbol{\tau_{wp}}[\mathbf{0}]^4 + 6\boldsymbol{\tau_{wp}}[\mathbf{0}]^5) - (\boldsymbol{\theta_{wp}}[\mathbf{0}] - \theta_{init})$

$\boldsymbol{P_{or}} \leftarrow$ Algorithm 5($\boldsymbol{\tau_{wp}}, \boldsymbol{\tau_{or}}$) // Orange rectangle, refer to Figure 6.6

for $i < size(\boldsymbol{\tau_{wp}})$ **do**

    if $n > 1$ **and** $i > 0$ **then** // Changes the indexes of α_{ik} and β_i (refer Figure 6.6)

        $aux \leftarrow \boldsymbol{\tau_{wp}}[\boldsymbol{n - i - 1}]$

        $\boldsymbol{\tau_{wp}}[\boldsymbol{n - i - 1}] \leftarrow \boldsymbol{\tau_{wp}}[\boldsymbol{n - 1}]$

        $\boldsymbol{\tau_{wp}}[\boldsymbol{n - 1}] \leftarrow aux$

        $aux \leftarrow \boldsymbol{\theta_{wp}}[\boldsymbol{n - i - 1}]$

        $\boldsymbol{\theta_{wp}}[\boldsymbol{n - i - 1}] \leftarrow \boldsymbol{\theta_{wp}}[\boldsymbol{n - 1}]$

        $\boldsymbol{\theta_{wp}}[\boldsymbol{n - i - 1}] \leftarrow aux$

    $\boldsymbol{N_{um}} \leftarrow \boldsymbol{N_{um}} + \boldsymbol{P_{or}}[\boldsymbol{i}] \times$ Algorithm 4($\boldsymbol{\tau_{or}}[\mathbf{0 : n - 1}], \boldsymbol{\tau_{wp}}[\mathbf{0 : n - 1}], \boldsymbol{\theta_{wp}}[\mathbf{0 : n - 1}]$)

// compute the same function for all waypoints and with array τ_{wp} changed)

Algorithm 5 Orange part of denominator and numerator (refer 6.5 and 6.6)

Input: $\boldsymbol{\tau_{wp}}$ // array of $\tau = [\tau_1, ..., \tau_N]$ where the changes are computed

$\boldsymbol{\tau_{or}}$ // original array of $\tau = [\tau_1, ..., \tau_N]$

Output: $\boldsymbol{P_{or}}$ // array with the Orange part of the denominator or numerator

$i, n \leftarrow size(\boldsymbol{\tau_{wp}})$

while $i > 0$ **do**

  if i is even **then**

    $Eq \leftarrow -$Algorithm 6($\boldsymbol{\tau_{wp}}[\boldsymbol{i}], \boldsymbol{\tau_{or}}[\boldsymbol{n}]$) // Equation $-\alpha_{ik}$

  else

    $Eq \leftarrow$ Algorithm 6 ($\boldsymbol{\tau_{wp}}[\boldsymbol{i}], \boldsymbol{\tau_{or}}[\boldsymbol{n}]$) // Equation α_{ik}

  i = i - 1

  $\boldsymbol{P_{or}}.append(Eq)$

Algorithm 6 Equation α_{ik}

Input: σ // value of array $\tau = [\tau_1, ..., \tau_N]$ from Algorithm 5

γ // value of array $\tau = [\tau_1, ..., \tau_N]$ from Algorithm 5

Output: Eq

$Eq1 \leftarrow (-10\gamma^3 + 15\gamma^4 - 6\gamma^5)(1-\sigma)^5 + (+20\gamma^3 - 35\gamma^4 + 15\gamma^5)(1-\sigma)^4 + (-10\gamma^3 + 20\gamma^4 - 10\gamma^5)(1-\sigma)^3$

$Eq2 \leftarrow 0$

if $\sigma \leq \gamma$ **then**

  $Eq2 \leftarrow (\gamma - \sigma)^5$

$Eq \leftarrow Eq1 + Eq2$

6.1.4 Waypoints time

Now that each Lagrange multiplier of $\pi = [\pi_1, \pi_2, ..., \pi_N]$ is already defined, we rely on equation 5.27 to determine the time of each waypoint $[t_1, t_2, ..., t_N]$. Applying the costate conditions (refer to equation 6.12) at these points to equation 5.27, we obtain a general equation of the Hamiltonian continuity at a waypoint n:

$$H^{n-1}(t_n) = H^n(t_n) \tag{6.49}$$

where H^{n-1} corresponds to the Hamiltonian function of the segmentation trajectory after the waypoint $(n-1)$ and before the waypoint n, and H^n corresponds to the Hamiltonian of the segmentation trajectory after the waypoint n and before the waypoint $n+1$, or target position θ_{wp}.

Using equation 6.49 for every waypoint results in the system 6.50, where the unique unknown variables are times $[t_1, ..., t_N]$, when the robots passes through waypoints $[\theta_{wp1}, ..., \theta_{wpN}]$.

$$\begin{cases} \pi_1 v_1(t_1) & = 0 \\ \pi_2 v_2(t_2) & = 0 \\ ... & \\ \pi_N v_N(t_N) & = 0 \end{cases} \tag{6.50}$$

Note that the system 6.50 is for a unique joint, however, the system is easily extended to deal with K joints. Each joint trajectory's motion is independent from the others and, therefore, they are calculated separately. On the other hand, all robot joints must have a coordinated motion, i.e., all joints must start and finish the movement simultaneously. Furthermore, they must reach the waypoints at the same time. Thus, we can extend the system 6.50 and calculate the time of each waypoint considering all joints.

$$\begin{cases} \sum_{i=1}^{K} \pi_1^i v_1^i(t_1) = f_1 = 0 \\ \sum_{i=1}^{K} \pi_2^i v_2^i(t_2) = f_2 = 0 \\ ... \\ \sum_{i=1}^{K} \pi_N^i v_N^i(t_N) = f_N = 0 \end{cases} \tag{6.51}$$

$$\begin{cases} f_1 = \overbrace{\pi_1^1 v_1^1(t_1)}^{\text{waypoint 1 and joint 1}} + \pi_1^2 v_1^2(t_1) + ... + \overbrace{\pi_1^K v_1^K(t_1)}^{\text{waypoint 1 and joint K}} \\ f_2 = \pi_2^1 v_2^1(t_2) + \pi_2^2 v_2^2(t_2) + ... + \pi_2^K v_2^K(t_2) \\ \dots \\ f_N = \underbrace{\pi_N^1 v_N^1(t_N)}_{\text{waypoint N and joint 1}} + \pi_N^2 v_N^2(t_N) + ... + \underbrace{\pi_N^K v_N^K(t_N)}_{\text{waypoint N and joint K}} \end{cases} \tag{6.52}$$

In this dissertation, the nonlinear system 6.51 is solved by means of an open-source Python library *SciPy*, which addresses a wide range of models as optimization, integration, interpolation, ODE solvers and many other tasks common in science and engineering. Specifically, from the *SciPy* library, we use the *fsolve* function, which returns the roots of equations 6.51 given a starting estimate. The Initial Guess

(IG), or starting estimate is determined by the joint average of the travel path in each waypoint.

$$IG_n = \frac{1}{K}\sum_{k=1}^{K}\left\lfloor \frac{\sum_{i=0}^{n}||\theta_{[i+1],k} - \theta_{i,k}||}{\sum_{i=0}^{N}||\theta_{[i+1],k} - \theta_{i,k}||} \right\rceil \tag{6.53}$$

where K is the number of joints of the robot, N is the number of waypoints, $\theta_{0,k}$ is the initial joint position θ_{init} of the joint k, θ_N is the final joint position θ_{tar} of the joint k and $\theta_{i,k}$ is the waypoint i of the joint k.

The function *fsolve* gets as input the system of equations 6.51 and an array of initial guesses for of all waypoints $IG_{wp} = [IG_1, IG_2, ..., IG_N]$, determined by equation 6.53. The acceptable root must lie between 0 and 1. Furthermore, the time of each waypoint must be sequential, i.e $[\tau_1 < \tau_2 < ... < \tau_N]$. If *fsolve* returns more than one possible solution, the solution selected is the one that minimises the energy of the system through equation 6.54.

$$En = \sqrt{(f_1)^2 + (f_2)^2 + (...)^2 + (f_N)^2} \tag{6.54}$$

After having the solution of system 6.51, we can substitute the time of each waypoint $[t_1, t_2, ..., t_N]$ in equations 6.26 and, finally, obtain the final expression of $\theta(t)$ for the entire movement.

6.2 Time parametrization

The trajectory generated in this dissertation is time normalised and time is also discretised into number of steps. Hence, the total movement duration, T, the number of steps, N_{steps}, and the size of each time step, Δt, need to be computed.

6.2.1 Total time

Rosenbaum et al. (2001) determined the optimal duration time, T, of a human-like movement based on Fitts' Law of joint angular displacement, $\Delta\theta$. Recently, Gulletta et al. (2021), extended the Rosenbaum et al. (2001) formulation and added the term $N_{steps}\frac{\Delta\theta_{max,k}}{w_{max,k}}$, which guarantees that the maximum joints velocity is respected, and, therefore, never reached (equation 6.55). The optimal time is the one that minimises the travel cost, which is defined by the sum of the costs of the movements of the individual joints going from the starting posture to the next goal posture (equation 6.57).

$$T_k = N_{steps}\frac{\Delta\theta_{max,k}}{w_{max,k}} + \delta_k \ln(1 + \Delta\theta_k) \tag{6.55}$$

$$\Delta\theta_k = \sum_{i=1}^{N_{steps}} |\theta_{i,k} - \theta_{i-1,k}| \tag{6.56}$$

$$\Delta\theta_{max,k} = \underset{i=1,\ldots,N_{steps}}{max} |\theta_{i,k} - \theta_{i-1,k}| \tag{6.57}$$

where δ_k is the expense factor of the k^{th} joint, N_{steps} is the number of time steps in the movement, $w_{max,k}$ is the maximum angular velocity of the k^{th} joint.

This approach grants an optimal movement time and at the same time ensures the minimum cost of travel without violating the limits of kinematics. Besides, since the robot must have a coordinate movement, i.e, all joints start and stop a movement at the same time, the optimal time is computed as a weighted average of the optimal duration for each joint (Equation 6.58).

$$T = \frac{\sum_{k=1}^{n} \delta_k \Delta\theta_k T_k}{\sum_{k=1}^{n} \delta_k \Delta\theta_k} \tag{6.58}$$

6.2.2 Number of steps

The number of steps, N_{steps}, is determined according to the weight average of the distance between each waypoint (equation 6.59).

$$N_{steps} = \sum_{i=0}^{Nwp} \left\lfloor N_m + (N_M - N_m)\frac{||\theta_{wp_{[i+1]}} - \theta_{wp_i}||}{||\theta_M - \theta_m||} \right\rceil \tag{6.59}$$

where N_M and Nm are the acceptable maximum and minimum number of steps, respectively. Nwp is the total number of waypoints. θ_M is the maximum limit of the joints, and $theta_m$ is the minimum limit of the joints. θ_{wp_i} is the joint value in the i^{th} waypoint. θ_{wp_0} is equal to θ_{init}, and represents the initial joint position. $\theta_{wp_{[Nwp+1]}}$ is equal to θ_{tar}, and represents the final joint position.

6.2.3 Time step

An approximated time step is determined based on a standard trajectory with null boundary conditions, which can be defined by equation 6.47 with no constraints, therefore, with null Lagrange multiplier $\pi_k = 0$, and is given by equation 6.60:

$$\Delta t = \frac{T}{N_{steps}} \tag{6.60}$$

The resultant step size Δt should provide a trajectory that does not reach the joint velocity and acceleration limits, w_{max}.

Part V

Validation of the Trajectory Planning

Chapter 7

Human-like Trajectory Planning with waypoints in a Human-Robot Collaboration Scene

In this dissertation, it is considered a quality inspection task to validate the proposed trajectory planner. Specifically, the user selects mandatory positions where the robot must cross for allowing the inspection of the selected eye angles. The process to validate the proposed method is first described in section 7.1. The trajectory is executed in a human-like manner, which enhances the productivity of the human-robot collaboration. Thus, metrics to evaluate the human-likeness of the UR10 robot movements are presented in section 7.2. Then, in section 7.3.1, the task to validate the proposed method, as well as the method to select waypoints are described. Finally, the results are individually analyzed in section 7.4, and then an overall discussion is presented in section 7.5.

Note that the results were obtained with a machine with a processor *Intel®Core^{TM} i7-7700HQ 2.80GHz*, running the *Ubuntu* 16.04 LTS 64-bits operator system, with graphics *NVIDIA GeForce GTX 1050M* and 16 GB RAM.

7.1 Validation Architecture

To ensure the integrity of the robotic platform and the workplace involved, the validation of the trajectory planning that this dissertation presents is performed in a simulation environment. The architecture of the trajectory planning process and its validation in a simulation environment is illustrated in Figure 7.1, which is essentially composed of 4 software modules, called: i) CoppeliaSim simulator; ii)Polyscope iii) Motion

Manager; iv) Motion Planner;

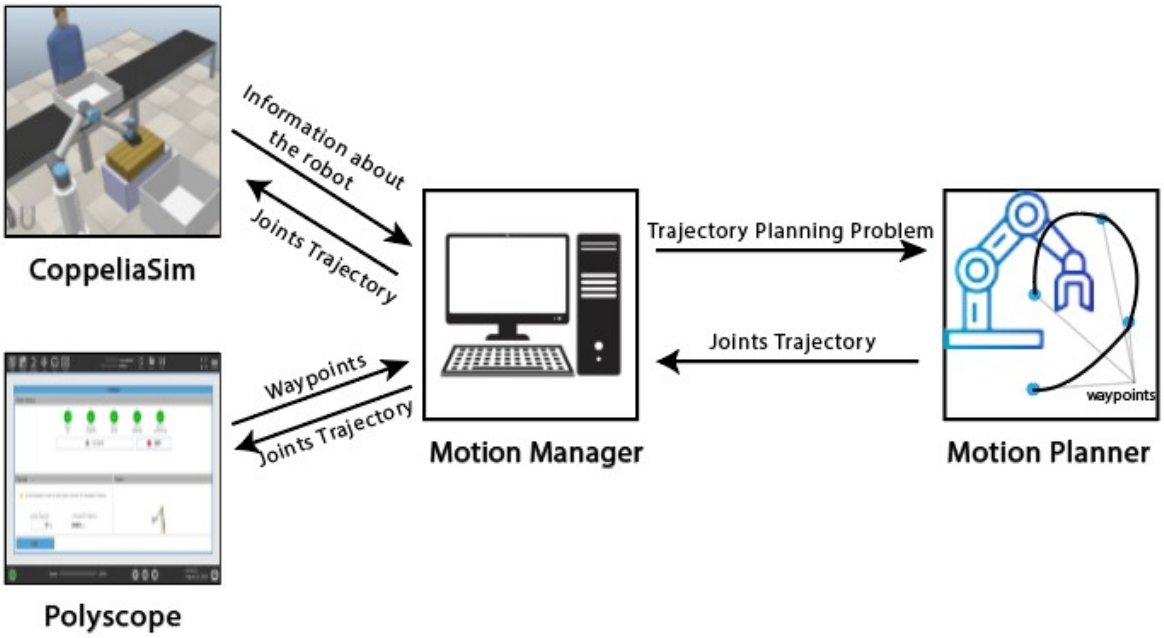


Figure 7.1: The validation of the generated trajectory is achieved by using the following modules: simulator CoppeliaSim; Polyscope; Motion Manager and Motion Planner.

The trajectory planning method that is here presented is implemented in the **H**uman-like **U**pper-limb **M**otion **P**lanner (**HUMP**) [Gulletta et al. (2021)]. The HUMP library is a motion planning algorithm that generates collision-free trajectories with human-like characteristics. Very briefly, the human-like obstacle avoidance is accomplished by the imposition of two movements: a *direct movement*, from the start posture to the target posture, and a *back-and-forth movement*, from the start posture to a bounce posture [Gulletta et al. (2021)]. In this dissertation is presented a completely new functionality, and therefore, none of these algorithms are used to accomplish the results further on demonstrated.

The communication between the different modules is achieved through the ROS (**R**obot **O**perating **S**ystem) framework from Quigley et al. (2008). The Motion Manager module is used to obtain the necessary scenario information for trajectory planning, from the kinematics of the robotic platform to the waypoints that the robot must cross. Note that the waypoints are defined in the Polyscope module, which is the Graphical User Interface embedded in Universal Robots' robots. Having obtained the data, this is sent to the Motion Planner module, or HUMP, which involves the implementation in C++ of the proposed method to generate the human-like trajectory. Then, this trajectory is sent back to the simulator *CoppeliaSim*, where the human-robot collaboration scenery is projected. Additionally, the joint trajectory is also sent to the *Polyscope* module, which has a motion simulator of the real robot and, thus, one can observe the

robot moving in the Universal Robot GUI.

7.2 Human-likeness Evaluation

As already referred endowing the UR10 robot, a collaborative robot, with the capability to generate human trajectories is the objective of this dissertation. Accordingly, with section 3.3, a human-like arm motion is composed by a minimization of the joints' acceleration, which results in a single peak bell-shaped hand velocity and a slightly curved hand trajectory between two points. Additionally, more velocity peaks may be observed due to high rate wrist rotation, normally occurring when there are objects in proximity. Therefore, we must have a methodology to analyze the robot movements and whether they correspond to what is expected. Although it is not possible to measure exactly the human-likeness of the robot's movements, it is possible to analyze quantitatively and qualitatively a set of metrics, which have been proposed in psychology and neuroscience [Gulletta et al. (2020)].

Some studies have been pursued to evaluate reaching movements in children with cerebral palsy [Chang et al. (2005)], and post-stroke patients [Chang et al. (2008)]. The metrics used are directly related to the smoothness of the movement and, thus, the human-likeness of it. These metrics are known as the Normalised Jerk Score (**NJS**) and the Number of Movement Units (**NMU**).

The variable NJS is formulated as:

$$NJS = \sqrt{\frac{1}{2}\left(\frac{T^5}{D^2}\right)\int\left(\left(\frac{d^3x_H}{dt^3}\right)^2 + \left(\frac{d^3y_H}{dt^3}\right)^2 + \left(\frac{d^3z_H}{dt^3}\right)^2\right)} \tag{7.1}$$

where (x_H, y_H, z_H) is the hand position in the Cartesian space, T is the total duration of the movement and D is the travelling distance of the hand.

According to Chang et al. (2005), the **NMU** variable is determined by: Firstly, searching for local minima and maxima in the velocity profile; secondly, finding an increase in velocity between the adjacent minimum and maximum that exceed a threshold of 10%. If the threshold of 10% is exceeded means that occurs a movement unit.

Usually, in reaching movement, the value of NJS metric is less than 100 [Chang et al. (2005)]. Yet, naturally, this value is higher when considering more complex movements [Gulletta et al. (2020)]. Similarly, the NMU metric is typically 1 in reaching movements; however, a higher NMU value is expected when high wrist rotations are necessary [Gulletta et al. (2020)].

The human-likeness of movements can also be evaluated qualitatively by the human operator. The expected movements are natural, smooth, and pleasant, which contribute to a better human-robot inter-

action by its predictability and, additionally, enhancing the operator's confidence and well-being [Schaal (2007)]. Therefore, a user study would be useful for a more concise evaluation of the human-likeness of robot movements.

7.3 Task: Quality inspection

The validation of the proposed trajectory generation method is performed in an assembler process with a quality inspection context. The scenario used is only a demonstration of how this method can be advantageous for several tasks in today's industry.

The purpose of this task, Figure 7.2, is to pick up the boards from the pallet and assemble them in the box on top of the conveyor. At the same time, the operator must inspect the board to guarantee that it is suitable and in perfect condition to be assembled. This requires, from the operator, a manual rotation of the board to verify all sides. According to its condition, the user or even assembles the board in the box or places it into the faulty box, which is closed to the palette.

