NAZARBAYEV UNIVERSITY

Model Predictive Control and Imitation Learning Algorithms for Robot Motion Planning in Physical Human-Robot Interaction

by

Aigerim Nurbayeva

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics Engineering

Date of Completion
October, 2024

Model Predictive Control and Imitation Learning Algorithms for Robot Motion Planning in Physical Human-Robot Interaction

by
Aigerim Nurbayeva

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Robotics Engineering

Department of Robotics and Mechatronics
School of Engineering and Digital Sciences
Nazarbayev University

October, 2024

Supervised by
Prof. Matteo Rubagotti (lead supervisor)
Prof. Almas Shintemirov (co-supervisor)
Prof. Gian Paolo Incremona (external co-supervisor)

Declaration

I, Aigerim Nurbayeva, declare that the research contained in this thesis, unless otherwise formally indicated within the text, is the author's original work. The thesis has not been previously submitted to this or any other university for a degree and does not incorporate any material already submitted for a degree.

Signature:

Date:

BLANK

Abstract

This PhD thesis focuses on the design and testing of safe robot motion planning algorithms for human-robot workspace sharing. These algorithms are based on the use of nonlinear model predictive control (NMPC), a model-based method for motion planning relying on numerical optimization. The contribution of the thesis can be split into two main areas. The first area consists of the approximation of NMPC laws using deep neural networks (DNNs), often referred to as "imitation learning". This is motivated by the fact that the execution of NMPC laws might require a considerable amount of time, which restricts the performance of the closed-loop system. Calculating the output of a DNN for a given input is instead a much faster process. Therefore, replacing the optimization solver of NMPC with a DNN can reduce computation times, thus improving performance. It is crucial, though, to suitably train the DNN to imitate the NMPC law in order to improve performance and at the same time guarantee safety. The final result obtained in this area consists of using the so-called dataset-aggregation approach for DNN training, together with properly designed safety filters, which ensure that the safety constraints imposed in the NMPC problem also hold for the robot motion generated by the DNN. The second area consists of the extension of a previously defined NMPC law in terms of stabilizing terminal constraints. The most common approach for guaranteeing closed-loop stability in an NMPC problem is the imposition of terminal constraints, i.e., the prediction of the system motion is required to satisfy certain conditions at the end of the prediction horizon. Specifically, in a previous approach, the "point terminal constraint" was used, in which the prediction of the robot motion had to exactly reach the desired goal configuration at the end of the prediction horizon. In this thesis, this condition is relaxed by imposing that a given set, rather than a given point, is reached in the state space for the predicted robot motion. The imposition of this new condition allows for an enlargement of the domain of attraction, i.e., the NMPC law can find a solution for reaching the goal configuration from a wider set of initial configurations. All the proposed motion planning strategies were tested experimentally on a UR5 collaborative manipulator.

BLANK

Acknowledgments

The thesis document presents my research work completed during the PhD program in Robotics Engineering at Nazarbayev University. I would like to gratefully acknowledge the following people who helped and supported me throughout the time I worked on this thesis project.

First, I would like to thank my lead supervisor, Prof. Matteo Rubagotti, for providing invaluable guidance and support throughout the thesis research. His expertise and insightful critiques have been invaluable not only to my research but also to my growth as a scholar. I am thankful for his patience and the considerable time he has invested in me. Thank you, professor, for being an exceptional mentor and for all the opportunities you have provided me.

I would also like to thank my co-supervisor, Prof. Almas Shintemirov, for giving me the opportunity to do research at Astana Laboratory for Robotic and Intelligent Systems (ALARIS) and for his valuable advice and guidance.

I am also grateful for the support of my external co-supervisor, Prof. Gian Paolo Incremona. His advice from our past collaborative project has been very helpful in determining the course and implementation of this work.

I am thankful for the support of all my lab mates at Robot Control and Learning (RCL) and ALARIS Labs. We had endless thoughtful conversations, knowledge sharing, a fun time, and moral support. Special thanks to Artemiy Oleinikov for his consistent support and invaluable assistance with all questions related to the NMPC controller.

Finally, I would like to especially thank my parents, sisters, and dear friend Anton Kim for all their advice and for always being my biggest supporter. Thank you!

BLANK

Copyright Statement

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Physical Human-Robot Interaction

For decades, traditional industrial robots have been used to accomplish large-scale, high-speed tasks where robots must be isolated from humans. While non-collaborative industrial robots are designed to replace humans to perform repetitive and/or dangerous tasks, collaborative robots (cobots) are designed to safely carry out functions in collaboration with human operators.

The field of industrial robotics is undergoing a paradigm shift as collaborative robots replace traditional heavy manipulators [6, 7]. Over the past 20 years, there have been changes in both academic and industrial research that have allowed humans and robots to actively collaborate and share their workspace on an increasing number of tasks [8, 9]. Physical human-robot interaction (pHRI) is an essential area of research in manufacturing that can potentially revolutionize the industry. Indeed, some works already refer to the "Industry 5.0" framework, focusing on further developing cooperation between humans and robots by implementing advanced technologies to expand human capabilities [10]. By combining the strengths of humans and robots, manufacturers can increase the efficiency, flexibility, safety, and profitability of their activities [11].

1.1.1 Safety Standards

The most crucial aspect of pHRI is to ensure safety, emphasizing the prevention of accidents that could harm humans. The ISO/TS 15066:2016 technical specification sets out principles and requirements for the design of human-robot collaboration (HRC) applications, providing a comprehensive framework for safety. This document sets out requirements and recommendations for the design, implementation, and operation of such robots in four main approaches:

- **Safety-rated monitored stop** ensures the robot halts its movement whenever the operator is within the collaboration workspace. This function is especially beneficial when loading and unloading end-effector and ergonomic devices, where the operator's presence is vital for safe operation. By preventing the robot from moving in the operator's presence, this function significantly reduces the risk of accidents, enhancing the overall safety.

- **Hand guiding** allows the robot to move only with the operator's direct participation, providing him/her with continuous control over the robot movements within the framework of a shared workspace. This applies to lifting assistance, ergonomic adaptation, and load positioning.

- **Speed and separation monitoring** ensures a safe distance between the human operator and the robot during joint tasks, allowing the robot to move only if the distance exceeds a minimum threshold. This function is particularly crucial when working on individual tasks in a shared area, as it eliminates the need for speed control and separation, thereby enhancing safety.

- **Power and force limiting** aims to minimize the mechanical load on human body parts due to potential contact with moving robot parts, final actuators, or workpieces. It is applicable in environments with the possibility of short-term and quasi-static physical contact, such as mixed media.

In addition, the standard contains recommendations for designing workspaces for collaboration in which humans and robots interact. This includes considerations regarding the workspace layout, safety zones, and barriers. The ISO/TS 15066:2016 technical specification is not just a document, but a crucial tool for promoting safe practices in collaborative robotics. By adhering to the recommendations outlined in this specification, we can ensure that collaborative robots can work harmoniously with humans, minimizing unnecessary risks to their safety.

1.1.2 Speed and Separation Monitoring

This thesis studies the use of speed and separation monitoring (henceforth, SSM [12–15]), which is one of the above-mentioned four approaches in the ISO/TS 15066:2016 technical specification. Rather than allowing the robot to move at full speed or not move at all based on the distance with the human, in this thesis the robot speed is continually modulated, dynamically adjusting the maximum allowable speed of the robot depending on the proximity of the human operator. As this distance decreases, the robot automatically slows down or stops as needed. Traditionally, SSM is applied to a fixed trajectory of the robot movement [12, 16]. However, in specific scenarios, the real-time redefinition of the robot trajectory is used to achieve a balance between performance and safety goals [4, 17–21].

This line of research is dictated by the desire to avoid frequent stops of robots, thereby increasing productivity. For example, in [17], a kinematic control strategy was introduced that allows a robot to adhere to the end effector path while applying SSM constraints (with a slightly simplified formulation compared to [12]), for both redundant and non-redundant manipulators. Similarly, in [18], the same SSM formulation was

applied to point-to-point motions using a method developed to identify waypoints along a given motion path to mitigate collisions and minimize travel time. Similarly, in [20], the SSM formulation from [12] was used to generate a set of arbitrary trajectories for this task to determine the optimal solution that could prevent the robot from stopping due to collisions with human operators.

1.2 Model Predictive Control for Workspace Sharing

Model predictive control (MPC) is a control approach that forecasts future behavior and optimizes control actions over a finite time horizon using a dynamic system model [22] in the presence of constraints on system states and control actions. The MPC algorithm solves a numerical optimal control problem over a finite time horizon and applies the first calculated control action to the system. This technique is known as *receding horizon*, as the time horizon is shifted forward as new state (or output) values are acquired. This allows the MPC law to adjust to continuously redefine the control actions, in a closed-loop fashion. MPC was first primarily employed in the petrochemical sector, with some applications in the chemical, pulp and paper, food, and mining sectors [23]. More recently, its use has expanded to systems with fast dynamics, like power converters and automobile powertrains [24].

MPC is a good candidate to solve general control problems in robotics (see, e.g., [25–28]), mainly because it can naturally allow for the imposition of inequality constraints on both inputs and states. In particular, it can be used to plan the motion of a robot manipulator in real time when a human operator shares its workspace. Planning the motion of a robot manipulator in real time in workspace-sharing scenarios necessitates meeting several requirements, including ensuring the operator's safety, avoiding fixed obstacles, and preserving the joint position and speed of all joints within predetermined bounds. In particular, [4, 19, 21, 29, 30] employed MPC for real-time motion planning of robot manipulators in the presence of people while ensuring safety via SSM.

The first work published on MPC with SSM constraints was [19], in which the tracking of a desired trajectory was performed via MPC. The space occupied by the human was delimited using separating planes; the SSM constraints were enforced by requiring that the robot speed be equal to zero at the end of the prediction horizon. The method defined in [19] was tested on a 7-DoF manipulator via Robot Operating System (ROS), and the space occupancy of the human was simplified by using a single cylindrical obstacle.

Since SSM constraints only slow down or stop the robot based on its distance from the operator, [4] introduced a nonlinear MPC strategy, based on the SSM constraints defined in [12], that redefines the robot trajectory to help it avoid obstacles. To

calculate the distance between the robot and the constraints, both the human and the robot were covered using spheres. The SSM constraints were calculated based on the distances between these spheres, and the robot trajectory to the goal point was planned using NMPC. The method of [4] was validated on a Kinova Gen3 manipulator, using prerecorded human motions to account for the presence of an operator. The approach of [4] was extended in [29] to account for how safety was perceived by the human, and in [30] to exploit multiple predictions of the human motion, according to a scenario-based NMPC approach.

An alternative method for SSM-based MPC method is that of [21]. In this work, a human-aware Cartesian MPC motion planning algorithm was used in conjunction with a safe motion unit, based on the distance between robot end effector and human wrist. The primary function of the safe motion unit is to prevent physical harm to humans by adjusting the robot motion in real-time. It acts as a safeguard that monitors the robot movement and ensures that it does not exceed safe operational parameters. The method proposed in [21] was tested on a Franka Emika Panda robot in the presence of a human operator. One of the weaknesses of all NMPC based algorithms is that the periodic recalculation of updated robot trajectories requires carrying out computations that necessarily cause delays in the implementation of the trajectory itself; this leads to performance degradation proportional to the computation time. This problem of MPC implementations in systems with fast dynamics has been studied beyond the field of robotics, and a possible solution is approximating the MPC via neural networks.

1.3 Approximating MPC via neural networks

Even though MPC can be a powerful and versatile technique for producing a safe robot motion for pHRI, the regular recalculation of updated robot trajectories demands calculations at every sampling instant. Instead of the ideal scenario where an MPC law may be assessed immediately, this causes temporal delays that approach and occasionally surpass the sampling period, degrading the closed-loop system performance [4, 19, 21]. This might be problematic for industrial applications, as SSM already places a conservative restriction on robot speed. Additionally, a further decline in productivity brought on by the motion planning algorithm calculation time could make MPC (and in particular NMPC) less applicable for manufacturing.

A possibility for improving the performance of MPC laws is using machine learning tools. MPC and machine learning have been applied in many works, particularly in the last ten years. Machine learning may be used in various parts of the MPC law in learning-based MPC strategies [31]. Learning the system dynamics and automatically modifying the system model based on data is one approach [32–34]. To achieve a desirable behavior of the closed-loop system in the particular task, a second idea

concentrates on learning other aspects of the MPC law besides the system dynamics, such as the cost function or the constraints [35–37]. Using MPC approaches to provide safety for learning-based controllers is the third idea [38, 39]. The fourth concept involves obtaining a high-quality initialization for an NMPC solver by utilizing neural network-provided approximation NMPC solutions [40]. In this approach, the artificial neural network is trained to predict the optimal control policy based on offline NMPC simulations. This prediction is used as the initial guess for solving the NMPC problem, thereby accelerating the convergence to an optimal solution. It is crucial to find a good initial guess for the NMPC law because it involves solving a nonconvex optimization problem, where the quality of the initial guess can significantly impact the speed and success of finding an optimal solution. The final and fifth option is to use neural networks to directly learn an approximation of an existing MPC law, which is usually not learning-based. This method differs fundamentally from the other four research paths stated in that it does not need the online solution of an optimal control problem, and the resultant controller is no longer an MPC controller. To the best of our knowledge, this was most likely the first option to be presented (in [41]), which describes a method for controlling nonlinear systems using a MPC strategy. It incorporates a neural network to approximate the optimal control policy, making the approach more computationally feasible for complex nonlinear systems.

Following this last approach, to approximate an NMPC law, [42] proposed using a long short-term memory (LSTM) deep neural network (DNN). The authors of [43] suggested creating data pairs using multi-stage NMPC, which were then utilized to train DNNs to learn a robust NMPC strategy. DNNs were trained to mimic hybrid MPC for domestic heating systems in [44]. Simulated MPC law imitation using DNNs was implemented for resonant power converters in [45]. A learning-based multistage MPC framework was presented in [46] for systems with time-varying plant-model mismatches, and online plant-model mismatches were learned to use Gaussian processes. In [47] a process for defining neural approximations of NMPC laws with various topologies, such as reduced-order and decentralized controllers, was described. Recent research has examined the theoretical characteristics of deep neural network (DNN) approximations of MPC laws in several publications, including [48–51], with focus on linear parameter-varying systems [51], nonlinear systems with polytopic constraints [48] and linear systems [49, 50]. Neural approximations of MPC controllers have been proposed for several applications, including control of solar parabolic-trough plants [52], control of resonant power converters [45], building control [53], set point tracking for robot manipulators [54] and diesel engine air path control [55].

Scholars with expertise in reinforcement learning have examined a comparable methodology, often known as *MPC-guided policy search*. In this case, the goal is to train a DNN to mimic an MPC policy to convert a reinforcement learning problem

into a supervised learning problem (see, e.g., [56]). This method strongly emphasizes using measurements of the available output variables to train the DNN, while full-state feedback was used to execute the MPC strategy. In other words, *(deep) imitation learning* refers to using neural networks to approximate MPC rules. The earliest use of this method was to simulate the activities of a human operator, or *expert*. In this case, an MPC controller is the expert to mimic (see, for example, [57–59]).

1.4 Research Questions

The thesis work starts from the NMPC approach for SSM-based motion planning of robot manipulators introduced in [4], and aims at improving it in two main directions.

The first direction is to use approximations of the NMPC law based on DNNs, following the principles described in Section 1.3. By virtue of this approximation, computation times can be reduced, thereby improving performance. However, approximating MPC laws always brings the risk of violating the related constraints, and this would be especially critical in the case of SSM constraints.

The second direction is to extend the approach of [4] beyond the point terminal constraints method. Indeed, all the stabilizing strategies of SSM-based robot motion planning mentioned in Section 1.2 impose a constraint at the end of the prediction horizon, which requires the predicted robot configuration to be at a given value. If one could impose the final predicted robot configuration to belong to a set rather than coinciding with a specific value, this would increase the so-called domain of attraction. In other words, for a fixed value of the prediction horizon, the robot would be able to plan a trajectory to reach the goal configuration for a wider set of initial configurations. Designing an SSM-based stabilizing NMPC law for robot motion planning with set terminal constraint requires proving the stability properties of the closed-loop system, which can be a challenging problem.

Two research questions can be formulated related to the above-mentioned directions:

Q1. Is it possible to improve the performance of SSM-based NMPC laws, at the same time guaranteeing human safety, by emulating their behavior through DNNs?

Q2. Is it possible to define and design an SSM-based NMPC law with guaranteed stability properties and based on a set terminal constraint, which increases the domain of attraction compared to the point terminal constraint approach?

1.5 Thesis Outline

The remainder of the thesis is structured as follows:

- Chapter 2 of the thesis provides the description of an NMPC approach for SSM-based motion planning of robot manipulators similar to that proposed in [4], which constitutes the basis for the developments described in the following chapters.
- Chapter 3 presents an "offline" approach for deep imitation learning, which is designed to mimic the behavior of NMPC law via pre-recorded data. Two NMPC approaches, designed through various toolboxes and six deep imitation learning methods are designed and contrasted.
- Chapter 4 describes a more advanced deep imitation learning technique known as dataset aggregation (in short, DAgger), which is also designed to mimic the NMPC behavior. The main difference with respect to the techniques detailed in Chapter 3 is that the DNN in this approach is trained through online training, which leads to better exploring the state space compared to the offline approach.
- Chapter 5 shifts the focus to introducing an NMPC approach for SSM-based motion planning with set terminal constraint. The focus is on proving stability and evaluating the domain of attraction on a case study compared to the NMPC law described in Chapter 2.

1.6 Related Publications

The following publications are related to the thesis

[1] A. Nurbayeva, A. Shintemirov, and M. Rubagotti, "Deep imitation learning of nonlinear model predictive control laws for safe physical human-robot interaction," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 8384–8395, 2022.

[2] A. Nurbayeva, and M. Rubagotti, "Safely imitating predictive control policies for real-time human-aware manipulator motion planning: A dataset aggregation approach," *Under review*.

[3] A. Nurbayeva, and M. Rubagotti, "Nonlinear Model Predictive Control with Set Terminal Constraint for Safe Robot Motion Planning via Speed and Separation Monitoring," *Conditionally accepted under minor revision on Control Engineering Practice*.

The submitted paper [3] provides the theoretical foundations and experimental results for Chapters 2 and 5. Meanwhile, the research articles [1] and [2] contribute both the background material and the empirical findings pertaining to the themes of efficiency and safety, as elaborated upon in Chapters 3 and 4.

During my PhD I also worked on a related topic, which is teleoperation of robot manipulators with obstacle avoidance via deep reinforcement learning, the results of which were published in

[60] M. Rubagotti, B. Sangiovanni, A. Nurbayeva, G. P. Incremona, A. Ferrara and A. Shintemirov, "Shared control of robot manipulators with obstacle avoidance: A deep reinforcement learning approach," *IEEE Control Systems Magazine*, vol. 43, no. 1, pp. 44–63, 2023.

As this topic is not directly related to pHRI, this paper is not described within this thesis.

1.7 Notation

A sequence of integer values between n_1 and n_2 included is indicated as $\mathbb{N}_{[n_1,n_2]}$. Given a vector $v \in \mathbb{R}^{n_v}$, $\|v\|$ represents its Euclidean norm and, given a matrix $M \in \mathbb{R}^{n_v \times n_v}$, one has that $\|v\|_M^2 \triangleq v'Mv$. Also, given two vectors, $v_1, v_2 \in \mathbb{R}^{n_n}$, we define the shorthand notation $\|v_1\|_{v_2} \triangleq \|v_1 - v_2\|$. In the Euclidean space $\mathbb{R}^{n_v}$, the ball of radius r centered at $\bar{v}$ is defined as $\mathcal{B}_r(\bar{v}) \triangleq \{v \in \mathbb{R}^{n_v} : \|v\|_{\bar{v}} \leq r\}$. Given a set $\mathcal{S} \subseteq \mathbb{R}^{n_\mathcal{S}}$, $\mathrm{int}(\mathcal{S})$ represents its interior. The minimum and maximum eigenvalue of a symmetric positive (semi)definite matrix M are indicated as $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$, respectively. The $n \times n$ identity matrix is referred to as $\mathbb{I}_n$.

In the thesis, the following comparison functions will be used, defined in [22].

Definition 1.1. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}$-function (in brief, $f(\cdot) \in \mathcal{K}$) if it is continuous, positive definite and strictly increasing, and $f(0) = 0$.

Definition 1.2. A function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a $\mathcal{K}_\infty$-function (in brief, $f(\cdot) \in \mathcal{K}_\infty$) if it is a $\mathcal{K}$-function and $\lim_{s \to +\infty} f(s) = +\infty$.

A comprehensive glossary of all used terms is reported in Appendix A.

Chapter 2

Nonlinear MPC for safe human-robot workspace sharing

This chapter[1] provides a description of an NMPC approach for SSM-based motion planning of robot manipulators similar to that proposed in [4], which constitutes the basis for the developments described in the following chapters. These, in turn, represent the main contributions of this thesis. The main differences between the content of this chapter and that of [4] are the introduction of Assumption 2.1 – which defines a maximum acceptable distance between the human and the goal configuration of the robot – and the introduction of a stability proof based on comparison functions, which provides preliminary results to be used in the remainder of the thesis. The theory presented in this chapter is applicable to any serial manipulator, and the focus will be on the general formulation experimentsrather than on a specific case study.

The robot space occupancy, its kinematics and system dynamics used in the NMPC law are described in Section 2.1. The inequality constraints imposed in the NMPC problem are described in Section 2.2, and include limits on joint angles and speeds, avoidance of fixed obstacles and SSM constraints. The NMPC cost function is typically made of the sum of cost terms related to specific time instants along the prediction horizon, each referred to as *stage cost*; these terms are introduced in Section 2.3. Section 2.4 provides some details on the NMPC prediction and on admissible sequences, concepts that will be useful to formulate the NMPC problem. The latter will then be detailed in Section 2.5 for the case of point terminal constraints, and a proof of closed-loop stability will be given. Finally, a summary of the content of the chapter will be given in Section 2.6.

[1] Part of the text in this chapter is adapted from [3], conditionally accepted for publication in Control Engineering Practice (Elsevier).

2.1 Robot Modeling

2.1.1 State space model

The state space of the system is described by the vector of joint positions, directly referred to as $x \in \mathbb{R}^n$. The space of control variables is defined using a kinematic model of the robot, by assuming to be able to directly impose joint velocities, described by a vector $u \in \mathbb{R}^n$, where each joint speed $u_i \in \mathbb{R}$ represents the time derivative of the corresponding joint position x_i, for $i \in \mathbb{N}_{[1,n]}$. In addition to simplifying computations compared to a detailed dynamic model, this approach allows one to use the generated values of u as joint velocity references for low-level controllers. The system dynamics are given by

$$x(t+1) = f(x(t), u(t)), \tag{2.1}$$

where $t \in \mathbb{N}_{\geq 0}$ is the discrete-time index, and $f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ can be expressed in the following simplified form:

$$f(x(t), u(t)) = x(t) + Tu(t), \tag{2.2}$$

where T is the sampling interval. Two main reasons dictate the use of this simplified model, already highlighted in [4] and [30]. The first reason is that the NMPC controller solves a nonlinear numerical optimization problem at each sampling time, and this problem becomes more complex if a detailed and realistic dynamic model with joint torques as inputs is used. Instead, the simple model used in this thesis limits the complexity of the NMPC problem. The second reason is that for most commercial manipulators, the joint torques cannot be used as actual inputs, and instead, velocity references should be given, which are then tracked by inner control loops (whose implementation details are usually unavailable for the user). The robot task involves steering x to a desired goal configuration $x_g \in \mathbb{R}^n$ in the joint space. The associated "goal speed" $u_g \in \mathbb{R}^n$, which would allow the robot to remain at x_g once its value is reached, is defined as a vector of zeros in $\mathbb{R}^n$. Indeed, one can easily verify that $f(x_g, u_g) = x_g$.

2.1.2 Space occupancy and robot kinematics

The volume occupied by both the robot and human in the fixed Cartesian reference frame centered at the robot base and referred to as $O - xyz$ is over-approximated via a union of spheres (n_h spheres for the human and n_r for the robot). Each of them has a constant radius, while its center is assumed to be attached to a given location of either the human or the robot and moves together with it. Each sphere $\mathcal{S}_{h,j}$ on the human, $j \in \mathbb{N}_h \triangleq \mathbb{N}_{[1,n_h]}$, has center $p_{h,j} \in \mathbb{R}^3$ and radius $R_{h,j}$, whereas each sphere $\mathcal{S}_{r,i}$ on the robot, $i \in \mathbb{N}_r \triangleq \mathbb{N}_{[1,n_r]}$, has center $p_{r,i} \in \mathbb{R}^3$ and radius $R_{r,i}$. Following a common

terminology in robotics, we refer to the sphere centers as *test points*. For the sake of compactness, we refer to the vector given by all human test points (which, assuming the corresponding radii as constant and known, defines the human pose) as

$$\mathcal{H} = \begin{bmatrix} p_{h,1}^T & p_{h,2}^T & \dots & p_{h,n_h}^T \end{bmatrix}^T \in \mathbb{R}^{3n_h}. \tag{2.3}$$

To calculate the value of each $p_{r,i}$ and the Cartesian velocity vectors $v_i \in \mathbb{R}^3$ corresponding to the same points, one can employ forward and differential kinematics. More precisely, we can write

$$p_{r,i} = \mathscr{K}_{f,i}(x), \tag{2.4}$$

where $\mathscr{K}_{f,i}(\cdot) : \mathbb{R}^n \to \mathbb{R}^3$ is obtained via forward kinematics, and

$$v_i = \mathscr{K}_{d,i}(x, u), \tag{2.5}$$

where $\mathscr{K}_{d,i}(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^3$, is obtained via differential kinematics. The expressions of $\mathscr{K}_{f,i}(\cdot)$ and $\mathscr{K}_{d,i}(\cdot,\cdot)$ depend on the employed robot. A specific realization of these functions is described in Appendix B for the UR5 manipulator that will be used in all case studies. In this context, both the robot and the human are modeled as time-varying functions. However, within the NMPC prediction horizon, the human is treated as static, meaning that their state remains constant over the prediction period.

2.2 Inequality constraints

Following a common approach in industrial robotics, bounds are imposed on joint speeds and positions to avoid damaging the robot. These bounds are typically expressed independently for the speed and position of each joint, thus resulting in box constraints. Overall, the described constraints can be expressed as

$$x \in \Theta, \tag{2.6}$$

$$u \in \Omega, \tag{2.7}$$

with $\Theta \subseteq \mathbb{R}^n$ and $\Omega \subseteq \mathbb{R}^n$ being closed boxes including x_g and u_g, respectively, in their interiors. Both Ω and Θ are bounded sets, as all joint speeds must be limited. Limits can be imposed in principle on all joint variables (if they are not needed to avoid self-collisions, these bounds can be imposed as loose as needed to allow the manipulator to perform any needed motion).