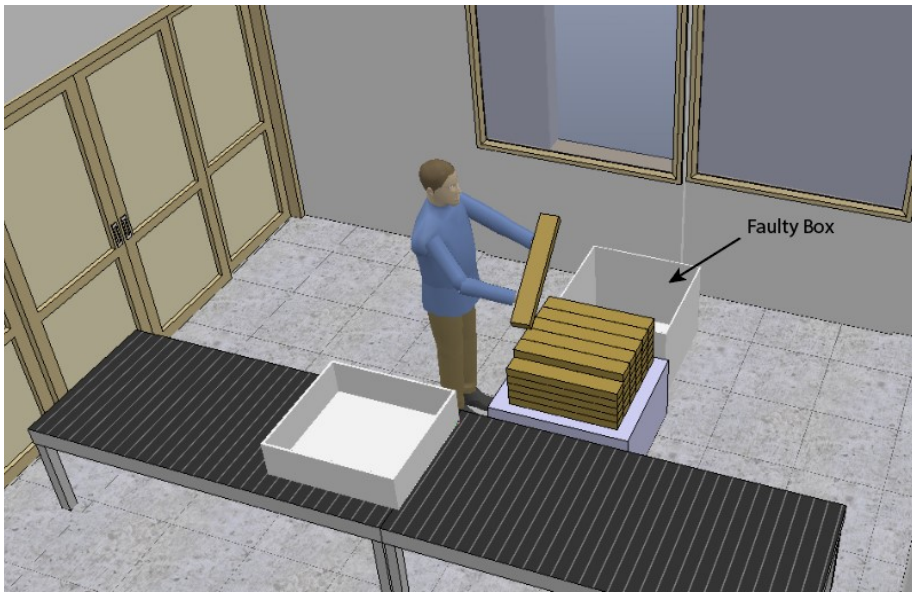


Figure 7.2: Quality inspection scene

Considering the human inspection, there are several properties of an object that may transform it into an inadequate one for the following task, such as: texture, painting, size, format, and more. Naturally, the requisites vary according to the next task, which could be an assembler process, painting, etc.

This task is tedious, heavy, and repetitive, which may negatively contribute to the human's happiness and well-being. As already explained, when a human performs constantly repetitive movements, it may result in long term diseases. Therefore, to maintain the healthy life of the human workers, and at the same

time to improve the productivity of the task, we propose the implementation of the UR10 collaborative robot with a vacuum gripper in the scene, Figure 7.2. Since the worker is working close to the robot and both collaborate to complete the task, the human-likeness of robot movements has a tremendous impact on the operator confidence, security, and productivity.

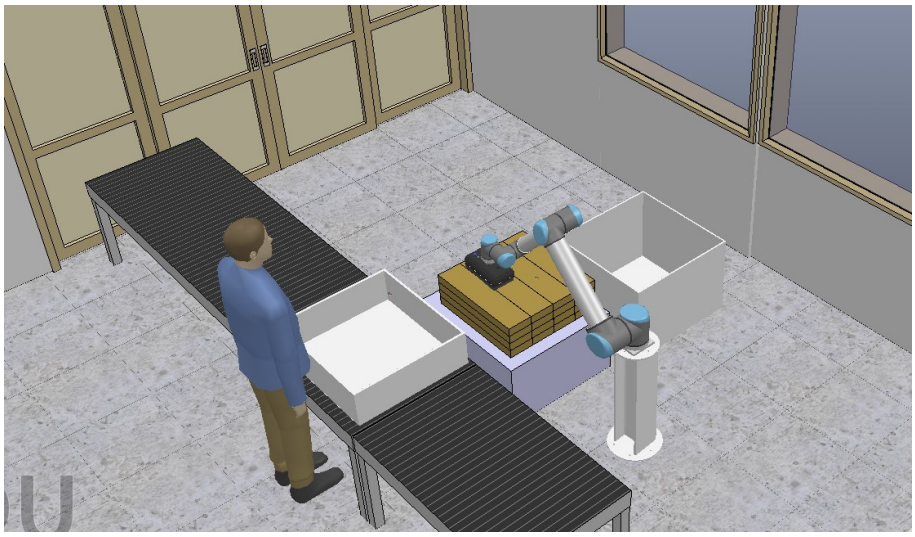


Figure 7.3: Quality inspection scene with human-robot collaboration

Figure 7.3 illustrates the validation task of the proposed method. In this scene, we can observe the UR10 robot, two boxes, a pallet of boards, and a conveyor. The robot performs the critical and heavy tasks before handling by the operator. This means that the main task of the robot is to constantly manipulate the boards. On the other hand, the operator deals with the cognitive task, which is the definition of mandatory points where the robot must pass during his trajectory. Thus, after picking up the board, the robot shows it to the worker in the angles that he has selected. If the boards are suitable, the robot releases them and the operator mounts the box on the conveyor. Otherwise, the robot places it in the "faulty" box and repeats the task for the following boards. Another important aspect is that, since it is the worker who sets the mandatory crossing points and he knows the workspace environment, he can set strategic points to avoid collisions if the workspace is restricted.

This task must be divided into 3 different movements (pick the plank; show to the operator; place it), since it demands an interruption between them and the action of the vacuum gripper, which are thoroughly described in subsection 7.3.1. The selection of the waypoints is explained in detail in subsection 7.3.2.

7.3.1 Task Description

In the validation task introduced in section 7.3, the human operator and the robot collaborate to accomplish the task. The robot deals with all transportation and manipulation of objects. The operator, in turn, is responsible for the cognitive tasks, which is the quality inspection and based on that decides where the object should be placed. The user has the most important role, which is the definition of the waypoints, i.e. the definition of mandatory points in the robot trajectory.

As presented in section 3.3, Martins de Sá (2018) divides a task into different movements: *Pick*, *Place* and *Move*, and each movement is composed of different phases and states. For the case of this project, since we deal with waypoints, all movements are classified as a *Move* movement. Note that a *Move* movement corresponds to the trajectory from an initial to a final position, taking no account of the grasp and ungrasp phases. When considering objects, this movement is equivalent to the *transport* phase, where the robot transports the object from a position to another. This is easily seen when using a vacuum gripper since it only demands action at the end of the movement. However, when considering a finger gripper, the waypoints also include the gripper aperture, and thus the *grasp* and *ungrasp* phases are embedded in the waypoints selection.

Hence, considering the task described in section 7.3, one can divide it in three phases: movement to pick the plank; movement to show the plank to the operator; movement to place the plank;

Algorithm 7 Procedure to **pick** the plank

1: Home posture
2: Go to N waypoints
3: Go to Goal posture - Plank position from the pallet
4: Activate Vacuum Gripper

Algorithm 8 Procedure to **show** the plank

1: Final posture of Pick Movement (Algorithm 7 line 4)
2: Go to N waypoints
3: Go to Goal posture - Last waypoint

Algorithm 9 Procedure to **place** the plank

1: Final posture of Show Movement (Algorithm 8 line 3)
2: Go to N waypoints
3: Go to Goal posture - Faulty or Good Box
4: Deactivate Vacuum Gripper
5: Go to home posture

Firstly, considering the *Pick* movement (Algorithm 7), the operator must set the home posture of the task, where the robot returns after each task cycle. The goal posture is also interpreted as waypoints, however, since all boards are in different positions, the goal posture of the picking movement in each task cycle must be different from the previous ones. For a pallet of 24 boards, one must have 24 goal postures and perform 24 task cycles. Secondly, in the *show* movement, the robot starts its trajectory from the final position of the picking movement (Algorithm 7 line 4), and crosses the waypoints defined by the operator. The last waypoint defined is considered as the goal posture of the movement. Lastly, regarding the *place* movement, the *Motion Planner* plans a trajectory starting from the final posture of *show* movement (Algorithm 8 line 3) to the goal posture - Faulty or Good Box - and passing through N waypoints defined by the operator. At the end of the movement, the robot releases the plank by deactivating the vacuum gripper and then returns to the home posture of the task (Algorithm 7 line 1).

7.3.2 Waypoints definition

The key of this dissertation is the generation of a trajectory that passes through waypoints, which are totally independent of the task. To define waypoints for each movement, as explained in Figure 7.1, the user must use the *Polyscope* software built-in the UR10 teach pendant. By using this software the user has the possibility to define waypoints by manipulating physically the robot or even using the built-in joystick to move the joints individually or controlling the end-effector pose and consequently the joints position. In order to physically grab the robot arm and pull it to the desired pose, the user must hold down the *Freedrive* button (Figure 7.4).

The definition of waypoints demands the correlation between the *Polyscope* and *Motion Manager* models, where the first allows the robot manipulation and the second deals with the acquisition of those waypoints in the joint space. The communication between both modules is held by the ROS framework.

Figure 7.5 illustrates the Graphical User Interface built in the Motion Manager module, where it is possible to obtain the waypoints. Regarding this platform, the button *Add waypoint* is used to define the

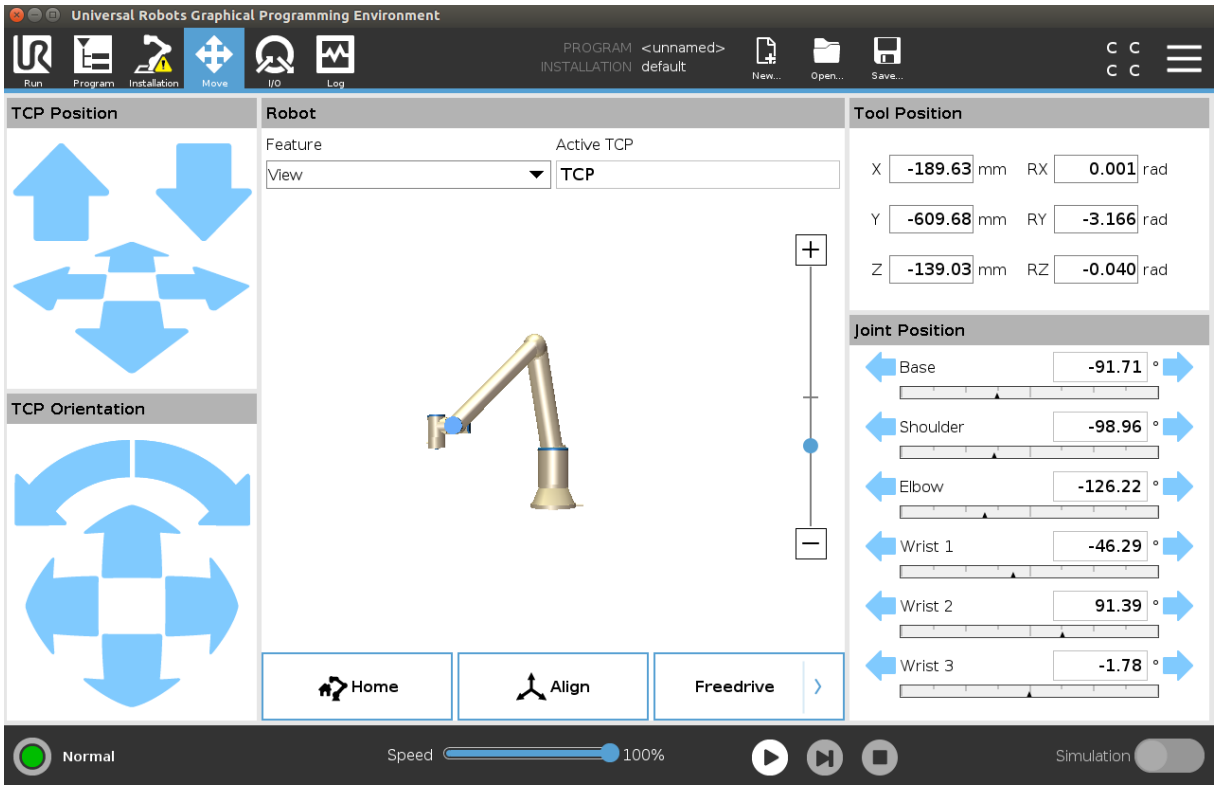


Figure 7.4: Universal Robots Graphical Programming Environment

waypoints of a movement, by enabling the connection to *Polyscope*, reads the robot's joints values and displays the result in the box above. The operator must define the action of the gripper at the end of the movement with the button *Vacuum On/Off*. Once these steps are defined the user must give a name to the movement (e.g. *pick*), and the button, *Add Movement* merges the waypoints, name and gripper action to the movement. The procedure should be repeated until all movements are determined, and once concluded, the user must click on the button *End Task*. Then, the waypoints are defined and ready to be sent to the *Motion Planner* module, which computes the joints trajectory.

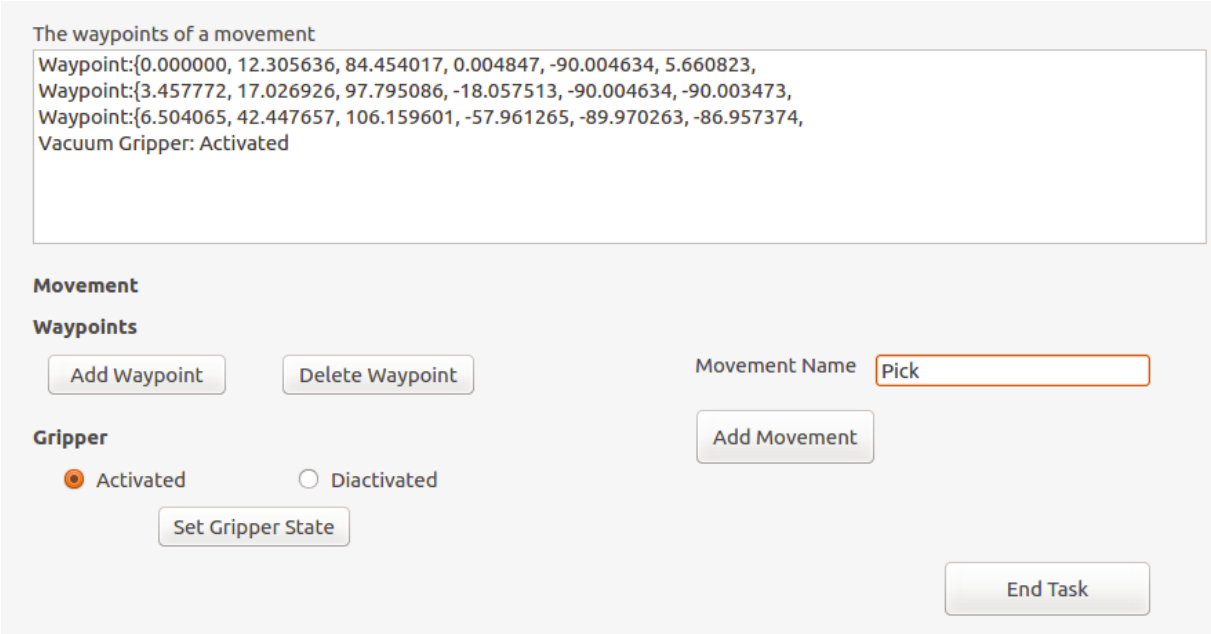


Figure 7.5: Motion Manager Environment for defining waypoints

7.4 Movements and Results Achieved

There are an infinite possible set of waypoints the user could define to accomplish the task 7.3.1. Programming a task through waypoints is flexible, intuitive, and easy to use. The user has a direct intervention on the robot trajectory and can easily program a new task by physically manipulating the robot. When defining waypoints, it should be taken into account the environment scene, i.e. the obstacles in the workplace; the robot joint limits; the complexity of the task; and the critical positions of the trajectory.

The first step is the definition of a home posture, table 7.1, where the robot starts and returns in each task cycle, Figure 7.6.

Table 7.1: Home posture of the UR10 robot for the task 7.3.1

$\boldsymbol{Joints}$	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$
$\boldsymbol{Deg}$	-180	-80	-130	-60	90	0

The experiments in the following subsections are only an example of how the proposed method could be applied. The movement *Pick* is defined by 3 waypoints and the vacuum action activated at the end of the movement; the movement *Show* is defined by 6 waypoints, with 3 critical angle inspection points;

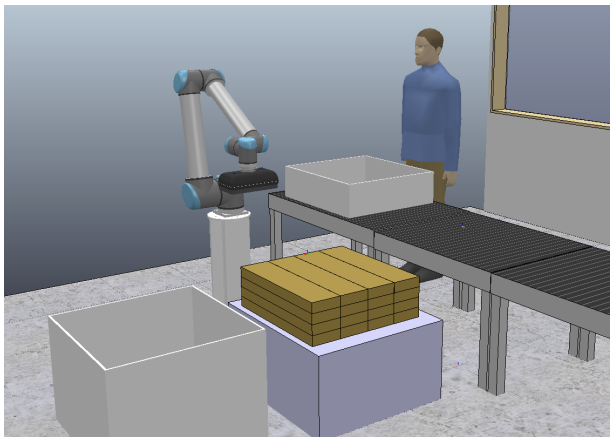


Figure 7.6: Posture where the robot starts and returns in each task cycle

and finally, the movement *place* is also composed of 3 waypoints followed by the vacuum deactivation.

7.4.1 Pick Movement

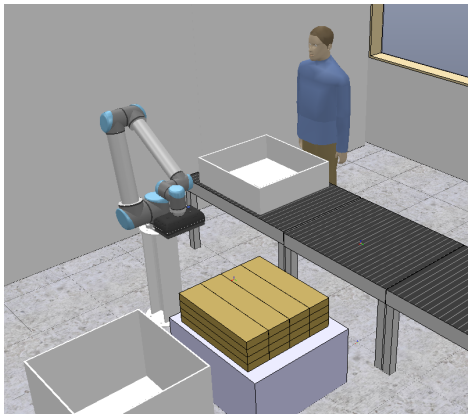
In order to plan the *Pick* movement, simple sequences of waypoints, Figure 7.7, were addressed to maintain the focus on the human-likeness of the robot movement. As explained in the section 7.3.1, the home pose is considered the first waypoint (Figure 7.7a), and the board position is the last waypoint (Figure 7.7c). The second waypoint, Figure 7.7b, guarantees that the robot approaches the board in the z-axis. In this dissertation, the waypoint that corresponds to the board position is defined manually by the user. Further on, in section 8.1, it will be explained how this could be automated with no human action required.

Table 7.2: Waypoints to accomplish the show movement in the task 7.3.1

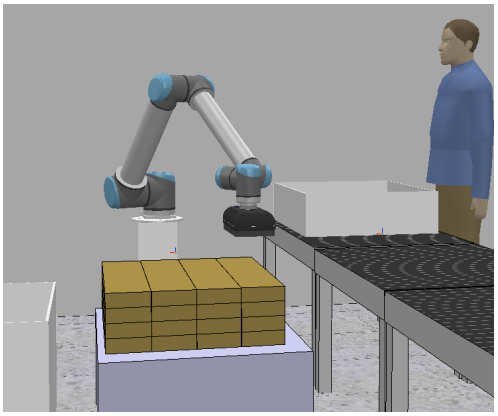
	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$
$\boldsymbol{\theta_{wp1}}$	-180	-80	-130	30	90	0
$\boldsymbol{\theta_{wp2}}$	-155.71	-41.12	-86.43	38.05	89.79	-66.71
$\boldsymbol{\theta_{wp3}}$	-155.71	-51.24	-87.32	49.05	89.79	-66.71

The defined waypoints, table 7.6, are the input of the Motion Planner module. Firstly is determined the duration of the movement by means of equation 6.58. Then, the time when the robot passes through each waypoint, equation 6.51, is computed followed by the Human-like trajectory, equation 6.48.