Another set of constraints can be imposed on the robot test points to ensure that all robot spheres remain outside the space occupied by static obstacles:

$$p_{r,i} = \mathscr{K}_i(x) \in \mathcal{P}_i, \tag{2.8}$$

where $\mathcal{P}_i$, $i \in \mathbb{N}_{[1,n_r]}$, are suitably defined sets in the Cartesian space. Constraints (2.8) are expressed directly, depending on the system state x. It is assumed that all inequalities that compose (2.8) are defined such that

$$\mathscr{K}_i(x_g) \in \operatorname{int}(\mathcal{P}_i). \tag{2.9}$$

For example, suppose a manipulator has to avoid colliding with a wall. In that case, x_g should correspond to a configuration where the manipulator is at a non-zero distance from the same wall to avoid damage.

The last category of constraints we impose in the thesis is related to human safety. Equation (4) in [61], which, in turn, implements a particular form of the ISO/TS 15066 SSM criterion specified in equation (4) in [11], is the basis for the SSM algorithms. This requirement ensures that the robot can stop before a collision, even when the human and robot are moving toward each other (with the human moving at his/her maximum possible speed $\bar{v}_H$). In order to do that, we need to define a formulation that calculates the distance between the robot's i-th sphere and the human,

$$d_{ih} = \min_{j \in \mathbb{N}_h} \{d_{ij} - (R_{r,i} + R_{h,j})\}, \tag{2.10}$$

where a collision is indicated by a negative d_{ih}, and $R_{r,i}$ and $R_{h,j}$ represent the radii of the spheres that occupy the robot and human body, respectively. In (2.10), $d_{ij} = \|p_{r,i} - p_{h,j}\|$ is the Euclidean distance between the corresponding robot and human test points. By using (2.10), the following equation can be formulated, which has to be satisfied for each v_i:

$$\bar{d}_{ih} + \bar{d}_{ir} \leq d_{ih} + \epsilon_s, \tag{2.11}$$

where $\bar{d}_{ih}$ and $\bar{d}_{ir}$ are the maximum distances that $\mathcal{S}_{h,j}$ and $\mathcal{S}_{r,i}$ can travel until one can achieve $v_i = 0$. And ϵ_s is the maximum error with which the employed motion capture system detects the positions of the test points of the human. We can define $\bar{d}_{ih}$ as $\bar{d}_{ih} \triangleq \bar{v}_H(T_{dr} + \frac{v_i}{\bar{a}_r})$ using the time interval $T_{dr} \triangleq T + T_r$ needed for the detection of the human position (assumed to be equal to the sampling interval T) and the ensuing reaction (depending on used algorithm and hardware and denoted as T_r). On the other hand, $\frac{v_i}{\bar{a}_r}$ represents the time required to bring the robot to a controlled stop (where $\bar{a}_r$ is the maximum deceleration that the robot can impose on its test points). Additionally, $\bar{d}_{ir} \triangleq v_i T_{dr} + \frac{v_i^2}{2\bar{a}_r}$ is defined, where $\frac{v_i^2}{2\bar{a}_r}$ is the distance required to halt the robot and $v_i T_{dr}$ represents the distance traveled by the robot while the control input is calculated. For all $i \in \mathbb{N}_r$, one may solve (2.11) as an equation to get the maximum speed the robot can go at given the present value of d_{ih}. However, these SSM formulations cannot be directly applied to our NMPC problem. This limitation arises because the variable v_i must always be squared in the formulations. Indeed, the non-differentiability of the square root function (needed to calculate v_i) at the origin would pose difficulties for the

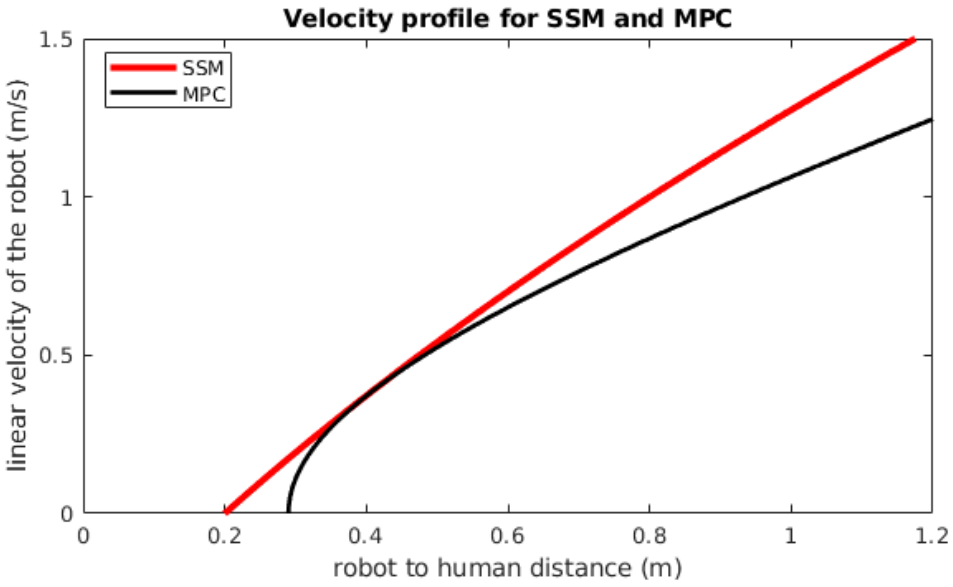

Figure 2.1: Velocity modulation profiles for the SSM and NMPC controllers (figure adapted from [4]).

optimization solver. Therefore, in the thesis, the SSM formulation is modified to enable changes in the robot trajectory as demonstrated in [4], as

$$\|v_i\|^2 \leq \alpha^2 \left(d_{ij}^2 - \rho_{ij}^2\right), \; i,j \in \mathbb{N}_r \times \mathbb{N}_h \tag{2.12}$$

where $\rho_{ij} \triangleq R_{r,i} + R_{h,j} + \bar{d}$; also, $\alpha \in \mathbb{R}_{>0}$ and $\bar{d} \in \mathbb{R}_{>0}$ are tuning parameters.

These parameters are determined based on the above-mentioned value $\bar{v}_H$, and on other parameters, including ϵ_s, $\bar{a}_r$, T and T_r. This modified version of the original SSM parameters is a more conservative version of (2.11); in other words, the speed v_i is always lower or equal than the threshold determined directly in (2.11). More precisely, it is necessary to choose values for α and $\bar{d}$ that ensure the linear velocity of the NMPC controller never exceeds the SSM limits and remains below the tangent line of the SSM, as depicted in Figure 2.1.

It is important to clarify that $\bar{d}$ depends on the detection/reaction time of the robot and represents the minimum distance between any two spheres $\mathcal{S}_{h,j}$ and $\mathcal{S}_{r,i}$ for which the robot would be able to stop, in the worst case scenario, before a collision with the human happens. Indeed, even for $\|v_i\| \to 0^+$, condition (2.12) cannot be satisfied when the distance between $\mathcal{S}_{h,j}$ and $\mathcal{S}_{r,i}$ is smaller than $\bar{d}$, as this implies $d_{ij}^2 - \rho_{ij}^2 < 0$. This is not a problem of the more conservative constraints defined in (2.12), as the same problem would occur using (2.11) directly.

Our formulation can easily show that the SSM constraints can be expressed as a function of x and u, in addition to $\mathcal{H}$. Indeed, given the definition of d_{ij} and the robot kinematics functions defined in (2.4) and (2.5), we can rewrite each of the $n_h \times n_r$

conditions in (2.12) as

$$\|\mathscr{K}_{d,i}(x,u)\|^2 \leq \alpha^2 \left(\|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2 - \rho_{ij}^2\right). \tag{2.13}$$

In short, (2.13) can be expressed as $u \in \mathbb{U}_{\text{SSM}}(x, \mathcal{H})$, as the range of joint speeds allowed by the SSM constraints depends on robot and human poses. However, as will be specified more clearly in the remainder of the paper, we will assume a constant value of $\mathcal{H}$ along the prediction horizon; thus, $\mathcal{H}$ will be considered as a set of fixed parameters, and the dependence on it will not need to be explicitly stated. Therefore, we will refer to the SSM conditions as

$$u \in \mathbb{U}_{\text{SSM}}(x). \tag{2.14}$$

The constant value of $\mathcal{H}$ is assumed to satisfy the following condition, which will be considered guaranteed in the remainder of the thesis.

Assumption 2.1. *There exists $\epsilon \in \mathbb{R}_{>0}$ such that all sub-vectors $p_{h,j}$ of $\mathcal{H}$ satisfy*

$$\|\mathscr{K}_{f,i}(x_g) - p_{h,j}\|^2 - \rho_{ij}^2 \geq \epsilon, \tag{2.15}$$

for all $i \in \mathbb{N}_r$.

Assumption 2.1 requires that the SSM constraint (2.13) be satisfied when $x = x_g$ and $u = u_g$, with the distance between $\mathcal{S}_{h,j}$ and $\mathcal{S}_{r,i}$ being greater than $\bar{d} + \epsilon$; this can be easily verified after noticing that $\|\mathscr{K}_{d,i}(x_g, u_g)\| = 0$. All constraints defined in (2.6) and (2.8) are imposed only on the state. Therefore, there exists a set $\mathbb{X} \subseteq \mathbb{R}^n$ such that their simultaneous satisfaction can be compactly expressed as

$$x \in \mathbb{X}. \tag{2.16}$$

Constraints (2.7) and (2.14) defined on the input are respectively independent and dependent on the state. Therefore, as the input constraints depend in general on the state via the SSM constraints, we express (2.7) and (2.14) together as

$$u \in \mathbb{U}(x) \tag{2.17}$$

where $\mathbb{U}(x) \triangleq \Omega \cap \mathbb{U}_{\text{SSM}}(x)$.

Given the requirement expressed in (2.9) and the fact that $x_g \in \text{int}(\Theta)$, one can see that $x_g \in \text{int}(\mathbb{X})$. One can also notice that $u_g \in \text{int}(\mathbb{U}(x_g))$. Indeed, $u_g \in \text{int}(\Omega)$ was directly required when introducing Ω; also, Assumption 2.1 implies $\|\mathscr{K}_{f,i}(x_g) - p_{h,j}\|^2 - \rho_{ij}^2 > 0$, but this corresponds to strictly satisfying (2.13) for $u = u_g$, or equivalently that $u_g \in \text{int}(\mathbb{U}_{\text{SSM}}(x_g))$.

2.3 Stage cost

To define the NMPC cost function, we first recall that $p_{r,n_r} = \mathscr{K}_{f,n_r}(x)$, and define the corresponding reference value as $p^*_{r,n_r} \triangleq \mathscr{K}_{f,n_r}(x_g)$. As p_{r,n_r} represents the center of the last robot sphere, it is implicitly assumed (without loss of generality) that it covers the robot end effector. Also, we introduce the position of a given test point p_{h,j_γ} (which is part of $\mathcal{H}$, assumed constant) for a fixed index $j = j_\gamma$, on the human operator. We can, therefore, introduce the following function:

$$\varphi(x, x_g) \triangleq \exp\left\{-\beta \frac{\left\|\mathscr{K}_{f,n_r}(x) - p_{h,j_\gamma}\right\|^2}{\left\|\mathscr{K}_{f,n_r}(x) - p^*_{r,n_r}\right\|^2}\right\}, \tag{2.18}$$

in which $\beta \in \mathbb{R}_{>0}$ is a tuning parameter while $\exp\{\cdot\}$ represents the natural exponential function. The tuning parameters β and γ can be chosen through a trial-and-error approach. This involves observing the robot motion to ensure it slows down appropriately as the human gets closer, while also ensuring that the robot is not repulsed too much, allowing it to reach the goal configuration. Notice that $\varphi(x, x_g)$, for a fixed human pose $\mathcal{H}$, tends to zero for any fixed value of $p_{h,j_\gamma} \neq p^*_{r,n_r}$ (condition guaranteed by Assumption 2.1) as x approaches x_g, since

$$\lim_{x \to x_g} \|\mathscr{K}_{f,n_r}(x) - \mathscr{K}_{f,n_r}(x_g)\| = 0. \tag{2.19}$$

The cost function that will be minimized in the considered NMPC problems includes the sum, along the prediction horizon, of stage costs defined as follows, for any given pair (x, u):

$$\ell(x, u) \ = \|x - x_g\|^2_Q + \|u\|^2_R + \gamma\varphi^2(x, x_g) \tag{2.20}$$

where $Q = Q' \in \mathbb{R}^{n\times n}$ and $R = R' \in \mathbb{R}^{n\times n}$ are both symmetric and positive definite matrices. The first two terms in the cost function penalize, as usual, the distance to the reference and the control energy (notice that an equivalent expression for the second term is $\|u - u_g\|^2_R$), whereas the third term increases when the robot approaches the human operator and is aimed at generating trajectories in which the speed is not excessively reduced due to the need to satisfy the SSM constraints. When the robot reaches the goal configuration x_g and a zero joint speed $u = u_g$ is applied, the stage cost satisfies $\ell(x_g, u_g) = 0$ for any fixed value of $\mathcal{H}$.

2.4 Predictions and admissible sequences

While the actual time evolution of the system state x is determined based on (2.1)-(2.2), we refer to closed-loop state values predicted by NMPC as x_u, and to the corresponding discrete-time instants along the prediction horizon as k, with $k = 0, \ldots, N-1$ (N

being the prediction horizon length). In all NMPC problems, we will impose that the closed-loop system dynamics satisfy $x_u(0) = x_0$ (i.e., the predicted closed-loop system state x_u, at a time $k = 0$ corresponding to the current time instant t, is equal to the measured state $x_0 = x(t)$) and $x_u(k+1) = f(x_u(k), u(k))$ (i.e., the prediction happens using the model defined in (2.1)-(2.2)).

From a given initial condition x_0, the admissible sequences of control inputs $u(0)$, $u(1), \ldots, u(N-1)$, indicated in short as $u(\cdot)$, are those that lead to satisfying $x_u(k) \in \mathbb{X}$ for $k = 0, \ldots, N$ and $u(k) \in \mathbb{U}(x_u)$ for $k = 0, \ldots, N-1$. We refer to sequences that satisfy these conditions as $u(\cdot) \in \mathbb{U}^N(x_0)$.

2.5 NMPC problem with point terminal constraint

In most stabilizing NMPC formulations, terminal constraints are imposed on the state at the end of the prediction horizon. These constraints guarantee that the system meets specific operational criteria or reaches a desirable steady state by ensuring it acts as intended at the end of the prediction horizon.

A point terminal constraint mandates a particular value for the state variables at the last time step of the prediction horizon. Explicitly requiring that the robot configuration x be precisely guided into the intended value x_g at the end of the prediction horizon is the simplest way to create stabilizing terminal conditions. For this purpose, additional restrictions are applied, defining a more limited set of valid sequences $u(\cdot) \in \mathbb{U}^N(x_0)$ also to incorporate the condition $x_u(N) = x_g$. These allowed sequences are designated in this thesis as $u(\cdot) \in \mathbb{U}^N_{x_g}(x_0)$. Therefore, the NMPC problem is designed as follows:

$$\text{minimize } J_N(x_0, u(\cdot)) \triangleq \sum_{k=0}^{N-1} \ell(x_u(k), u(k)) \tag{2.21a}$$

$$\text{with respect to } u(\cdot) \in \mathbb{U}^N_{x_g}(x_0) \tag{2.21b}$$

$$\text{subj. to } x_u(0) = x_0,\ x_u(k+1) = f(x_u(k), u(k)). \tag{2.21c}$$

The value of $\mathcal{H}$ utilized to describe cost function and constraints is measured at the current instant t and kept constant throughout the prediction horizon: in practice, this value can change at each sampling time instant, however, we will assume it to remain constant in order to prove stability properties. The optimal sequence of control moves obtained by solving (2.21) is denoted by $u^*(\cdot) = \{u^*(0), u^*(1), \ldots, u^*(N-1)\}$, and the related optimal cost by $J^*_N(x_0)$. The NMPC feedback policy is then established by solving the system dynamics (2.1) using $u(t) = u^*(0)$ by the so-called receding-horizon concept.

The following technical lemma provides preliminary results to demonstrate the stability characteristics of the described NMPC law.

Lemma 2.1. *If Assumption 2.1 is satisfied for the system dynamics specified in* (2.2)*, the constraints introduced in* (2.16)-(2.17)*, and the stage cost formulated in* (2.20)*, then the following conditions are satisfied:*

(i) *the dynamics $f(x,u)$ and the stage cost $\ell(x,u)$ are continuous with respect to both x and u, for all $x,u \in \mathbb{R}^n$;*

(ii) *Ω introduced in* (2.7) *is a compact set;*

(iii) *there exists a ball $\mathcal{B}_\nu(x_g)$ with $\nu \in \mathbb{R}_{>0}$ defined such that, for all $x \in \mathcal{B}_\nu(x_g) \cap \mathbb{X}$, one has that $u_x \in \mathbb{U}(x)$, with u_x satisfying $f(x,u_x) = x_g$;*

(iv) *there exists a function $\tilde{\alpha}_2(\cdot) \in \mathcal{K}_\infty$ (see Definition 1.2) such that, for all $x \in \mathcal{B}_\nu(x_g) \cap \mathbb{X}$, one has that $\ell(x,u_x) \leq \tilde{\alpha}_2(\|x\|_{x_g})$.*

Proof. We start by proving (i). From (2.2), it is immediate to see that $f(x,u)$ is linear (and therefore continuous) in both its arguments. The stage cost in (2.20) is composed of three terms, of which the first two are quadratic (and thus continuous) concerning x and u, respectively; the third term, defined in (2.18), is made of the composition of continuous functions (in particular, one can easily verify that this is always true for the forward kinematics $\mathscr{K}_{f,n_r}(x)$), and is therefore also continuous. We can conclude that $\ell(x,u)$ is also continuous with respect to both its arguments, which shows that condition (i) holds.

As for (ii), Ω is introduced as a set that has to be both closed and bounded, and this coincides with the definition of a compact set.

To prove (iii), we start by noticing that the only expression of u_x such that $f(x,u_x) = x_g$ is, by definition of $f(x,u)$ in (2.2), $u_x \triangleq \frac{1}{T}(x^* - x)$. To verify that $u_x \in \mathbb{U}(x)$, one needs to recall that $\mathbb{U}(x) = \Omega \cap \mathbb{U}_{\mathrm{SSM}}(x)$. Therefore, we will proceed by determining two different balls with radii ν_1 and ν_2, respectively, in the state space, such that $x \in \mathcal{B}_{\nu_1}(x_g) \cap \mathbb{X}$ implies $u_x \in \Omega$, and $x \in \mathcal{B}_{\nu_2}(x_g) \cap \mathbb{X}$ implies $u_x \in \mathbb{U}_{\mathrm{SSM}}(x)$. Then, we will obtain the value of ν as $\nu = \min\{\nu_1, \nu_2\}$.

We first focus on determining ν_1. Given the expression of u_x, one has that $\|u_x\|_{u_g} = \frac{1}{T}\|x\|_{x_g}$, and thus $x \in \mathcal{B}_{\nu_1}(x_g) \Leftrightarrow u_x \in \mathcal{B}_{\bar{r}}(u_g)$, with $\bar{r} \triangleq \nu_1/T$. Since Ω was defined in Section 2.2 including u_g in its interior, there exists a positive value $\bar{r}$ satisfying $\mathcal{B}_{\bar{r}}(u_g) \subseteq \Omega$. This means that the value of ν_1 that we need is $\nu_1 = \bar{r}T$, as

$$x \in \mathcal{B}_{\bar{r}T}(x_g) \cap \mathbb{X} \Rightarrow u_x \in \mathcal{B}_{\bar{r}}(u_g) \subseteq \Omega. \tag{2.22}$$

The value of ν_2 cannot be directly found with a similar rationale, as Assumption 2.1 guarantees that u_g is in the interior of $\mathbb{U}_{\mathrm{SSM}}(x_g)$, but not in the interior of $\mathbb{U}_{\mathrm{SSM}}(x)$ as x varies. Therefore, as a first step to find ν_2, we define the following scalar function:

$$\phi_{f,ij}(x) \triangleq \|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2. \tag{2.23}$$

As $\mathscr{K}_{f,i}(x) : \mathbb{R}^n \to \mathbb{R}^3$ represents the forward kinematics of a manipulator, each of its components includes linear combinations of sine and cosine functions of single components (or of the product of multiple components) of x. Sine and cosine functions are Lipschitz continuous, and the same property extends to their linear combination [62]. Therefore, each component of $\mathscr{K}_{f,i}(x)$ is Lipschitz continuous, and since the composition of Lipschitz continuous functions is also Lipschitz continuous [62], we can conclude that $\phi_{f,ij}(x)$ is Lipschitz continuous (remember that $p_{h,j}$ is constant by assumption). More precisely, for all $x, x_g \in \mathbb{X}$ and all $i, j \in \mathbb{N}_r \times \mathbb{N}_h$, there exists $L_{f,ij} \in \mathbb{R}_{>0}$ such that

$$|\phi_{f,ij}(x) - \phi_{f,ij}(x_g)| \leq L_{f,ij}\|x\|_{x_g}. \tag{2.24}$$

This implies that, defining $L_f \triangleq \max_{i,j} L_{f,ij}$, for all $x, x_g \in \mathbb{X}$ and all $i, j \in \mathbb{N}_r \times \mathbb{N}_h$ the following holds:

$$\phi_{f,ij}(x_g) - L_f\|x\|_{x_g} \leq \phi_{f,ij}(x) \leq \phi_{f,ij}(x_g) + L_f\|x\|_{x_g}. \tag{2.25}$$

We now introduce a second scalar function

$$\phi_{d,i}(x) \triangleq \|\mathscr{K}_{d,i}(x, u_x)\|^2 = \left\|\mathscr{K}_{d,i}\left(x, \tfrac{1}{T}(x_g - x)\right)\right\|^2, \tag{2.26}$$

defined using $\mathscr{K}_{d,i}(x, u)$ in the specific case in which $u = u_x$, which always holds in part (iii) of this lemma. It is always possible (see, e.g., [63]) to express

$$\mathscr{K}_{d,i}(x, u) = J_{\text{lin},i}(x) \cdot u, \tag{2.27}$$

where $J_{\text{lin},i}(x) \in \mathbb{R}^{3\times n}$ is the part of the so-called *geometric Jacobian* of the manipulator that accounts for the linear motion of a test point i. Similarly to $\mathscr{K}_{f,i}(x)$, each element of the matrix $J_{\text{lin},i}(x)$ contains linear combinations of sine and cosine functions or products of components, or single components, of x. As $u = u_x$ in our case is an affine function of x, all considerations about Lipschitz continuity made for $\phi_{f,ij}(x)$ also hold for $\phi_{d,i}(x)$. Thus, we can conclude that, for all $x, x_g \in \mathbb{X}$ and all $i \in \mathbb{N}_r$, there exists $L_d \in \mathbb{R}_{>0}$ such that

$$\phi_{d,i}(x_g) - L_d\|x\|_{x_g} \leq \phi_{d,i}(x) \leq \phi_{d,i}(x_g) + L_d\|x\|_{x_g}. \tag{2.28}$$

From (2.13), one can see that $u_x \in \mathbb{U}(x)$ if and only if

$$\|\mathscr{K}_{d,i}(x, u_x)\|^2 - \alpha^2\|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2 \leq -\alpha^2\rho_{ij}^2. \tag{2.29}$$

Based on the first inequality in (2.25) and on the second inequality in (2.28), for all $x, x_g \in \mathbb{X}$ and all $i \in \mathbb{N}_r$, the following holds:

$$\|\mathscr{K}_{d,i}(x, u_x)\|^2 - \alpha^2\|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2 = \phi_{d,i}(x) - \alpha^2\phi_{f,ij}$$

$$\leq \phi_{d,i}(x_g) + L_d\|x\|_{x_g} - \alpha^2\phi_{f,ij}(x_g) + \alpha^2 L_f\|x\|_{x_g}$$
$$\leq -\alpha^2\|\mathscr{K}_{f,i}(x_g) - p_{h,j}\|^2 + (L_d + \alpha^2 L_f)\|x\|_{x_g}, \quad (2.30)$$

as $\phi_{d,i}(x_g) = \|\mathscr{K}_{d,i}(x_g, u_g)\|^2 = 0$. If we define $\nu_2 \in \mathbb{R}_{>0}$ such that $\|x\|_{x_g} \leq \nu_2$, (2.30) implies

$$\|\mathscr{K}_{d,i}(x, u_x)\|^2 - \alpha^2\|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2 \leq -\alpha^2\|\mathscr{K}_{f,i}(x_g) - p_{h,j}\|^2 + (L_d + \alpha^2 L_f)\nu_2. \quad (2.31)$$

A sufficient condition to guarantee that $u_x \in \mathbb{U}(x)$ is then $(L_d+\alpha^2 L_f)\nu_2 - \alpha^2\|\mathscr{K}_{f,i}(x) - p_{h,j}\|^2 \leq -\alpha^2\rho_{ij}^2$, or equivalently

$$\nu_2 \leq \frac{\alpha^2\left(\|\mathscr{K}_{f,i}(x_g) - p_{h,j}\|^2 - \rho_{ij}^2\right)}{L_d + \alpha^2 L_f}. \quad (2.32)$$

By means of Assumption 2.1, (2.32) is satisfied if we set $\nu_2 = \alpha^2\epsilon/(L_d + \alpha^2 L_f)$. Part (iii) is thus proven, with

$$\nu = \min\{\nu_1, \nu_2\} = \min\left\{\bar{r}T, \alpha^2\epsilon/(L_d + \alpha^2 L_f)\right\}. \quad (2.33)$$

As for (iv), the stage cost in (2.20) is defined by three terms, so $\tilde{\alpha}_2(\cdot)$ can be found as the sum of the upper bound on each of these terms. For the first term, one can see that

$$\|x - x_g\|_Q^2 \leq \lambda_{\max}(Q)\|x\|_{x_g}. \quad (2.34)$$

The second term also depends on u; remembering that in part (iv) of this lemma we always assume $u = u_x$, we obtain

$$\|u_x\|_R^2 = \left\|\tfrac{1}{T}(x_g - x)\right\|_R^2 \leq \tfrac{1}{T^2}\lambda_{\max}(R)\|x\|_{x_g}. \quad (2.35)$$

To determine an upper bound on the third term of (2.20), we examine the two squared norms in the exponent of $\varphi(x, x_g)$. Based on the Lipschitz continuity analysis in part (iii) of the proof and on Assumption 2.1, we can determine a lower bound on $\phi_{f,n_r j_\gamma}(x) = \left\|\mathscr{K}_{f,n_r}(x) - p_{h,j_\gamma}\right\|^2$ as

$$\phi_{f,n_r j_\gamma}(x) \geq \phi_{f,n_r j_\gamma}(x_g) - L_f\|x\|_{x_g} \geq -\nu\|x\|_{x_g} + \rho_{n_r j_\gamma}^2 + \epsilon, \quad (2.36)$$

for all $x, x_g \in \mathcal{B}_\nu(x_g) \cap \mathbb{X}$

Regarding the second squared norm in the exponent of $\varphi(x, x_g)$, we define it as $\phi_f^*(x) \triangleq \left\|\mathscr{K}_{f,n_r}(x) - p_{r,n_r}^*\right\|$. The same considerations hold for this function as for functions $\phi_{f,ij}$, although the Lipschitz constant is generally different from L_f. We can therefore claim the existence of a new constant L_r such that, for all $x, x_g \in \mathbb{X}$,

$$\phi_f^*(x_g) - L_r\|x\|_{x_g} \leq \phi_f^*(x) \leq \phi_f^*(x_g) + L_r\|x\|_{x_g}. \quad (2.37)$$

The second inequality in (2.37) provides us with an upper bound on $\phi_f^*(x)$, which, noticing that $\phi_f^*(x_g) = 0$ by construction, can be expressed as

$$\left\| \mathscr{K}_{f,n_r}(x) - p^*_{r,n_r} \right\| \leq L_r \|x\|_{x_g}. \tag{2.38}$$

An upper bounding function on the third term in (2.20), for all $x \in \mathcal{B}_\nu(x_g) \cap \mathbb{X}$, can therefore be expressed as

$$\gamma \varphi^2(x, x_g) \leq \Phi(\|x\|_{x_g}) \tag{2.39}$$

where

$$\Phi(\|x\|_{x_g}) \triangleq \gamma \exp\left\{ -2\beta \frac{(-\nu\|x\|_{x_g} + \rho^2_{n_r j_\gamma} + \epsilon)^2}{L_r^2 \|x\|^2_{x_g}} \right\}. \tag{2.40}$$

$\Phi(\cdot)$ is a $\mathcal{K}$-function of its argument [22, Def 2.13], while the upper bounding functions defined in (2.34) and (2.35) are $\mathcal{K}_\infty$-functions. This implies that $\tilde{\alpha}_2(\cdot) \in \mathcal{K}_\infty$ can be defined as

$$\tilde{\alpha}_2(\|x\|_{x_g}) \triangleq \left(\lambda_{\max}(Q) + \tfrac{1}{T^2} \lambda_{\max}(R) \right) \|x\|^2_{x_g} + \Phi(\|x\|_{x_g}), \tag{2.41}$$

which concludes the proof. ■

Based on conditions (i)-(iv) guaranteed by Lemma 2.1, the properties of the NMPC strategy with point terminal constraints are proven by the following theorem.