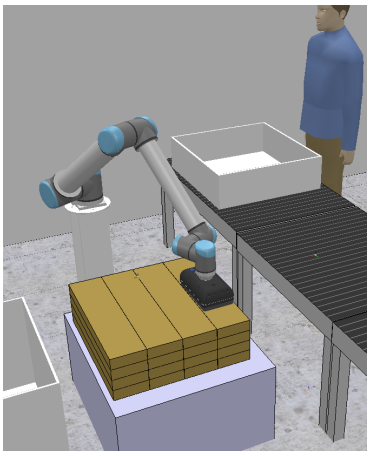
As we can observe from Figure 7.8, the executed movement to pick the board presents similar properties to the experiments in upper-limb movements. The movement is defined by a uni-modal velocity shape



(a) First waypoint



(b) Second waypoint



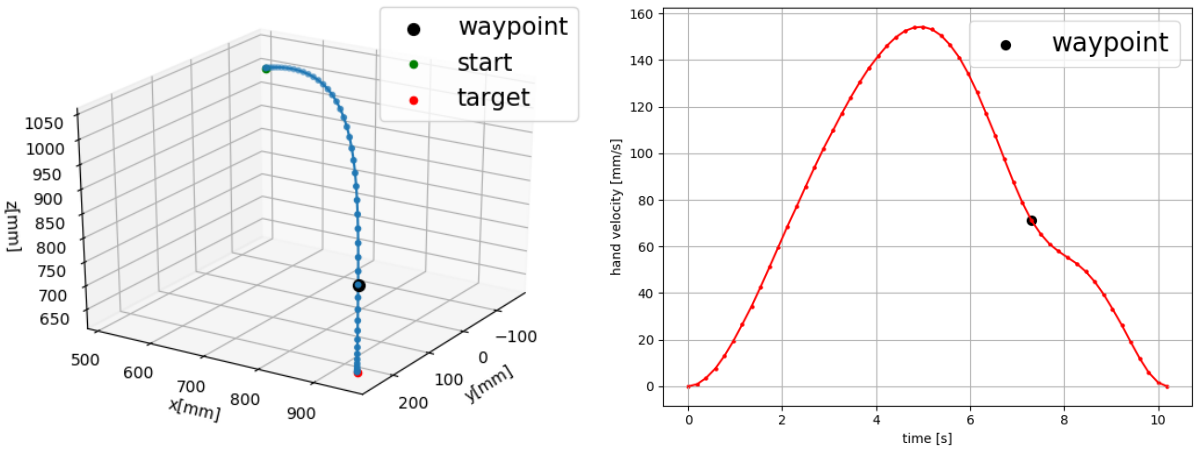
(c) Third waypoint

Figure 7.7: Waypoints sequence of the pick movement

profile (Figure 7.8b) and a curved path (Figure 7.8a). Particularly, all joints present a smooth movement with no jerk pikes, as demonstrated by Figure 7.9. Quantitatively, considering the human-like evaluation metrics introduced in section 7.2, the Normalised Jerk Score (**NJS**) is 41.75 and the Number of Movement Units (**NMU**) is 1, which confirms the human-likeness and the smoothness of the movement. The total duration of the movement is 9.9 seconds, and the planning solving time is 0.49 seconds. Furthermore, the robot achieves the waypoint θ_{wp2} at time 7.3 seconds.

Table 7.3: Results of the movement pick planning

NMU	NJS	Solving Time (ms)	Movement Duration (sec)
1	41.75	490	9.9

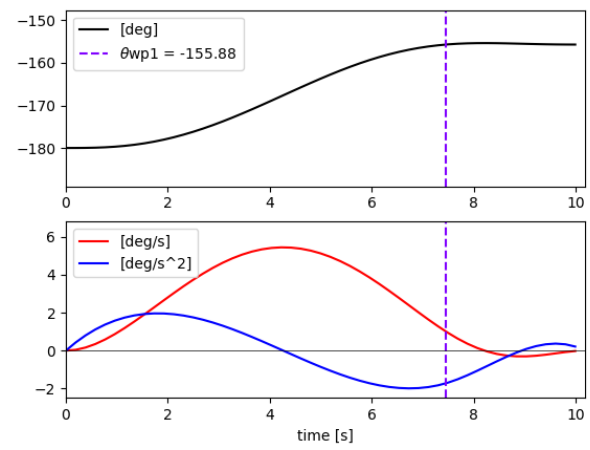


(a) Hand position

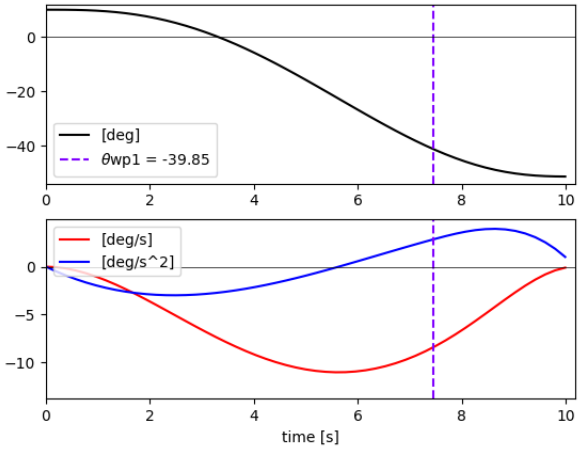
(b) Hand velocity

Figure 7.8: Position and velocity of the hand during the pick movement. The waypoint position is marked by the dashed line.

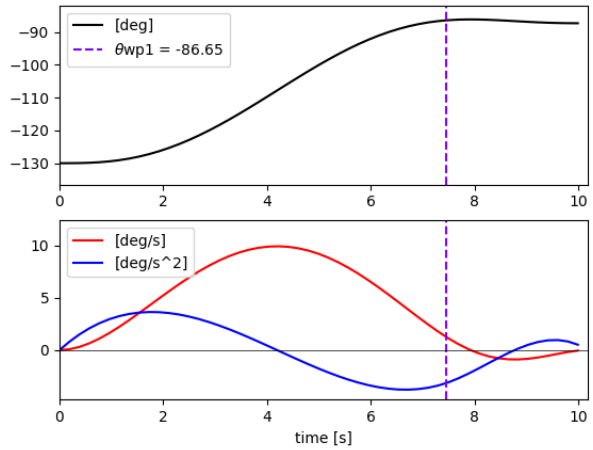
Regarding the precision of the planner, the robot achieved the waypoint with an error in the average of 0.96%. More specifically, table 7.5 shows the absolute error between the joints **Expected** and **Calculated**, $[0.17, 1.27, 0.22, 1.95, 0, 0.46]$, with an average of 0.68 degrees. This error corresponds to a distance of the robot hand position on a scale of $[2.94, 4.46, -17.17]$mm, table 7.4. Note that the first and last waypoint, i.e, the initial and final pose, are always accomplished with 100% of precision.



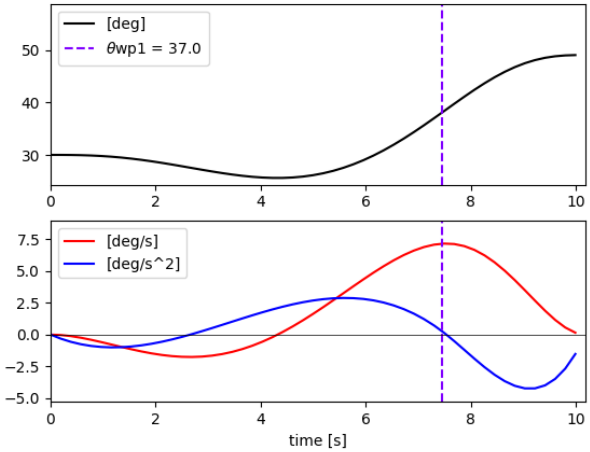
(a) Joint 1



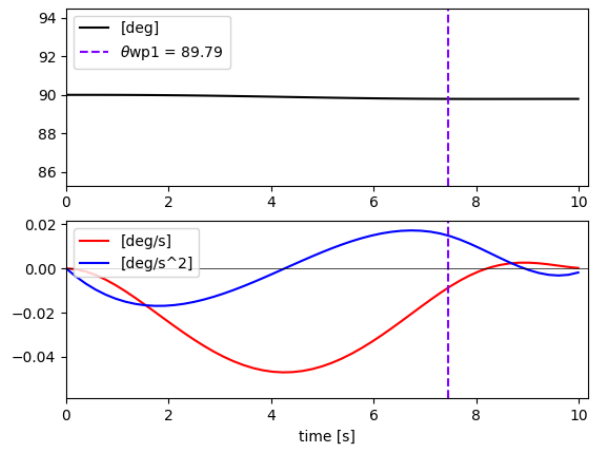
(b) Joint 2



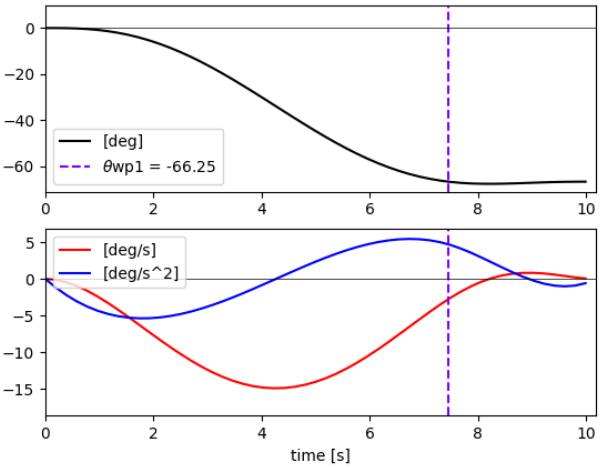
(c) Joint 3



(d) Joint 4



(e) Joint 5



(f) Joint 6

Figure 7.9: Joint position(black line), velocity(red line) and acceleration(blue line) profile during the pick movement. The waypoint is marked with a dashed line.

Table 7.4: Comparison between the expected and the calculated robot hand pose at the waypoints

Waypoint		x_e[mm]	y_e[mm]	z_e[mm]	Roll(γ)	Pitch(β_e)	Yaw(α_e)
	Expected	955.29	250.39	789.45	1.59	0.0	3.13
θ_{wp2}	**Calculated**	952.35	245.93	806.62	1.58	0.0	3.13
	Absolute Error	2.94	4.46	-17.17	0.01	0.0	0.0

Table 7.5: Comparison between the expected and the calculated robot joints values at the waypoints

Waypoint		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	**Error °**
	Expected	-155.71	-41.12	-86.43	38.05	89.79	-66.71	
θ_{wp2}	**Calculated**	-155.88	-39.85	-86.65	37.00	89.79	-66.25	0.53
	Absolute Error	0.17	1.27	0.22	1.05	0.0	0.46	

7.4.2 Show Movement

To plan the show movement, the waypoints in table 7.2 were defined. As explained in section 7.3.1, the first waypoint of the show movement (Figure 7.10a) is the last waypoint of the Pick movement. The second waypoint, Figure 7.10b, is a safety waypoint to define a retreat phase in the z-axis and avoid collisions when removing the board from the palette. The following waypoints are set in critical positions to inspect the boards. Thus, the third and fourth waypoints, Figure 7.10c and 7.10d, are set to inspect the left and right lateral of the board, respectively. The fifth waypoint, Figure 7.10e, is defined to observe the top of the board, and, finally, the sixth waypoint, Figure 7.10e, allows the inspection of the button part of the board.

Table 7.6: Waypoints to accomplish the show movement in the task 7.3.1

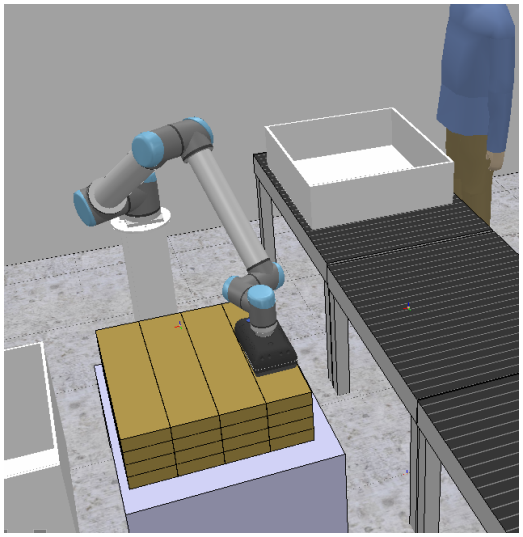
	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\boldsymbol{\theta_4}$	$\boldsymbol{\theta_5}$	$\boldsymbol{\theta_6}$
$\boldsymbol{\theta_{wp1}}$	-155.71	-51.24	-87.32	49.05	89.79	-66.71
$\boldsymbol{\theta_{wp2}}$	-154.46	-41.97	-85.03	35.65	89.77	-65.46
$\boldsymbol{\theta_{wp3}}$	-105.79	-27.80	-65.86	12.11	113.33	30.02
$\boldsymbol{\theta_{wp4}}$	-48.82	-24.91	-73.13	30.95	78.15	-31.63
$\boldsymbol{\theta_{wp5}}$	-68.88	-28.02	-63.02	2.51	86.55	19.51
$\boldsymbol{\theta_{wp6}}$	-64.71	-35.60	-79.25	148.72	67.02	-15.35

The defined waypoints, table 7.2, are the input of the Motion Planner module. Firstly is determined the duration of the movement by means of equation 6.58. Then, the time when the robot passes through each waypoint, equation 6.51, is computed followed by the Human-like trajectory, equation 6.48.

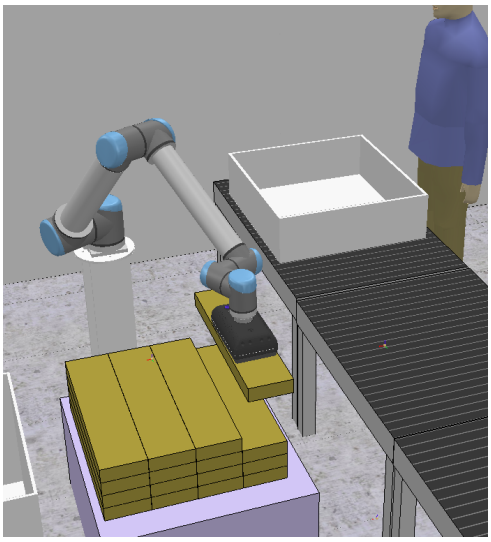
As we can observe from Figure 7.11, the executed movement to pick the board presents similar properties to the experiments in upper-limb movements. The movement is defined by a multi bell-shaped velocity profile (Figure 7.11b) and a curved path (Figure 7.11a). Particularly, all joints present a smooth movement with no jerk pikes and a bell-shaped velocity profile, as demonstrated by Figure 7.12. Quantitatively, considering the human-like evaluation metrics introduced in section 7.2, the Normalised Jerk Score (**NJS**) is 349.76 and the Number of Movement Units (**NMU**) is 1, which confirms the human-likeness and the smoothness of the movement. The successive bell-shaped peaks in the hand velocity profile are caused by the high rate of change in hand orientation demanded by the waypoints. A similar event is often observed in humans, especially when avoiding collisions with obstacles [Gulletta et al. (2020)]. Additionally, the total duration of the movement is 9.9 seconds, and the planning solving time is 1.26 seconds. Furthermore, the robot achieves θ_{wp2} at 3.7 seconds, θ_{wp3} at 12.15 seconds, θ_{wp4} at 20.59 seconds and θ_{wp5} at 24.82 seconds.

Table 7.7: Results of the movement pick planning

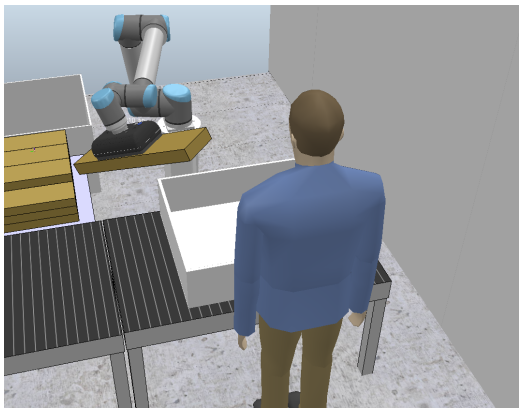
NMU	**NJS**	**Solving Time (ms)**	**Movement Duration (sec)**
1	349.76	1245	32.21



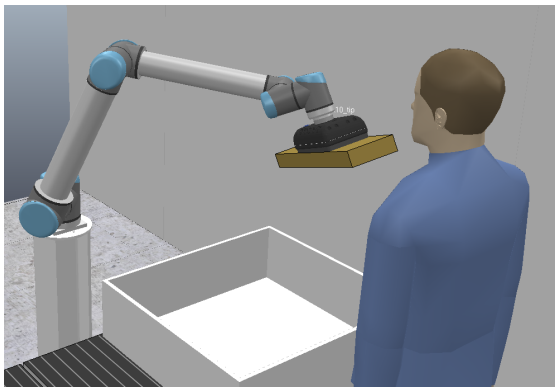
(a) First waypoint



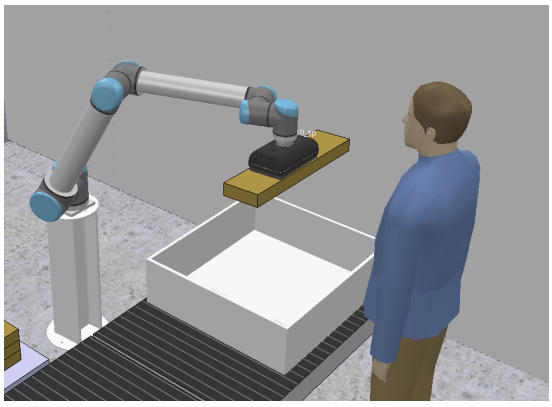
(b) Second waypoint



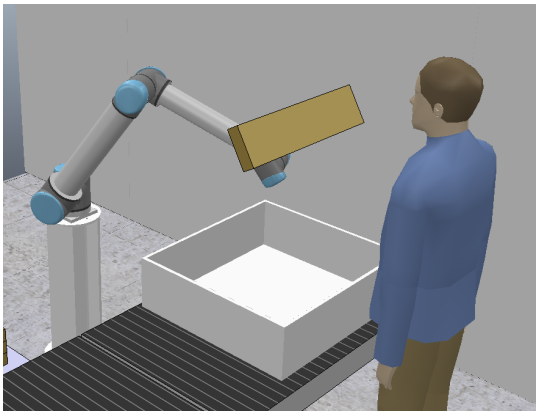
(c) Third waypoint



(d) Fourth waypoint

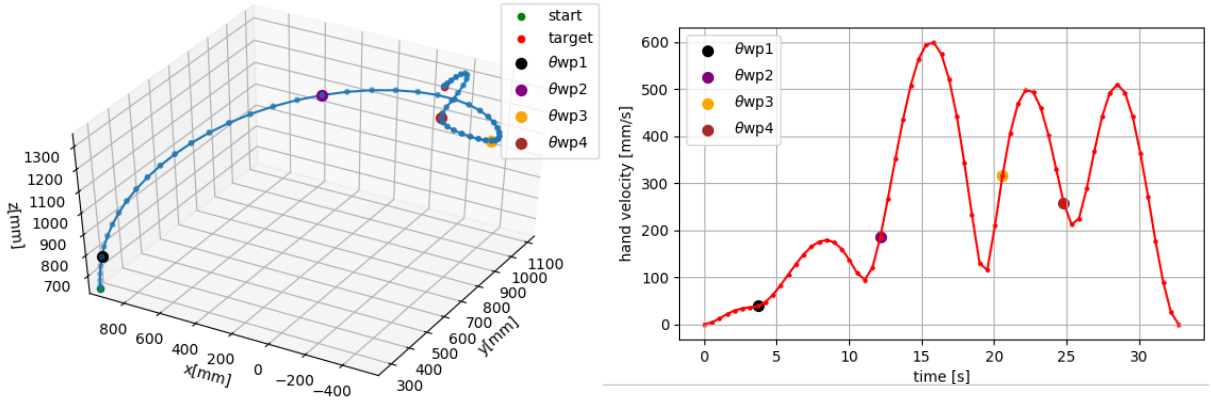


(e) Fifth waypoint



(f) Sixth waypoint

Figure 7.10: Waypoints sequence of the show movement

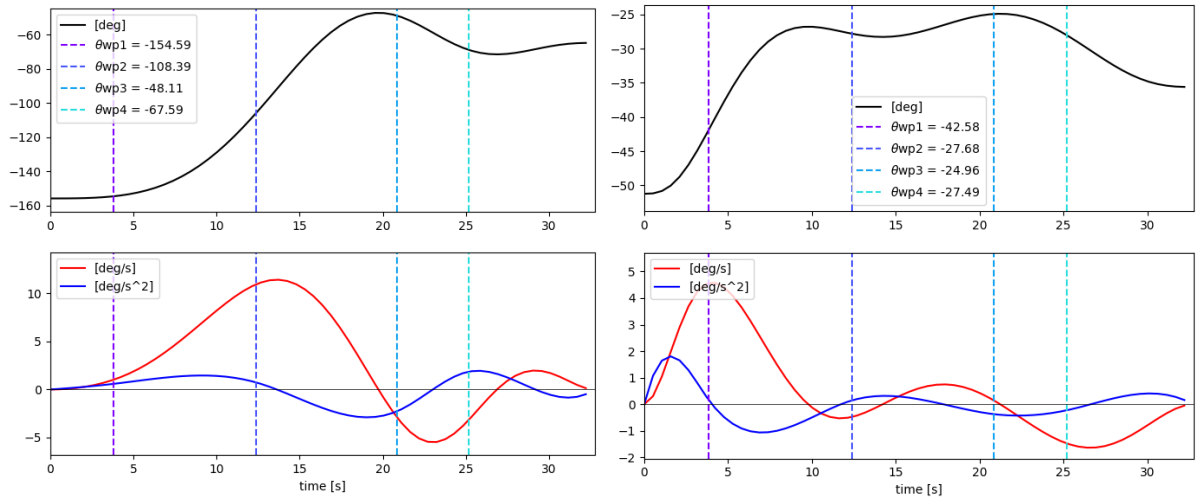


(a) Hand position

(b) Hand velocity

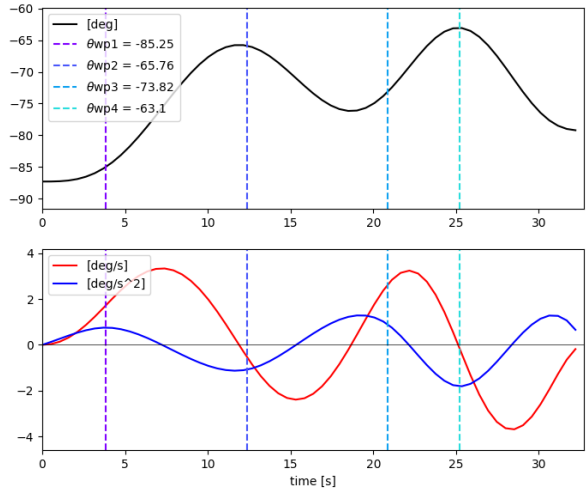
Figure 7.11: Position and velocity of the hand during the show movement.

Regarding the precision of the planner, the robot achieved the waypoints θ_{wp2},θ_{wp3},θ_{wp4} and θ_{wp5} with an error in average of 0.43%, 1.05%, 1.3% and 1.05%, respectively. More specifically, table 7.9 shows, for each waypoint, the absolute error between the **Expected** and **Calculated** value of all joints. The joints error in second waypoint is $[0.13, 0.61, 0.22, 1.01, 0.12, 0.49]$, which corresponds to a distance of the robot hand position in a scale of $[1.18, 2.40, 10.58]$mm, table 7.8. The joints error in the third waypoint is $[3.4, 0.12, 0.1, 2.11, 0.49, 0.09]$, which corresponds to a distance of the robot hand position in a scale of $[38.9, 22.84, 1.32]$mm. The joints error in the fourth waypoint is $[0.71, 0.05, 0.69, 3.96, 0.2, 2.17]$, which corresponds to a distance of the robot hand position in a scale of $[15.13, 0.94, 3.12]$mm. The joints error in the fifth waypoint is $[1.29, 0.53, 0.08, 1.23, 0.7, 2.47]$, which corresponds to a distance of the robot hand position in a scale of $[15.47, 19.91, 0.47]$mm. Note that the first and last waypoint, i.e, the initial and final pose, are always accomplished with 100% of precision.

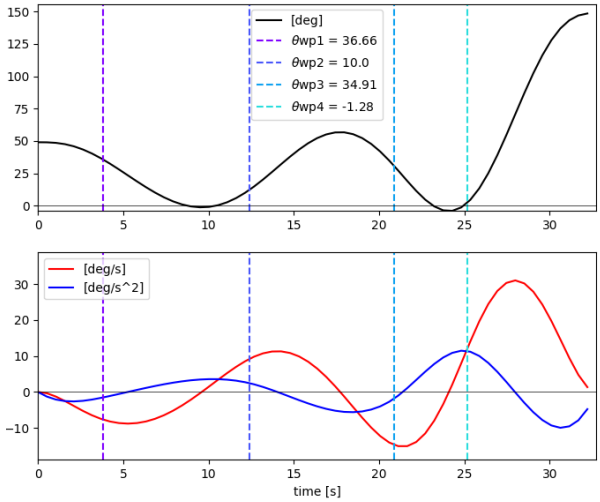


(a) Joint 1

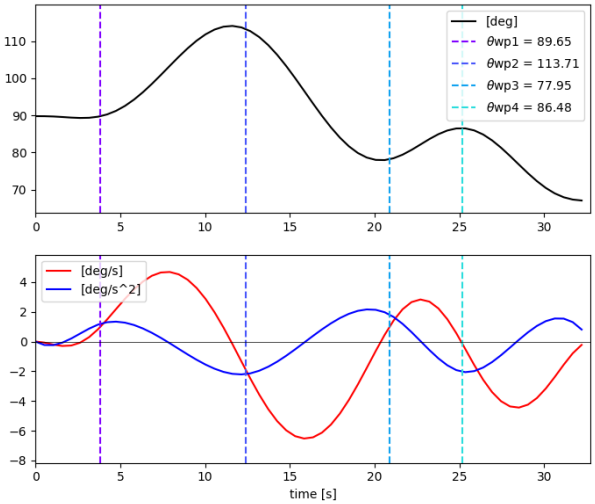
(b) Joint 2



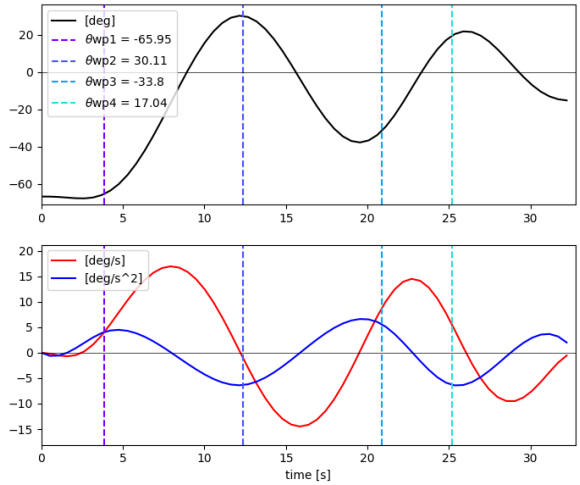
(c) Joint 3



(d) Joint 4



(e) Joint 5



(f) Joint 6

Figure 7.12: Joint position(black line), velocity(red line) and acceleration(blue line) profile during the show movement. The waypoints are marked by dashed lines.

Table 7.8: Comparison between the expected and the calculated robot hand pose at the waypoints

Waypoint		x_e[mm]	y_e[mm]	z_e[mm]	Roll(γ)	Pitch(β_e)	Yaw(α_e)
	Expected	952.86	272.65	784.11	1.59	0.01	-3.12
θ_{wp2}	**Calculated**	954.04	270.25	773.53	1.59	0.01	-3.13
	Absolute Error	1.18	2.40	10.58	0	0	0.01
	Expected	352.44	937.76	1216.48	0.82	0.07	-2.72
θ_{wp3}	**Calculated**	391.34	914.92	1215.16	0.77	0.10	-2.72
	Absolute Error	38.9	22.84	1.32	0.5	0.03	0
	Expected	-508.99	893.96	1216.07	2.85	-0.23	2.76
θ_{wp4}	**Calculated**	-524.12	893.02	1219.19	2.9	-0.26	2.71
	Absolute Error	15.13	0.94	3.12	0.05	0.03	0.05
	Expected	-188.94	978.17	1209.31	1.60	-0.04	-3.09
θ_{wp5}	**Calculated**	-204.41	958.26	1209.78	1.67	0.01	3.07
	Absolute Error	15.47	19.91	0.47	0.07	0.03	0.02

Table 7.9: Comparison between the expected and the calculated robot joints values at the waypoints.

Waypoint		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	**Error °**
	Expected	-154.46	-41.97	-85.03	35.65	89.77	-65.46	
θ_{wp2}	**Calculated**	-154.59	-42.58	-85.25	36.66	89.65	-65.95	0.43
	Absolute Error	0.13	0.61	0.22	1.01	0.12	0.49	
	Expected	-105.79	-27.80	-65.86	12.11	113.22	30.02	
θ_{wp3}	**Calculated**	-108.39	-27.68	-65.76	10.00	113.71	30.11	1.05
	Absolute Error	3.4	0.12	0.1	2.11	0.49	0.09	
	Expected	-48.82	-24.91	-73.13	30.95	78.15	-31.63	
θ_{wp4}	**Calculated**	-48.11	-24.96	-73.82	34.91	77.95	-33.80	1.3
	Absolute Error	0.71	0.05	0.69	3.96	0.20	2.17	
	Expected	-68.88	-28.02	-63.02	2.51	86.55	19.51	
θ_{wp5}	**Calculated**	-67.59	-27.49	-63.10	-1.28	86.48	17.04	1.05
	Absolute Error	1.29	0.53	0.08	1.23	0.7	2.47	

7.4.3 Place Movement

The place movement is performed after the show movement and in accordance with what the operator decides. More specifically, if the inspected board is in good conditions, the operator approves it and the board is placed in the box in front of the operator. Otherwise, if the board is faulty, it should be placed into the other box next to the robot. Thus, the place movement is divided into two types: *Place Approved* and *Place Faulty*. The *Place Approved* is the movement to place the approved board after inspection, and the *Place Faulty* is the movement to place the rejected board in the inspection phase.

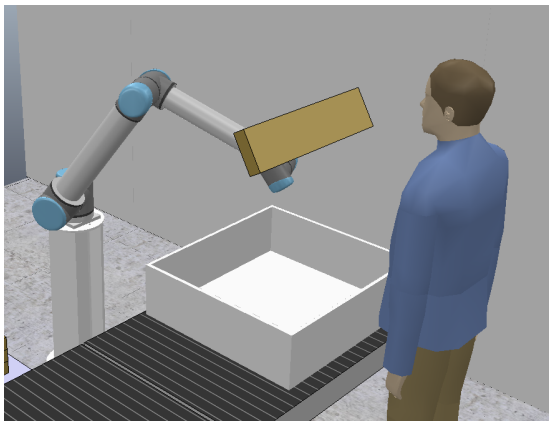
To plan both movements, simple sequences of waypoints, Figure 7.14 and 7.17, were addressed to maintain the focus on the human-likeness of the robot movement. As explained in section 7.3.1, the first waypoint of the place movement (Figure 7.13a and 7.16a) is the last waypoint of the show movement. Therefore, the movements *Place Approval* and *Place Faulty* have the first waypoint in common. Both movements have similar waypoints, i.e. the second waypoint (Figure 7.13b and 7.16b) in both movements is for safety reasons, which addresses the approach phase, in the z-axis, and guarantees no collisions with

the box. The third waypoint, 7.13c and 7.16c, is set to release the board in that position. When the robot achieves the last waypoint, the vacuum action is deactivated.

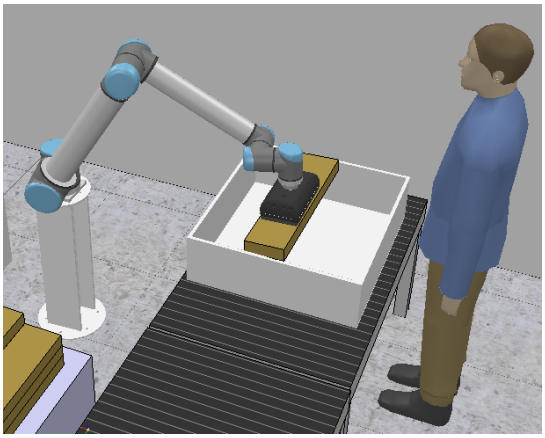
Place Approval

Table 7.10: Waypoints to accomplish the *Place Approved* movement in the task 7.3.1

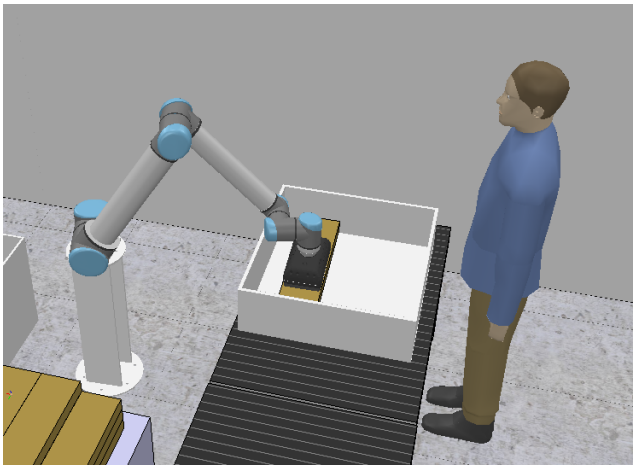
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{wp1}	-64.71	-35.60	-79.25	148.72	67.02	-15.35
θ_{wp2}	-72.30	-19.58	-92.11	19.99	88.95	16.12
θ_{wp3}	-72.86	-34.29	-104.37	46.96	88.96	16.40



(a) First waypoint



(b) Second waypoint



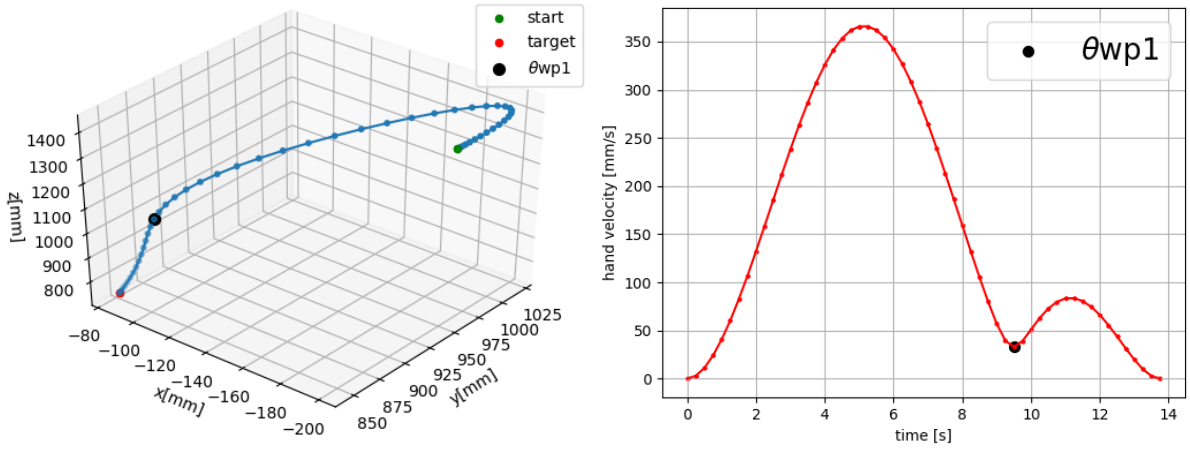
(c) Third waypoint

Figure 7.13: Waypoints sequence of the place approved movement

As we can observe from Figure 7.14, the executed movement to place the board presents similar properties to the experiments in upper-limb movements. The movement is defined by a bell-shape velocity profile (Figure 7.14b) and a curved path (Figure 7.14a). Particularly, all joints present a smooth movement with no jerk pikes, also with bell-shaped velocity profile, as demonstrated by Figure 7.15. Quantitatively, considering the human-like evaluation metrics introduced in section 7.2, the Normalised Jerk Score (**NJS**) is 70.30, which confirms the smoothness of the movement. The Number of Movement Units (**NMU**) is 1, which is the same as the observed in upper-limb movements. The total duration of the movement is 13.5 seconds, and the planning solving time is 0.53 seconds. Furthermore, the robot achieves the second waypoint, θ_{wp2}, at 9.50 seconds.

Table 7.11: Planning results of the movement place

NMU	NJS	Solving Time (ms)	Movement Duration (sec)
1	70.30	530	13.5

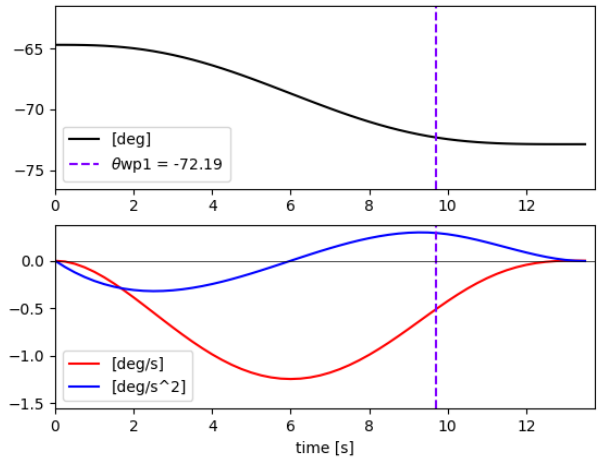


(a) Hand position

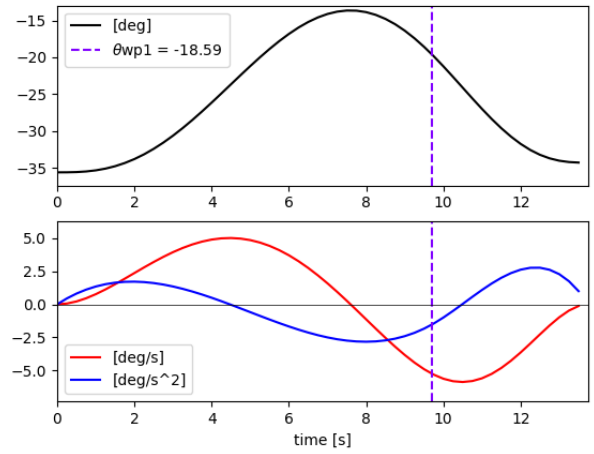
(b) Hand velocity

Figure 7.14: Position and velocity of the hand during the *Place Approved* movement.

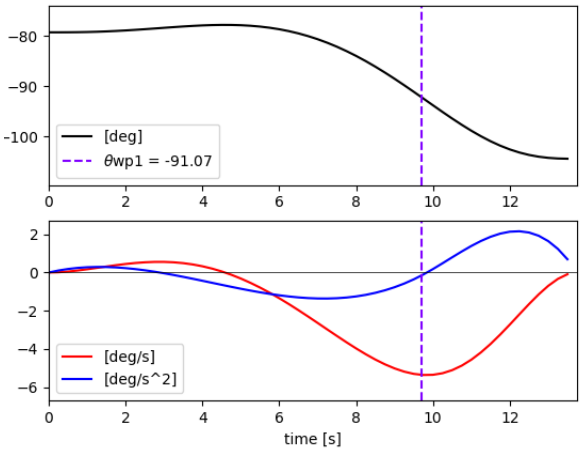
Regarding the precision of the planner, the robot achieved the waypoint with an error in the average of 0.6%. More specifically, table 7.13 shows the absolute error between the joints **Expected** and **Calculated**, $[0.11, 1.01, 1.04, 0.99, 0.18, 0.28]$, with an average of 0.6 degrees. This error corresponds to a distance of the robot hand position on a scale of $[1.25, 0.87, 24.8]$mm, table 7.12. Note that the first and last waypoint, i.e, the initial and final pose, are always accomplished with 100% of precision.