Theorem 2.2. *Consider the closed-loop system* (2.2) *with input* $u(t)$ *determined via the receding-horizon principle from the solution of the NMPC problem* (2.21)*, whose constraints are formulated in* (2.16)-(2.17)*, and the stage cost is introduced in* (2.20)*. Then, if Assumption 2.1 holds, the set* $\mathbb{X}_F$ *of states for which a solution of* (2.21) *exists is a positively invariant set, and* x_g *is an asymptotically stable equilibrium point for the above-mentioned closed-loop system with domain of attraction* $\mathbb{X}_F$*.*

Proof. Regarding recursive feasibility, according to [22, Lem. 5.3], $\mathbb{X}_F$ is positively invariant for the closed-loop system (2.2) with input $u(t)$ obtained through an NMPC problem with point terminal constraints in the form (2.21), if there exists a control input $u_* \in \mathbb{U}(x_g)$ such that $f(x_g, u_*) = x_g$. In our formulation, choosing $u_* = u_g$ implies $f(x_g, u_*) = x_g$, as already specified in Section 2.1. The definition of Ω in Section 2.2 implies $u_g \in \Omega$, whereas $u_g \in \mathbb{U}_{\text{SSM}}(x_g)$ is guaranteed by Assumption 2.1. These two conditions imply $u_g \in \mathbb{U}(x_g)$. As the required assumptions for proving [22, Lem. 5.3] are satisfied, recursive feasibility is guaranteed.

According to [22, Thm. 5.5], the closed-loop system (2.2) with input $u(t)$ determined via an NMPC problem with point terminal constraints in the form (2.21) is asymptotically stable if the following conditions hold:

(a) there exists $u_* \in \mathbb{U}(x_g)$ such that $f(x_g, u_*) = x_g$ and $\ell(x_g, u_*) = 0$;

(b) there exist $\mathcal{K}_\infty$-functions $\alpha_1(\cdot)$, $\alpha_2(\cdot)$ and $\alpha_3(\cdot)$ such that

$$\alpha_1(\|x\|_{x_g}) \leq J_N^*(x) \leq \alpha_2(\|x\|_{x_g}) \tag{2.42}$$

and $\ell(x, u) \geq \alpha_3(\|x\|_{x_g})$ for all $x \in \mathbb{X}_F$ and all $u \in \mathbb{R}^n$.

As for condition (a), choosing again $u_* = u_g$ also implies $\ell(x_g, u_*) = 0$, as already specified in Section 2.3. This property and those already described in the recursive feasibility proof guarantee the satisfaction of condition (a). Regarding condition (b), as the initial state x cannot be modified by the NMPC problem, the first term $\|x - x_g\|_Q^2$ in the stage cost can be used to determine a global $\mathcal{K}_\infty$ lower bounding function, as $\|x - x_g\|_Q^2 \geq \lambda_{\min}(Q)\|x\|_{x_g}^2$. Therefore, we can choose $\alpha_1(\|x\|_{x_g}) = \lambda_{\min}(Q)\|x\|_{x_g}^2$. According to [22, Prop 5.7], since conditions (i)-(iv) here formulated in Lemma 2.1 hold, then the existence of a suitable upper bounding function $\alpha_2(\cdot) \in \mathcal{K}_\infty$ is guaranteed for all $x \in \mathbb{X}_F$. Finally, given the formulation of the stage cost, one can set $\alpha_3(\|x\|_{x_g}) = \alpha_1(\|x\|_{x_g})$, and therefore condition (b) holds as well. Asymptotic stability is therefore proven. ■

2.6 Chapter summary

This chapter focused on the theoretical formulation of an NMPC law for SSM-based motion planning of a serial manipulator in the presence of a human operator. Its contribution consists of the presented stability analysis, which is why simulation and experimental results are not present in this chapter. The NMPC law here introduced will constitute the foundation for the algorithms presented in the remainder of the thesis. These algorithms will be compared on specific case studies with the NMPC algorithm of this chapter.

It is important to note that the tuning parameters of the robot may need to be adjusted differently depending on the specific scenario. In the scenario presented in this thesis, the goal is to bring the robot to the goal configuration while avoiding all the obstacles, including human operator. Therefore, careful tuning is essential. The system must be calibrated by evaluating all relevant parameters to ensure that obstacles are effectively avoided and the robot successfully reaches the goal.

BLANK

Chapter 3

Imitation learning: an offline approach

This chapter[1] describes the design and implementation of DNN-based motion planning schemes that aim to imitate the NMPC law for pHRI described in Chapter 2. Three distinct strategies for the data collection process were designed, each utilizing different NMPC approaches: the first NMPC law sets the human position as constant (and equal to the currently measured value) throughout the prediction horizon; the second approach utilizes prerecorded future human motions; the third strategy adds a past measurement of the human position to the second approach to implicitly get velocity information. Subsequently, DNNs are trained through those data and implemented for the UR5 universal robot manipulator with six degrees of freedom. The main task of the robot is to reach two different target configurations while satisfying all constraints.

Due to the fact that all the proposed methods were trained based on pre-recorded data generated by applying the NMPC law, there may be certain aspects of the environment that were not thoroughly explored during training. Therefore, the method described in this section is referred to as "offline". This can increase the chances of constraint violations in situations in which the robot finds itself at a state value never experienced during training, which is why the next chapter will cover other imitation learning methods that can better explore the workspace.

The structures of DNN approaches, data collection, and training procedures are described in Sections 3.1 and 3.2. Section 3.3 covers implementation and comparison of the suggested DNN-based techniques with two distinct NMPC implementations.

3.1 DNNs for imitation learning

The proposed imitation learning process uses a feed-forward neural network with several hidden layers between the input and output layers. Feed-forward DNNs are specifically chosen for approximating NMPC solutions because they efficiently handle the main problem: mapping current system states to control actions. And they have a simpler structure with a unidirectional flow of data, which typically requires less computational resources compared to more complex architectures. The simplicity and efficiency

[1]The text, algorithms, tables and figures in this chapter have been adapted from [1]: © 2023 IEEE.

make feed-forward DNNs a natural fit for this task, offering a good balance between computational cost and performance. Each neuron computes the weighted sum of the outputs from the preceding layer and applies a nonlinear function, sometimes known as *activation function*, to the result. After being determined initially randomly, the weight values are adjusted through training until the required performance is attained [64]. Using a robot simulator or a real robot, the DNN must be taught to mimic the input-output behavior of the NMPC scheme, described in (2.21), using prerecorded NMPC solutions derived for a specific set of human motions. The agent is designed using DNNs, according to the deep imitation learning method [65–67]. The extended state s of the system, which consists of human position $\mathcal{H}$ and robot state x, is provided to the input layer of the DNN. The DNN implements a deterministic policy $u^{DNN}(s)$. This policy determines the DNN output as an action u^{DNN} at a given extended state value, which is provided to the robot joints as reference joint velocities.

The NMPC motion planning system described in Chapter 2 serves as the "expert" to be imitated. At any given value of s, it produces an action $u^{NMPC}(s)$ that is still made of joint speed references. In general, the set of extended states s for which an NMPC law was calculated is referred to as $\mathcal{S}$. The corresponding sets of NMPC-generated and DNN-generated actions are instead referred to as $\mathcal{U}^{NMPC}$ and $\mathcal{U}^{DNN}$, respectively.

A quadratic loss function

$$L(\mathcal{U}^{NMPC},\mathcal{U}^{DNN}) = \sum_{j=1}^{n_s} \left(u^{DNN}(s_j) - u^{NMPC}(s_j)\right)^2. \tag{3.1}$$

may be computed for n_s realizations s_j of the state, and associated DNN and NMPC actions. The DNN policy at episode i, namely $u_i^{DNN}(s)$ – which belongs to the set Π of all possible policies implementable by the given DNN – is updated for a given number of training episodes N_e by first obtaining the corresponding loss function L_i, and then updating the DNN to obtain an updated policy $u_{i+1}^{DNN}(s)$ through a single back-propagation step. The optimal policy $u_*^{DNN}(s)$ is chosen as the version of $u_i^{DNN}(s)$ that minimizes L on a given validation set of states, different from the training set.

Using pre-recorded NMPC data allows one to directly associate the value of u^{NMPC} determined for specific values of s. This removes the influence of NMPC calculation delays in the acquired data. Moreover, datasets created especially for human motion in industrial activities are accessible (see, for example, [68]), which can help DNN designers save time and effort by generating and annotating a pre-existing dataset.

If appropriate data are unavailable for the particular application, we still advise to record the human motion to create an ad-hoc dataset, which one can use to train the DNN. It can be impractical to generate the NMPC data with real-time readings of human motion, since it would take a long time. Additionally, it may be necessary to repeat the training procedure many times to fine-tune the parameters of the NMPC

scheme. One more note about creating NMPC data: if an accurate robot simulator is available, utilizing it instead of the actual robot would save even more time.

3.2 DNN structures

This thesis considers different DNN topologies. All architectures trained using pre-recorded data are indicated by the letter "O" in the names, which stands for "offline".

- When using O-DNN-0, the mimicked NMPC technique, also known as NMPC-0, is based only on the values of $\mathcal{H}$ and x (i.e., s) that are presently measured, since the NMPC law is typically assuming (see Chapter 2) that the human posture remains constant across the prediction horizon. To accomplish this, the DNN is designed using the same input data, as follows:

$$u^{O-DNN-0}(s) \approx u^{NMPC-0}(s), \tag{3.2}$$

  where s is the state of the system that contains from x and $\mathcal{H}$, and $u^{O-DNN-0}(s)$ and $u^{NMPC-0}(s)$ are the functions that define u for O-DNN-0 and NMPC-0, respectively.

- O-DNN-P (where the letter P stands for "prediction") is trained to mimic an NMPC scheme (specifically, NMPC-P) that uses the actual human position, both present and future, throughout the whole prediction horizon, denoted as $\bar{\mathcal{H}}$. This NMPC dataset can only be constructed offline using a human motion that has already been recorded. However, the O-DNN-P approximation only takes into account $\mathcal{H}$, the present human stance, which results in

$$u^{O-DNN-P}(s) \approx u^{NMPC-P}(\bar{s}), \tag{3.3}$$

  where $\bar{s} \triangleq \{x, \bar{\mathcal{H}}\}$.

- In O-DNN-PS (where "S" stands for "sequence"), both the present $\mathcal{H}$ and past $\mathcal{H}^-$ positions of the human are used as input data. This provides indirect information on human speed data, which may be connected to the future human motion:

$$u^{O-DNN-PS}(s^-) \approx u^{NMPC-PS}(\bar{s}), \tag{3.4}$$

  where $s^- \triangleq \{x, \mathcal{H}, \mathcal{H}^-\}$.

All proposed DNN approaches are trained according to Algorithm 1, where $\mathcal{S}$ and $\mathcal{U}^{NMPC}$ sets are determined using the NMPC policy before beginning training the DNN. Subsequently, at each episode i, the DNN generates actions $\mathcal{U}^{DNN}$ corresponding to states in $\mathcal{S}$. Then, according to the outcomes of the loss function detailed in Equation 3.1,

Algorithm 1 Offline training: O-DNN

Generate $\mathcal{S}, \mathcal{U}^{NMPC}$ via NMPC
Set $u_0^{DNN}(s)$ to any policy in Π
for $i \leftarrow 0$ to N_e **do**
  Run policy u_i^{DNN} on $\mathcal{S}$ to get $\mathcal{U}^{DNN}$
  Calculate $L(\mathcal{U}^{NMPC}, \mathcal{U}^{DNN})$
  Determine $u_{i+1}^{DNN}(s)$ via back-propagation
end for
Return best policy $u_*^{DNN}(s)$ based on validation.

DNN policy $u_i^{DNN}(s)$ is updated and recorded in a set of policies Π. Upon completion of the training, the algorithm identifies and returns the optimal policy $u_*^{DNN}(s)$, as determined through validation.

Large numbers of components are usually used to represent the quantity of information in $\mathcal{H}$, which might result in a high dimension of the DNN input layer, particularly in the O-DNN-PS situation. A smaller input layer size might be possible with fewer variables needed to define $\mathcal{H}$, which could result in improved performance for the same quantity of training data. According to this notion, a symmetrical autoencoder [69] is trained to get a reduced-dimensionality description $\mathcal{H}_E$ of the human position. Autoencoder is a type of artificial neural network primarily used for data compression and feature learning. It works by learning to compress data into a lower-dimensional representation (encoder) and then reconstructing it back to its original form (decoder). The encoder part consists of one or more layers of neurons that progressively reduce the dimensionality of the input. The final layer of the encoder is often referred to as the latent space, and it represents the compressed version of the input data. The decoder is the mirror image of the encoder. It takes the encoded data from the latent space and attempts to reconstruct the original input. The output of the decoder should be as close as possible to the original input, which is how the network learns to capture the most important features in the latent space. Autoencoders are a powerful tool which can capture the most important features of data, making them invaluable in many machine learning tasks. The trained encoder is then inserted into each of the three models that were previously mentioned. The versions of the DNNs using the encoder would still rely on the formulas given in (3.2)-(3.4), although with $\mathcal{H}_E$ substituting $\mathcal{H}$ in the left-hand side, and would be referred to as O-E-DNN-0, O-E-DNN-P, and O-E-DNN-PS, respectively, where 'E' stands for "encoder".

We suggest training a distinct DNN for every goal configuration x_g, as the implemented NMPC technique addresses a regulation problem with a specific goal configuration. This makes the training process simpler because x_g does not need to be provided as extra DNN input.

3.3 Case Study

The DNN-based controllers described in this chapter were implemented, together with the original NMPC scheme explained in Chapter 2, on a Universal Robot UR5, a collaborative robot with six degrees of freedom ($n = 6$). The robot task was to sequentially reach two distinct goal configurations, the first of which was provided by

$$x_g = \begin{bmatrix} 0.0 & -2.3 & -1.1 & -1.2 & -1.2 & 0.5 \end{bmatrix}' \triangleq x_A,$$

and the second by

$$x_g = \begin{bmatrix} 3.0 & -1.6 & -1.7 & -1.7 & -1.7 & 1.0 \end{bmatrix}' \triangleq x_B.$$

To determine the location and speed of the robot test points, the forward and differential kinematics were formulated using standard homogeneous transformation matrices based on the Denavit-Hartenberg parameters of the robot (details are explained in Appendix B). A total of $n_r = 7$ test points and associated spheres were used to represent the space occupied by the robot (also discussed in detail in Appendix B). Upper bounds on the magnitudes of the joint speeds were introduced to prevent harm to the robot, defining the set Ω presented in Section 2 as

$$\Omega = \{u \ : \ \|u\|_\infty \leq 1 \text{ rad/s}\}. \tag{3.5}$$

The UR5 was mounted on a table with a surface height of 1.2 m; this was the only obstacle considered fixed, so it was simple to apply avoidance constraints using the provided sphere radii.

To train DNN schemes based on a representative and reliable sequence of human actions, it was decided to use the large set of human movement data to solve industrial-like tasks (for example, screwing a bolt in three different places, untying a knot, and placing loads of various weights on shelves) presented in [68]. Thirteen different human subjects took part in the process of recording the dataset, and each subject performed the prescribed actions five times.

Since the human operator could not move the legs above the table surface, only the upper part of the human body had to be covered with $n_h = 14$ spheres to avoid possible collisions. In all subjects, the radii of the spheres were adjusted to the same value, which was $R_{h,j} = 0.55$ m for $j = 1, 13$, $R_{h,j} = 0.60$ m for $j = 2, 14$, $R_{h,j} = 0.50$ m for $j = 3, 4, 5, 6, 7, 8$, $R_{h,j} = 0.45$ m for $j = 9, 10$ and $R_{h,j} = 0.48$ m for $j = 11, 12$.

The human speed limit of $\bar{v}_H = 2$ m/s was selected according to the guidelines provided in ISO 13855 standard [12, Sec. V], and also $\varepsilon_s = 10^{-3}$ m (which is a reasonable tolerance for a high-precision motion capture system) was set as the maximum position error in determining the location of the center of the spheres on the human frame.

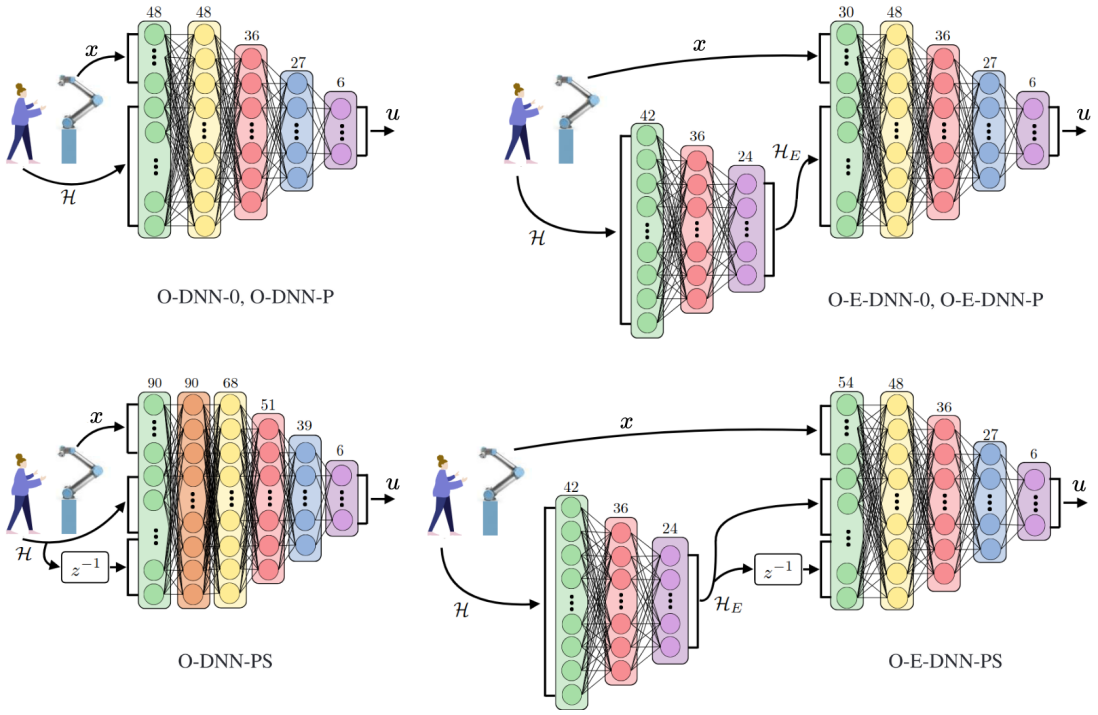

Figure 3.1: Schematics of the proposed DNNs. O-DNN-0 and O-DNN-P have an identical structure, and the same holds for O-E-DNN-0 and O-E-DNN-P. The layer's dimensions are written on their top, and the z^{-1} block represents the delay in discrete time unit required to provide both $\mathcal{H}$ and $\mathcal{H}^-$ (or both $\mathcal{H}_E$ and $\mathcal{H}_E^-$) to the DNN. It is displayed as a separate neural network for the cases when the encoder is present (figure adapted from [1], © 2023 IEEE).

Ultimately, the value of $T_{dr} = 0.1$ s was chosen as the detection-reaction time; this is equivalent to twice the NMPC sampling time ($T_s = 50$ ms); in fact, it is reasonable to assume that T_s represents the maximum delay for obtaining a human pose (detection time), and the NMPC law is calculated during the subsequent sampling instant.

Through trial and error, we determined $Q = 10 \cdot I_{6\times6}$, $R = I_{6\times6}$, and $\gamma = 500$ for the cost function (2.20), while $\beta = 3.0$ was set in (2.18). A sampling interval equivalent to the sampling period was used to discretize the system dynamics, and sequential quadratic programming (SQP) and direct multiple inclusion were used to solve the NMPC problem (2.21) using the Gauss-Newton approximation to the Hessian of the Lagrangian.

First, we used the ACADO Toolbox [70], setting the number of SQP steps to 10 for NMPC and using full condensing for solving each SQP step via the active-set solver qpOASES [71]. As a second option, we used the *acados* toolbox [72], setting the same number of SQP steps and using again full condensing for solving each SQP step, but now using the interior-point solver HPIPM [73].

During the preliminary test experiments, it was found that none of the considered DNNs could accurately lead the robot to the specified goal configurations. Therefore, it was decided to train two different DNNs for each target, depending on the value

of $\delta_x = \|x - x_g\|_\infty$. A subset of the NMPC dataset generated for the entire robot movement between different global configurations was used to train a single DNN that was active when $\delta_x > 0.18$ rad. Another subset of the NMPC dataset generated for a sequence of random initial values of x, defined using $\delta_x \leq 0.2$ rad, was used to train another DNN.

The *acados* version of the NMPC law showed the best results regarding task execution time and minimizing the NMPC cost function during experimental testing on a real UR5. As a result, we decided to use *acados* to create the datasets needed to train various DNNs. Two different datasets were required because the standard and pre-release versions of the NMPC were NMPC-0 and NMPC-P. A high-precision URSim industrial simulator was used to collect datasets to create the NMPC-P dataset and save time compared to obtaining training data at the experimental facility.

In both scenarios, the movements of twelve different participants from [68] were used; each subject performed three different sequences of movements five times. With a sampling interval of 50 ms, the values of x, $\bar{\mathcal{H}}$ and u were recorded when running NMPC. Approximately $1.35 \cdot 10^6$ data points were obtained for each dataset. Of these, approximately $1.14 \cdot 10^6$ (corresponding to the movements of ten subjects) were used to train DNNs, and approximately $2.1 \cdot 10^5$ (corresponding to the movements of two subjects) were used for validation.

The structure of the different DNNs is depicted in Fig. 3.1. All DNNs had an output layer of 6 neurons, one for each component of u. The size of the input layer was equal to $n_x + 3n_h = 6 + 42 = 48$ for O-DNN-0 and O-DNN-P and to $n_x + 2 \cdot 3n_h = 6 + 2 \cdot 42 = 90$ for O-DNN-PS, due to the presence in the latter of both $\mathcal{H}$ and $\mathcal{H}^-$. After several attempts, it was decided to train an autoencoder with a latent space size of 24, corresponding to the encoder's output layer size. Consequently, the size of the input layer was equal to $6+24 = 30$ for O-E-DNN-0 and O-E-DNN-P and to $6 + 2 \cdot 24 = 54$ for O-E-DNN-PS. For every DNN, the numbers and sizes of the hidden layers were determined through trial and error and can be observed in Fig. 3.1. Initially, we experimented with employing a similar number of neurons in all hidden layers. However, this approach did not yield the desired results. Then we tested a feedforward neural network with three hidden layers, which was structured to progressively reduce the dimensionality of neurons in hidden layers. For example, for O-DNN-0 and O-DNN-P the first hidden layer is initialized with 48 neurons, matching the dimensionality of the input layer. This choice ensures that the network fully captures the complexity of the input data. Subsequently, the number of neurons in each successive hidden layer is reduced by approximately 25%, resulting in 36 neurons in the second layer and 27 neurons in the third layer. This gradual reduction is intended to encourage the network to learn increasingly abstract and compact representations of the data, thereby enhancing its ability to generalize. The final output layer consists of 6 neurons,

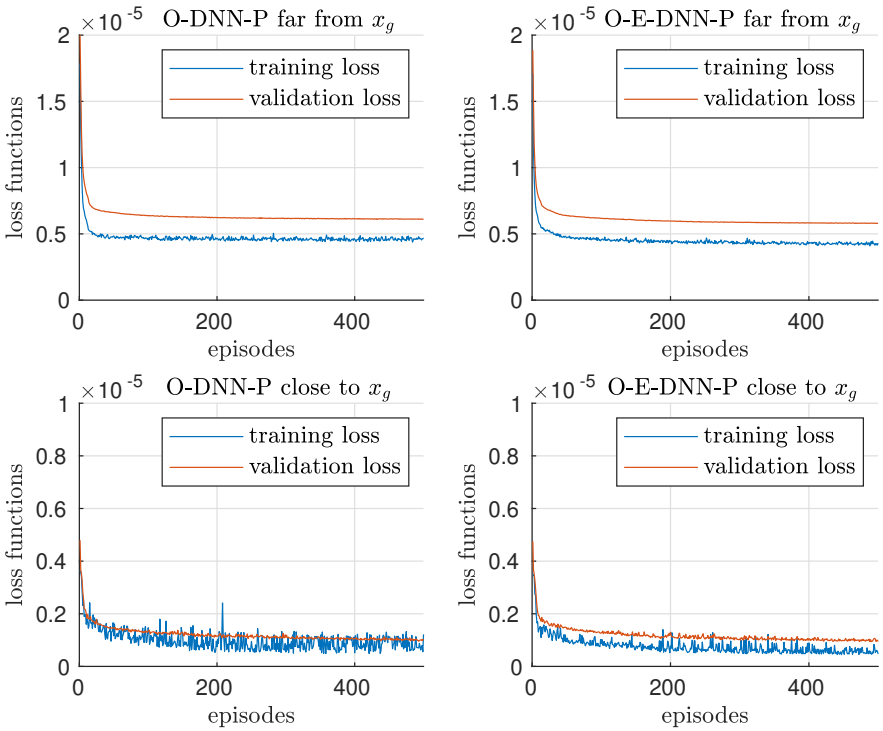

Figure 3.2: Learning curves for O-DNN-P (left) and O-E-DNN-P (right), distinguishing the cases in the neighborhood of ("close to") x_g and outside the same neighborhood ("far from x_g"). For simplicity, we only show the case when $x_g = x_B$. For the remaining DNNs, similar curves were found, including the instances when $x_g = x_A$ (figure adapted from [1], © 2023 IEEE).

corresponding to the dimensionality of the target output. To automatically satisfy the criterion (3.5), the output layer activation functions were configured as hyperbolic tangents. The PyTorch Python library served as the basis for creating the suggested DNNs. The Adam optimizer, with a learning rate equal to 10^{-3}, was used for training, and dropout [74] was used during training to avoid overfitting, with a dropout rate of 25% for each hidden layer. As examples, the resulting learning curves are shown in Fig. 3.2 for O-DNN-P and O-E-DNN-P, when $x_g = x_B$. As $x_g = x_B$, each episode consisted of a full robot motion in the suitable NMPC dataset, either from x_A to x_B (for the DNN that accounted for the case "far from x_g") or from a configuration with $\delta_x \leq 0.02$ rad to x_B (for the DNN that accounted for the case "near x_g").