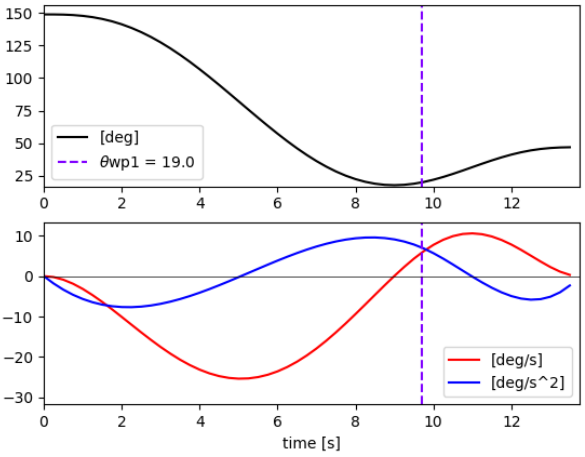
(a) Joint 1



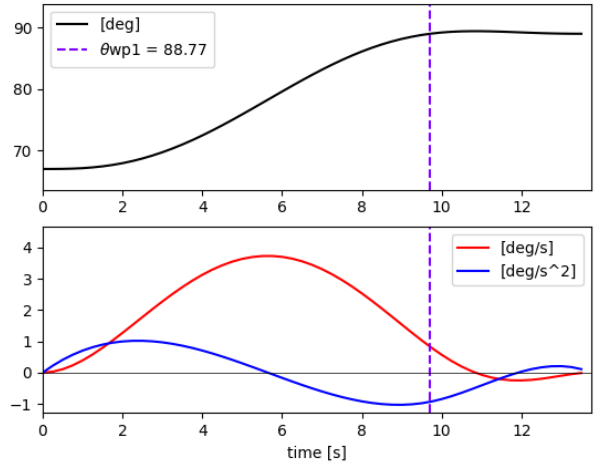
(b) Joint 2



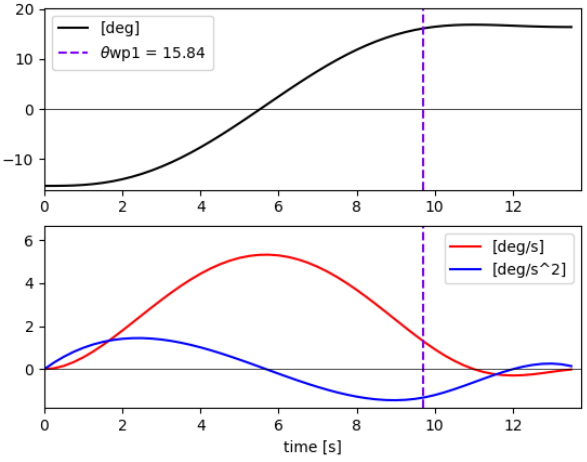
(c) Joint 3



(d) Joint 4



(e) Joint 5



(f) Joint 6

Figure 7.15: Joint position (black line), velocity (red line) and acceleration (blue line) profile during the *Place Approved* movement. The waypoint θ_{wp2} is marked by a dashed line.

Table 7.12: Comparison between the expected and the calculated robot hand pose at waypoint θ_{wp2} of the *Place Approved* movement.

Waypoint		x_e[mm]	y_e[mm]	z_e[mm]	Roll(γ)	Pitch(β_e)	Yaw(α_e)
	Expected	-97.61	857.45	1037.85	1.6	0.02	3.12
θ_{wp2}	**Calculated**	-98.86	858.32	1062.43	1.61	0.01	3.12
	Absolute Error	1.25	0.87	24.8	0.01	0.01	0.0

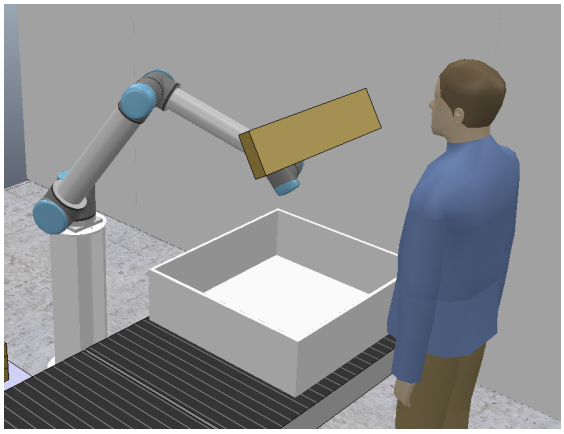
Table 7.13: Comparison between the expected and the calculated robot joints values at waypoint θ_{wp2} of the *Place Approved* movement.

Waypoint		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	**Error °**
	Expected	-72.30	-19.58	-92.11	19.99	88.95	16.12	
θ_{wp2}	**Calculated**	-72.19	-18.59	-91.07	19.00	88.77	15.84	0.6
	Absolute Error	0.11	1.01	1.04	0.99	0.18	0.28	

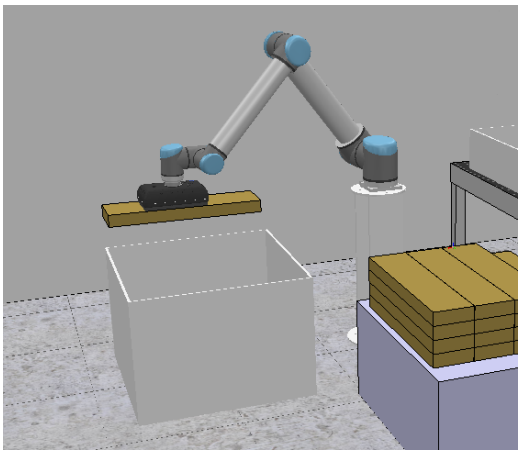
Place Faulty Movement

Table 7.14: Waypoints to accomplish the *Place Faulty* movement in the task 7.3.1

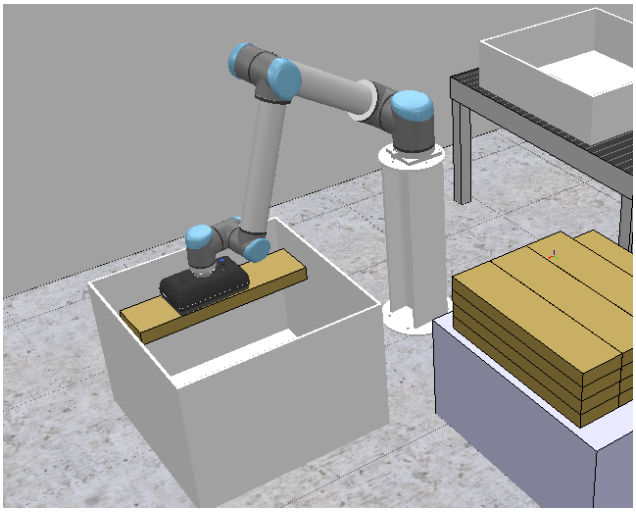
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
θ_{wp1}	-64.71	-35.60	-79.25	148.72	67.02	-15.35
θ_{wp2}	-261.10	-33.62	-107.53	52.41	87.85	-84.63
θ_{wp3}	-261.10	-55.80	-104.68	71.74	87.85	-84.63



(a) First waypoint



(b) Second waypoint



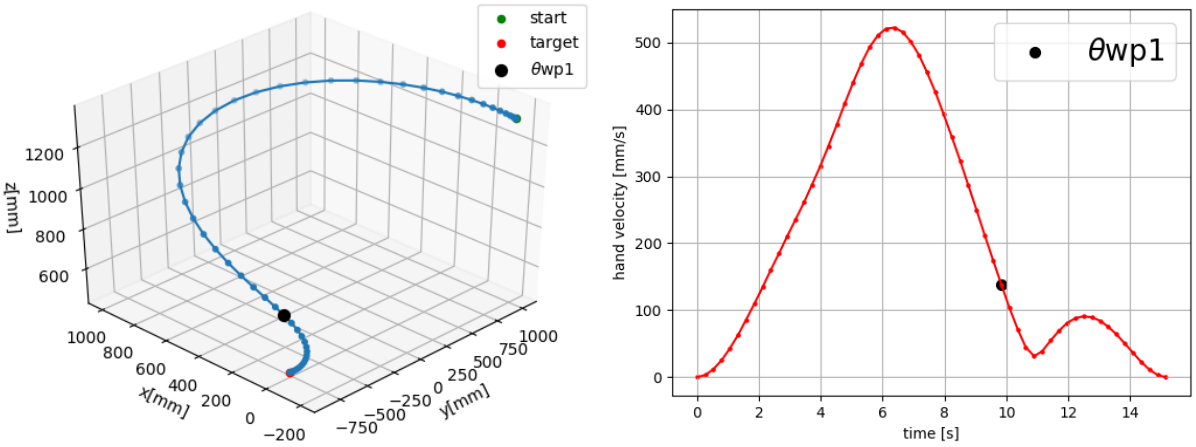
(c) Third waypoint

Figure 7.16: Waypoints sequence of the place approved movement

As we can observe from Figure 7.17, the executed movement to place the board presents similar properties to the experiments in upper-limb movements. The movement is defined by a bell-shape velocity profile (Figure 7.17b) and a curved path (Figure 7.17a). Particularly, all joints present a smooth movement with no jerk pikes, also with bell-shaped velocity profile, as demonstrated by Figure 7.18. Quantitatively, considering the human-like evaluation metrics introduced in section 7.2, the Normalised Jerk Score (**NJS**) is 185.36 and the the Number of Movement Units (**NMU**) is 1, which is the same as the observed in upper-limb movements and confirms the smoothness of the movement. The total duration of the movement is 14.9 seconds, and the planning solving time is 0.54 seconds. Furthermore, the robot achieves the second waypoint, θ_{wp2}, at 9.84 seconds.

Table 7.15: Planning results of the movement *Place Faulty*

NMU	NJS	Solving Time (ms)	Movement Duration (sec)
1	185.36	540	14.9



(a) Hand position

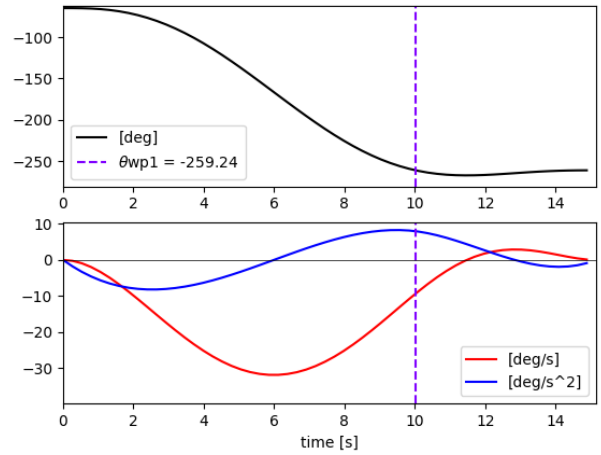
(b) Hand velocity

Figure 7.17: Position and velocity of the hand during the *Place Faulty* movement.

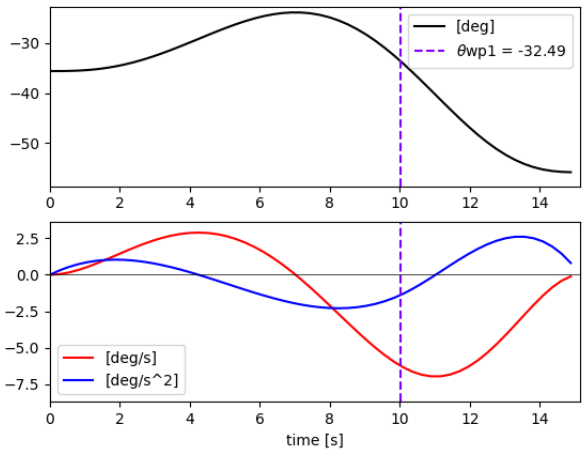
Regarding the precision of the planner, the robot achieved the waypoint with an error in the average of 0.81%. More specifically, table 7.17 shows the absolute error between the joints **Expected** and **Calculated**, $[1.86, 1.13, 0.12, 0.91, 0.2, 0.66]$, with an average of 0.81 degrees. This error corresponds to a distance of the robot hand position on a scale of $[27.11, 4.56, 16.98]$mm, table 7.16. Note that the first and last waypoint, i.e, the initial and final pose, are always accomplished with 100% of precision.

Table 7.16: Comparison between the expected and the calculated robot hand pose at waypoint θ_{wp2} of the *Place Faulty* movement.

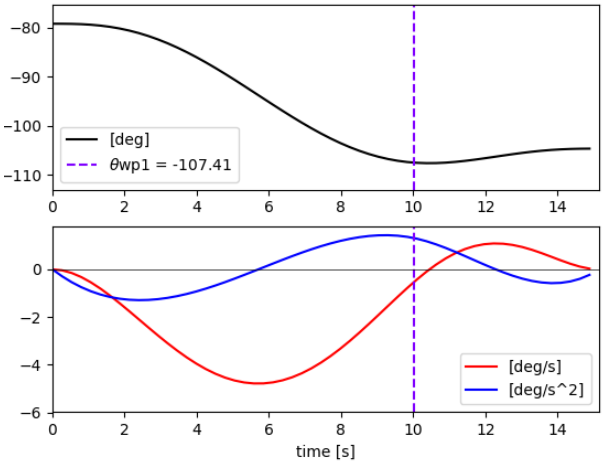
Waypoint		x_e[mm]	y_e[mm]	z_e[mm]	Roll(γ)	Pitch(β_e)	Yaw(α_e)
	Expected	-42.91	-834.79	742.98	0.06	0.04	3.12
θ_{wp2}	**Calculated**	-15.80	-839.35	759.96	0.08	0.04	3.10
	Absolute Error	27.11	4.56	16.98	0.02	0.00	0.02



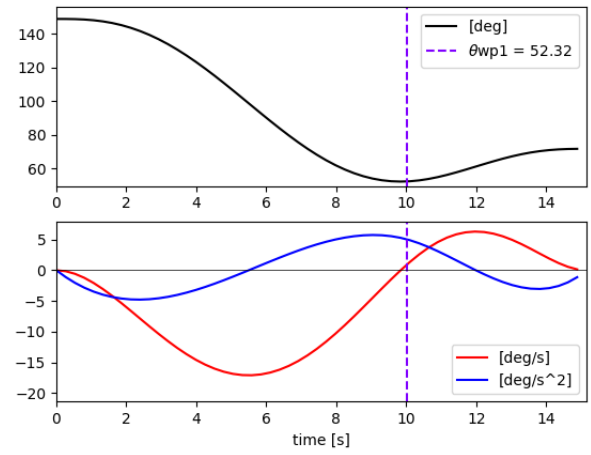
(a) Joint 1



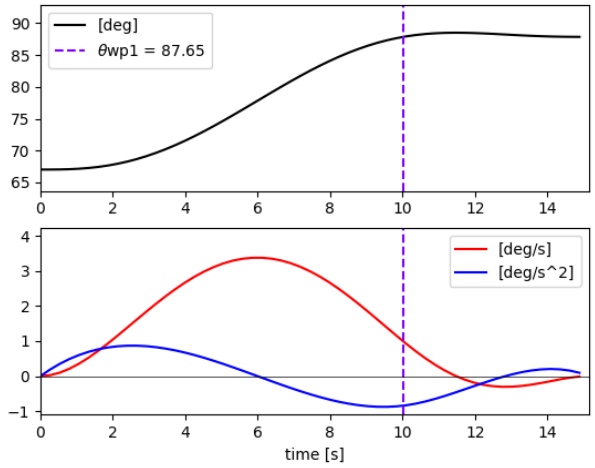
(b) Joint 2



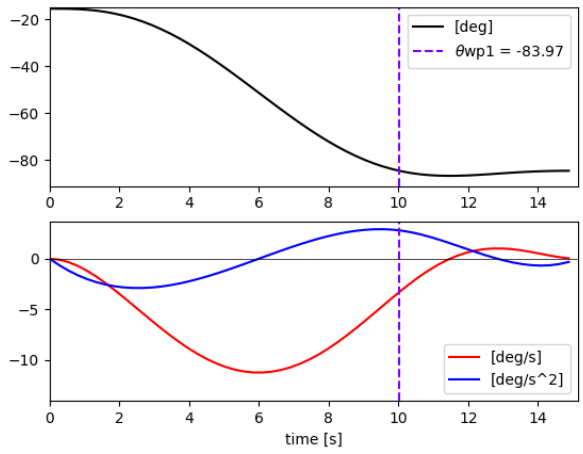
(c) Joint 3



(d) Joint 4



(e) Joint 5



(f) Joint 6

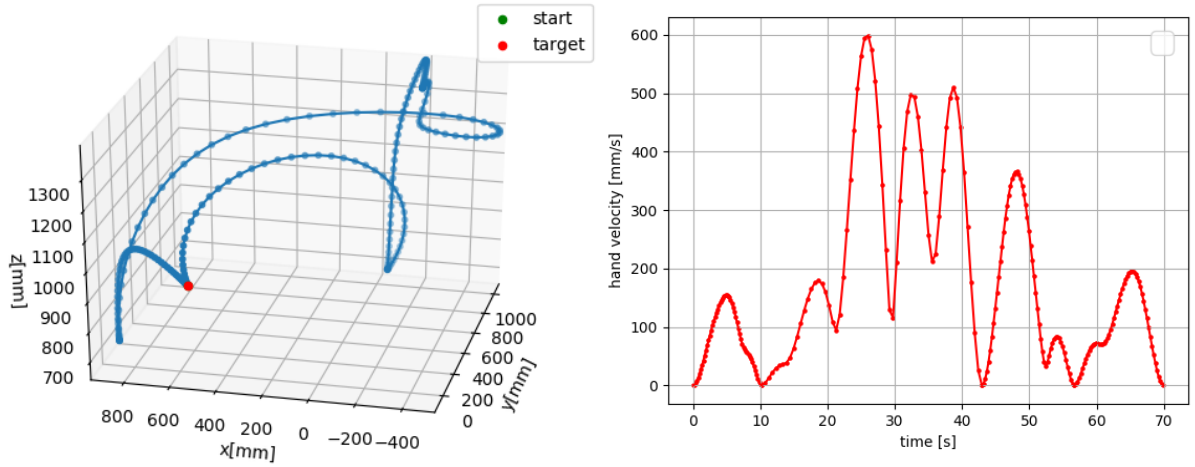
Figure 7.18: Joint position (black line), velocity (red line) and acceleration (blue line) profile during the *Place Faulty* movement. The waypoint θ_{wp2} is marked by a dashed line.

Table 7.17: Comparison between the expected and the calculated robot joints values at waypoint θ_{wp2} of the *Place Faulty* movement.

Waypoint		θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	**Error °**
	Expected	-261.10	-33.62	-107.53	52.41	87.85	-84.63	
θ_{wp2}	**Calculated**	-259.24	-32.49	-107.41	52.32	87.65	-83.97	0.81
	Absolute Error	1.86	1.13	0.12	0.91	0.2	0.66	

7.4.4 Task Results

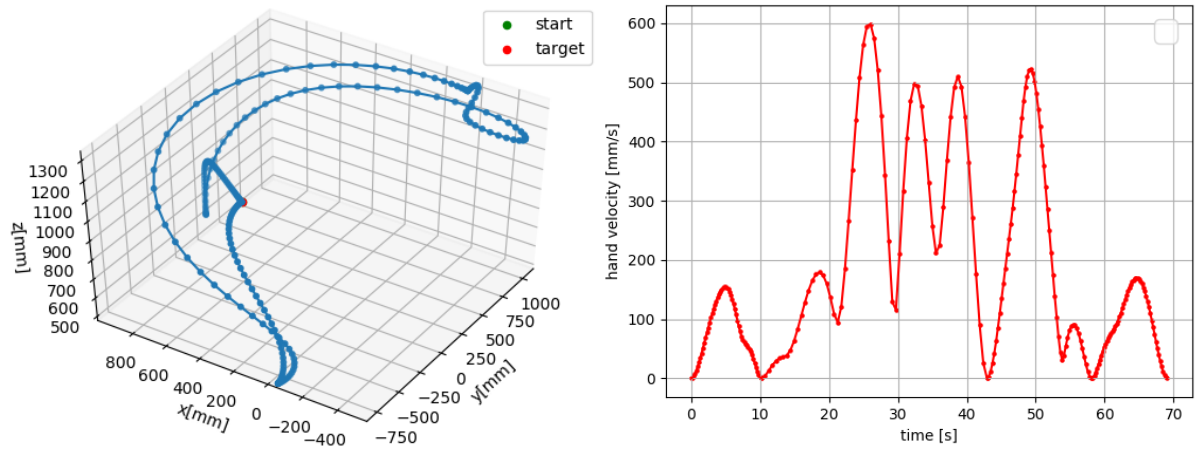
The concatenation of the movements defines the overall task. In Figure 7.19 and Figure 7.20, one can observe hand velocity profile and trajectory since the home pose until the placement of the board in the box in front of the operator, and in the faulty box, respectively. Since the robot starts and finishes the task in the home posture, the initial and final hand position correspond to the same point in the space (Figure 7.19a and 7.20a). The velocity hand profile, represented in Figure 7.19b and Figure 7.20b, clearly shows that vacuum activation and deactivation defines the phases of the movement. These occurrences are the points in time during the entire task execution when the hand come to rest.



(a) Hand position

(b) Hand velocity

Figure 7.19: Position and velocity of the hand during the task



(a) Hand position

(b) Hand velocity

Figure 7.20: Position and velocity of the hand during the task

7.5 Discussion

The validation of the trajectory planning through waypoints method is processed in a quality inspection task, where the user selects the critical points of the robot posture. The waypoints allow the human operator to easily program a new task with no necessary previous deep knowledge about robot kinematics and trajectory planning. The resultant movements are, as pretended, smooth, pleasant, intuitive, and easily understandable. Besides, this method is efficient, robust, and highly flexible since it can compute a trajectory crossing N waypoints for a robot with K degrees of freedom.

The trajectory planner achieves similar results observed to those seen in upper-limb experiments in areas of psychology and neuroscience. In summary, these properties are associated with bell-shaped velocity profile [Flash and Hogan (1985); Rosenbaum et al. (1995, 2001); Wada and Kawato (2004)]; decomposition of hand movement in sub-movements [Burdet et al. (2013)]; a low value of Normalised Jerk Score (NJS), which corresponds to a smooth movement of the hand [Flash and Hogan (1985)]; the existence of one movement unit (NMU), or even more in case of significant rotation of the joint wrist, or when avoiding obstacles [Rosenbaum et al. (2001),Gulletta et al. (2020)].

The robot's hand and joints performs a smooth movement with bell-shaped velocity profiles in the entire task. The maximum NJS obtained was 349.76 in the *show* movement, which is very acceptable due to the complexity of the task. Interestingly, in movement *show* we observe more than one bell shaped peak in the hand velocity profile, however we still obtain a NMU of 1. This happens due to the threshold of 10% defined to determine number of movement units. Nevertheless, more than one bell-shaped peak are

usually observed when considering obstacle avoidance. Here, we do not deal with obstacles, but we have waypoints, which are constraints in the path. The reason for the peaks in velocity profile is the high rate of wrist rotation demanded by the waypoints (Figure 7.10). Hence, these movements follow human-like metrics observed in human experiments; however, we can not affirm that the movements are absolutely human-like, since there are no experiments of human's movements considering waypoints.

By planning different movements, we conclude the results presented are directly affected by: the complexity of the movement; the number of waypoints; selection and disposition in the space of the waypoints; kinematics of the robot, specifically, the number of degrees of freedom. The complexity of the system increases significantly as the number of waypoints increases, and, naturally, the degrees of freedom. This can be easily understood when analysing sections 6.1.3 and 6.1.2. The algorithms are defined by recursive processes, which compute the Lagrange multipliers and when all waypoints are achieved. Thus, the system becomes more numerically expensive with more waypoints and degrees of freedom. Note that to our best knowledge, this approach is the most efficient. Therefore, a good selection of waypoints is of utmost importance to avoid non-critical trajectory positions and thus optimise performance. Besides the number of waypoints that may overload the system, the user must also be aware of their position in space. For safety, and no risk of collision, no waypoint should be set close to objects. In other words, the user must take into consideration the planning precision, i.e., the planner is acceptably precise, but still not completely.

Also important, the resultant joint trajectories from the planner are not monotonic. This can be observed in figures 7.9, 7.12, 7.15, and 7.18. For instance, Figure 7.15b illustrates this clearly; the initial and final joint positions are -35, while the waypoint is -20. What would be expected for a monotonic function is a constant growth of the function followed by a constant decrease after the waypoint. However, before the waypoint, the joint increases to -15 and then decrease to -20. In a two-point trajectory planning, the result is a monotonic function, where the joint value during the whole path constantly increases or decreases. However, the goal here is different; there are mandatory points to be passed through, and this restricts the planning method. The whole trajectory is computed with the concatenation of multiple trajectory segmentation's, however, each segmentation takes into account a coefficient (Lagrange multiplier) that represents all waypoints of the trajectory (refer to section 6.1.1 and equation 6.27). Thus, the planner returns the optimal path with the lowest joint jerk variations from the initial to the final positions passing through some mandatory points in between. Hence, to achieve this, a non-monotonic function may be necessary. The major practical consequence of the non-monotonic joint evolution is the risk of exceeding the robot joints limits. To avoid this event, a safety high-level verification is addressed to the planner.

Furthermore, another point to be considered is the execution time expended by this method. For instance, the *show* movement (refer to 7.4.2) takes 32.21 seconds, and the overall task to pick, inspect, and place the board takes approximately 70 seconds to be completed. Therefore, the resulting time is presumably expensive. Still, the execution time is influenced by many factors such as: computer Central Processor Unit (CPU) used to obtain the numeric results; the number of waypoints; selection and disposition in the space of the waypoints. Probably, a human operator would not need 70 seconds to perform such a task.

Hence, a user study would be necessary to get human experiments using waypoints for a more reasoned and critique description regarding the execution time and the overall robot motion.

Part VI

Conclusion

Chapter 8

Conclusion and Future Work

The emergence of industry 4.0 demands more autonomy and flexibility of the current industry. As a consequence, robotics has advanced and changed according to the needs of society. To address these needs, the research community proposed collaborative robots, which are are safe, easy-to-use and easy-to-program, and foremost are capable of sharing the workspace with human operators. Its programming is mainly achieved by recording and executing thereafter the demonstrations of human operators, which facilitates the operators effort when programming a task. These robots are considerably advantageous compared to traditional robots, and attend many requisites of industry 4.0. However, industry 5.0 is already coming and many authors predict that robots will live and constantly cooperate with humans in their daily routines. Thus, the performance of more pleasant, smooth, and predictable movements are required [Gulletta et al. (2020)].

The trajectory planning methods (chapter 3) currently used in industry are focused on the efficiency and computational cost. However, the industry still does not have any method capable of computing human-like trajectories, and more specifically, human-like trajectories through waypoints.

The proposed trajectory planning method presented in chapter 6 includes a set of characteristics associated with the behaviour of a human's arm observed in experiments. Considering the research of Martins de Sá (2018) and Gulletta et al. (2021), which defined all movements in pick, place and move, here we consider each movement through waypoints as a *move* movement. The planner takes inspiration from: (i) kinesthetic teaching programming method and (ii) upper-limb kinematic features observed in human arm movements. Particularly, this dissertation extends the Flash and Hogan (1985) approach, which presents a mathematical model that predicts experiments of coordination of voluntary human arm movements in unconstrained point-to-point motion and curved motions (through a mandatory waypoint in between). The proposed method considers N waypoints in between the initial and final points (instead of

only 1 waypoint), transfers the human's movements knowledge to robotics and minimzes the variation of K joints acceleration (instead of only the variation of hand acceleration). Mathematically, as explained in chapter 5, the trajectory is accomplish by solving an optimization problem where the criteria function is the time integral of the third derivative of the joints position, with the addition of Lagrange multipliers, which are constants calculated and represent the restrictions imposed by the waypoints. This problem is known as a dynamic optimization problem with interior point equality constraints, and it is solved using concepts of dynamic optimization theory and optimal control theory, especially, Pontryagin method [Pontryagin (1986)]. All code related to this dissertation, including the planner and simulation modules, is readily available for free access [1].

The properties observed in upper-limb experiments in areas of psychology and neuroscience are associated with bell-shaped velocity profile [Flash and Hogan (1985); Rosenbaum et al. (1995), Rosenbaum et al. (2001), Wada and Kawato (2004)]; decomposition of hand movement in sub-movements [Burdet et al. (2013)]; a low value of Normalised Jerk Score (NJS), which corresponds to a smooth movement of the hand [Flash and Hogan (1985)]; the existence of one movement unit (NMU), or even more in case of significant rotation of the joint wrist, or when avoiding obstacles [Rosenbaum et al. (2001),Gulletta et al. (2020)].

The validation of the proposed method (chapter 7) is accomplished in the context of an assembler process with quality inspection. Previously, the operator's job was to take boards from a pallet and assemble them in the box on top of the conveyor. At the same time, the operator also had to inspect them to ensure they were suitable and in perfect condition for assembly. Thus, to improve the operators' quality of life, and optimize the production time and quality, we implement a collaborative robot UR10 (chapter 4) capable of generating predictable and understandable trajectories. The robot becomes responsible for constantly manipulating the boards, which are repetitive and heavy tasks; the operator, on the other hand, deals with the cognitive task, which is the definition of mandatory points where the robot must pass during his trajectory. In this scene, these mandatory points are defined as eye-angles to observe and inspect the boards without the need to rotate them manually.

The programming approach through waypoints is intuitive, easy-to-use, and essentially easy-to-program. It is so simple as defining mandatory robot postures in the trajectory by manipulating the robot physically or by the help of a joystick, that even a non-expert operator with no previous knowledge regarding programming methods or robot kinematics are demanded for programming a new task.

Regarding the objectives mentioned, the planner allows the performance of smooth, fluent and in-

[1] https://github.com/JoaoQPereira

tuitive movements through the previously defined waypoints by the operator, as one can confirm from the obtained movements for good [2] and faulty [3] boards in the Coppellia simulation environment. Thus, human satisfaction, well-being, and productivity are enhanced by the performance of comprehensible and predictable during the task.

Despite the good results of the hand profile, we can not absolutely affirm that the resultant movements are human-like since there are no experiments in humans with waypoints. Nevertheless, as already mentioned, the resultant movements include essential properties of human's movements.

By planning different movements, we conclude that the results presented are directly affected by: the complexity of the movement; the number of waypoints; selection and disposition in the space of the waypoints; kinematics of the robot, specifically, the number of degrees of freedom. The complexity of the system increases significantly as the number of waypoints and the degrees of freedom increases.

8.1 Future Work

The accomplished results are highly satisfactory, however, there are some points to improve and to complete in future work. Starting from the validation of the trajectory planner, in this dissertation it was only possible to realise experiments in a simulation environment scene. Thus, experiments in reality with the real robot would be necessary and interesting for a future work. Still, the transfer from the simulation to reality should be straightforward and the expected results are the same.

Regarding the accuracy of the planner, the error of the joint position is sometimes considerable in the first joint, which then propagates the error to the others joints. The maximum observed was an error of 3.4 degrees in θ_1. This error, for many tasks is irrelevant; however, for a task where the precision and accuracy are mandatory, this is at least inconvenient.

Additionally, the calculation of the time when the robot passes through each waypoint is determined by using the library *fsolve*, implemented in python. This solution is practical and effective, however a complete solution in *c++* would be robust and faster.

The selection of waypoints has a high impact on the resultant robot movement. The waypoints can be defined through the joystick embedded in the UR10 UI poyscope, or by physically manipulating the robot and setting mandatory positions during the task. Thus, a good knowledge regarding the workplace and robot kinematics are useful when defining waypoints. Therefore, a user study would be necessary to evaluate the intuitiveness, efficiency, and usability of the programming method through waypoints.

[2] https://youtu.be/aX8ifqhSzqA
[3] https://youtu.be/J-V-WMKI5mE

Since there are no experiments of human movements through waypoints, we could not completely affirm the human-likeness of the resultant trajectories. Thus, it would be necessary observe the behaviour of human's movements through waypoints and a user study to evaluate the co-worker's confidence and psychology stress when working along side the robot and, also important, evaluate if the movements are smooth and predictable.

Bibliography

Abdel-Malek, K., Mi, Z., Yang, J., and Nebel, K. (2006). "Optimization-based trajectory planning of the human upper body." *Robotica*, 24(6), 683–696.

Aggogeri, F., Amici, C., and Pellegrini, N. (2020). "Dual control for jerk-driven robotics in rehabilitative planar applications." *Micromachines*, 11(2), 141.

Akgun, B., Cakmak, M., Yoo, J. W., and Thomaz, A. L. (2012). "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective." *HRI'12 - Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, 391–398.

Albrecht, S., Ramírez-Amaro, K., Ruiz-Ugalde, F., Weikersdorfer, D., Leibold, M., Ulbrich, M., and Beetz, M. (2011). "Imitating human reaching motions using physically inspired optimization principles." *IEEE-RAS International Conference on Humanoid Robots*, 602–607.

Andersen, R. S. (2018). "Kinematics of a UR5." *Aalborg University, May 2018*, 1–12.

Barre, P.-J., Bearee, R., Borne, P., and Dumetz, E. (2005). "Influence of a jerk controlled movement law on the vibratory behaviour of high-dynamics systems." *Journal of Intelligent and Robotic Systems*, 42(3), 275–293.

Bearee, R., Barre, P.-J., and Hautier, J.-P. (2005). "Vibration reduction abilities of some jerk-controlled movement laws for industrial machines." *IFAC Proceedings Volumes*, 38(1), 796–801.

Ben-Ari, M., Mondada, F., Ben-Ari, M., and Mondada, F. (2018). "Kinematics of a Robotic Manipulator." *Elements of Robotics*, 267–291.

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

Bloss, R. (2016). "Collaborative robots are rapidly providing major improvements in productivity, safety, programing ease, portability and cost while addressing many new applications." *Industrial Robot: An International Journal*.

Bobrow, J. E., Dubowsky, S., and Gibson, J. S. (1985). "Time-optimal control of robotic manipulators along specified paths." *The international journal of robotics research*, 4(3), 3–17.

Boesl, D. B. and Liepert, B. (2016). "4 robotic revolutions-proposing a holistic phase model describing future disruptions in the evolution of robotics and automation and the rise of a new generation 'r'of robotic natives." *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 1262–1267.

Bohlin, R. and Kavraki, L. E. (2000). "Path planning using lazy prm." *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 1, IEEE, 521–528.

Bortot, D., Born, M., and Bengler, K. (2013). "Directly or on detours? how should industrial robots approximate humans?." *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 89–90.

Brooks, R. A. and Lozano-Perez, T. (1985). "A subdivision algorithm in configuration space for findpath with rotation." *IEEE Transactions on Systems, Man, and Cybernetics*, (2), 224–233.

Bryson, A. E. and Ho, Y.-C. (1975). *Applied Optimal Control*, Vol. 21. Taylor Francis, New York.

Burdet, E., Franklin, D. W., and Milner, T. E. (2013). *Human Robotics: Neuromechanics and Motor Control*. The MIT Press, London.

Carfi, A., Villalobos, J., Coronado, E., Bruno, B., and Mastrogiovanni, F. (2019). "Can Human-Inspired Learning Behaviour Facilitate Human–Robot Interaction?." *International Journal of Social Robotics*, (April).

Chachuat, B. (2016). "OPTIMIZATION From Theory to Practice." (August).

Chang, J. J., Wu, T. I., Wu, W. L., and Su, F. C. (2005). "Kinematical measure for spastic reaching in children with cerebral palsy." *Clinical Biomechanics*, 20(4), 381–388.

Chang, J.-J., Yang, Y.-S., Lan-Yuen, G., Wen-Lan, W., and Fong-Chin, S. (2008). "Differences in Reaching Performance Between Normal Adults and Patients Post Stroke-A Kinematic Analysis." *Journal of Medical and Biological Engineering*, 3–8.

Clarke, R. (1994). "Asimov's laws of robotics: Implications for information technology. 2." *Computer*, 27(1), 57–66.

Constantinescu, D. and Croft, E. A. (2000). "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths." *Journal of robotic systems*, 17(5), 233–249.

Craig, . (1986). *Introduction to robotics : mechanics & control*. Addison-Wesley Pub. Co.„ Reading, Mass. Includes bibliographies and index.

Dehghani, S. and Bahrami, F. (2020). "3D human arm reaching movement planning with principal patterns in successive phases." *Journal of Computational Neuroscience*, 48(3), 265–280.

Deloitte (2005). "Made-to-order: The rise of mass personalisation.

El Zaatari, S., Marei, M., Li, W., and Usman, Z. (2019). "Cobot programming for collaborative industrial tasks: An overview." *Robotics and Autonomous Systems*, 116(June), 162–180.

Elbanhawi, M. and Simic, M. (2014). "Sampling-based robot motion planning: A review." *Ieee access*, 2, 56–77.

Emadi Andani, M. and Bahrami, F. (2012). "COMAP: A new computational interpretation of human movement planning level based on coordinated minimum angle jerk policies and six universal movement elements." *Human Movement Science*, 31(5), 1037–1055.

FarzanehKaloorazi, M. H. and Bonev, I. A. (2018). "Singularities of the typical collaborative robot arm." *Proceedings of the ASME Design Engineering Technical Conference*, 5B-2018, 1–7.

Fischer, K., Kirstein, F., Jensen, L. C., Krüger, N., Kuklínski, K., Der Wieschen, M. V., and Savarimuthu, T. R. (2016). "A comparison of types of robot control for programming by demonstration." *ACM/IEEE International Conference on Human-Robot Interaction*, 2016-April, 213–220.

Fitts, P. M. (1954). "Journal of Experimental Psychology.." *Journal of Experimental Psychology*, 47(6), 381–391.

Flash, T. and Hogan, N. (1985). "The coordination of arm movements: An experimentally confirmed mathematical model." *Journal of Neuroscience*, 5(7), 1688–1703.

Garcia, N., Rosell, J., and Suarez, R. (2019). "Motion Planning by Demonstration with Human-Likeness Evaluation for Dual-Arm Robots." *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11), 2298–2307.

Gasparetta, A., Boscario, P., Lanzutti, A., and Vidoni, R. (2015). *Motion and Operation Planning of Robot Systems, Chapter 1 - Path Planning and Trajectory Planning Algorithms: A General Overview*, Vol. 29, <http://link.springer.com/10.1007/978-3-319-14705-5>.

Gasparetto, A. and Scalera, L. (2019). "A brief history of industrial robotics in the 20th century." *Advances in Historical Studies*, 8(1), 24–35.

Gasparetto, A. and Zanotto, V. (2008). "A technique for time-jerk optimal planning of robot trajectories." *Robotics and Computer-Integrated Manufacturing*, 24(3), 415–426.

Geraerts, R. and Overmars, M. H. (2004). "A comparative study of probabilistic roadmap planners." *Algorithmic Foundations of Robotics V*, Springer, 43–57.

Giles, N. and Hatzel, M. (2013). "Innovative human-robot cooperation in BMW Group Production." *BMW Groups*.

Gladden, M. E. (2019). "Who will be the members of society 5.0? towards an anthropology of technologically posthumanized future societies." *Social Sciences*, 8(5), 148.

Gulletta, G., Costa e Silva, Eliana Erlhagen, W., Meulenbroek, R., Pires Costa, M. F., and Bicho, E. (2021). "A Human-like Upper-limb Motion Planner: generating naturalistic movements for humanoid robots." *International Journal of Advanced Robotic Systems*, (April), In Press.

Gulletta, G., Erlhagen, W., and Bicho, E. (2020). "Human-like arm motion generation: A review." *Robotics*, 9(4), 1–48.

Gupta, A. K., Arota, S. K., and Westcott, J. R. (2015). *Industrial Automation and Robotics: An Introduction*. Mercury Learning Information, Boston.

Han, L. and Amato, N. (2001). "A kinematics-based probabilistic roadmap method for closed chain systems: Li han, texas a nancy m. amato, texas a." *Algorithmic and Computational Robotics*, AK Peters/CRC Press, 243–251.

Hanc, J. (2017). "The original Euler's calculus-of-variations method: Key to Lagrangian mechanics for beginners." *European Journal of Physics, Submitted*, (September), 1–16.

Hawkins, K. P. (2013). "Analytic Inverse Kinematics for the Universal Robots UR-5/UR-10 arms." 1–5.

Hettich, R., Kaplan, A., and Tichatschke, R. (2009). *Semi-infinite programming: numerical methodsSemi-infinite Programming: Numerical Methods*. Springer US, Boston, MA, <https://doi.org/10.1007/978-0-387-74759-0$_5$88>.