3.4 Experimental Results

In the UR5 experiments, the NMPC and DNN techniques were applied on a laptop running Robot Operating System (ROS) equipped with an i7-9750H CPU, 16 GB RAM, and an NVIDIA GeForce GTX 1660 Ti. The NMPC controllers were implemented as distinct C++ nodes in ROS after being produced as C code routines using either ACADO or *acados*.

Through the Universal Robots driver, the computer transmitted the values of u, which serve as the reference for the UR5 inner control loops, to the robot at a frequency of 20 Hz. The UR5 was given the task of moving from x_A to x_B or vice versa, using the simulated motion of the 13^{th} human subject in the dataset, which was never utilized for training or validation, to test the three created algorithms on our experimental setup.

For each investigated motion planning algorithm, 25 separate experiments were conducted using 25 distinct segments of the recorded human motion to evaluate the performance of each algorithm. The various algorithms were compared using four distinct metrics:

- The task completion time T_c needed to move from x_A to x_B and back.
- The evaluation of the NMPC cost J_{exp} obtained by summing the terms $\|x - x_g\|_Q^2 + \|u\|_R^2 + \gamma\varphi^2(x, x_g)$ over all sampling instants of the task, based on real measurements rather than only over the prediction horizon and based on predictions as in (2.21a).
- The average path length L_p traveled by the end effector, defined similarly to J_{exp}, by summing the terms

  $$\|p_{n,n_r}(k+1) - p_{n,n_r}(k)\|$$

  for all sampling instants.
- The overall control effort U, calculated as for J_{exp}, but only accounting for the control term $\|u_\ell(k)\|_R^2$.

The results can be seen in Table 3.1, which also shows the average and maximum computation times for each controller, indicated as $\hat{\tau}$ and $\bar{\tau}$, respectively. In terms of comparing the two NMPC implementations (both for NMPC-0, as NMPC-P cannot be implemented in real-time), *acados* outperformed ACADO in T_c and J_{exp}; both obtained around the same term L_p, but ACADO's U was lower.

Interestingly, with the settings used, the *acados* computation time was higher and occasionally exceeded the sampling interval $T_s = 50$ ms. Despite this computation-induced delay, *acados* achieved an overall performance that we can consider better than ACADO. Therefore, as mentioned above, for this reason *acados* was used to build the NMPC dataset for DNN training and validation.

Except for U, where O-DNN-0 outperformed the other DNNs without an encoder, O-DNN-P produced the best performance across all metrics. It did not appear that using the previous value $\mathcal{H}^-$ of the human stance would benefit O-DNN-PS in any noticeable way. Comparable outcomes were shown for the DNNs with encoder, where O-E-DNN-P beat the other structures for every metric except J_{exp}, which O-E-DNN-0 reduced. Once more, there was no advantage that O-E-DNN-PS had over the equivalent DNNs

that did not use $\mathcal{H}^-$. One explanation for this in both situations could be that human motion could be predicted with reasonable accuracy based solely on the present human stance. Adding more parameters made the task more complex. Instead, the encoder's presence further reduced the DNN complexity, allowing for an overall improvement in performance. As anticipated, the computation durations for the DNNs displayed in Table 3.1 are significantly less than those of the NMPC schemes. This factor helped the DNNs beat the NMPC schemes, along with the capacity to implicitly account for information on the human motion prediction, except for the O-DNN-0 and O-E-DNN-0 situations. O-E-DNN-P appears to be the best option considering all these factors, yet more research into the robot's movements is required before making any firm judgments. With and without an encoder, O-DNN-P and O-E-DNN-P demonstrated the best performance in their respective categories, as shown in Figs. 3.3-3.9. When employing these DNNs, we present the temporal evolutions of the robot joint angles and speeds and also draw comparisons with the *acados* NMPC case. Two consecutive robot moves are depicted in the data. The robot travels from x_A to x_B and back in both scenarios, with the second motion involving the human operator being closer. The dashed vertical lines show the end times for each motion.

The time evolution of the joint positions x_i, which are the elements of x, for each of the three motion planning systems is displayed in Figure 3.3. First, the more "regular" time evolution of the joint angles in the first motion recalls, for NMPC and O-DNN-P at least, the response of a stable linear time-invariant system to a step input; in contrast, less regular time evolution of the joint angles in the second motion results from the UR5 having to deal with the operator's proximity. Regarding task completion time, O-DNN-P and O-E-DNN-P appear to function similarly; however, NMPC causes a slower motion.

The three motion planning systems are represented by the joint speeds u_i (components of u) as a function of time in Figure 3.4. The joint speed range is within

Table 3.1: Average values of the computation times and considered measures for a total of 25 experiments, adapted from [1], © 2023 IEEE.

	T_c(s)	J_{exp}	L_p(m)	U	$\bar{\tau}$(ms)	$\hat{\tau}$(ms)
ACADO	22.74	6588.8	**2.59**	**74.09**	32.5	14.0
acados	**17.58**	**5311.7**	2.60	78.17	162.2	39.9
O-DNN-0	15.71	5098.7	2.57	**62.04**	24.6	1.7
O-DNN-P	**15.60**	**5081.3**	**2.55**	63.09	11.8	1.8
O-DNN-PS	16.12	5348.4	2.56	64.81	22.9	1.9
O-E-DNN-0	15.74	**4871.8**	2.55	63.33	26.0	2.8
O-E-DNN-P	**15.56**	4993.8	**2.54**	**61.54**	26.5	2.2
O-E-DNN-PS	15.82	5057.3	2.59	62.28	39.6	3.2

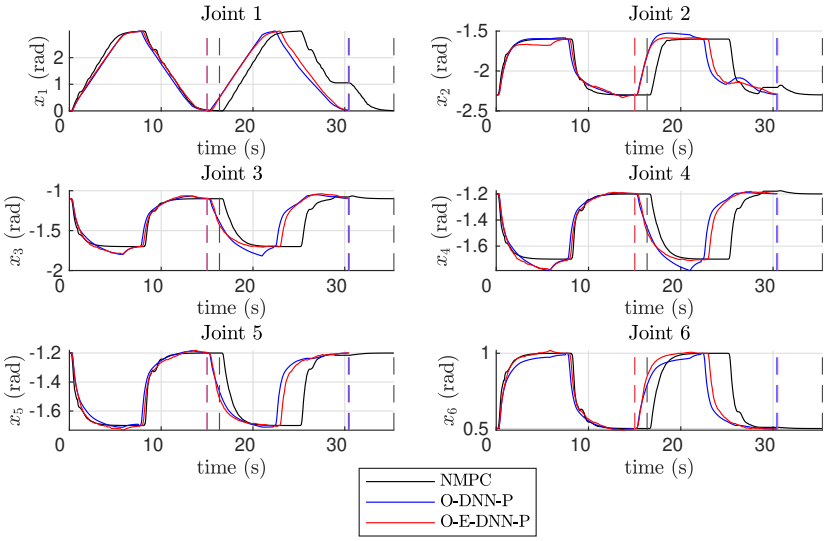

Figure 3.3: Time evolution of joint positions x_i for NMPC with *acados*, O-DNN-P and O-E-DNN-P. The vertical lines with dashes denote the instants in time when the joint position colored in the same way finishes the cycle from x_A to x_B and back. (figure adapted from [1], © 2023 IEEE).

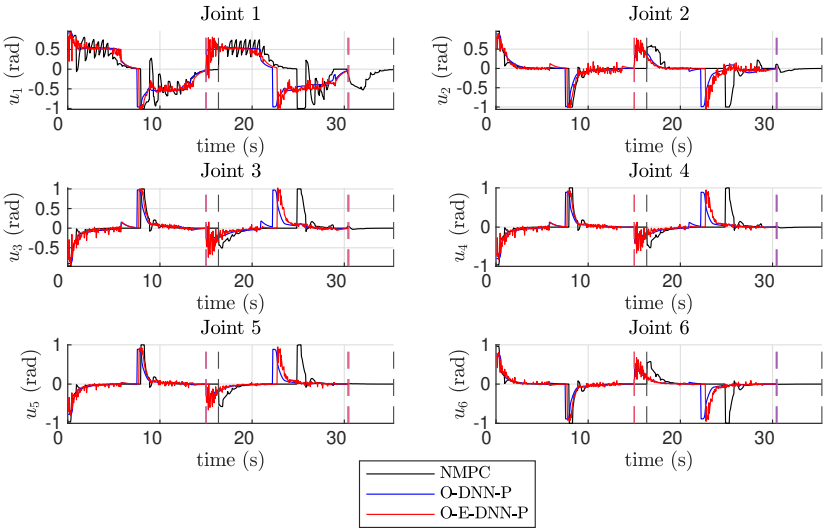

Figure 3.4: Time evolution of joint speeds u_i for NMPC with *acados*, O-DNN-P and O-E-DNN-P, corresponding to the joint positions reported in Fig. 3.3. The vertical lines with dashes denote the instants in time when the joint position colored in the same way finishes the cycle from x_A to x_B and back. (figure adapted from [1], © 2023 IEEE).

the specified limit of 1 rad/s for all three systems, and the speed value range is quite close. This indicates that both DNNs were able to accurately mimic the aggressiveness of the NMPC behavior in terms of the control variables; in fact, a comparison with NMPC would have been highly challenging if the DNNs had been able to complete tasks faster at the cost of using comparatively higher joint speeds. It is also essential to note that the joint speeds for O-DNN-P appear smoother than those for NMPC. This is probably because NMPC introduces significant temporal delays through the

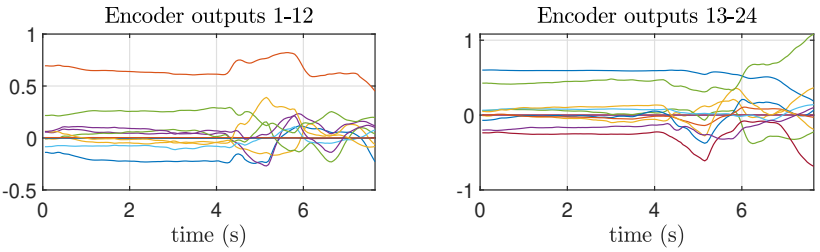

Figure 3.5: Time evolutions of the 24 components of the encoder outputs $\mathcal{H}_E$ for O-E-DNN-P, corresponding to the first 7.65 s of Figs. 3.3 and 3.4 (robot motion from x_A to x_B). (figure reprinted from [1], © 2023 IEEE).

computation of the control law, whereas there are none in DNN-P.

In contrast to NMPC, O-E-DNN-P introduced oscillations at higher frequencies. These might result from quick changes in the elements of $\mathcal{H}_E$; this is untrue for our case study, though, as these have generally been found to be smooth. For simplicity, the time evolution of all 24 encoder outputs is reported in Fig. 3.5 for the first 7.65 s of the O-E-DNN-P motion reported in Figs. 3.3 and 3.4, which corresponds to the robot moving from x_A to x_B. An alternative explanation would be that the autoencoder performed poorly; however, this is likewise untrue in this instance, as executing the entire autoencoder might yield a reasonable reconstruction of $\mathcal{H}$. The result, shown for the same time interval of Fig. 3.5, is visible in Fig. 3.6, where the three components of $p_{h,j}$, namely $p^x_{h,j}$, $p^y_{h,j}$ and $p^z_{h,j}$, are shown for test points located at the right shoulder ($j = 3$), at the right elbow ($j = 7$) and at the right hand ($j = 11$). Consequently, the low-dimensionality representation of $\mathcal{H}$ in $\mathcal{H}_E$ is most likely the cause of the observed speed oscillations. This representation allowed the decoder part of the autoencoder to approximately reconstruct $\mathcal{H}$ from $\mathcal{H}_E$. Still, it also caused the DNN to produce an output with slight but frequent variations.

For each of the three algorithms, the time evolution of the test point speeds is presented in Figs. 3.7-3.9. These numbers likewise show the same oscillating phenomenon noted for the joint speeds. Specifically, the quick changes in the speed v_7 of the relevant test point for O-E-DNN-P, shown in Fig. 3.9, resulted in a minor SSM constraint violation after roughly 16 s. Since the DNN parameters depend on training data imitation of the NMPC technique, minor violations of the constraints might be expected. It has also been noted that when inadequate training or unexpected human behavior, such as human motions that deviate significantly from those taught during training, substantially larger violations of the SSM limitations may appear. Safety may consistently be enforced by stopping the robot should these conditions be broken.

The behavior of the other DNNs is qualitatively similar to those of O-DNN-P (for O-DNN-0 and O-DNN-PS) and O-E-DNN-P (for O-E-DNN-0 and O-E-DNN-PS). The best option for this case study appeared to be O-DNN-P because the encoder-based DNN methods produced rapid oscillations of the joint speeds, which occasionally

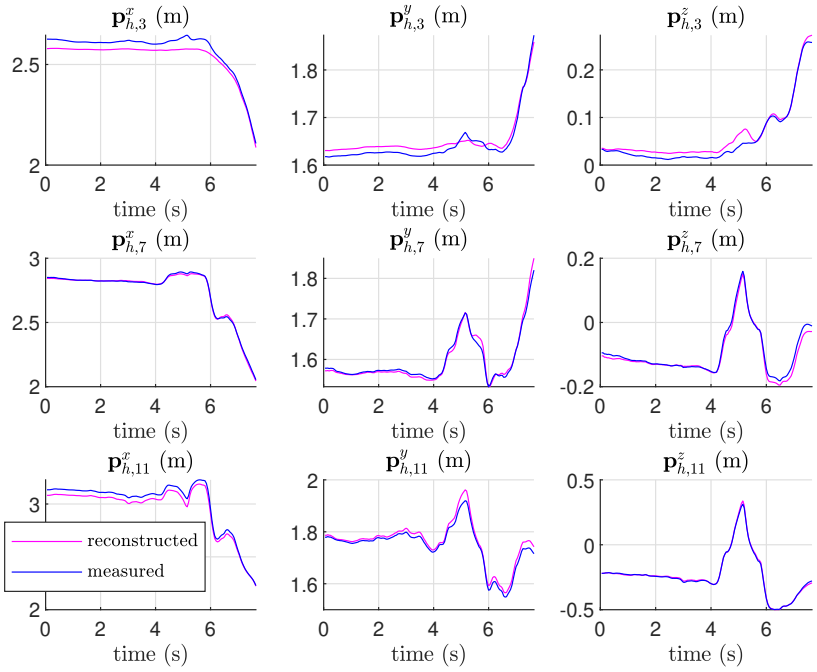

Figure 3.6: Time evolution of inputs and outputs of the autoencoder, corresponding to all components of $\mathcal{H}$, for the same time interval of Fig. 3.5. In each row, we show the three components of $p_{h,j}$, namely $p^x_{h,j}$, $p^y_{h,j}$ and $p^z_{h,j}$, for test points located at the right shoulder ($j = 3$), at the right elbow ($j = 7$) and at the right hand ($j = 11$). (figure reprinted from [1], © 2023 IEEE).

resulted in minor violations of the SSM constraints.

3.5 Chapter summary

Six distinct DNN-based motion planning schemes were designed and implemented in this chapter with the goal of mimicking a safe (in accordance with the SSM principle) NMPC scheme for pHRI. The experimental findings demonstrate that the DNN-based schemes effectively compensate the induced performance loss of the NMPC scheme due to its computational delay, and they further enhance performance by utilizing the future human motion data obtained during training. The results from experiments with the real robot demonstrated that the O-E-DNN-P algorithm outperformed all other offline algorithms in terms of task completion time. However, since the O-E-DNN-P algorithm violated constraints during the experiments, it cannot be considered a safe and reliable approach. In contrast, the O-DNN-P algorithm adhered to all imposed SSM constraints, leading us to conclude that O-DNN-P is the most effective approach among the offline learned algorithms. Like other learning-based techniques, the disadvantage of all implemented DNN-based schemes is that the motion planning algorithm could not accurately plan the robot movements if the human moved differently from the training set. Therefore, particular actions must be utilized to ensure that human motion

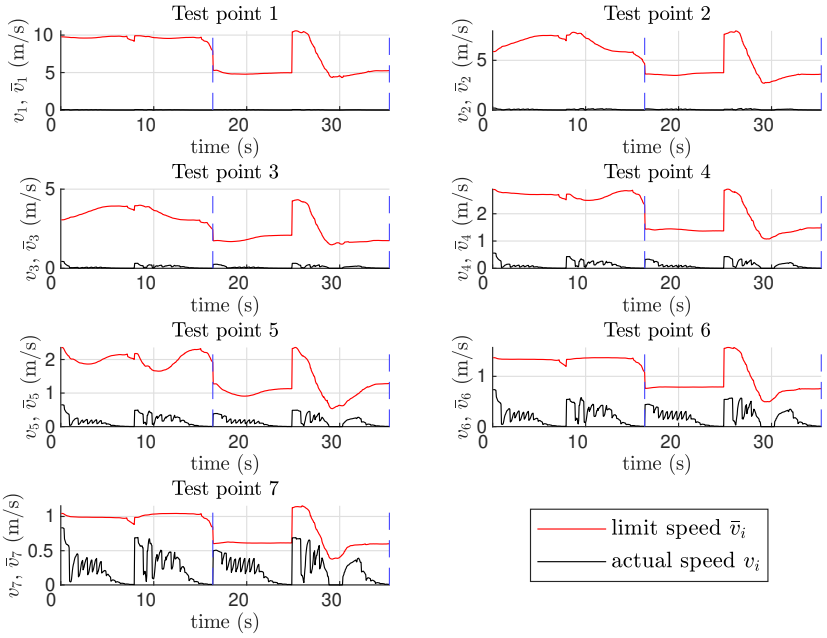

Figure 3.7: Time evolution of robot test point speeds v_i for the *acados* NMPC scheme, with corresponding limit speed $\bar{v}_i$, for the robot motions reported in Figs. 3.3 and 3.4. The dashed vertical blue lines indicate task completion (figure reprinted from [1], © 2023 IEEE).

follows exact instructions. The fact that the DNN-based schemes successfully applied their learned behavior to a novel subject that had never been seen before is noteworthy, though, as it indicates that the suggested algorithms may be somewhat generalizable across various human operators. Meanwhile, training the DNN models on prerecorded data might not lead to exploring the environment as well as one would want. Therefore, the controller may not be aware of certain parts of the environment where the robot did not move during the data collection procedure. That is why the method described in this section is called “offline”. To explore the workspace better, we will investigate different methods of imitation learning approach in the next chapter.

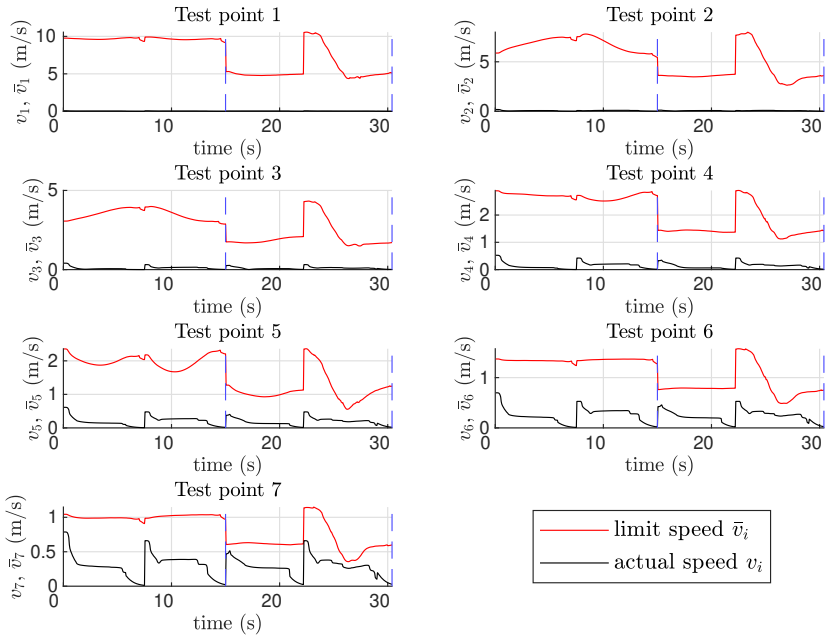

Figure 3.8: Time evolution of robot test point speeds v_i for the O-DNN-P scheme, with corresponding limit speed $\bar{v}_i$, for the robot motions reported in Figs. 3.3 and 3.4. The dashed vertical blue lines indicate task completion (figure reprinted from [1], © 2023 IEEE).

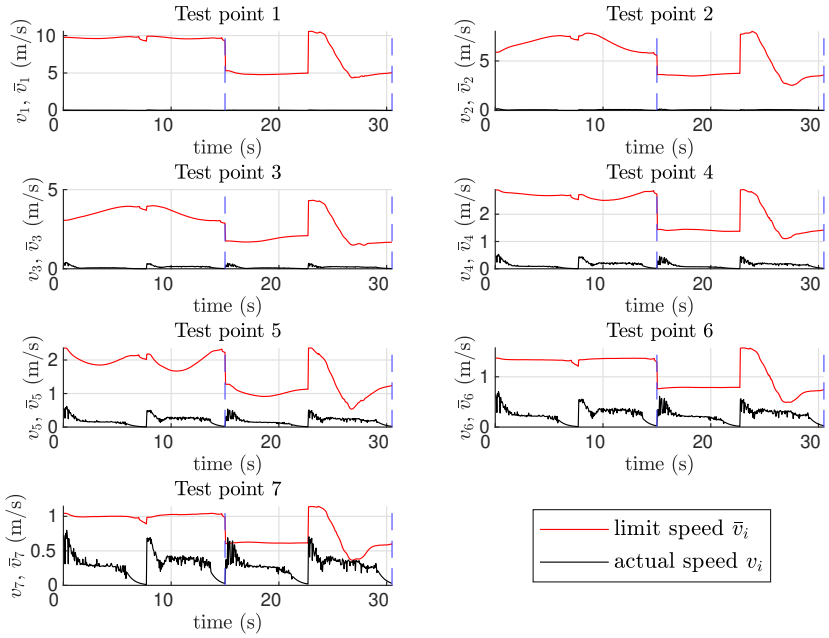

Figure 3.9: Time evolution of robot test point speeds v_i for the O-E-DNN-P scheme, with corresponding limit speed $\bar{v}_i$, for the robot motions reported in Figs. 3.3 and 3.4. The dashed vertical blue lines indicate task completion (figure reprinted from [1], © 2023 IEEE).

BLANK

Chapter 4

Imitation learning: a DAgger approach

This chapter describes another imitation learning approach known as “DAgger”. If the offline imitation learning approach described in Chapter 3 is used to train DNNs, the environment might not be explored sufficiently well. Therefore, this chapter investigates the DAgger method, which further explores the environment during the training phase. Moreover, to ensure safety in pHRI tasks using DNNs, so-called “safety filters” are implemented and tested for the same robot as in Chapter 3.

The general information about the DAgger method and algorithm structures are described in Section 4.1. The use of safety filters and encoders is explained in Section 4.2. The O-DNN methods covered in Chapter 3 are modified to make them comparable with the DAgger method. All the designed approaches are implemented and tested in Sections 4.3-4.4.

4.1 The DAgger method

Based on DAgger, this section examines an NMPC imitation learning framework for human-robot workspace sharing. When using the offline approach, the *expert* policy would be run by observation-action pairs. These pairs would then be sent to the imitation learning algorithm to train the DNN. However, when tested, this could cause the trained policy to get data quite different from the observations obtained during the expert policy training, leading to insufficient performance.

The DAgger method was introduced by Ross et al. in [75] to address this problem in a generic imitation learning environment. It trains the agent online iteratively (i.e., new states are created using the current policy $u_i^{DNN}(s)$) on a continually aggregating dataset. To avoid using an entirely "untrained" DNN, we utilize a variant of DAgger in our work, where the first N_0 episodes still use states created using NMPC. It helps to prevent generating several data points that would be unusable for training. Sets $\mathcal{S}$, $\mathcal{U}^{DNN}$, and $\mathcal{U}^{NMPC}$ are initially started as empty sets, as detailed in Algorithm 2. Next, fresh NMPC-generated data $\mathcal{S}_i$ and $\mathcal{U}_i^{NMPC}$ are aggregated to $\mathcal{S}$ and $\mathcal{U}^{NMPC}$, respectively, for the first N_0 episodes (initialization). The resultant set $\mathcal{S}$ is then subjected to the DNN policy $u_i^{DNN}(s)$, which produces the associated set of actions $\mathcal{U}^{DNN}$. The policy

$u_{i+1}^{DNN}(s)$ is generated by calculating the loss function L_i at the i^{th} episode. To compare the efficiency between the DAgger method and the offline approach, the same loss function defined in (3.1) can be employed.

After initialization, a similar DAgger method is still utilized until the last episode N_e, though now the new states $\mathcal{S}_i$ are always generated via DNN instead of NMPC. This should allow for a more profitable exploration of the state space as compared to Algorithm 1. The resulting DNN that executes the optimal approach with this training technique will be referred to as DA-DNN, where "DA" stands for "DAgger".

Algorithm 2 DAgger: DA-DNN

$\mathcal{S} \leftarrow \emptyset, \mathcal{U}^{DNN} \leftarrow \emptyset, \mathcal{U}^{NMPC} \leftarrow \emptyset$
Set $u_0^{DNN}(s)$ to any policy in Π
for $i \leftarrow 0$ to N_0 **do**
  Generate $\mathcal{S}_i, \mathcal{U}_i^{NMPC}$ via NMPC
  Aggregate datasets: $\mathcal{S} \leftarrow \mathcal{S} \bigcup \mathcal{S}_i$
  Aggregate datasets: $\mathcal{U}^{NMPC} \leftarrow \mathcal{U}^{NMPC} \bigcup \mathcal{U}_i^{NMPC}$
  Run policy $u_i^{DNN}(s)$ on $\mathcal{S}$ to get new $\mathcal{U}^{DNN}$
  Calculate $L_i(\mathcal{U}^{NMPC}, \mathcal{U}^{DNN})$
  Determine $u_{i+1}^{DNN}(s)$ via back-propagation
end for
for $i \leftarrow N_0 + 1$ to N_e **do**
  Generate $\mathcal{S}_i$ via π_i
  Run NMPC on $\mathcal{S}_i$ to get $\mathcal{U}_i^{NMPC}$
  Aggregate datasets: $\mathcal{S} \leftarrow \mathcal{S} \bigcup \mathcal{S}_i$
  Aggregate datasets: $\mathcal{U}^{NMPC} \leftarrow \mathcal{U}^{NMPC} \bigcup \mathcal{U}_i^{NMPC}$
  Run policy $u_i^{DNN}(s)$ on $\mathcal{S}$ to get new $\mathcal{U}^{DNN}$
  Calculate $L_i(\mathcal{U}^{NMPC}, \mathcal{U}^{DNN})$
  Determine $u_{i+1}^{DNN}(s)$ via back-propagation
end for
Return best policy $u_*^{DNN}(s)$ based on validation.

Compared to Algorithm 2, where this set is initially empty and grows at each episode, the DNN policy taught using Algorithm 1 can rely on a considerably more extensive collection of states $\mathcal{S}$ right away. Therefore, to execute the offline method and adequately evaluate its performance compared to the DA-DNN approach, we utilized a dataset-aggregation methodology for the offline method as well. In practice, this was done by prolonging the initialization phase of Algorithm 2 to the last episode N_e, as seen in Algorithm 3. The resultant DNN is called ODA-DNN, where the abbreviation "ODA" stands for "offline DAgger."

4.2 Safety filters

As already seen in Chapter 3, as the obtained DNNs approximate the NMPC behavior, violations of the NMPC constraints may occur, especially when further simplifications

Algorithm 3 Offline DAgger: ODA-DNN

$\mathcal{S} \leftarrow \emptyset, \mathcal{U}^{DNN} \leftarrow \emptyset, \mathcal{U}^{NMPC} \leftarrow \emptyset$
Set $u_0^{DNN}(s)$ to any policy in Π
for $i \leftarrow 0$ to N_e **do**
  Generate $\mathcal{S}_i, \mathcal{U}_i^{NMPC}$ via NMPC
  Aggregate datasets: $\mathcal{S} \leftarrow \mathcal{S} \bigcup \mathcal{S}_i$
  Aggregate datasets: $\mathcal{U}^{NMPC} \leftarrow \mathcal{U}^{NMPC} \bigcup \mathcal{U}_i^{NMPC}$
  Run policy $u_i^{DNN}(s)$ on $\mathcal{S}$ to get new $\mathcal{U}^{DNN}$
  Calculate $L_i(\mathcal{U}^{NMPC}, \mathcal{U}^{DNN})$
  Determine $u_{i+1}^{DNN}(s)$ via back-propagation
end for
Return best policy $u_*^{DNN}(s)$ based on validation.

are introduced (for example, by inserting encoders). This can be particularly problematic when violating SSM constraints, which would affect safety. As a potential solution, one can avoid directly using the u joint speed references selected by $u_*^{DNN}(s)$ when testing the algorithm. Alternatively, one may compute u by solving a nonlinear program in which the decision variable a belongs to the set $\mathcal{U}^{SSM}(s)$ of control actions that satisfy the SSM conditions for that specific state, as follows:

$$u = \arg \min_{a \in \mathcal{U}^{SSM}(s)} \left(a - u_*^{DNN}(s)\right)^2 \tag{4.1}$$

If $u_*^{DNN}(s)$ already satisfies the SSM constraints, then one would get $u = u_*^{DNN}(s)$, as $\left(a - u_*^{DNN}(s)\right)^2 = 0$. This method, referred to as *safety filter*, is different from the predictive safety filter method proposed in [39]. The proposed method solves a more straightforward optimization problem to find the present action without prediction. In contrast, in [39], an NMPC problem is solved to determine a safe control action, which is why the related approach is known as *predictive safety filter*.

When safety filters are included, the above-mentioned algorithms will be called O-S-DNN, ODA-S-DNN, and DA-S-DNN, respectively; "S" stands for "safety filter." DNN layers are similar to DNN structures described in Section 3.3. The encoder presence in the algorithms above will be indicated by adding the letter "E" to their acronyms. For instance, O-SE-DNN will be the name given to the O-S-DNN version with an encoder.

4.3 Case Study

All the DNN-based controllers described above were designed and implemented for the same UR5 manipulator, which is required to reach the goal configurations outlined in Section 3.3. The robot and human have the same amount of spheres located at the same places as described in Section 3.3.

All training techniques considered (i.e., O-DNN, DA-DNN, and ODA-DNN, along with their variants with a safety filter and/or encoder, totaling 12 methods) also utilized

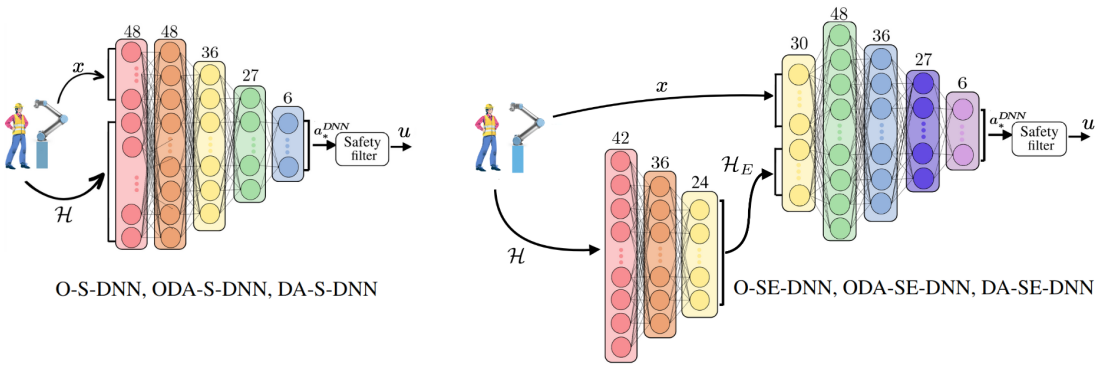

Figure 4.1: Schematics of the DNNs designed in the case study. O-S-DNN, ODA-S-DNN, and DA-S-DNN share the same structure, and the same holds for O-SE-DNN, ODA-SE-DNN, and DA-SE-DNN. In the cases where the encoder is present, it is represented as a separate neural network. The same structure applies to the cases without safety filters, in which u would be taken equal to the DNN output u_*^{DNN} (figure reprinted from [2]).

the same DNN structure as in Section 3.3, with the only difference being that the neural network output must be tested through a safety filter, when the latter is present. The sizes of the hidden layers with safety filters were established according to Fig. 4.1. The O-DNN-P data collection method is used to train the ODA-DNN since the O-DNN-P method showed better results than other O-DNN algorithms in Section 3.4.

4.4 Results and Discussion

While the initialization of DA-based techniques comprised the first $N_0 = 30$ episodes, all methods were trained for $N = 2000$ episodes. For instance, the learning curves for DA-E-DNN and ODA-E-DNN are displayed in Fig. 4.2. Since the quantity of data for each episode differs from that of ODA-E-DNN and DA-E-DNN, we do not display learning curves for O-DNN-P. The possibility of making this comparison is one of the primary motivations behind introducing ODA-based algorithms. Both techniques were trained on data from the aforementioned training set and, in parallel, run on the validation set. Fig. 4.2 illustrates that ODA-E-DNN initially converged more quickly than DA-E-DNN, most likely due to the need for DA-E-DNN to explore a larger portion of the extended state space. However, as the zoomed regions in Fig. 4.2 show, the training and validation values of the loss function for DA-E-DNN converged to lower values than their corresponding values for ODA-E-DNN. Specifically, taking into account the mean of the previous 100 episodes—which helps offset oscillations in the loss function, ODA-E-DNN demonstrated training and validation losses that were 12.4% and 18.5%, larger, respectively, than those of DA-E-DNN. This indicates that DAgger can enhance the efficacy of the offline imitation learning strategy.

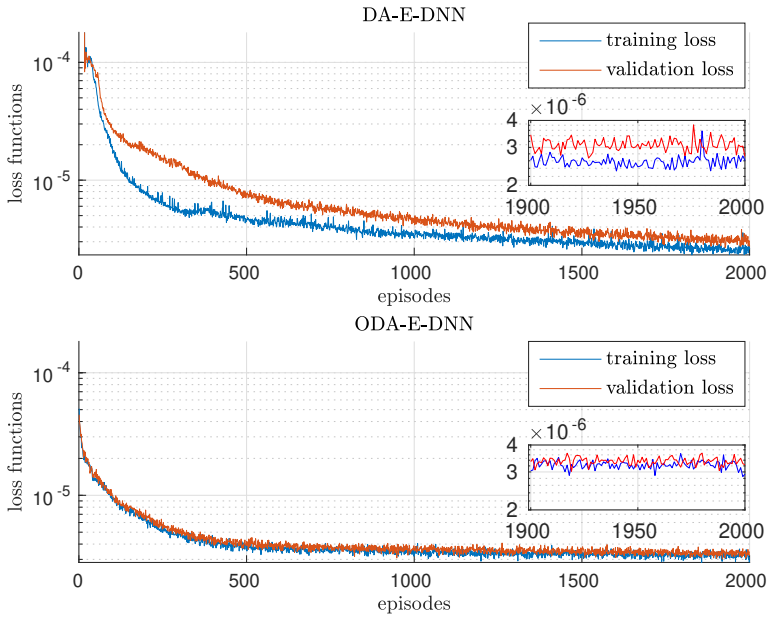

Figure 4.2: Learning curves for DA-E-DNN and ODA-E-DNN, employing a semilogarithmic scale. For the sake of simplicity, we only show the case when $x_g = x_B$ and $\|x - x_B\|_2 \geq 0.18$ rad (figure reprinted from [2]).

An open-source toolbox for algorithmic differentiation and nonlinear optimization, CasADi, was used to develop the predictive safety filter [76]. Instead of using the actual robot during training, the URsim industrial simulator was used to speed up the process and reduce the possibility of harming the robot, particularly when utilizing DA-based algorithms to explore the state space with DNN-generated regulations.

50 distinct test sets were used to evaluate the methods. To assess how the robot would respond to a partially different human behavior, the human subjects used for testing were, on average, 10 cm closer to the robot than the human subjects used for training.

The experiments are summarized in Table 4.1, where the numbers in bold indicate the best outcome (among methods with or without safety filter) for each metric under consideration. Among techniques without safety filter, DA-E-DNN and among methods with safety filter, DA-SE-DNN, produced the best values of J_{exp} and T_c. Compared to $\mathcal{H}$, $\mathcal{H}_E$ represents information more compactly, reducing the number of DNN parameters. For most methods, this probably results in an improvement in J_{exp} and an increase in computation time; however, T_c remains unaffected. Regarding T_c, DA-based approaches consistently outperformed similar O-based or ODA-based methods; the same was true for J_{exp}, except for techniques without encoder and safety filter. This demonstrates that the suggested method performs better than the offline method suggested in Chapter 3. As expected, the value of T_c achieved with DA-SE-DNN was larger than that obtained with DA-E-DNN because the safety filter perturbed the real DNN output to impose SSM limitations, limiting the robot speed. We should note that the safety filters take

approximately 1 ms to execute their calculations. Therefore, we can be confident that the slower motions of the safety algorithms are not due to the computation time of the safety filters themselves. Instead, these slower motions are designed to ensure that the robot satisfies the safety constraints. On the other hand, J_{exp} would be increased by the slower convergence to the goal configuration (via terms $(x - x_f)^T Q (x - x_g)$); the results in Table 4.1 indicate that this second effect was more prominent, as J_{exp} obtained with DA-SE-DNN was higher than that obtained with DA-E-DNN.

Notably, some approaches failed to achieve the desired configuration in at least some of the experimental trials. These approaches, including a safety filter, are indicated with an asterisk in Table 4.1. This problem may have arisen because the robot reached configurations in areas of the state space it had never encountered during training when utilizing the safety filter. This caused the DNN to provide action values with no apparent meaning. The issue solely pertained to O-based and ODA-based techniques, so it was not possible to achieve a more thorough workspace exploration because the NMPC expert generated all state trajectories. DA-based techniques, however, were not affected by this issue.

Even though DA-E-DNN appears to be the best approach thus far, the lack of a safety filter may violate SSM constraints and compromise safety. We further investigate the DA-E-DNN and DA-SE-DNN results and compare them with the imitated NMPC expert (which, when used in practice, cannot rely on precise human predictions, as when creating training data) to gain a deeper understanding of this problem.

In Figs. 4.3 and 4.4, the time evolutions of the robot joint angles and speeds, respectively, for these approaches are provided and compared. These plots show two robot motions from x_A to x_B and back. The dashed lines show two distinct robot

Table 4.1: Average values of the considered measures for 50 experiments, reprinted from [2].

	J_{exp}	T_c(s)	$\bar{\tau}$(ms)	$\hat{\tau}$(ms)
NMPC	6257.2	18.9	119	43.8
O-DNN-P	6078.5	17.9	2.30	0.78
ODA-DNN	5680.8	17.1	**1.90**	**0.73**
DA-DNN	5850.5	16.5	3.00	0.81
O-E-DNN-P	5793.9	18.2	6.50	1.35
ODA-E-DNN	5429.7	17.5	6.20	1.30
DA-E-DNN	**5378.7**	**15.3**	29.4	1.40
O-S-DNN-P*	6913.6	19.8	2.30	0.70
ODA-S-DNN*	7254.0	19.4	3.60	0.67
DA-S-DNN	5875.2	16.8	**2.20**	**0.72**
O-SE-DNN-P*	6973.8	20.4	29.7	1.55
ODA-SE-DNN*	6793.6	19.2	31.1	1.60
DA-SE-DNN	**5722.8**	**16.0**	32.4	1.06

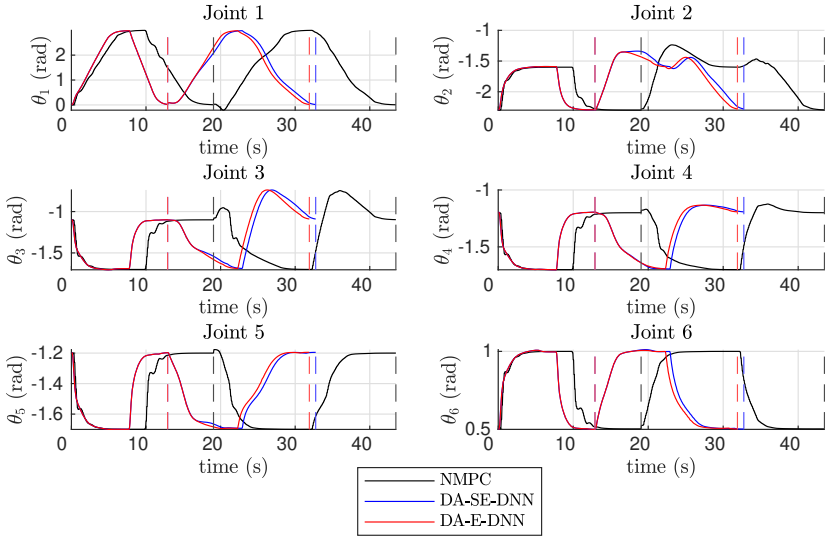

Figure 4.3: Time evolution of joint positions x_i for NMPC, DA-E-DNN and DA-SE-DNN. The vertical dashed lines denote the time instants at which the joint position depicted by the particular color accomplishes the process from x_A to x_B and back (figure reprinted from [2]).

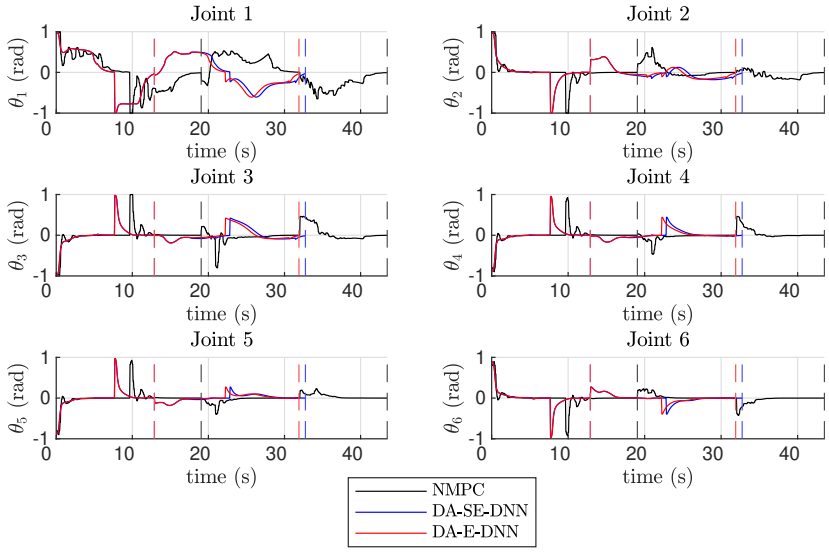

Figure 4.4: Time evolution of joint speeds u_i for NMPC, DA-E-DNN and DA-SE-DNN, similarly to the joint positions illustrated in Fig. 4.3. The vertical dashed lines denote the time instants at which the joint position depicted by the particular color accomplishes the process from x_A to x_B and back (figure reprinted from [2]).

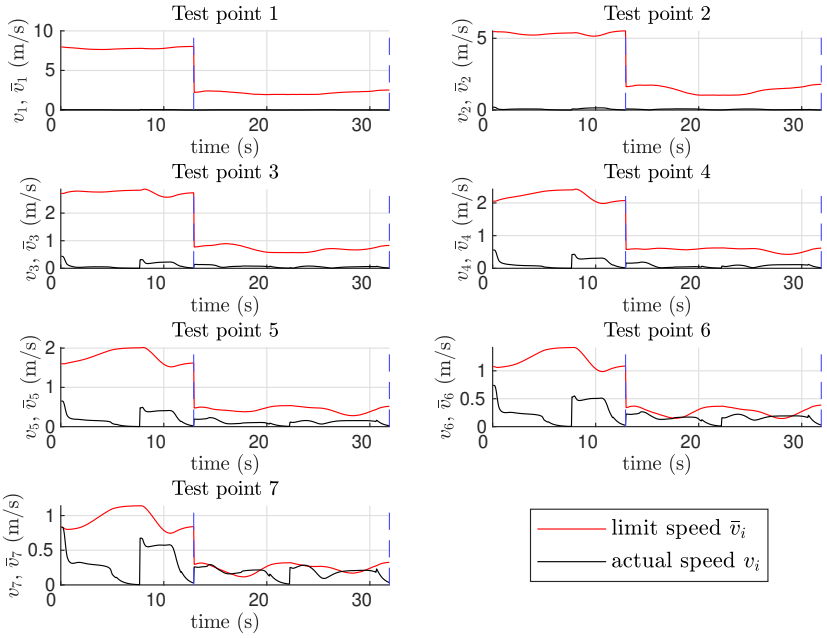

Figure 4.5: Time evolution of linear velocities of robot test points v_i for the DA-E-DNN method, together with a corresponding limit speed $\bar{v}_i$, for the robot motions shown in Figs.4.3 and 4.4. The vertical, dashed blue lines indicate task completion (figure reprinted from [2]).

actions: the first involves the human working away from the robot, while the second consists of the person moving near the robot. Regarding T_c, Fig. 4.3 supports the results of Table 4.1: both DNN-based techniques produced faster motions than NMPC, with DA-E-DNN surpassing DA-SE-DNN. Fig. 4.4 illustrates how similar the speed value range is for all three approaches, indicating that each DNN was appropriately trained to replicate the behavior of NMPC.

The time evolution of the test point speed, where the SSM constraints are used to impose safety, is displayed in Figs 4.5 and 4.6 for DA-E-DNN and DA-SE-DNN, respectively, along with the corresponding SSM limits. Fig. 4.5 illustrates that during the second motion, when the human comes closer to the robot, DA-E-DNN violates the SSM constraints for test points 6 and 7. Meanwhile, due to the presence of the safety filter, DA-SE-DNN satisfies the same constraints.

In conclusion, DA-SE-DNN offers the best trade-off between safety and performance.

4.5 Chapter summary

In this chapter, the DAgger imitation learning approach was developed and compared against offline imitation learning algorithms. Minor modifications were also made to offline-designed algorithms to facilitate a direct comparison with the DAgger approach. Safety filters, were also designed and implemented. Experimental results indicate that

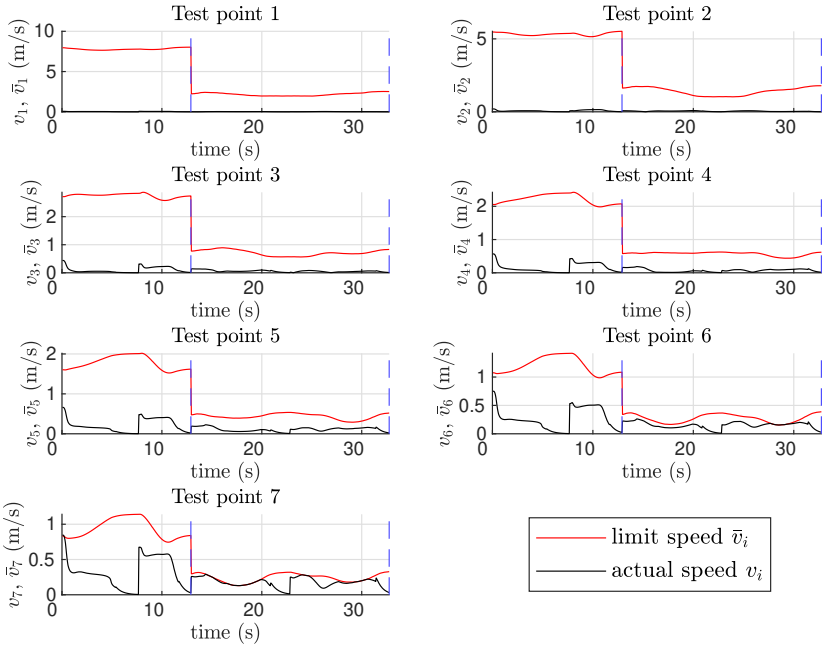

Figure 4.6: Time evolution of linear velocities of robot test points v_i for the DA-SE-DNN method, together with a corresponding limit speed $\bar{v}_i$, for the robot motions shown in Figs.4.3 and 4.4. The vertical, dashed blue lines indicate task completion (figure reprinted from [2]).

the DAgger approach outperforms other imitation learning strategies that mimic NMPC, and NMPC itself. We can conclude that the DAgger approach can approximate NMPC solutions while enabling efficient and safe interactions in pHRI tasks. In this section, we also identified a weakness of DNN-based approaches: they cannot guarantee the stability of the system. As highlighted in the results, some offline imitation learning approaches failed to bring the robot to its goal configuration in certain cases. However, all the algorithms developed using the DAgger approach successfully brought the robot to its goal configuration. Overall, the task completion time is decreased by approximately 19% using the DA-E-DNN approach compared to NMPC, and by 15.34% using the DA-SE-DNN approach. Nonetheless, the stability of the system in DNN-based approaches is not guaranteed, which will be a focus of our future work.

In the subsequent chapter, further analysis of NMPC will be conducted to design better SSM-based controllers, mainly aimed at enlarging the domain of attraction.

BLANK

Chapter 5

Nonlinear MPC with set terminal constraint

Chapters 3 and 4 have described how the NMPC law introduced in Chapter 2 can be approximated using DNNs, and how this can lead to a performance improvement. The present chapter,[1] instead, focuses on improving performance by directly modifying the control law described in Chapter 2. Specifically, the terminal constraint requesting that the robot be at the goal configuration at the end of the prediction horizon is substituted with a less conservative terminal constraint, in which the robot configuration is allowed to belong to an ellipsoidal set. The expected result is an enlargement of the domain of attraction, i.e., the set of initial values from which the robot configuration will converge to the goal point, satisfying all the imposed constraints.

The theoretical formulation of the proposed NMPC algorithm is presented in Section 5.1, together with the related stability proofs and a description of the procedure to determine the parameters of the NMPC problem. Section 5.2 describes the case study, whose related experimental results are presented and discussed in Section 5.3.

5.1 Theoretical formulation

5.1.1 NMPC problem formulation

In this section, we will focus on improving the NMPC approach by expanding the domain of attraction $\mathbb{X}_F$. The domain of attraction refers to the set of initial points from which the robot configuration can converge to the goal configuration while satisfying all imposed constraints, due to recursive feasibility and closed-loop stability. In the control law discussed in Section 2.4), it was imposed that the configuration coincides with the goal point at the end of the prediction horizon, ensuring that the system asymptotically converges to the goal configuration. With the new approach, we introduce a terminal set $\mathbb{X}_T \subseteq \mathbb{X}$ that contains the goal at its center. The robot is required to reach inside this set by the end of the prediction horizon. This approach allows the robot to reach the goal configuration from a broader range of initial points, thereby increasing the domain of attraction.

[1] Part of the text in this chapter is adapted from [3], conditionally accepted for publication in Control Engineering Practice (Elsevier).

To improve the domain of attraction, compared to the control law obtained from (2.21), it is now imposed that the state at the end of the prediction horizon belongs to a terminal set containing x_g, rather than exactly coinciding with x_g. As a consequence, the new set of admissible sequences is defined by further constraining $u(\cdot) \in \mathbb{U}^N(x_0)$ (defined in Section 2.4) to include condition $x_u(N) \in \mathbb{X}_T$ instead of the point terminal constraint used in (2.21); we denote these admissible sequences as $u(\cdot) \in \mathbb{U}^N_{\mathbb{X}_T}(x_0)$. A terminal cost $F(\cdot) : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ has also to be added to the cost function.

To determine the expressions of $\mathbb{X}_T$ and $F(x)$, we will follow the instructions provided in [22, Rem. 5.15], which can generally be used to prove the stability and recursive feasibility of NMPC with set terminal constraints. The main idea of the remark is generally focused on finding the ellipsoid around the goal configuration. Following these instructions, we introduce $P \in \mathbb{R}^{n\times n}$ as the only symmetric and positive definite solution of the discrete-time algebraic Riccati equation

$$P = A'PA - (A'PB)(R + B'PB)^{-1}(B'PA) + Q. \tag{5.1}$$

In (5.1), Q and R are the same weight matrices introduced in (2.20), whereas the stabilizable pair (A, B) summarizes the linear system dynamics (2.2), and is thus defined by $A \triangleq \mathbb{I}_n$ and $B = T\mathbb{I}_n$. Setting $\sigma \in \mathbb{R}_{>1}$, one can determine a suitable value for another parameter $\eta \in \mathbb{R}_{>0}$, ideally as large as possible (the procedure to determine it will be explained in Section 5.1.3), and define

$$\mathbb{X}_T \triangleq \{x \in \mathbb{R}^n \ : \ (x - x_g)'P(x - x_g) \leq \eta\}, \tag{5.2}$$

$$F(x) \triangleq \sigma(x - x_g)'P(x - x_g). \tag{5.3}$$

Using these formulations for $\mathbb{X}_T$ and $F(x)$, the optimal control problem takes the following form:

$$\text{minimize } J_{NF}(x_0, u(\cdot)) \triangleq \sum_{k=0}^{N-1} \ell(x_u(k), u(k)) + F(x_u(N)) \tag{5.4a}$$

$$\text{with respect to } u(\cdot) \in \mathbb{U}^N_{\mathbb{X}_T}(x_0) \tag{5.4b}$$

$$\text{subj. to } x_u(0) = x_0,\ x_u(k+1) = f(x_u(k), u(k)). \tag{5.4c}$$

As in the case of the NPMC algorithm described in Chapter 2, the value of $\mathcal{H}$ measured at the current instant n is used for all time instants in (5.4). The optimal sequence of control moves related to (5.4) is denoted again by $u^*(\cdot) = \{u^*(0), \ldots, u^*(N-1)\}$, and the optimal cost by $J^*_{NF}(x_0)$. As in the point terminal constraint case, we apply the receding horizon principle by setting $u(t) = u^*(0)$ as the closed-loop NMPC policy.