Hockstein, N. G., Gourin, C., Faust, R., and Terris, D. J. (2007). "A history of robots: from science fiction to surgical robotics." *Journal of robotic surgery*, 1(2), 113–118.

Hsu, D., Kindel, R., Latombe, J.-C., and Rock, S. (2002). "Randomized kinodynamic motion planning with moving obstacles." *The International Journal of Robotics Research*, 21(3), 233–255.

Hsu, D. and Sun, Z. (2004). "Adaptively combining multiple sampling strategies for probabilistic roadmap planning." *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, Vol. 2, IEEE, 774–779.

Huang, J., Hu, P., Wu, K., and Zeng, M. (2018). "Optimal time-jerk trajectory planning for industrial robots." *Mechanism and Machine Theory*, 121, 530–544.

Intelligence, M. (2020). "Global collaborative robot market." 25.

Jazar, R. N. (2008). "Theory of Applied Robotics." *Journal of Chemical Information and Modeling*, Vol. 53, 287.

Kang, S. B. and Ikeuchi, K. (1994). "Determination of motion breakpoints in a task sequence from human hand motion." *Proceedings - IEEE International Conference on Robotics and Automation*, (pt 1), 551–556.

Karaman, S. and Frazzoli, E. (2011). "Sampling-based algorithms for optimal motion planning." *The international journal of robotics research*, 30(7), 846–894.

Karel, C. (1920). *R.U.R. - Rossum's Universal Robots*. Aventinum.

Kavraki, L. E., Svestka, P., Latombe, J.-C., and Overmars, M. H. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE transactions on Robotics and Automation*, 12(4), 566–580.

Kebria, P. M., Al-Wais, S., Abdi, H., and Nahavandi, S. (2017). "Kinematic and dynamic modelling of UR5 manipulator." *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, 4229–4234.

Khatib, O. (1986). "Real-time obstacle avoidance for manipulators and mobile robots." *Autonomous robot vehicles*, Springer, 396–404.

Kirk, D. E. (1970). "Optimal Control Theory An Introduction Englewood Cliffs New Jersey.

Koppenborg, M., Nickel, P., Naber, B., Lungfiel, A., and Huelke, M. (2017). "Effects of movement speed and predictability in human–robot collaboration." *Human Factors and Ergonomics In Manufacturing*, 27(4), 197–209.

Kucuk, S. (2017). "Optimal trajectory generation algorithm for serial and parallel manipulators." *Robotics and Computer-Integrated Manufacturing*, 48(April), 219–232.

Kuffner, J. J. and LaValle, S. M. (2000). "Rrt-connect: An efficient approach to single-query path planning." *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, Vol. 2, IEEE, 995–1001.

Kunz, T. and Stilman, M. (2011). "Turning Paths Into Trajectories Using Parabolic Blends." *Georgia Institute of Technology*.

Kurt, R. (2019). "Industry 4.0 in Terms of Industrial Relations and Its Impacts on Labour Life." *Procedia Computer Science*, 158, 590–601.

Kyriakopoulos, K. J. and Saridis, G. N. (1988). "Minimum jerk path generation." *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, IEEE, 364–369.

Lan, J., Xie, Y., Liu, G., and Cao, M. (2020). "A Multi-Objective Trajectory Planning Method for Collaborative Robot." *eletronics*.

LaValle, S. M. (1998). "Rapidly-exploring random trees: A new tool for path planning.

LaValle, S. M. (2006). "Planning algorithms." *Planning Algorithms*, 9780521862, 1–826.

LaValle, S. M. and Kuffner Jr, J. J. (2000). "Rapidly-exploring random trees: Progress and prospects.

Liberzon, D. (2011). *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, USA.

Liu, S., Wang, Y., Wang, X. V., and Wang, L. (2018). "Energy-efficient trajectory planning for an industrial robot using a multi-objective optimisation approach." *Procedia Manufacturing*, 25(July), 517–525.

Liu, X., Qiu, C., Zeng, Q., Li, A., and Xie, N. (2020). "Time-energy optimal trajectory planning for collaborative welding robot with multiple manipulators." *Procedia Manufacturing*, 43, 527–534.

Lozano-Perez, T. (1990). "Spatial planning: A configuration space approach." *Autonomous robot vehicles*, Springer, 259–271.

Lu, S., Ding, B., and Li, Y. (2020). "Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation." *Advances in Mechanical Engineering*, 12(3), 1687814020913667.

Lynch, K. M. and Park, F. C. (2017). *Modern Robotics*. Cambridge University Press.

Mansour, R. and Waldemar, K. (2004). *Human-Robot Interaction*, Vol. 53. Taylor & Francis, London.

Martins de Sá, S. F. (2018). "Planeamento de Movimentos Compreensíveis pelo Humano para o Robô Sawyer." M.S. thesis, Universidade do Minho, Universidade do Minho, <http://hdl.handle.net/1822/59244>.

Meirovitch, Y., Bennequin, D., and Flash, T. (2016). "Geometrical Invariance and Smoothness Maximization for Task-Space Movement Generation." *IEEE Transactions on Robotics*, 32(4), 837–853.

Nahavandi, S. (2019). "Industry 5.0—a human-centric solution." *Sustainability*, 11(16).

Neto, P., Pires, J. N., and Moreira, A. P. (2010). "3D CAD-based robot programming for the SME shop-floor." *20th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM*.

Nguiadem, C., Raison, M., and Achiche, S. (2020). "Motion Planning of Upper-Limb Exoskeleton Robots: A Review." *Applied Sciences*.

Niku, S. B. (2020). *Introduction to robotics: analysis, control, applications*. John Wiley & Sons.

Oosterwyck, N. V. (2018). "Real Time Human Robot Interactions and Speed Control of a Robotic Arm for Collaborative Operations Real Time Human-Robot Interactions and Speed Control of a Robotic Arm for Collaborative Operations Nick Van Oosterwyck." Ph.D. thesis, Ph.D. thesis.

Park, G. R. and Kim, C. H. (2010). "Constructing of optimal database structure by imitation learning based on evolutionary algorithm." *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, 2698–2703.

Perumaal, S. S. and Jawahar, N. (2013). "Automated trajectory planner of industrial robot for pick-and-place task." *International Journal of Advanced Robotic Systems*, 10.

Piazzi, A. and Visioli, A. (2000). "Global minimum-jerk trajectory planning of robot manipulators." *IEEE Transactions on Industrial Electronics*, 47(1), 140–149.

Pilz, G. and KG, C. (2019). "Motion blending." *GitHub repository*, <https://github.com/ros-planning/moveit/tree/master/moveit_planners/pilz_industrial_motion_planner>.

Pontryagin, L. S. (1986). *The Mathematical Theory of Optimal Processes*, Vol. 4 - The Ma.

Quigley, M., Gerkey, B., Conley†, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2008). "ROS: an open-source Robot Operating System." *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 4754–4759.

Rojas, R. A., Garcia, M. A. R., Wehrle, E., and Vidoni, R. (2019). "A variational approach to minimum-jerk trajectories for psychological safety in collaborative assembly stations." *IEEE Robotics and Automation Letters*, 4(2), 823–829.

Rosenbaum, Ruud G.J., Meulenbroek, D. A., Jansen, C., Vaughan, J., and Vogt, S. (2001). "Multijoint grasping movements: Simulated and observed effects of object location, object size, and initial aperture." *Experimental Brain Research*, 138(2), 219–234.

Rosenbaum, D., D.Loukopoulos, L., Vaughan, J., E.Engelbrecht, S., and Meulenbroek, R. G. (1995). "Planning Reaches by Evaluating Stored Postures." *American Psychological Association*.

Saito, H., Tsubone, T., and Wada, Y. (2006). "Movement time planning in human movement with via-points." *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, (4), 1208–1211.

Schaal, S. (2007). "The new robotics—towards human-centered machines." *HFSP Journal*, 1(2), 115–126.

Short, A., Pan, Z., Larkin, N., and van Duin, S. (2016). "Recent progress on sampling based dynamic motion planning algorithms." *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, IEEE, 1305–1311.

Simon, D., Isik, C., et al. (1991). "Optimal trigonometric robot joint trajectories.." *Robotica*, 9(4), 379–386.

Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. (2006). *Robot modeling and control*.

Sun, J. D., Cao, G. Z., Li, W. B., Liang, Y. X., and Huang, S. D. (2017). "Analytical inverse kinematic solution using the D-H method for a 6-DOF robot." *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017*, 714–716.

Sung, C., Kagawa, T., and Uno, Y. (2013). "Whole-body motion planning for humanoid robots by specifying via-points." *International Journal of Advanced Robotic Systems*, 10(7), 300.

Sung, C., Kagawa, T., and Uno, Y. (2015). "Synthesis of humanoid whole-body motion with smooth transition." *Advanced Robotics*, 29(9), 573–585.

Tlach, V., Kuric, I., Ságová, Z., and Zajačko, I. (2019). "Collaborative assembly task realization using selected type of a human-robot interaction." *Transportation Research Procedia*, 40, 541–547.

Todorov, E. and Jordan, M. I. (1998). "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements." *Journal of Neurophysiology*, 80(2), 696–714.

Tsarouchi, P., Makris, S., and Chryssolouris, G. (2016). "Human–robot interaction review and challenges on task planning and programming." *International Journal of Computer Integrated Manufacturing*, 29(8), 916–931.

Tsuzuki, Y. and Ogihara, N. (2018). "A recurrent neural network model for generation of humanlike reaching movements." *Advanced Robotics*, 32(15), 837–849.

Universal Robots (2015). "Universal Robots. UR10e/CB3 User manual." *Universal Robots*, I–55 – I–58.

Uno, Y., Kawato, M., and Suzuki, R. (1989). "Formation and control of optimal trajectory in human multijoint arm movement." *Biological Cybernetics*, 61(2), 89–101.

Villani, V., Pini, F., Leali, F., and Secchi, C. (2018). "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications." *Mechatronics*, 55(February), 248–266.

Vysocky, A. and Novak, P. (2016). "Human - Robot collaboration in industry." *MM Science Journal*, 2016-June(December 2017), 903–906.

Wada, Y. and Kawato, M. (1995). "A theory for cursive handwriting based on the minimization principle." *Biological Cybernetics*, 73(1), 3–13.

Wada, Y. and Kawato, M. (2004). "A via-point time optimization algorithm for complex sequential trajectory formation." *Neural Networks*, 17(3), 353–364.

Wang, C., Peng, L., Hou, Z. G., Li, J., Luo, L., Chen, S., and Wang, W. (2019). "Kinematic Redundancy Analysis during Goal-Directed Motion for Trajectory Planning of an Upper-Limb Exoskeleton Robot." *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 5251–5255.

Williams, R. L. (2013). "Simplified robotics joint-space trajectory generation with a via point using a single polynomial." *Journal of Robotics*, 2013.

Wolpert, D. M. and Ghahramani, Z. (2012). "Computational Motor Control: ERN." *SpringerReference*.

Yokoyama, H., Saito, H., Kurai, R., Nambu, I., and Wada, Y. (2018). "Investigation of isochrony phenomenon based on the computational theory of human arm trajectory planning." *Human Movement Science*, 61(January), 52–62.

Zamalloa, I., Kojcev, R., Hernández, A., Muguruza, I., Usategui, L., Bilbao, A., and Mayoral, V. (2017). "Dissecting robotics-historical overview and future perspectives." *arXiv preprint arXiv:1704.08617*.

Zanchettin, A. M., Rocco, P., Bascetta, L., Symeonidis, I., and Peldschus, S. (2011). "Kinematic motion analysis of the human arm during a manipulation task." *Joint 41st International Symposium on Robotics and 6th German Conference on Robotics 2010, ISR/ROBOTIK 2010*, 2, 1252–1257.

Zhao, J., Xie, B., and Song, C. (2014). "Generating human-like movements for robotic arms." *Mechanism and Machine Theory*, 81, 107–128.

Appendices

Appendix A

Revisiting Universal Robot's Kinematics

Revisiting Universal Robot's Kinematics

João G. Cunha[1*], João Q. Pereira[2*], Estela Bicho[2]

***Abstract*— Literature related to robot kinematics is often abundant and meticulous. However, when faced with the challenge of developing a kinematics solution for Universal Robots from scratch, we found this was not the case. We found Universal Robot's documentation to be confusing, and that literature regarding this subject was scarce. Prompted by these considerations, we present a comprehensive literature review related to Universal Robots kinematics. Secondly, we provide an analytical solution for their inverse kinematics using the modified Denavit-Hartenberg convention. Lastly, we test the solution in MATLAB, and visualise it in CoppeliaSim through a remote API. The main goal of this work is to provide an explicit and transparent guide into Universal Robots kinematics, by expanding on the literature we found and describing every part of our analysis exhaustively.**

***Index Terms*— Kinematics Solution, Collaborative Robotics, Universal Robots, Robotics Simulation**

I. INTRODUCTION

Regarding collaborative robots, Universal Robots is, to date, the largest industrial player and the front-runner in market share [1]. Most straightforward industrial applications can be resolved with the use of the included controller, and programming interface - Polyscope. However, when a closer human-robot collaboration is needed, the aforementioned programming approach comes short in terms of flexibility and adaptability. In these cases, roboticists are required to model the robot's behaviour - which cannot be programmed in Polyscope. To do this, one needs a custom controller that often requires its own kinematic solver.

Kinematics consists in studying the motion of a certain system disregarding the forces or torques that cause it [2]. To develop a kinematics solution for the Universal Robots, we conducted a literature review and found the available documentation was scarce and inadequately detailed [3].

In this sense, this paper presents an in-depth analysis of the Universal Robots kinematics. Specifically, the contributions of this paper are fourfold: (i) a literature review of papers related to this subject; (ii) forward kinematic solution based on the modified Denavit-Hartenberg convention; (iii) geometrical analysis of the inverse kinematics problem; and (iv) validation of the proposed analysis using MATLAB and CoppeliaSim (kinematic models are available in an online repository). Accompanying this paper is also a video detailing how to use our kinematic models[1].

*These authors contributed equally to this work.

[1]João G. Cunha is with Association Collaborative Laboratory in Digital Transformation - DTx, Campus de Azurém, University of Minho, 4800-058 Guimarães, Portugal {firstname.lastname}@dtx-colab.pt

[2]João Q. Pereira and Estela Bicho with are with Centre Algoritmi, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal {firstname.lastname}@dei.uminho.pt

[1] Universal Robots Kinematics: Visualisation in CoppeliaSim with MATLAB API.

II. RELATED WORK

Although the use of the UR's collaborative robots is widespread in both industrial and academic context, publications regarding their kinematic modelling is relatively limited (six publications, from 2013 to 2019).

The common approach to solve the UR's kinematics across literature is to use the conventional D-H method, and obtain an analytical solution for the inverse kinematics based on geometrical approaches [4]–[9]. In the author's opinion, the most detailed publication is [8], as the authors provide drawings which ease reader comprehension. However, most publications are inadequately detailed and are of difficult understanding.

Additionally, the authors in [7] present another approach for kinematic modelling based on the Product of Exponentials (POE) analysis. They establish a comparison between it and the conventional D-H method, concluding that although the modelling process is simpler with the POE method, not all configurations are solvable due to the inviability to decompose some into solvable sub-problems.

Besides the authors in [5], which provide the kinematic and dynamic model of the UR5 robot implemented in MATLAB, and a robot model for SimMechanics, all other papers lack validation via code of their analyses.

To the best of the authors knowledge, the lack of detail and clear explanation of the Universal Robots kinematics is a common problem. Regarding the wide usage of the UR robots, we argue this paper could be helpful for the community, providing an in-depth analysis of their kinematic model. Our approach can be abstracted to any of the Universal Robots product range, and has been validated by means of simulation. In addition, the code related to this work is readily available for free access[2].

III. FORWARD KINEMATICS

Forward kinematics consists of finding the pose of the robot's tip $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e]^T$, from its joint positions $q = [\theta_1, \theta_2, \theta_2, \theta_3, \theta_4, \theta_4, \theta_5, \theta_6]^T$. For every vector of joint values, q, the tip's pose always exists and is unique [10].

A. Assign the Reference Frames

To derive the Denavit-Hartenberg parameters, one must first position the manipulator in its home pose, and secondly attribute a reference frame $\{x_i, y_i, z_i\}$ for every joint. In our case, the manipulator's home pose was defined as all joint values equal to zero, which determine an upright pose (for the CoppeliaSim robot model). Reference frames {0-6} are represented in Fig. 1 (where the x, y and z components are

[2] Jgocunha/universal-robots-kinematics.

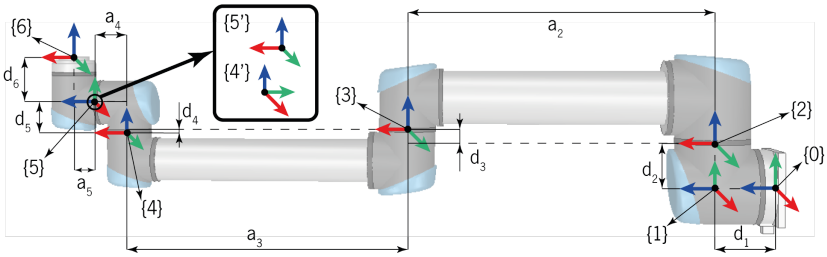


Fig. 1: Reference frames along the UR10 e-series structure.

represented with red, green, and blue arrows, respectively). To attribute these reference frames we followed three basic rules: i) the right hand rule; ii) the location of the frame is at the centre of the corresponding joint; and iii) the z-axis must indicate the rotational axis of the joint [11].

Reference frame $\{0\}$ is the origin of the robot, and is located on the centre of its base. From henceforth, reference frames $\{1, 2, 3, 4, 5, 6\}$ all follow the above stated rules. Due to the order of multiplication of the transformations for the modified Denavit-Hartenberg convention, it was necessary to add two auxiliary reference frames: $\{4'\}$, and $\{5'\}$ located at the centre of joint 5.

B. Denavit-Hartenberg Parameters

After attributing the reference frames, it is possible to compute the transformations that occur between them, using the D-H parameters mentioned previously, Table I.

C. Computing the Individual Transformation Matrices

Using the Denavit-Hartenberg parameters and the modified Denavit-Hartenberg matrix, it is possible to compute the following transformation matrices: ${}^0T_1$, ${}^1T_2$, ${}^2T_3$, ${}^3T_4$, ${}^4T_5$, and ${}^5T_6$. Like the homogeneous matrix, the modified Denavit-Hartenberg matrix is a transformation matrix from a system of coordinates to another. The modified Denavit-Hartenberg homogeneous transformation ${}^{i-1}T_i$, from frame $i-1$ to i is defined in Eq. 2 and 3.

D. Computing the Complete Transformation Matrix

Having defined the homogeneous transformation matrices along the robotic arm, the transformation from the robot base to robot-tip is given by:

$$ {}^0T_6 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_{4'}\,{}^{4'}T_5\,{}^5T_{5'}\,{}^{5'}T_6 \tag{1} $$

TABLE I: Denavit-Hartenberg transformations between reference frames

${}^{i-1}\boldsymbol{T_i}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{a_{i-1}}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
$\{0\} \rightarrow \{1\}$	0	0	d_1	θ_1
$\{1\} \rightarrow \{2\}$	$-\pi/2$	0	d_2	$\theta_2 - \pi/2$
$\{2\} \rightarrow \{3\}$	0	a_2	d_3	θ_3
$\{3\} \rightarrow \{4\}$	0	a_3	d_4	θ_4
$\{4\} \rightarrow \{4'\}$	0	a_4	d_5	$\pi/2$
$\{4'\} \rightarrow \{5\}$	$\pi/2$	0	0	θ_5
$\{5\} \rightarrow \{5'\}$	$-\pi/2$	0	0	$-\pi/2$
$\{5'\} \rightarrow \{6\}$	0	a_5	d_6	θ_6

IV. INVERSE KINEMATICS

Obtaining vector $q = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$ that makes the robot's tip pose equal to $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e]^T$ is called inverse kinematics, and is not as linear to solve as the forward kinematics problem [12]. For the Universal Robots there are eight possible inverse kinematic solutions for a certain tip pose, Fig. 2.