5.1.2 Proving closed-loop stability

The following lemma provides a result that will be instrumental in proving the subsequent theorem.

Lemma 5.1. *Given $\zeta \in \mathbb{R}_{(0,1)}$, there exists $\delta \in \mathbb{R}_{>0}$, such that $\|x\|_{x_g} \leq \delta$ implies*

$$\zeta \ell(x, u_a(x)) \geq \gamma \varphi^2(x, x_g), \tag{5.5}$$

where

$$u_a(x) = -K(x - x_g), \tag{5.6}$$

in which $K \triangleq (R+B'PB)^{-1}B'PA$ is the gain associated to the discrete-time algebraic Riccati equation (5.1)*.*

Proof. The proof will be made of three steps. As a first step, we prove that, given $D \in \mathbb{R}_{>0}$, one can find $\delta_1 \in \mathbb{R}_{>0}$ such that $\|x\|_{x_g} \leq \delta_1$ implies

$$\gamma \varphi^2(x, x_g) \leq D\|x\|_{x_g}^3. \tag{5.7}$$

Given the existence of the upper bound $\Phi(\|x\|_{x_g})$ as $\gamma\varphi^2(x, x_g)$ defined in (2.40), a sufficient condition that implies (5.7) is

$$\psi(y) \triangleq Dy^3 - \Phi(y) \geq 0 \tag{5.8}$$

where $y \triangleq \|x\|_{x_g}$. By calculating the Taylor series of $\psi(y) : \mathbb{R} \to \mathbb{R}$ around $y = 0$, one would obtain $\psi(y) = Dy^3 + \mathcal{O}(y^4)$. This implies that, for a sufficiently small value of y (namely, $y \leq \delta_1$ for some $\delta_1 \in \mathbb{R}_{>0}$), (5.7) holds.

As a second step, we show that, for any given $\zeta \in \mathbb{R}_{(0,1)}$ and $D \in \mathbb{R}_{>0}$, one can find $\delta_2 \in \mathbb{R}_{>0}$ such that

$$D\|x\|_{x_g}^3 \leq \zeta \ell(x, u_a(x)) \tag{5.9}$$

for $\|x\|_{x_g} \leq \delta_2$. Given the expression of $\ell(x, u_a(x))$ in (2.20) and the formulation of the auxiliary control law in (5.6), the following holds true for all $x \in \mathbb{R}^n$:

$$\ell(x, u_a(x)) \geq \lambda_{\min}(\tilde{Q})\|x\|_{x_g}^2, \tag{5.10}$$

with $\tilde{Q} = Q + K'RK$. Therefore, (5.9) is implied by the more conservative condition

$$D\|x\|_{x_g}^3 \leq \zeta \lambda_{\min}(\tilde{Q})\|x\|_{x_g}^2 \tag{5.11}$$

which, excluding the trivial case $x = x_g$ for which it is always satisfied, is equivalent to $D\|x\|_{x_g} \leq \zeta\lambda_{\min}(\tilde{Q})$, or $\|x\|_{x_g} \leq \delta_2$, with $\delta_2 \triangleq \zeta\lambda_{\min}(\tilde{Q})/D$.

The third and last step consists of showing that, given any $D \in \mathbb{R}_{>0}$ and $\zeta \in \mathbb{R}_{(0,1)}$, one can find $\delta \in \mathbb{R}_{>0}$ such that

$$\gamma\varphi^2(x, x_g) \leq D\|x\|_{x_g}^3 \leq \zeta\ell(x, u_a(x)) \tag{5.12}$$

for $\|x\|_{x_g} \leq \delta$. This is obtained by merging the results of the first two steps, choosing $\delta = \min\{\delta_1, \delta_2\}$. As this result holds for any $D \in \mathbb{R}_{>0}$, the existence of δ such that inequality (5.5) holds for $\|x\|_{x_g} \leq \delta$ is proven. ■

The following theorem proves the stability properties of the NMPC approach with a set terminal constraint.

Theorem 5.2. *Consider the closed-loop system defined by* (2.2) *with input* $u(t)$ *determined via the receding-horizon principle from the solution of the NMPC problem* (5.4)*, whose constraints are formulated in* (2.16)-(2.17)*, and the stage cost is introduced in* (2.20)*. Then, if Assumption 2.1 holds, for any given* $\sigma \in \mathbb{R}_{>1}$ *there exists* $\eta \in \mathbb{R}_{>0}$ *such that the set* $\mathbb{X}_F$ *of states for which a solution of* (5.4) *exists is a positively invariant set, and* x_g *is an asymptotically stable equilibrium point for the above-mentioned closed-loop system with domain of attraction equal to* $\mathbb{X}_F$*.*

Proof. As for recursive feasibility, from [22, Lem. 5.11], we can claim that $\mathbb{X}_F$ is positively invariant for the above-mentioned closed-loop system if $\mathbb{X}_T \subseteq \mathbb{X}$ and there exists an auxiliary control law $u_a(x) \in \mathbb{U}(x)$ such that $x \in \mathbb{X}_T \Rightarrow f(x, u_a(x)) \in \mathbb{X}_T$. In our formulation, as auxiliary control law, we choose $u_a(x)$ already defined in (5.6). The value of P used to obtain the gain matrix K is related to the solution of the infinite-horizon linear-quadratic optimal control problem,

$$\text{minimize } J_\infty(x_0, u(\cdot)) \triangleq \sum_{k=0}^{\infty} \|x_u(k) - x_g\|_Q^2 + \|u(k)\|_R^2 \tag{5.13a}$$

$$\text{with respect to } u(\cdot) \in (\mathbb{R}^n)^\infty \tag{5.13b}$$

$$\text{subj. to } x_u(0) = x_0,\ x_u(k+1) = f(x_u(k), u(k)), \tag{5.13c}$$

where $(\mathbb{R}^n)^\infty$ represents the infinite-length sequences of unconstrained control inputs. Indeed, the minimum of problem (5.13) is given by

$$J_\infty^*(x_0) \triangleq \|x_0 - x_g\|_P^2, \tag{5.14}$$

and the associated optimal feedback control law is equal to $u_a(x)$, as defined in (5.6). Based on the fundamental concept of infinite-horizon dynamic programming, we know that, for a generic state value $x \in \mathbb{R}^n$,

$$J_\infty^*(x) = \|x - x_g\|_Q^2 + \|u_a(x)\|_R^2 + J_\infty^*(f(x, u_a(x))), \tag{5.15}$$

where $f(x, u_a(x)) = Ax + Bu_a(x)$. After noticing from (2.20) that

$$\|x - x_g\|_Q^2 + \|u_a(x)\|_R^2 = \ell(x, u_a(x)) - \gamma\varphi^2(x, x_g), \tag{5.16}$$

equation (5.15) can be rewritten as

$$J_\infty^*(f(x, u_a(x))) = J_\infty^*(x) - \ell(x, u_a(x)) + \gamma\varphi^2(x, x_g). \tag{5.17}$$

Applying the result of Lemma 5.1 to equation (5.17) we obtain that, for all $\sigma \in \mathbb{R}_{>1}$ – notice that for any $\zeta \in \mathbb{R}_{(0,1)}$ one can find a corresponding constant $\sigma \triangleq 1/(1-\zeta) \in \mathbb{R}_{>1}$, and vice versa – there exists $\delta \in \mathbb{R}_{>0}$ such that $\|x\|_{x_g} \leq \delta$ implies

$$-\ell(x, u_a(x)) + \gamma\varphi^2(x, x_g) \leq -\frac{1}{\sigma}\ell(x, u_a(x)) \tag{5.18}$$

and thus, from (5.17),

$$J_\infty^*(f(x,u_a(x))) - J_\infty^*(x) \leq -\frac{1}{\sigma}\ell(x,u_a(x)) \tag{5.19}$$

for all $\|x\|_{x_g} \leq \delta$. Exploiting the fact that $J_\infty^*(x) \geq \lambda_{\min}(P)\|x\|_{x_g}^2$ – remember that $P \succ 0$ and thus $\lambda_{\min}(P) > 0$ – the largest sublevel set of $J_\infty^*(x)$ contained in the ball defined by $\|x\|_{x_g} \leq \delta$ can be found as the set of all, $x \in \mathbb{R}^n$ such that

$$J_\infty^*(x) \leq \delta^2 \lambda_{\min}(P). \tag{5.20}$$

The terminal set $\mathbb{X}_T$ introduced in (5.2) is also defined as a sublevel set of $J_\infty^*(x)$. We define the value of η in (5.2) as the largest real number in $(0, \delta^2\lambda_{\min}(P)]$ such that $\mathbb{X}_T \subseteq \mathbb{X}$ and $x \in \mathbb{X}_T \Rightarrow u_a(x) \in \mathbb{U}(x)$. The condition $\eta \leq \delta^2\lambda_{\min}(P)$ guarantees that (5.19) holds for all $x \in \mathbb{X}_T$, and thus $x \in \mathbb{X}_T \Rightarrow f(x,u_a(x)) \in \mathbb{X}_T$, whereas $\mathbb{X}_T \subseteq \mathbb{X}$ and $x \in \mathbb{X}_T \Rightarrow u_a(x) \in \mathbb{U}(x)$ are directly required by [22, Lem. 5.11], as mentioned at the beginning of the proof. Notice that, as $x_g \in \mathrm{int}(\mathbb{X})$ and $u_g \in \mathrm{int}(\mathbb{U}_{\mathrm{SSM}}(x_g))$ (see the considerations in the last paragraph of Section 2.2), then it is always possible to determine η as a strictly positive value. This passage completes the part of the proof on recursive feasibility.

Regarding closed-loop stability, according to [22, Thm. 5.13], the above-mentioned closed-loop system is asymptotically stable if, in addition to satisfying the requirements for recursive feasibility, the following conditions hold:

(a) the auxiliary control law $u_a(x) \in \mathbb{U}(x)$ already defined in (5.6) guarantees $F\left(f(x,u_a(x))\right) + \ell(x,u_a(x)) \leq F(x)$;

(b) there exist $\mathcal{K}_\infty$-functions $\alpha_1(\cdot)$, $\alpha_2(\cdot)$ and $\alpha_3(\cdot)$ such that

$$\alpha_1(\|x\|_{x_g}) \leq J_{NF}^*(x) \leq \alpha_2(\|x\|_{x_g}) \tag{5.21}$$

and $\ell(x,u) \geq \alpha_3(\|x\|_{x_g})$ for all $x \in \mathbb{X}_F$ and all $u \in \mathbb{R}^n$.

To prove condition (a), we first notice that $F(x)$ in (5.3) can be written as $\sigma J_\infty^*(x)$. Thus, $F\left(f(x,u_a(x))\right) + \ell(x,u_a(x)) \leq F(x)$ is immediately satisfied, as it is equivalent to (5.19). To prove (b), given the expressions of $\ell(x,u)$ in (2.20) and $J_{NF}^*(x)$ in (5.4a), $\alpha_1(\|x\|_{x_g}) = \alpha_3(\|x\|_{x_g}) = \lambda_{\min}(Q)\|x\|_{x_g}^2$ satisfy the required conditions. According to [22, Prop. 5.14(ii)] $\alpha_2(\cdot)$ in (5.21) exists if condition (a) above is satisfied, and in addition, the following holds:

(i) the dynamics $f(x,u)$, the stage cost $\ell(x,u)$ is continuous with respect to both x and u, for all $x, u \in \mathbb{R}^n$, and the terminal cost $F(x)$ is continuous with respect to x for all $x \in \mathbb{X}_T$;

(ii) Ω introduced in (2.7) is a compact set;

(iii) $\mathbb{X}_T$ contains a ball $\mathcal{B}_\nu(x_g)$ with $\nu \in \mathbb{R}_{>0}$;

(iv) there exists a function $\tilde{\alpha}_2(\cdot) \in \mathcal{K}_\infty$ such that $F(x) \leq \tilde{\alpha}_2(|x|_{x_g})$.

We begin by proving (i). The continuity of the dynamics $f(x,u)$ and of the stage cost $\ell(x,u)$ with respect to both x and u was proven in part (i) of Lemma 2.1. The terminal cost given in (5.3) is a quadratic function of x, which ensures its continuity for all $x \in \mathbb{R}^n \supset \mathbb{X}_T$. Condition (ii) was also already proven, and precisely in part (ii) of Lemma 2.1. Condition (iii) is satisfied by choosing,

$$\nu = \sqrt{\frac{\eta}{\sigma \lambda_{\max}(P)}}, \tag{5.22}$$

noticing that $\eta, \sigma, \lambda_{\max}(P) > 0$. Indeed, with this definition of ν, $x \in \mathcal{B}_\nu(x_g)$, or equivalently $\|x\|_{x_g} \leq \nu$, implies $\sigma \|x - x_g\|_P^2 \leq \sigma \lambda_{\max}(P) \|x\|_{x_g}^2 \leq \eta$, i.e., $x \in \mathbb{X}_T$. Finally, (iv) is satisfied by noticing that $\sigma x' P x \leq \sigma \lambda_{max}(P) \|x\|_{x_g}^2 \triangleq \tilde{\alpha}_2(\|x\|_{x_g})$. ■

5.1.3 Determining the value of η

Theorem 5.2 states the existence of $\eta \in \mathbb{R}_{>0}$, but it does not provide an explicit procedure to determine its value. To determine it, the following steps – based on the derivations in the proofs of Lemma 5.1 and Theorem 5.2 – can be followed. Firstly, the designer of the control law has to fix a value of $\sigma \in \mathbb{R}_{>1}$, as already stated in Theorem 5.2. Afterward, a suitable value of δ can be determined based on condition (5.5).

Even if this value cannot be obtained analytically, for a fixed value of p_{h,j_γ} one can consider an equally-spaced sequence of n_δ values δ_i (with $i \in \mathbb{N}_{[1,n_\delta]}$) in the interval $[0, \delta_{ub}]$, for a given $\delta_{ub} \in \mathbb{R}_{>0}$. For each value of this sequence, one can generate a countable set Δ_i, made of a large number of randomly-generated values of x, all defined such that $\|x\|_{x_g} = \delta_i$. The estimate of δ is thus chosen as the largest δ_i such that (5.5) is satisfied for all $x \in \Delta_j$, for all $j \in \mathbb{N}_{[1,i]}$.

(an illustrative example of this procedure will be shown in the considered case study). The obtained value of δ is then used to determine the upper bound on η, equal to $\delta^2 \lambda_{\min}(P)$, discussed in the proof of Theorem 5.2. In the same proof, it is specified that the obtained value of η should be set such that $\mathbb{X}_T \subseteq \mathbb{X}$ and $x \in \mathbb{X}_T \Rightarrow u_a(x) \in \mathbb{U}(x)$. As all of these constraints are purely functions of x, we can express each inequality that composes them as $g_i(x) \leq 0$, for $i \in \mathbb{R}_{[1,n_c]}$, n_c being the total number of inequality constraints. Let us refer to the largest ellipsoid satisfying the single constraint. $g_i(x) \leq 0$ for all its points as $\mathbb{X}_{T,i} \triangleq \{x \in \mathbb{R}^n \ : \ (x - x_g)' P (x - x_g) \leq \eta_i\}$, with $\eta_i \in \mathbb{R}_{>0}$. This ellipsoid can be determined by numerically solving the following optimization problem:

$$\text{minimize } (x - x_g)' P (x - x_g) \tag{5.23a}$$

$$\text{with respect to } x \in \mathbb{R}^n \tag{5.23b}$$

$$\text{subj. to } g_i(x) \leq 0. \tag{5.23c}$$

Once the minimizer, namely, $x^*_{(i)}$, is found, the corresponding minimum is $\eta_i \triangleq (x^*_{(i)} - x_g)' P (x^*_{(i)} - x_g)$. The desired value of η that satisfies all constraints is then obtained as $\eta = \min\{\delta^2 \lambda_{\min}(P), \eta_i\}$, for $i \in \mathbb{R}_{[1,n_c]}$. Notice that the optimization problem (5.23) is a quadratic program when simple bounds on joint positions and speeds are imposed by $g_i(x)$, and becomes a nonlinear program in more complex cases, such as those deriving from SSM constraints.

5.2 Case Study

In the case study, the method proposed in Chapter 2, as well as the NMPC with Set Terminal Constraint described in Chapter 5, were applied to the UR5 robot (see, Fig. **??**). This is an identical robot model used in the experiments presented in Section 3.3 and Section 4.3. According to these sections, the identical spheres located at the same points to cover both the robot and the human were utilized.

With reference to Section 2.1, the system dynamics in (2.1)-(2.2) was defined with $T = 250$ ms. The constraints for the NMPC problems described in Section 2.2 were formulated as follows. Each joint angle (see (2.6)) was limited in the range $\pm 2\pi$ rad, resulting in $\Theta = \{x \in \mathbb{R}^n \ : \ \|x\|_\infty \leq 2\pi\}$. Also, the joint speeds (see (2.7)) were limited in the range ± 1.2 rad/s, resulting in $\Omega = \{u \in \mathbb{R}^n \ : \ \|u\|_\infty \leq 1.2\}$. Setting the goal configuration (in radians) as

$$x_g = \begin{bmatrix} 0.000 & -2.000 & -1.220 & -1.518 & -1.588 & 0.500 \end{bmatrix}',$$

one can notice that Θ and Ω include x_g and u_g, respectively, in their interiors, as required. The inequality constraints defined in (2.8) are imposed to ensure that the robot remains above the table. Specifically, it is imposed that the z-coordinate of each robot test point $p_{r,i}$, for $i = \mathbb{N}_{[1,7]}$ be above the table level of a quantity equal to the corresponding sphere radius $R_{r,i}$. It was verified that when at the goal configuration x_g, (2.9) holds. Based on the robot characteristics and on an estimated maximum human speed $\bar{v}_H = 2$ m/s (value used in the ISO/TS 15066 standard [12]), the $n_r \times n_h = 98$ SSM constraints (2.13) were defined with $\alpha = 0.15$ and $\bar{d} = 0.29$. The reader is referred to [4] if interested in more details on how these parameters can be determined. We considered a fixed position $\mathcal{H}$ of the human operator acquired from the human motion dataset presented in [68], defined such that Assumption 2.1 holds.

Regarding the NMPC cost function discussed in Section 2.3, the stage cost (2.20) was defined as in the previous chapters, with $Q = 10 \cdot \mathbb{I}_6$, $R = \mathbb{I}_6$, $\gamma = 500$ and $\beta = 3$.

The above-mentioned parameters are sufficient to design the NMPC law with point terminal constraint described in Section 2.5. Instead, designing the NMPC law with the

set terminal constraint described in this chapter requires defining $\mathbb{X}_T$ and $F(x)$. The latter was defined with $P = 68.4429 \cdot \mathbb{I}_6$, resulting from the corresponding discrete-time algebraic Riccati equation (5.1), and $\sigma = 3$, to which corresponds $\zeta = 0.67$. Also, the control gain corresponding to P used to define $u_a(x)$ was equal to $K = 2.9221 \cdot \mathbb{I}_6$, implying $\tilde{Q} = 18.5389 \cdot \mathbb{I}_6$, as defined in (5.10). Concerning Remark 5.1.3, the value of η depends on the human position h_v, which influences the SSM constraints. In order to take the most conservative case into account, η was calculated based on the value of h_v with the robot in its goal configuration.

To obtain the actual value of η to determine the expression of $\mathbb{X}_T$, one optimization problem in form (5.23) was solved using the CasADI tool [76] for each inequality constraint $g_i(x)$ defining sets $\mathbb{X}$ and $\mathbb{U}$, thus obtaining the corresponding value of η_i. Following the procedure detailed in Section 5.1.3, we set $\eta = 2.51283$. The numerical optimal control problems (2.21) and (5.4) were both solved by generating C-code routines using the *acados* toolbox [72]. As a consequence, the value of δ_{ub} could be chosen arbitrarily small, as long as it satisfied $\delta_{ub}^2 \lambda_{\min}(P) \geq 2.51283$, which led to setting $\delta_{ub} = 0.19161$. Always following the guidelines expressed in Subsection 5.1.3, the number of equally-spaced points considered in the interval $[0, \delta_{ub}]$, with a set spacing interval of $2 \cdot 10^{-4}$, was equal to $n_\delta = 2128$, and each set Δ_i was generated containing 10^5 random values. For all the 212.8 million considered values of x, condition 5.5 was satisfied, implying that $\eta = \min_{i \in [1, n_c]} \eta_i = 2.5128$.

5.3 Results and discussion

The main reason for designing an NMPC controller with a set terminal constraint was to enlarge the domain of attraction $\mathbb{X}_F$ compared to the point terminal constraint case. The evaluation of the domain of attraction of a closed-loop system with NMPC is usually performed by solving the NMPC problem for a set of initial conditions defined in a grid in the state space. In our case study, visualizing $\mathbb{X}_F$ would have been impossible, as the state space is six-dimensional in our case study. However, as the positions of the first joints influence the overall robot configuration more than that of the last joints, we decided to show a graphical evaluation of $\mathbb{X}_F$ for a grid of initial conditions of $x_1 \in [-2\pi, 2\pi]$ and $x_2 \in [-\pi, 0]$ (where x_i represents the i-th component of the vector x). The initial value of the subsequent components x_i, for $i \in \mathbb{N}_{[3,6]}$, was instead set equal to the corresponding component of x_g. The result is reported in Fig. 5.1, varying the prediction horizon from $N = 10$ down to $N = 1$. In Fig. 5.1, the black dot represents the first two coordinates of x_g, while the green points represent values of the state from which a feasible solution could be found for both NMPC formulations. At the yellow points, the solution could be found only for the set terminal constraint, while no solutions could be found for the red points. We only show values

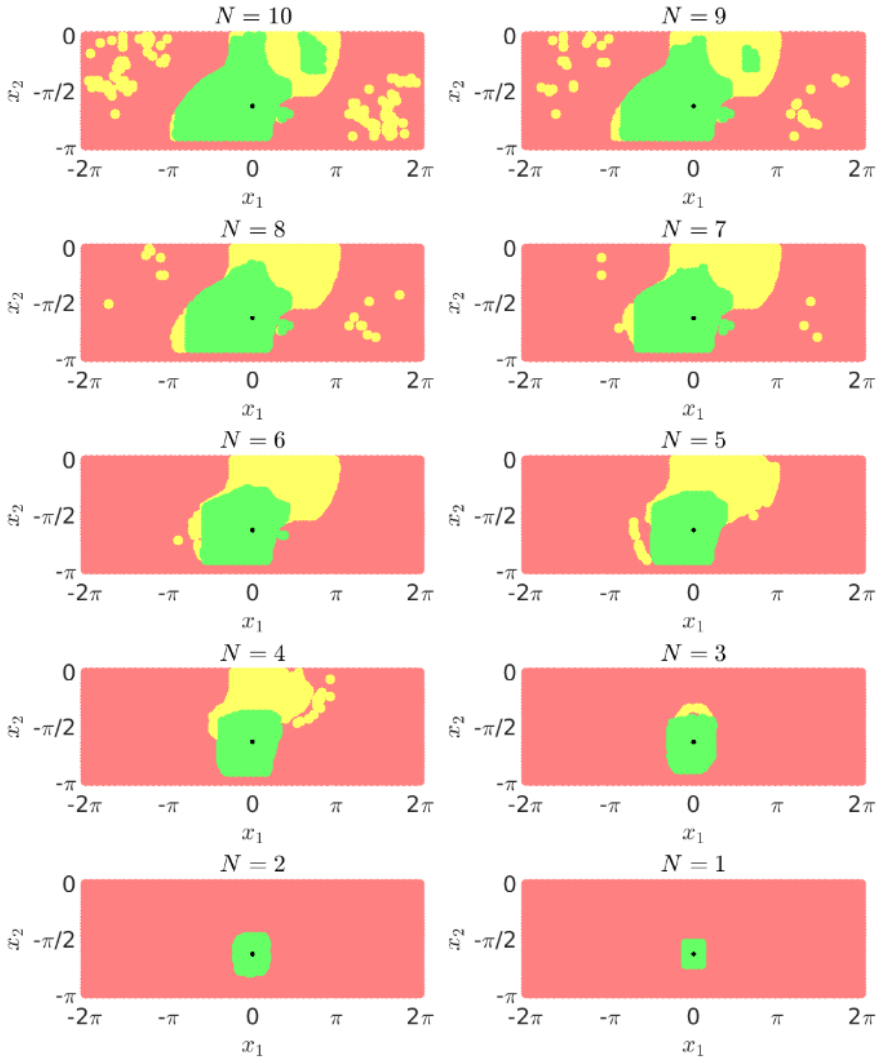

Figure 5.1: Evaluation, for the first two coordinates of the state vector, of the domain of attraction $\mathbb{X}_F$ for $N \in \mathbb{N}_{1,10}$, in the case of both set and point terminal constraint (green), set terminal constraint only (yellow). The red areas represent initial conditions from which no feasible solution to the NMPC problem could be determined for either NMPC formulation.

of $x_{\in}[-\pi, 0]$, as outside this interval, no feasible solutions could be found, mainly due to the immediate violation of the constraint requiring the whole robot frame to remain above the table. Even when the prediction horizon N is reduced, the controller with the set terminal constraint retains a larger domain of attraction than the controller with the point terminal constraint. It is worth mentioning that the fact that $\mathbb{X}_F$ for the set terminal constraint case seems disconnected is probably because these images only represent a two-dimensional "slice" of a six-dimensional space and also to the presence of several nonlinear and non-convex inequality constraints.

Table 5.1 provides detailed information about how the domain of attraction for both controllers, STC and PTC, increases as the prediction horizon length (N) increases. The table also shows the percentage improvement in the area related to the STC

domain of attraction (green and yellow portions in Fig. 5.1) compared to that related to the PTC domain of attraction (green portions). Notably, there is no percentage improvement when the prediction horizon length is less than 2, but the improvement remains significant as N increases (e.g., 50.54% for $N = 10$).

After evaluating the domain of attraction, we show the results of experiments with the actual UR5 robot. The experiments were conducted using the same used for the experiments described in the previous chapters, with an i7-9750H CPU, 16GB RAM and NVIDIA GeForce GTX 1660Ti using the Robot Operating System (ROS) environment. The control values, represented by u, were provided with the above-mentioned sampling interval $T = 250$ ms as a reference to two lower-level control loops, which acted with a shorter sampling interval of 50 ms and 1 ms, respectively. The internal low-level controller receives the velocity values generated by the NMPC controller and converts them into the necessary torque values to control the physical robot in real-time. Experiments were run starting from the initial configuration

$$x_0 = \begin{bmatrix} 2.619 & -0.958 & -1.220 & -1.518 & -1.588 & 0.500 \end{bmatrix}$$

(which is one of those visualized in Fig. 5.1, as its last four coordinates coincide with those of x_g) to x_g, with the human operator holding the pose $\mathcal{H}$ mentioned above. From this initial configuration, a feasible solution existed for both point terminal constraint and set terminal constraint cases. Therefore, we ran one experiment in each case to compare the performance. As a solution existed in both cases, we did not expect the set terminal constraint formulation to show any improved results compared to the point terminal constraint one. However, if N was reduced to 5 from the same initial condition, the NMPC problem with set terminal constraint was feasible, whereas that with point terminal constraint was not. Reducing the prediction horizon also reduced the complexity of the nonlinear program to be solved, and thus decreased the computation time needed to find a solution. This, in turn, reduced the computational

Table 5.1: Evaluation of the domain of attraction corresponding to Fig.5.1

N	STC	PTC	$\%$ improvement
10	181.06	120.27	50.54
9	173.27	108.08	60.32
8	168.94	99.29	70.15
7	157.32	90.12	74.57
6	135.21	78.19	72.93
5	111.66	65.50	70.47
3	68.52	50.56	35.52
3	34.23	33.35	2.63
2	17.21	17.21	0
1	6.28	6.28	0

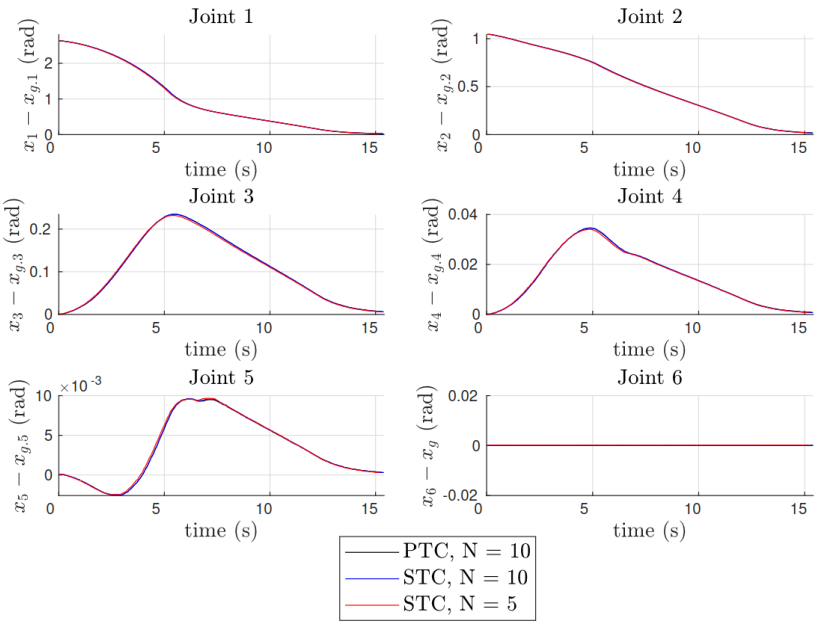

Figure 5.2: Time evolution of the difference between each component x_i of the state vector and the corresponding component $x_{g,i}$ of the goal configuration, for $i = \mathbb{N}_{[1,6]}$, for the three NMPC laws implemented in the experimental setup.

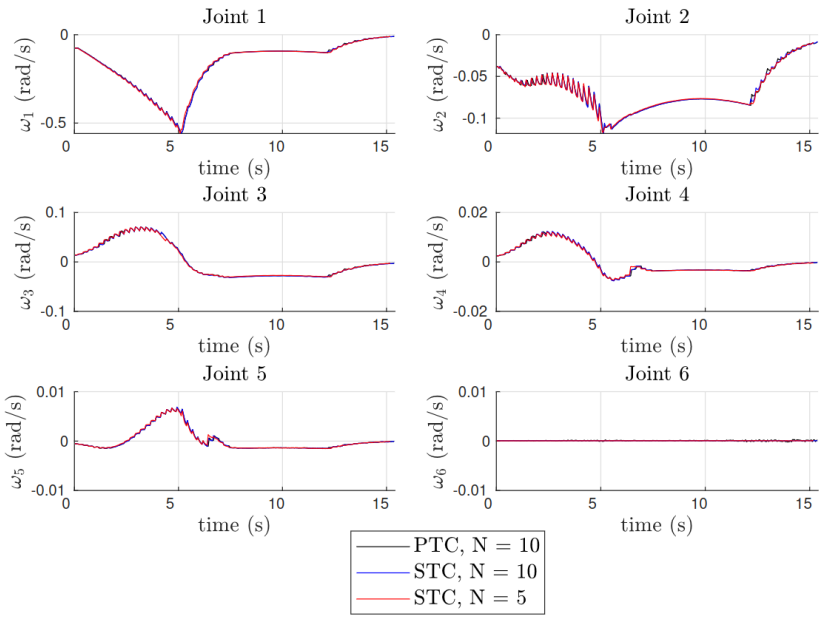

Figure 5.3: Time evolution of each component u_i of the control vector, for $i = \mathbb{N}_{[1,6]}$, for the three NMPC laws implemented in the experimental setup.

delay, which usually leads to a better performance of the closed-loop system. We also ran an experiment with $N = 5$ for the set terminal constraint case to test this hypothesis.

The results are visible in Figs. 5.2-5.6. Figures 5.2 and 5.3 show, respectively, the time evolutions of states (showing the value of $x_i - x_{g,i}$ for each joint, with $x_{g,i}$ representing the i-th coordinate of the goal configuration) and inputs (showing the value

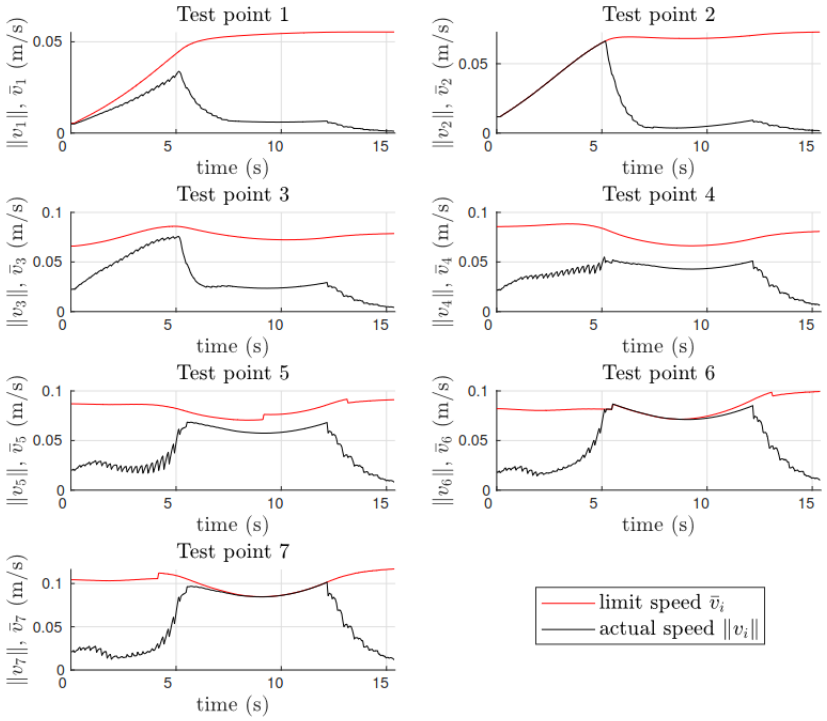

Figure 5.4: Time evolution of linear speeds $\|v_i\|$ of robot test points for the point terminal constraint (PTC) NMPC formulation, together with the corresponding limit speed $\bar{v}_i$, corresponding to the robot motions shown in Figs.5.2 and 5.3.

of u_i, i-th coordinate of u, for each joint) for all three cases. The goal configuration was reached in all cases with a similar time evolution of x and u, as the three lines are practically indistinguishable. Also, the SSM constraints, which ensure human safety, are satisfied in practice (in addition to being satisfied in the NMPC problem) along with the robot's motion in all three cases. This can be seen in Figs. 5.4-5.6, in which one can see the time evolution of the magnitude of the speed v_i of each robot test point, defined in (2.5), and the corresponding upper bound introduced by the SSM constraints, here indicated as

$$\bar{v}_i \triangleq \alpha^2 \min_{j \in \mathbb{N}_h} \left(d_{ij}^2 - \rho_{ij}^2 \right). \tag{5.24}$$

To measure J_{exp}, the evaluation method described in Section 3.4 is utilized. The task completion time T_c in this section represents the time employed to steer the robot state from x_0 to x_g. As expected, the values of both J_{exp} and T_c are very close for the two

Table 5.2: Comparison of the considered measures from experimental results with point terminal constraint (PTC) and set terminal constraint (STC)

PTC/STC	N	J_{exp}	T_c(s)	$\bar{\tau}$(ms)	$\hat{\tau}$(ms)
PTC	10	7169.6	15.3003	84.783	46.388
STC	10	7164.9	15.3003	57.958	40.364
STC	5	7094.9	15.1004	17.839	12.027

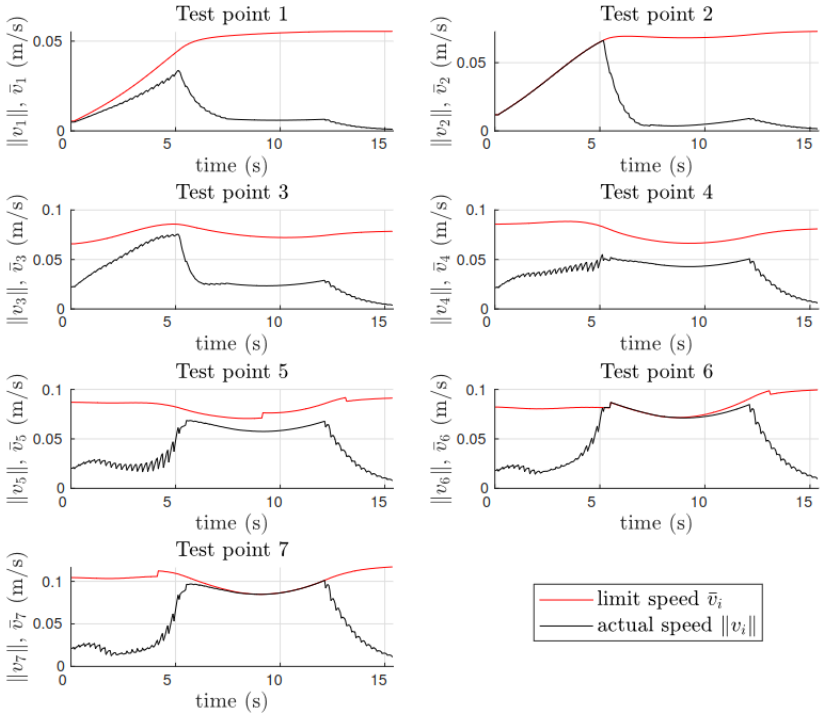

Figure 5.5: Time evolution of linear speeds $\|v_i\|$ of robot test points for the set terminal constraint NMPC formulation with $N = 10$, together with the corresponding limit speed $\bar{v}_i$, corresponding to the robot motions shown in Figs.5.2 and 5.3.

cases with $N = 10$, while a slight improvement can be observed for both measures in the case of set terminal constraint-based NMPC with $N = 5$. To explain this trend, one can refer to the values shown in the computation times. While all maximum values are way below the sampling time value $T = 250$ ms, the NMPC set terminal constraint case with $N = 5$ shows an average value much smaller than those of the other two NMPC algorithms. This shows that the proposed set terminal constraint-based NMPC strategy can allow a reduction of the value of N for the same task, leading to performance improvement. As an alternative to improving performance, the NMPC formulation based on set terminal constraints with $N = 5$ could be implemented on a cheaper computer, thus allowing for a reduction of economic costs.

It is worth mentioning that the fast oscillations of the robot speed observed in Figs. 5.3-5.6 do not significantly affect the robot motion.

5.4 Chapter summary

This chapter focused on modifying the NMPC law for SSM-based motion planning with point terminal constraint described in Chapter 2 to the case of set terminal constraint. First, the closed-loop stability of the considered system is proven. Then, it is shown that, in an experimental case study with a UR5 manipulator, the domain of attraction increases considerably in the case with a set terminal constraint.

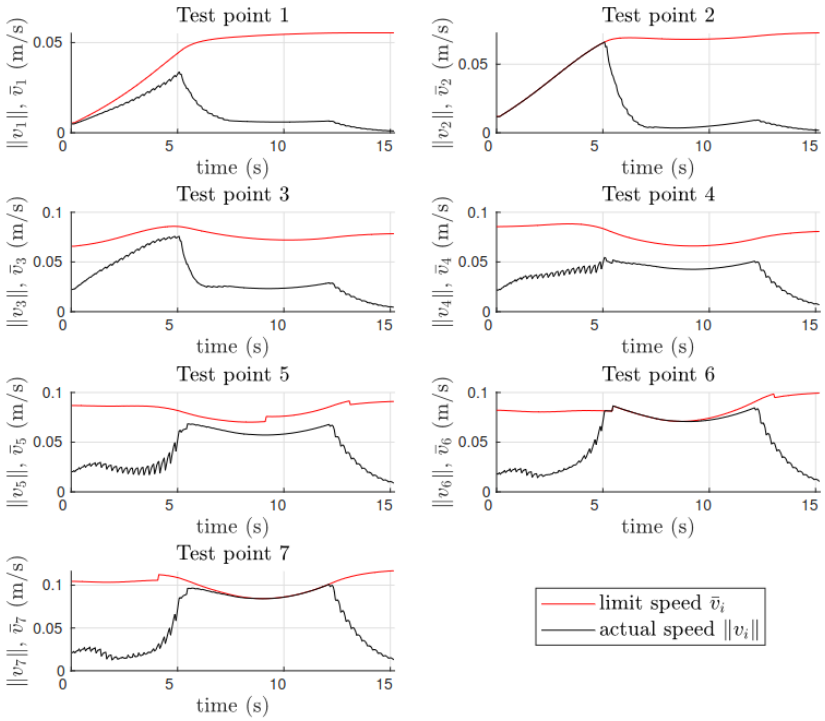

Figure 5.6: Time evolution of linear speeds $\|v_i\|$ of robot test points for the set terminal constraint NMPC formulation with $N = 5$, together with the corresponding limit speed $\bar{v}_i$, corresponding to the robot motions shown in Figs.5.2 and 5.3.

Chapter 6

Conclusions

6.1 Addressing the Research Questions

To summarize the contribution of this thesis, we focus on the research questions defined in Chapter 1. We report them in the following and respond to them.

6.1.1 Answering the first research question

The first research question was the following: "Is it possible to improve the performance of SSM-based NMPC laws, at the same time guaranteeing human safety, by emulating their behavior through DNNs?"

Six different DNN-based motion planning schemes were designed and implemented in Chapter 3 to imitate a safe – according to the SSM principle – NMPC scheme for pHRI. The experimental results show that the DNN-based schemes successfully compensate for the performance degradation caused by the computational delay of the NMPC scheme and rely on the information on future human motion acquired during training to further improve performance. The drawback of all DNN-based schemes, as in all learning-based methods, is that, should the human move with a considerably different pattern compared to training data, the motion planning algorithm would not be able to properly plan the robot motion. Thus, particular attention must be paid to ensure that the human motion happens according to precise instructions. It is essential to notice, though, that the DNN-based schemes were able to learn their behavior on a number of different subjects and successfully apply it to a new subject never seen before; this shows that, to some extent, the proposed algorithms possess the ability to generalize between different human operators.

Deep imitation learning of NMPC laws for safe pHRI via a dataset aggregation approach has also been designed and implemented in Chapter 4. Compared to offline imitation learning techniques, the main contributions consist of dataset aggregation-based training and safety filters. Experimental results showed that the DA-SE-DNN method, employing an encoder (to condense the information provided as input to the DNN), aggregation-based training, and a safety filter, provided the best trade-off between safety and performance.

In conclusion the answer to the research question is "yes", as performance improvement with respect to the NMPC law was observed for several DNN structures,

and the presence of the safety filters ensured the satisfaction of the SSM constraints.

6.1.2 Answering the second research question

The second research question was the following: “Is it possible to define and design an SSM-based NMPC law with guaranteed stability properties and based on a set terminal constraint, which increases the domain of attraction compared to the point terminal constraint approach?”

The answer to the second research question is also “yes”. Indeed, the NMPC law defined in Chapter 5 guarantees closed-loop stability under suitably defined assumptions, and it has been shown that the domain of attraction can be considerably extended, based on the considered case study.

6.2 Limitations and Future Work

One of the main limitations of this research study is related to deep imitation learning approaches. As discussed earlier, these approaches ensure safe interaction in pHRI solely through the use of safety filters. However, the safety filter can considerably modify the DNN output, in some cases leading the system state at locations from which the newly-generated DNN outputs will not be coherent with the NMPC strategy. Safety filters are crucial in the proposed imitation learning approach because the DNNs alone cannot inherently guarantee the safety concerning SSM constraints. This effect was shown to be mitigated when the DAgger approach was used, but it can present itself again in case quite different human motions are introduced. Therefore, as already mentioned, the human motion must adhere to that observed during the training phase. In case a new type of human motion has to be introduced, then new NMPC data can be generated to update the DNN parameters. Future research could be directed towards the definition of DNN-based approaches to imitate SSM-based NMPC laws, providing strict guarantees of recursive feasibility and stability.

Regarding the NMPC law, its main limitation consists in the fact that the human operator remains static throughout the interaction period, in order to meet the assumptions required to guarantee recursive feasibility. This assumption raises the question of whether it is possible to develop a control strategy based on the assumption that a human will move and the system has only partial information about the human future locations. Can stability be guaranteed under these conditions? To partially answer this question, an NMPC controller capable of ensuring stability and recursive feasibility in scenarios involving human movement was already proposed in [30]. However, the conditions required in [30] to ensure stability are not easy to translate into clear guidelines for the human operator, and this can be the object of future research. In future work, we aim to focus on DNN-based approaches that provide strict guarantees

of constraint satisfaction and stability. Additionally, we plan to further study the STC NMPC approach to design a system that can guarantee stability in an NMPC control strategy, based on the assumption that the human will move.

Appendices

BLANK

Appendix A

Glossary

A.1 Acronyms

The following acronyms, here listed in alphabetical order, are used in this thesis:

- DAgger: dataset aggregation
- DNN: deep neural network
- NMPC: nonlinear model predictive control
- pHRI: physical human-robot interaction
- SSM: speed and separation monitoring

Furthermore, Table A.1 describes all the acronyms related to the imitation learning algorithms developed in the thesis. In the table:

- **Training Mode (TM)** refers to the methodology by which the algorithm is trained, distinguished as either "offline" or "online".
- **Future Human Motions (Future HM)** indicates whether the data is generated from the NMPC model that utilizes future human motion trajectories in its prediction horizon.
- **Past Human Motions (Past HM)** denotes whether the historical motion data of humans are utilized as input during the training process.
- **Encoder** specifies whether the neurons responsible for processing information about human motions are encoded during the training phase.
- **Safety** defines whether the algorithm ensures safety based on SSM.
- **Dataset Aggregation (DAgger)** represents the data collection method, in which the data is added in portions at each step of the training

Table A.1: Developed imitation learning algorithms

Algorithm	TM	Future HM	Past HM	Encoder	Safety	DAgger
O-DNN-0	offline	no	no	no	no	no
O-DNN-P	offline	yes	no	no	no	no
O-DNN-PS	offline	yes	yes	no	no	no
O-E-DNN-0	offline	no	no	yes	no	no
O-E-DNN-P	offline	yes	no	yes	no	no
O-E-DNN-PS	offline	yes	yes	yes	no	no
ODA-DNN	offline	yes	no	no	no	yes
DA-DNN	online	yes	no	no	no	yes
ODA-E-DNN	offline	yes	no	yes	no	yes
DA-E-DNN	online	yes	no	yes	no	yes
O-S-DNN-P	offline	yes	no	no	yes	no
ODA-S-DNN	offline	yes	no	no	yes	yes
DA-S-DNN	online	yes	no	no	yes	yes
O-SE-DNN-P	offline	yes	no	yes	yes	no
ODA-SE-DNN	offline	yes	no	yes	yes	yes
DA-SE-DNN	online	yes	no	yes	yes	yes

A.2 Sets and Spaces

$\mathcal{P}$ Set of test points of the robot in the Cartesian space, p. 11

$\mathcal{S}$ Set of joint positions of the robot and human positions, p. 26

$\mathcal{B}_\nu(x_g)$ Ball with radius ν, centered at x_g, p. 17

$\mathcal{S}_{h,j}$ Sphere that occupies the human body, $j \in \mathbb{N}_r \triangleq \mathbb{N}_{[1,n_h]}$, p. 10

$\mathcal{S}_{r,i}$ Sphere that occupies the robot surface, $i \in \mathbb{N}_r \triangleq \mathbb{N}_{[1,n_r]}$, p. 10

Ω Closed box including u_g in the interior, p. 11

Π Set of DNN policies, p. 24

Θ Closed box including x_g in the interior, p. 11

$\mathbb{U}_{\text{SSM}}(x)$ SSM constraint set, p. 17

$\mathbb{U}(x)$ Control constraint set, p. 17

$\mathbb{X} \subseteq \mathbb{R}^n$ Set of joint positions of the robot, p. 14

$\mathbb{X}_F$ Feasible set for terminal constraint set, p. 52

$\mathbb{X}_T \subseteq \mathbb{X}$ Terminal set, p. 49

A.3 Variables

$\alpha \in \mathbb{R}_{>0}$ Tuning parameter for SSM formulation, p. 13

β Tuning parameter for $\ell(x, u)$, p. 15

$\bar{a}_r$ Maximum deceleration value of the robot, p. 12

$D \in \mathbb{R}_{>0}$ Scalar parameter essential for stability analysis, p.51

$\delta \in \mathbb{R}_{>0}$ Radius of feasible ball around x_g, p. 51

$\delta_x \in \mathbb{R}_{>0}$ Radius of ball around x_g, within which data is collected to train DNNs for achieving the goal state efficiently, p. 28

$\bar{d} \in \mathbb{R}_{>0}$ Safety margin parameter for SSM formulation, p. 13

$\bar{d_{ih}}$ Maximum distance that $\mathcal{S}_{h,j}$ can travel until $v_i = 0$, p. 12

$\bar{d_{ir}}$ Maximum distance that $\mathcal{S}_{r,i}$ can travel until $v_i = 0$, p. 12

$\epsilon \in \mathbb{R}_{>0}$ Scalar value that used in Assumption 14

$\epsilon_s \in \mathbb{R}_{>0}$ Maximum error with which the motion capture system detects $p_{r,i}$, p. 12

$\eta \in \mathbb{R}_{>0}$ Value of the optimal value function $J^*_\infty(x_0, u(\cdot))$, p. 50

$\gamma \in \mathbb{R}_{>0}$ Tuning parameter for $\ell(x, u)$, p. 15

$\mathcal{H}$ $\mathcal{S}_{h,j}$ positions, p. 11

$\bar{\mathcal{H}}$ Future positions of $\mathcal{S}_{h,j}$, p. 25

$\mathcal{H}^-$ Past positions of $\mathcal{S}_{h,j}$, p. 25

$\mathcal{H}_E$ Encoded $\mathcal{S}_{h,j}$ positions p. 26

$\mathcal{H}_E^-$ Encoded past positions of $\mathcal{S}_{h,j}$, p. 26

$\bar{v}_H$ Maximum possible speed of the human, p. 12

K Linear quadratic regulator gain, p. 12

k Discrete-time instant along the prediction horizon, p. 12

L_p Average path length traveled by the end effector, p. 30

N Prediction horizon length, p. 15

n_h Number of test points located on human body, p. 10

n_r Number of test points located on robot, p. 10

ν Radius of the ball $\mathcal{B}_\nu(x_g)$, p. 54

$P \in \mathbb{R}^{n \times n}$ Solution of the discrete-time algebraic Riccati equation, p. 50

$p^*_{r,n_r} \in \mathbb{R}^3$ Goal configuration in Cartesian Space, p. 15

$p_{h,j} \in \mathbb{R}^3$ Center of the test points on the human, p. 10

$p_{r,i} \in \mathbb{R}^3$ Center of the test points on the robot, p. 10

$\tilde{Q}$ Positive definite matrix, p. 51

$Q \in \mathbb{R}^{n \times n}$ Symmetric and positive definite matrix, p. 15

$R \in \mathbb{R}^{n \times n}$ Symmetric and positive definite matrix, p. 15

$R_{h,j} \in \mathbb{R}_{>0}$ Radius of the spheres that occupy human body p. 10

$R_{r,i} \in \mathbb{R}_{>0}$ Radius of the spheres on the robot p. 10

ρ_{ij} Summation value representing the combined radii and safety margin, p. 13

$\sigma \in \mathbb{R}_{>1}$ Scalar parameter essential for stability analysis, p. 52 $\|u\|_\infty$

$\mathcal{U}^{DNN}$ Control values generated by DNN, p. 25

$\mathcal{U}^{NMPC}$ Control values generated by NMPC, p. 25

U Control effort, p. 16

$u_g \in \mathbb{R}^n$ Goal joint velocities, p. 10

$u^*(\cdot)$ Optimal Control Sequence, p. 16

$u_a(x)$ Auxiliary control law, p. 51

$u_x \in \mathbb{U}(x)$ Control value that brings the robot to x_g, p. 17
$v_i \in \mathbb{R}^3$ Cartesian velocity vectors, p. 11
$x \in \mathbb{R}^n$ Joint positions of the robot, p. 10
$x_g \in \mathbb{R}^n$ Goal joint positions, p. 10
$\mathbb{X}_F$ Domain of attraction, p. 56
$\mathbb{X}_T$ Terminal set, p. 49
$\zeta \in \mathbb{R}_{(0,1)}$ Scalar parameter essential for stability analysis p. 51

A.4 Functions

$\alpha_1(\cdot) \in \mathcal{K}_\infty$ Lower bound of a Lyapunov function, p. 17
$\alpha_2(\cdot) \in \mathcal{K}_\infty$ Upper bound of a Lyapunov function, p. 17
$\alpha_3(\cdot) \in \mathcal{K}_\infty$ Upper bound of a Lyapunov function, p. 17
$\tilde{\alpha}_2(\cdot) \in \mathcal{K}_\infty$ upper bounding function, p. 17
$f(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ System dynamics, p. 10
$F(\cdot) : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ Terminal cost, p. 49
$g(x)$ Inequality constraints, p. 54
$J_N^*(x)$ Finite horizon optimal cost function, p. 21
$J_N(x_0, u(\cdot))$ Finite horizon cost function, p. 16
$J_\infty^*(x_0, u(\cdot))$ Infinite-horizon optimal cost function, p. 16
$J_\infty(x_0, u(\cdot))$ Infinite-horizon cost function, p. 52
J_{exp} The evaluation of the NMPC cost during the experiment, p. 30
$J_{NF}(x_0, u(\cdot))$ Finite horizon optimal cost of NMPC with set terminal constraint, p. 50
$\mathscr{K}_{d,i}(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^3$ Differential kinematics function, p. 11
$\mathscr{K}_{f,i}(\cdot) : \mathbb{R}^n \to \mathbb{R}^3$ Forward kinematics function, p. 11
$\lambda_{\max}(Q)$ Maximum eigenvalue of Q matrix, p. 54
$\lambda_{\min}(Q)$ Minimum eigenvalue of Q matrix, p. 54
$\ell(x,u)$ Stage cost of NMPC problem, p. 15
$L(\mathcal{U}^{NMPC}, \mathcal{U}^{DNN})$ Quadratic loss function, p. 24
$\Phi(\|x\|_{x_g})$ Upper bounding function on the third term of the loss function, p. 51
$\psi(y) : \mathbb{R} \to \mathbb{R}$ Upper bound function, p. 51
$u^{DNN}(s)$ DNN policy, p. 25
$u^{NMPC}(s)$ NMPC policy, p. 25
$\varphi(x, x_g)$ Component of the loss function related to human position, p. 15

Appendix B

UR5 kinematics

The robot forward kinematics can be developed utilizing the Denavit-Hartenberg (DH) parameters, which characterize the connection between two successive coordinate frames in a robotic manipulator's kinematic chain[1]. They are often used to simulate and regulate the motion of robotic arms and manipulators in mechanical and robotics engineering. Specifically, DH parameters are described using the following four aspects:

- Link Length (a): The length from the origin of the previous frame to the origin of the next frame, measured along the prior z-axis, between the axes along the standard normal.