Our approach to solving this particular inverse kinematics problem is inspired by the aforementioned literature, and uses a geometrical and analytical approach. For the computation of each joint angle, a detailed visual and mathematical demonstration is provided, in order to expedite comprehension.

$$ {}^{i-1}T_i = Rot_{x(\alpha_{i-1})} Trans_{x(a_{i-1})} Trans_{z(d_i)} Rot_{z(\theta_i)} \tag{2} $$

$$ {}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & \alpha_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

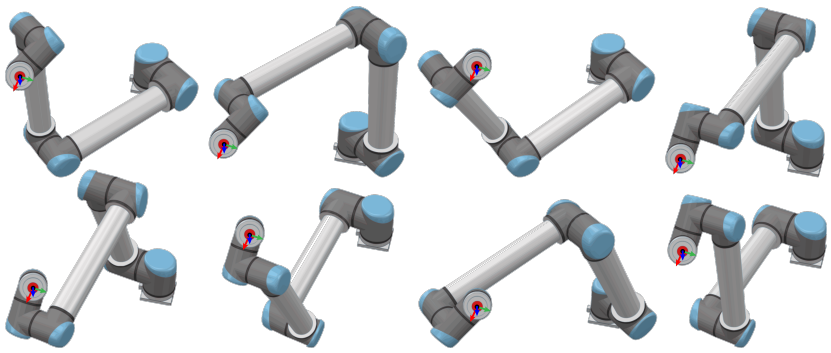


Fig. 2: The eight inverse kinematic solutions $(2\theta_1 \cdot 2\theta_5 \cdot \theta_6 \cdot 2\theta_3 \cdot \theta_2 \cdot \theta_4)$ for a random generated pose: $[x_e, y_e, z_e, \gamma_e, \beta_e, \alpha_e] = [0.730\ m, -0.292\ m, 0.643\ m, -9.357°, -46.676°, 29.751°]$.

A. *Computing* θ_1

To calculate θ_1, we first determine the value of ${}^0P_5$ (the position of frame $\{5\}$ in reference to frame $\{0\}$). Considering the fourth column of ${}^0T_6$, ${}^0P_6 = [p_x, p_y, p_z, 1]^T$ (the position of the robot's tip), we can infer that ${}^0P_5$ consists in a translation of $-d_6$ in the z-axis from the 6^{th} frame. Algebraically, this translation can be written as:

$$ {}^0P_5 = {}^0T_6 \begin{bmatrix} 0 & 0 & -d_6 & 1 \end{bmatrix}^T \tag{4} $$

To visualise the appearance of θ_1, we can consider an overhead view of the robot, where $\theta_2 = \pi/2$ and $\theta_1 \neq 0$, Fig. 3. This allows us to see θ_1 and represent the robot in the x-y plane of frame $\{0\}$ (useful for this analysis).

Empirically, observing Fig. 3, the value of θ_1 is equal to the subtraction of $\theta_1^{'}$ by π, Eq. 5. Whereas the value of $\theta_1^{'}$ is related to the sum of angle ψ with ϕ and $\pi/2$, Eq. 6.

$$ \theta_1^{'} = \psi + \phi + \frac{\pi}{2} \tag{5} $$

$$ \theta_1 = \theta_1^{'} - \pi \tag{6} $$

Angle ψ is created using ${}^0P_5$ and its x-y components which form triangle 1. ψ can now be directly calculated using the tangent function, since both the values of ${}^0P_{5x}$ and ${}^0P_{5y}$ are known from the previously calculated value of ${}^0P_5$. Thus, ψ is equal to:

$$ \psi = \text{atan2}({}^0P_{5y}, {}^0P_{5x}) \tag{7} $$

Angle ϕ can be obtained using the values of ${}^0P_5$ and d_2, d_3, d_4, d_5, which form triangle 2. θ_1 has two possible solutions which are dependant on the configuration of the shoulder joint (joint 2) - left or right.

$$ \phi = \pm \arccos\left(\frac{d_2 + d_3 - d_4 + d_5}{\sqrt{{}^0P_{5x}^2 + {}^0P_{5y}^2}} \right) \tag{8} $$

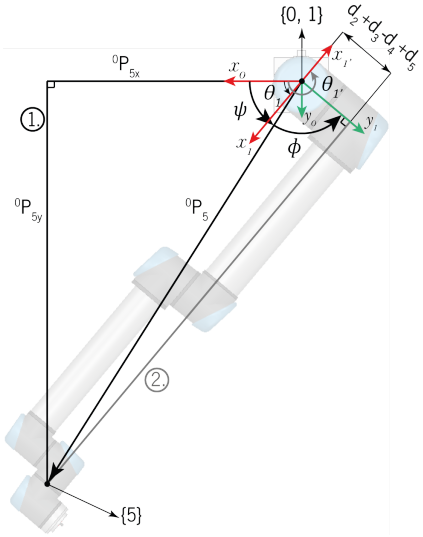


Fig. 3: Visualising θ_1.

B. *Computing* θ_5

Angle θ_5 is defined as the angle between the x-axis of frames $4'$ and 5. However, when the robot is in its pre-defined home pose, these two axis overlap, *i.e.* have the same orientation - making it difficult to extract any correlations. To do this, first we set θ_4 to $\pi/2$ and then give a value ($\neq 0$) to θ_5, resulting in Fig. 4.

Since θ_1 is already known, it is possible to calculate the transformation matrix from frame $\{1\}$ to frame $\{6\}$, ${}^1T_6$. Using its fourth column we obtain ${}^1P_6$, and consequently its y component ${}^1P_{6y}$. As it is possible to see in Fig. 4, ${}^1P_{6y}$ only depends on θ_5. Analysing this figure, one can conclude that ${}^1P_{6y}$ is determined by the sum of the robot's links along the y_1 referential (until frame $\{5\}$) and the length created by

θ_5 in the same direction (d_i), resulting in Eq. 9.

$$\begin{aligned} {}^1P_{6y} &= d_i + d_2 + d_3 - d_4 + d_5 \\ \text{where, } d_i &= d_6 \cos(\theta_5) \end{aligned} \tag{9}$$

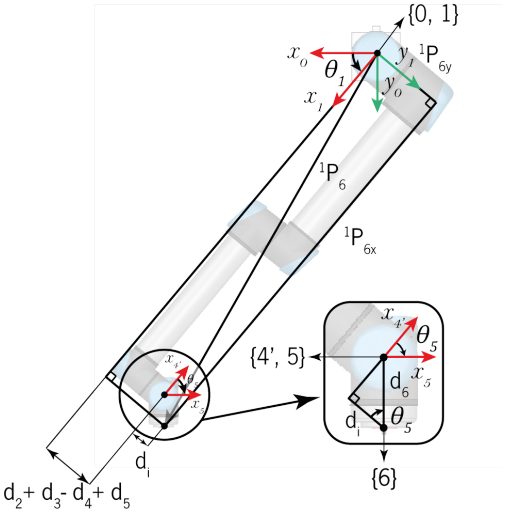


Fig. 4: Visualising θ_5.

Therefore, by solving Eq. 9, the expression of θ_5 is given by Eq. 10. θ_5 also has two possible values which are determined by the wrist being up or down.

$$\theta_5 = \pm \arccos\left(\frac{{}^1P_{6y} - (d_2 + d_3 + d_4 + d_5)}{d_6}\right) \tag{10}$$

C. Computing θ_6

The calculation of θ_6 is based on the analysis of y_1 seen from frame {6} - ${}^6y_1$. By inspecting the orientation of the reference frames along the robot's structure present in Fig. 1 and abstracting the translations between frame {6} and {1}, one can conclude that the orientation of y_1 only depends on the values of θ_5 and θ_6 (since y_1 will always be parallel to the z-axes of frames $\{2, 3, 4, 4', 5'\}$). This means we can discover the value of θ_6 as a function of θ_5.

To examine the relation between ${}^6y_1$ and the other reference frames, we can represent ${}^6y_1$ using a spherical coordinate system - where the azimuthal angle is θ_6 and the polar angle is θ_5. From Fig. 5a, which portrays a unit sphere showing the radial distance (r), polar angle (ψ) and azimuthal angle (θ) of a point P, we can abstract Fig. 5b.

$$\text{Cartesian} \leftrightarrow \text{Spherical}$$
$$\begin{cases} x = p\cos(\theta)\sin(\psi) \\ y = p\sin(\theta)\sin(\psi) \\ z = p\cos(\theta) \end{cases} \tag{11}$$

Knowing that Eq. 11 translates the conversion from spherical to Cartesian coordinates, it is possible to obtain the $x,\ y,\ z$ components of ${}^6y_1$:

$${}^6y_1 = \begin{bmatrix} \sin(\theta_5)\cos(\theta_6) \\ \sin(\theta_5)\sin(\theta_6) \\ \cos(\theta_5) \end{bmatrix} \tag{12}$$

Since the value of θ_1 is already known (and consequently the transformation matrix ${}^1T_6$) one can explicitly deduce the value of vector ${}^6y_1$. Hence, the system present in Eq. 13 can be solved for θ_6:

$$\begin{cases} \sin(\theta_5)\cos(\theta_6) = {}^6y_{1_x} \\ \sin(\theta_5)\sin(\theta_6) = {}^6y_{1_y} \end{cases} \Rightarrow \begin{cases} \cos(\theta_6) = \left({}^6y_{1_x}\right) / \sin(\theta_5) \\ \sin(\theta_6) = \left({}^6y_{1_y}\right) / \sin(\theta_5) \end{cases} \tag{13}$$

$$\theta_6 = \operatorname{atan2}\left(\frac{-{}^6y_{1_y}}{\sin(\theta_5)}, \frac{{}^6y_{1_x}}{\sin(\theta_5)}\right) \tag{14}$$

However, Eq. 14 is still not entirely correct, as it does not consider all the transformations between frame {5} and {6}. Besides a transformation of θ_6 along the z-axis we considered an auxiliary frame {5'} that assumes a transformation of $-\pi/2$ along the z-axis first (refer to Table I). So, this means that there needs to be an adjustment to Eq. 14 that considers this transformation:

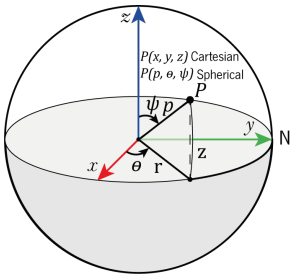
$$\theta_6 = -\pi/2 + \operatorname{atan2}\left(\frac{-{}^6y_{1_y}}{\sin(\theta_5)}, \frac{{}^6y_{1_x}}{\sin(\theta_5)}\right) \tag{15}$$

It is important to note that Eq. 15 only has a solution if $\sin(\theta_5) \neq 0$ (*i.e.* when $\theta_5 \neq 0°$, 180°or 360°). Otherwise, joints 2, 3, 4 and 6 are parallel and the solution for θ_6 is undetermined. When the axes of joints 4 and 6 become parallel a singularity occurs (more specifically a wrist singularity [3]) which cause these joints to spin 180° instantaneously.

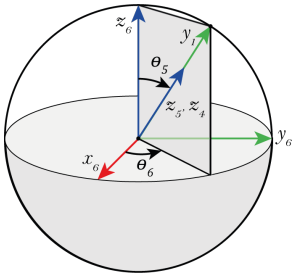
D. Computing θ_3

Concerning the calculation of θ_3 and θ_2 we considered a two link planar manipulator from frame {2} to frame {4} (links a_2 and a_3), refer to Fig. 6. This analysis is possible, since these three frames are parallel, which means the angle they form can be seen along the same two axes. Since the values of θ_1, θ_5 and θ_6 are already known, one can obtain the value of matrix ${}^1T_4$ through Eq. 16.

[3] Universal Robots Singularities: Visualisation in Polyscope.



(a) Representation of point P using spherical and Cartesian coordinates.



(b) Spherical representation of ${}^{6}y_1$, where the azimuthal angle is θ_6 and the polar angle is θ_5.

Fig. 5: Coordinate analysis of ${}^{6}y_1$.

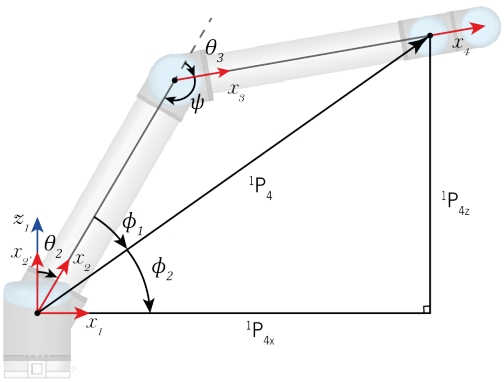


Fig. 6: Visualising θ_3 and θ_2.

$$\begin{cases} {}^{1}T_6 = inv({}^{0}T_1)\,{}^{0}T_6 \\ {}^{4}T_5 = {}^{4}T_{4'}\,{}^{4'}T_5 \\ {}^{5}T_6 = {}^{5}T_{5'}\,{}^{5'}T_6 \end{cases} \Rightarrow {}^{1}T_4 = {}^{1}T_6 inv({}^{4}T_5\,{}^{5}T_6) \quad (16)$$

Analysing Fig. 6, θ_3 is equal to the subtraction of ψ to π, Eq. 17. In its turn, ψ can be derived using the law of cosines and is given by Eq. 18. The two solutions for θ_3 correspond to the elbow being up or down.

$$\theta_3 = \pi - \psi \quad (17)$$

$$\cos(\psi) = \frac{{}^{1}P_4^2 - {a_3}^2 - {a_2}^2}{-2a_2a_3}$$
$$\psi = \pm \arccos\left(\frac{{}^{1}P_4^2 - {a_3}^2 - {a_2}^2}{-2a_2a_3}\right) \quad (18)$$

E. Computing θ_2

Angle θ_2 is defined as the angle between x_2 and $x_{2'}$, Fig. 6. Thus, the value of θ_2 is established by the subtraction of $\pi/2$ with the sum of the auxiliary angles ϕ_1 and ϕ_2, Eq. 21. These two angles can be obtained with the trigonometry concepts of arc tangent and the law of sines, respectively, as demonstrated in Eq. 19 and 20.

$$\phi_2 = \text{atan2}\left({}^{1}P_{4z}, {}^{1}P_{4x}\right) \quad (19)$$

$$\frac{a_3}{\sin(\phi_1)} = \frac{{}^{1}P_4}{\sin(\psi)} \Rightarrow \phi_1 = \arcsin\left(\frac{a_3\sin(\psi)}{{}^{1}P_4}\right) \quad (20)$$

$$\theta_2 = \frac{\pi}{2} - (\phi_1 + \phi_2) \quad (21)$$

F. Computing θ_4

By this stage all the joint angles except θ_4 are known. Being defined as the angle between x_3 and x_4, θ_4 can be obtained using the individual transformation matrix ${}^{3}T_4$ (more specifically using the x and y components of the first column, ${}^{3}X_4$). Refer to Eq. 22, and Fig. 7.

$$\theta_4 = \text{atan2}({}^{3}X_{4y}, {}^{3}X_{4x}) \quad (22)$$

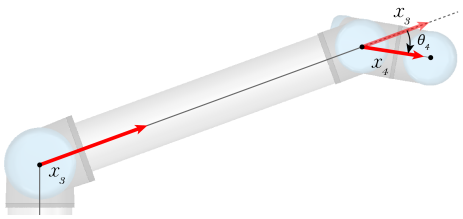


Fig. 7: Visualising θ_4.

V. VALIDATION

Both the forward and inverse kinematic algorithms were implemented in MATLAB. Our program is setup so as to validate the correct computation of both algorithms simultaneously, Fig. 8. First, the user selects which UR robot is to be controlled, and its target joint values - in a GUI. Secondly, the forward kinematics function computes the robot's tip pose for those joint values. Lastly, the inverse kinematics function is called, and 8 solutions are computed from the target tip pose. The 8 solutions are sent to CoppeliaSim[4], and the robot model is actuated accordingly.

The solutions were tested for a set of 10000 random target joint states. The maximum and average computation times[5]

[4]This is done via a remote API from the CoppeliaSim framework
[5]MATLAB script run in a Ryzen 5 3600 CPU at 4.28GHz.

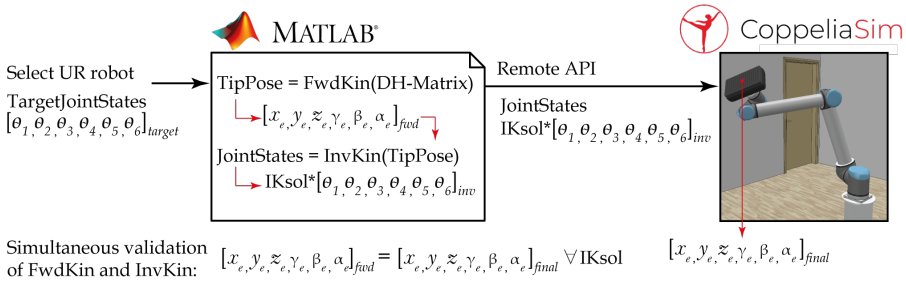


Fig. 8: Validation code structure and reasoning.

of the forward and inverse kinematic functions are provided in Table II.

TABLE II: Average and maximum computation times of the kinematics functions in seconds (s).

	Average	Maximum
Forward Kinematics	1.832571E-05	1.832571E-05
Inverse Kinematics	1.612797E-04	1.612797E-04

VI. CONCLUSION

In this paper, we present our kinematic modelling and analysis of the Universal Robots. First, a comprehensive literature review of other related papers is provided. Secondly, we present our forward kinematics solution based on the modified Denavit-Hartenberg convention and our inverse kinematics solution based on a geometrical analysis. Lastly, we prove the validity of our solutions by testing them in a simulated scene (CoppeliaSim + MATLAB). The code related to this work is available in an online repository, along with a video explaining how to use it.

REFERENCES

[1] Cobot Intelligence Inc., "Competitive Insights: Taking A Look At The Manufactures," p. 1, 2018. [Online]. Available: https://cobotintel.com/guide-to-collaborative-robots-market

[2] A. Renfrew, "Book Review: Introduction to Robotics: Mechanics and Control," *International Journal of Electrical Engineering & Education*, vol. 41, no. 4, pp. 388–388, 2004.

[3] Universal Robots, "Parameters for calculations of kinematics and dynamics," pp. 1–5, 2020. [Online]. Available: https://www.universal-robots.com/articles/ur-articles/parameters-for-calculations-of-kinematics-and-dynamics/

[4] K. P. Hawkins, "Analytic Inverse Kinematics for the Universal Robots," pp. 1–5, 2013. [Online]. Available: http://hdl.handle.net/1853/50782

[5] P. M. Kebria, S. Al-Wais, H. Abdi, and S. Nahavandi, "Kinematic and dynamic modelling of UR5 manipulator," *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pp. 4229–4234, 2017.

[6] J. D. Sun, G. Z. Cao, W. B. Li, Y. X. Liang, and S. D. Huang, "Analytical inverse kinematic solution using the D-H method for a 6-DOF robot," *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence, URAI 2017*, pp. 714–716, 2017.

[7] Q. Liu, D. Yang, W. Hao, and Y. Wei, "Research on kinematic modeling and analysis methods of UR robot," *Proceedings of 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference, ITOEC 2018*, vol. 1, no. Itoec, pp. 159–164, 2018.

[8] R. S. Andersen, "Kinematics of a UR5," *Aalborg University, May 2018*, pp. 1–12, 2018. [Online]. Available: http://rasmusan.blog.aau.dk/files/ur5{_}kinematics.pdf

[9] O. Abdelaziz, M. Luo, G. Jiang, and S. Chen, "Multiple configurations for puncturing robot positioning," *arXiv*, vol. 1, no. 4, pp. 1–19, 2019.

[10] S. Bruno, S. Lorenzo, V. Luigi, and O. Giuseppe, *Robotics: Modelling, Planning and Control*, 2019, vol. 53, no. 9.

[11] M. Ben-Ari, F. Mondada, M. Ben-Ari, and F. Mondada, "Kinematics of a Robotic Manipulator," *Elements of Robotics*, pp. 267–291, 2018.

[12] R. N. Jazar, *Theory of Applied Robotics*, 2008, vol. 53, no. 9.