- Link Twist (α): The angle, measured around the preceding x-axis, between the standard normal and the following z-axis.

- Link Offset (d): The length, measured along the old x-axis, along the common normal that connects the old z-axis to the new z-axis.

- θ: The angle, measured about the old z-axis, between the old and new x-axes.

These parameters allow the creation of a homogenous transformation matrix that characterizes the interaction between successive coordinate frames for each joint of a robotic manipulator. Given the joint angles, this matrix is then utilized to determine the end-effector orientation and location with respect to the base frame. Robotic manipulator kinematics may be conveniently and methodically modeled with the help of DH parameters.

Considering that the thesis uses the UR5 robot manipulator from Universal Robot, the following part of the appendix is specially designed for this robot.

The Denavit-Hartenberg parameters for any UR robot are readily available on the official website of Universal Robots [77]. These parameters are presented in the Table B.1. For the UR5 robot manipulator, parameters shown in Table B.1 are $a_2 = -0.425$, $a_3 = -0.39225$, $d_1 = 0.089459$, $d_4 = 0.10915$, $d_5 = 0.09465$, $d_6 = 0.0823$. The following homogeneous transformation matrices were built using the parameters given in the UR5 robots in Table B.1. To decrease the amount of space, the

[1] In this appendix, the notation partially differs from that employed in the main chapters of the thesis.

Table B.1: DH parameters of UR robot-manipulator

Joints	θ(rad)	a(m)	d(m)	α(rad)
Joint 1	0	0	d_1	$\pi/2$
Joint 2	0	a_2	0	0
Joint 3	0	a_3	0	0
Joint 4	0	0	d_4	$\pi/2$
Joint 5	0	0	d_5	$-\pi/2$
Joint 6	0	0	d_6	0

subsequent expressions are employed: $s_i = \sin(\theta_i)$ and $c_i = \cos(\theta_i)$ for $i = 1, ..., 5$.

$$H_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

$$H_2 = \begin{bmatrix} c_2 & -s_2 & 0 & -a_2c_2 \\ s_2 & c_2 & 0 & -a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.2}$$

$$H_3 = \begin{bmatrix} c_3 & -s_3 & 0 & -a_3c_2 \\ s_2 & c_2 & 0 & -a_3s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

$$H_4 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

$$H_5 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.5}$$

$$H_6 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.6}$$

In order to cover the whole robot body, additional matrices are used to shift the test point positions. The positions of the test points depicted in Figure B.1 were determined

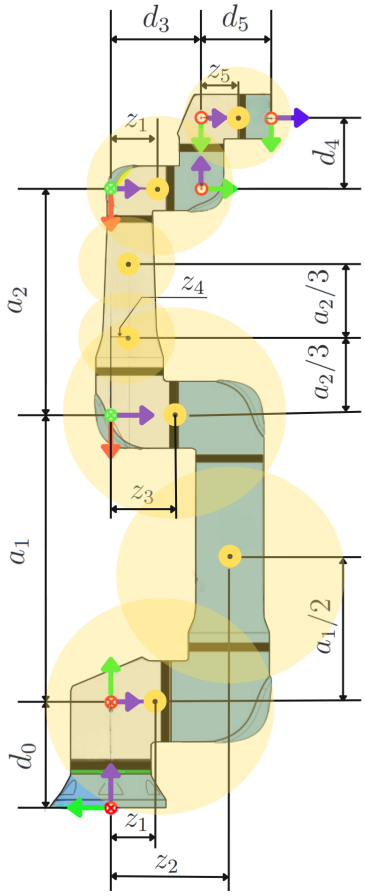

Figure B.1: Test points of the UR5, figure adapted from [5].

by utilizing the homogeneous transformation matrices as follows:

$$T_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.7}$$

where $i = 1, ..., 5$ and $z = \begin{bmatrix} 0.06 & 0.28 & 0.11 & 0.02 & 0.0322 \end{bmatrix}$. The test points locations in the Cartesian space are thus obtained as follows:

$$p_{r,1} = H_1 T_1, \tag{B.8}$$

$$p_{r,2} = \frac{H_1 H_2 T_2}{2}, \tag{B.9}$$

$$p_{r,3} = H_1 H_2 T_3, \tag{B.10}$$

$$p_{r,4} = H_1 H_2 T_4 + \frac{H_3 - H_2}{3}, \tag{B.11}$$

$$p_{r,5} = H_1 H_2 T_4 + 2\frac{H_3 - H_2}{3}, \tag{B.12}$$

$$p_{r,6} = H_1 H_2 H_3 T_1, \tag{B.13}$$

$$p_{r,7} = H_1 H_2 H_3 H_4 H_5 T_5. \tag{B.14}$$

After the computation of the aforementioned equations, we can get the exact positions of the test points. For the sake of simplicity, the following notation is used: $c_{23} = \cos(\theta_2 + \theta_3)$, $s_{23} = \sin(\theta_2 + \theta_3)$, $c_{234} = \cos(\theta_2 + \theta_3 + \theta_4)$, $s_{234} = \sin(\theta_2 + \theta_3 + \theta_4)$. The positions of the test points in the Cartesian space are the following:

$$p_1 = \begin{bmatrix} z_1 s_1 \\ -z_1 c_1 \\ d_1 \end{bmatrix} \tag{B.15}$$

$$p_2 = \begin{bmatrix} \frac{1}{2}a_2 c_1 c_2 + \frac{1}{2}z_2 s_1 \\ \frac{1}{2}a_2 c_2 s_1 - \frac{1}{2}z_2 c_1 \\ \frac{1}{2}a_2 s_2 + \frac{1}{2}d_1 \end{bmatrix} \tag{B.16}$$

$$p_3 = \begin{bmatrix} a_2 c_1 c_2 + z_3 s_1 \\ a_2 c_2 s_1 - z_3 c_1 \\ a_2 s_2 + d_1 \end{bmatrix} \tag{B.17}$$

$$p_4 = \begin{bmatrix} a_2 c_1 c_2 + z_4 s_1 + \frac{1}{3}a_3 c_1 c_2 c_3 - \frac{1}{3}a_3 c_1 s_2 s_3 \\ a_2 c_2 s_1 - z_4 c_1 + \frac{1}{3}a_3 c_2 c_3 s_1 - \frac{1}{3}a_3 s_1 s_2 s_3 \\ d_1 + \frac{1}{3}a_3 s_{23} + a_2 s_2 \end{bmatrix} \tag{B.18}$$

$$p_5 = \begin{bmatrix} a_2 c_1 c_2 + z_4 s_1 + \frac{2}{3}a_3 c_1 c_2 c_3 - \frac{2}{3}a_3 c_1 s_2 s_3 \\ a_2 c_2 s_1 - z_4 c_1 + \frac{2}{3}a_3 c_2 c_3 s_1 - \frac{2}{3}a_3 s_1 s_2 s_3 \\ d_1 + \frac{2}{3}a_3 s_{23} + a_2 s_2 \end{bmatrix} \tag{B.19}$$

$$p_6 = \begin{bmatrix} a_2 c_1 c_2 + z_1 s_1 + a_3 c_1 c_2 c_3 - a_3 c_1 s_2 s_3 \\ a_2 c_2 s_1 - z_1 c_1 + a_3 c_2 c_3 s_1 - a_3 s_1 s_2 s_3 \\ d_1 + a_3 s_{23} + a_2 s_2 \end{bmatrix} \tag{B.20}$$

$$p_7 = \begin{bmatrix} a_2 c_1 c_2 + d_4 s_1 - z_5 c_{234} c_1 s_5 + z_5 c_5 s_1 + d_5 s_{234} c_1 + a_3 c_1 (c_2 c_3 - s_2 s_3) \\ a_2 c_2 s_1 - d_4 c_1 - z_5 c_{234} s_1 s_5 - z_5 c_1 c_5 + d_5 s_{234} s_1 + a_3 s_1 (c_2 c_3 - s_2 s_3) \\ d_1 - s_5 (z_5 c_{23} s_4 + z_5 s_{23} c_4) + a_3 s_{23} + d_5 (s_{23} s_4 - c_{23} c_4) + a_2 s_2 \end{bmatrix} \tag{B.21}$$

In order to find the Cartesian velocity vectors $v_i \in \mathbb{R}^3$, the Matlab function diff was used as follows:

$$v_1 = \mathrm{diff}(p_1, \theta_1)u_1'; \tag{B.22}$$

$$v_2 = [\mathrm{diff}(p_2, \theta_1), \mathrm{diff}(p_2, \theta_2)][u_1 u_2]' \tag{B.23}$$

$$v_3 = [\mathrm{diff}(p_3, \theta_1), \mathrm{diff}(p_3, \theta_2)][u_1 u_2]' \tag{B.24}$$

$$v_4 = [\mathrm{diff}(p_4, \theta_1), \mathrm{diff}(p_4, \theta_2), \mathrm{diff}(p_4, \theta_3)][u_1 u_2 u_3]' \tag{B.25}$$

$$v_5 = [\mathrm{diff}(p_5, \theta_1), \mathrm{diff}(p_5, \theta_2), \mathrm{diff}(p_5, \theta_3)][u_1 u_2 u_3]' \tag{B.26}$$

$$v_6 = [\mathrm{diff}(p_6, \theta_1), \mathrm{diff}(p_6, \theta_2), \mathrm{diff}(p_6, \theta_3)][u_1 u_2 u_3]' \tag{B.27}$$

$$v_7 = [\mathrm{diff}(p_7, \theta_1), \mathrm{diff}(p_7, \theta_2), \mathrm{diff}(p_7, \theta_3), \mathrm{diff}(p_7, \theta_4), \mathrm{diff}(p_7, \theta_5)][u_1 u_2 u_3 u_4 u_5]' \tag{B.28}$$

The resulting speeds of the test points in the Cartesian space are as follows:

$$v_1 = \begin{bmatrix} u_1 z_1 c_1 \\ u_1 z_1 s_1 \\ 0 \end{bmatrix} \tag{B.29}$$

$$v_2 = \begin{bmatrix} \frac{1}{2}u_1(z_2 c_1 - a_2 c_2 s_1) - \frac{1}{2}u_2 a_2 c_1 s_2 \\ \frac{1}{2}u_1(z_2 s_1 + a_2 c_1 c_2) - \frac{1}{2}u_2 a_2 s_1 s_2 \\ \frac{1}{2}u_2 a_2 c_2 \end{bmatrix} \tag{B.30}$$

$$v_3 = \begin{bmatrix} u_1(z_3 c_1 - a_2 c_2 s_1) - u_2 a_2 c_1 s_2 \\ u_1(z_3 s_1 + a_2 c_1 c_2) - u_2 a_2 s_1 s_2 \\ u_2 a_2 c_2 \end{bmatrix} \tag{B.31}$$

In the following equations, the notation $cs = c_2 c_3 - s_2 s_3$ is utilized.

$$v_4 = \begin{bmatrix} u_1\left(z_4 c_1 - a_2 c_2 s_1 - \frac{1}{3}a_3 s_1 cs\right) - u_2 c_1\left(\frac{1}{3}a_3 s_{23} + a_2 s_2\right) - \frac{1}{3}u_3 a_3 s_{23} c_1 \\ u_1\left(z_4 s_1 + a_2 c_1 c_2 + \frac{1}{3}a_3 c_1 cs\right) - u_2 s_1\left(\frac{1}{3}a_3 s_{23} + a_2 s_2\right) - \frac{1}{3}u_3 a_3 s_{23} s_1 \\ u_2\left(\frac{1}{3}a_3 c_{23} + a_2 c_2\right) + \frac{1}{3}u_3 a_3 c_{23} \end{bmatrix} \tag{B.32}$$

$$v_5 = \begin{bmatrix} u_1\left(z_4 c_1 - a_2 c_2 s_1 - \frac{2}{3}a_3 s_1 cs\right) - u_2 c_1\left(\frac{2}{3}a_3 s_{23} + a_2 s_2\right) - \frac{2}{3}u_3 a_3 s_{23} c_1 \\ u_1\left(z_4 s_1 + a_2 c_1 c_2 + \frac{2}{3}a_3 c_1 cs\right) - u_2 s_1\left(\frac{2}{3}a_3 s_{23} + a_2 s_2\right) - \frac{2}{3}u_3 a_3 s_{23} s_1 \\ u_2\left(\frac{2}{3}a_3 c_{23} + a_2 c_2\right) + \frac{2}{3}u_3 a_3 c_{23} \end{bmatrix} \tag{B.33}$$

$$v_6 = \begin{bmatrix} u_1(z_1 c_1 - a_2 c_2 s_1 - a_3 s_1 cs - u_2 c_1(a_3 s_{23} + a_2 s_2) - u_3 a_3 s_{23} c_1 \\ u_1(z_1 s_1 + a_2 c_1 c_2 + a_3 c_1 cs - u_2 s_1(a_3 s_{23} + a_2 s_2) - u_3 a_3 s_{23} s_1 \\ u_2(a_3 c_{23} + a_2 c_2) + u_3 a_3 c_{23} \end{bmatrix} \tag{B.34}$$

Due to the long expressions required to express the speed of the 7th test point, the following notation is used in which, for the i-th test point, v_{i1}, v_{i2} and v_{i3} describe the components on the x, y and z axis of the Cartesian space, respectively.

$$\begin{aligned} v_{71} = {} & u_1(d_4c_1 - a_2s_1c_2 + z_5c_1c_5 - d_5s_{234}s_1 - a_3s_1cs + z_5c_{234}s_1s_5) \\ & + u_4c_1(d_5c_{234} + z_5s_{234}s_5) + u_3c_1(d_5c_{234} - a_3c_2s_3 - a_3c_3s_2 + z_5s_{234}s_5) \\ & - u_2c_1(a_2s_2 - d_5c_{234} + a_3(c_2s_3 + c_3s_2) - z_5s_{234}s_5) \\ & - u_5z_5(s_1s_5 + c_{234}c_1c_5) \end{aligned} \tag{B.35}$$

$$\begin{aligned} v_{72} = {} & u_1(d_4s_1 + a_2c_1c_2 + z_5c_5s_1 + d_5s_{234}c_1 + a_3c_1cs - z_5c_{234}c_1s_5) \\ & + u_4s_1(d_5c_{234} + z_5s_{234}s_5) + u_3s_1(d_5c_{234} - a_3c_2s_3 - a_3c_3s_2 + z_5s_{234}s_5) \\ & - u_2s_1(a_2s_2 - d_5c_{234} + a_3(c_2s_3 + c_3s_2) - z_5s_{234}s_5) \\ & + u_5z_5(c_1s_5 - c_{234}s_1c_5) \end{aligned} \tag{B.36}$$

$$\begin{aligned} v_{73} = {} & u_3(a_3c_{23} + d_5s_{234} - z_5c_{234}s_5) + u_2(a_3c_{23} + a_2c_2 + d_5c_{23}s_4 + d_5s_{23}c_4 \\ & - z_5c_{23}c_4s_5 + z_5s_{23}s_4s_5) + u_4(d_5s_{234} - z_5c_{234}s_5) - u_5z_5s_{234}c_5 \end{aligned} \tag{B.37}$$

Appendix C

Source Code

All codes can be found in https://github.com/moongerim/PhD_Thesis

BLANK

Bibliography

[1] A. Nurbayeva, A. Shintemirov, and M. Rubagotti, "Deep imitation learning of nonlinear model predictive control laws for safe physical human-robot interaction," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 8384–8395, 2022. © 2024 IEEE.

[2] A. Nurbayeva and M. Rubagotti, "Safely imitating predictive control policies for real-time human-aware manipulator motion planning: A dataset aggregation approach," *Under review*.

[3] A. Nurbayeva and M. Rubagotti, "Nonlinear model predictive control with set terminal constraint for safe robot motion planning via speed and separation monitoring," *Control Engineering Practice, conditionally accepted*.

[4] A. Oleinikov, S. Kusdavletov, A. Shintemirov, and M. Rubagotti, "Safety-aware nonlinear model predictive control for physical human-robot interaction," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5665–5672, 2021.

[5] "DH parameters for calculations of kinematics and dynamics." https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/. Accessed: 2024-02-02.

[6] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, collaborative robots: Challenges and opportunities.," in *Proc. Int. Joint Conference on Artificial Intelligence*, pp. 18–25, 2018.

[7] A. Weiss, A.-K. Wortmeier, and B. Kubicek, "Cobots in Industry 4.0: A roadmap for future practice studies on human–robot collaboration," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 4, pp. 335–345, 2021.

[8] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[9] E. Matheson, R. Minto, E. G. G. Zampieri, M. Faccio, and G. Rosati, "Human–robot collaboration in manufacturing applications: a review," *Robotics*, vol. 8, no. 4, p. 100, 2019.

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

[11] S. Haddadin and E. Croft, "Physical human–robot interaction," in *Springer Handbook of Robotics*, pp. 1835–1874, Springer, 2016.

[12] J. A. Marvel, "Performance metrics of speed and separation monitoring in shared workspaces," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 405–414, 2013.

[13] Int. Organization for Standardization, Geneva, Switzerland, *ISO-TS 15066: Robots and robotic devices – Collaborative robots*, 2016.

[14] M. J. Rosenstrauch, T. J. Pannen, and J. Krüger, "Human robot collaboration-using kinect v2 for ISO/TS 15066 speed and separation monitoring," *Procedia CIRP*, vol. 76, pp. 183–186, 2018.

[15] E. Kim, R. Kirschner, Y. Yamada, and S. Okamoto, "Estimating probability of human hand intrusion for speed and separation monitoring using interference theory," *Robotics and Computer-Integrated Manufacturing*, vol. 61, p. 101819, 2020.

[16] J. A. Marvel and R. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 144–155, 2017.

[17] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, 2015.

[18] A. M. Zanchettin, P. Rocco, S. Chiappa, and R. Rossi, "Towards an optimal avoidance strategy for collaborative robots," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 47–55, 2019.

[19] P. Zheng, P.-B. Wieber, and O. Aycard, "Online optimal motion generation with guaranteed safety in shared workspace," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9210–9215, 2020.

[20] P. Glogowski, A. Böhmer, A. Hypki, and B. Kuhlenkötter, "Robot speed adaption in multiple trajectory planning and integration in a simulation tool for human-robot interaction," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 1, pp. 1–20, 2021.

[21] M. Eckhoff, R. J. Kirschner, E. Kern, S. Abdolshah, and S. Haddadin, "An MPC framework for planning safe & trustworthy robot motions," in *Proc. International Conference on Robotics and Automation (ICRA)*, pp. 4737–4742, 2022.

[22] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2017.

[23] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[24] J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, pp. 415–424, 2011.

[25] M. Fnadi, W. Du, F. Plumet, and F. Benamar, "Constrained model predictive control for dynamic path tracking of a bi-steerable rover on slippery grounds," *Control Engineering Practice*, vol. 107, p. 104693, 2021.

[26] J. Cenerini, M. W. Mehrez, J.-w. Han, S. Jeon, and W. Melek, "Model predictive path following control without terminal constraints for holonomic mobile robots," *Control Engineering Practice*, vol. 132, p. 105406, 2023.

[27] M. H. Korayem, H. R. Adriani, and N. Y. Lademakhi, "Intelligent time-delay reduction of nonlinear model predictive control (NMPC) for wheeled mobile robots in the presence of obstacles," *ISA Transactions*, vol. 141, pp. 414–427, 2023.

[28] D. Saccani, L. Cecchin, and L. Fagiano, "Multitrajectory model predictive control for safe UAV navigation in an unknown environment," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 5, pp. 1982–1997, 2023.

[29] I. Tusseyeva, A. Oleinikov, A. Sandygulova, and M. Rubagotti, "Perceived safety in human–cobot interaction for fixed-path and real-time motion planning algorithms," *Scientific Reports*, vol. 12, no. 1, p. 20438, 2022.

[30] A. Oleinikov, S. Soltan, Z. Balgabekova, A. Bemporad, and M. Rubagotti, "Scenario-based model predictive control with probabilistic human predictions for human–robot coexistence," *Control Engineering Practice*, vol. 142, no. 105769, 2024.

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[32] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[33] M. Tanaskovic, L. Fagiano, R. Smith, and M. Morari, "Adaptive receding horizon control for constrained MIMO systems," *Automatica*, vol. 50, no. 12, pp. 3019–3029, 2014.

[34] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 2786–2793, 2017.

[35] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven MPC design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[36] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, 2019.

[37] M. Bujarbaruah, C. Vallon, and F. Borrelli, "Learning to satisfy unknown constraints in iterative MPC," in *Proc. IEEE Conf. on Decision and Control*, pp. 6204–6209, 2020.

[38] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[39] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[40] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, and A. Mitsos, "Accelerating nonlinear model predictive control through machine learning," *Journal of Process Control*, vol. 92, pp. 261–270, 2020.

[41] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443–1451, 1995.

[42] S. S. P. Kumar, A. Tulsyan, B. Gopaluni, and P. Loewen, "A deep learning architecture for predictive control," in *Proc. IFAC International Symposium on Advanced Control of Chemical Processes*, pp. 506–511, 2018.

[43] S. Lucia and B. Karg, "A deep learning-based approach to robust nonlinear model predictive control," in *Proc. IFAC Conference on Nonlinear Model Predictive Control*, pp. 511–516, 2018.

[44] Y. Löhr, M. Mönnigmann, M. Klaučo, and M. Kalúz, "Mimicking predictive control with neural networks in domestic heating systems," in *Proc. International Conference on Process Control*, pp. 19–24, 2019.

[45] S. Lucia, D. Navarro, B. Karg, H. Sarnago, and O. Lucia, "Deep learning-based model predictive control for resonant power converters," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 409–420, 2020.

[46] A. D. Bonzanini, J. A. Paulson, G. Makrygiorgos, and A. Mesbah, "Fast approximate learning-based multistage nonlinear model predictive control using gaussian processes and deep neural networks," *Computers & Chemical Engineering*, vol. 145, p. 107174, 2021.

[47] B. M. Åkesson and H. T. Toivonen, "A neural network model predictive controller," *Journal of Process Control*, vol. 16, no. 9, pp. 937–946, 2006.

[48] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, 2018.

[49] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020.

[50] H. H. Nguyen, T. Zieger, S. C. Wells, A. Nikolakopoulou, R. D. Braatz, and R. Findeisen, "Stability certificates for neural network learning-based controllers using robust control theory," in *Proc. American Control Conference*, pp. 3564–3569, 2021.

[51] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 2102–2114, 2021.

[52] S. Ruiz-Moreno, J. R. D. Frejo, and E. F. Camacho, "Model predictive control based on deep learning for solar parabolic-trough plants," *Renewable Energy*, vol. 180, pp. 193–202, 2021.

[53] J. Drgoňa, D. Picard, M. Kvasnica, and L. Helsen, "Approximate model predictive building control via machine learning," *Applied Energy*, vol. 218, pp. 199–216, 2018.

[54] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust MPC and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[55] R. Moriyasu, S. Nojiri, A. Matsunaga, T. Nakamura, and T. Jimbo, "Diesel engine air path control based on neural approximation of nonlinear MPC," *Control Engineering Practice*, vol. 91, p. 104114, 2019.

[56] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," in *Proc. International Conference on Robotics and Automation*, pp. 528–535, 2016.

[57] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020.

[58] M. Novak and T. Dragicevic, "Supervised imitation learning of finite-set model predictive control systems for power electronics," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 2, pp. 1717–1723, 2020.

[59] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation learning from MPC for quadrupedal multi-gait control," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 5014–5020, 2021.

[60] M. Rubagotti, B. Sangiovanni, A. Nurbayeva, G. P. Incremona, A. Ferrara, and A. Shintemirov, "Shared control of robot manipulators with obstacle avoidance: A deep reinforcement learning approach," *IEEE Control Systems Magazine*, vol. 43, no. 1, pp. 44–63, 2023.

[61] L. Dai, Y. Yu, D.-H. Zhai, T. Huang, and Y. Xia, "Robust model predictive tracking control for robot manipulators with disturbances," *IEEE T Ind Electron*, vol. 68, no. 5, pp. 4288–4297, 2021.

[62] K. Eriksson, D. Estep, C. Johnson, K. Eriksson, D. Estep, and C. Johnson, "Lipschitz continuity," in *Applied Mathematics: Body and Soul. Volume 1: Derivatives and Geometry in IR 3*, pp. 149–164, Springer, 2004.

[63] B. Siciliano *et al.*, *Robotics: Modeling, Planning and Control*. Springer, 2010.

[64] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[65] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, 2018.

[66] P. M. Kebria, A. Khosravi, S. M. Salaken, and S. Nahavandi, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 82–95, 2019.

[67] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, *et al.*, "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9651–9658, 2020.

[68] P. Maurice, A. Malaisé, C. Amiot, N. Paris, G.-J. Richard, O. Rochel, and S. Ivaldi, "Human movement and ergonomics: An industry-oriented dataset for collaborative robotics," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1529–1537, 2019.

[69] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[70] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - an open-source framework for automatic control and dynamic optimization," *Optimal Control: Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[71] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[72] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados–a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, pp. 1–37, 2021.

[73] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," in *Proc. IFAC World Congress*, pp. 6563–6569, 2020.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[75] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, JMLR Workshop and Conference Proceedings, 2011.

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

[77] "Universal robots," 2024. March, 28, 2024.