Technische Universität München
TUM School of Computation, Information and Technology

Hierarchical Optimization Control and Robust Imitation Learning for Manipulation

Yingbai Hu

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 11.04.2022 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.09.2022 angenommen.

Abstract

The manipulator is playing a crucial role in addressing the industrial, sporting, and healthcare fields, because of the high working efficiency and low energy consumption, etc. For robot manipulation, the high level of motion planning in task space and low level of control performance remains challenging topics, therefore attracting much attention from the robotic community. Specifically for the low-level control, the multi-task control is by no means a trivial problem for the redundant manipulator, such as end-effector's task, manipulability, and remote center of motion, because it is difficult to find a trade-off solution between different tasks when considering the priority of tasks. For high-level motion planning, imitation learning is a promising way to acquire similar intelligence and abilities as humans by transferring in task space, but it has its limits in the performance of complex tasks with small amounts of data, owing to compounding errors. Due to this fact, it raises a new challenge: how to improve robustness and adaptability of imitation learning. For example, when performing a new task not recorded in the dataset, such as obstacle avoidance, via-point and external force field, the robot can successfully complete the task without re-teaching.

In this thesis, we will work on the low-level multi-task hierarchical optimization control of redundant manipulators, and high-level motion learning by imitating human behaviors. Consequently, it is significant to design a hierarchical optimization strategy to control these tasks in terms of priority metrics. Specifically, we design the gradient descent-based parallel neural network to address the multi-task hierarchical optimization problem which combines the different priority objective functions. On the other hand, to reproduce excellent motor skills by cloning human skills, we proposed a novel policy improvement method to enhance the robustness and adaptability of supervisor learning for new tasks.

The major contribution of this thesis is to propose effective and novel algorithms or strategies to address these problems from motion learning to multi-task optimization. The main contents were published in international conferences or journals.

Zusammenfassung

Der Manipulator spielt aufgrund der hohen Arbeitseffizienz, des geringen Energieverbrauchs usw. eine entscheidende Rolle in den Bereichen Industrie, Sport und Gesundheit. Die Bewegungsplanung der Roboter im Aufgabenraum und die geringe Leistungsfähigkeit der Steuerung stellen nach wie vor eine Herausforderung dar, die in Fachkreisen der Robotik viel Aufmerksamkeit erhält. Besonders für die Steuerung auf niedriger Ebene ist die Multi-Task-Steuerung keineswegs ein triviales Problem für den redundanten Manipulator. Dies beinhaltet zum Beispiel die Aufgabe des Endeffektors, die Manipulierbarkeit und der entfernte Bewegungsschwerpunkt, da es schwierig ist, eine Kompromisslösung zwischen verschiedenen Aufgaben unter Berücksichtigung der Priorität dieser Aufgaben zu finden. Für die Bewegungsplanung auf hoher Ebene, ist Nachahmungslernen ein vielversprechender Weg um ähnliche Intelligenz und Fähigkeiten wie der Mensch durch Übertragung in den Aufgabenbereich zu erwerben, allerdings hat es seine Grenzen bei der Ausführung komplexer Aufgaben mit kleinen Datenmengen, da sich Fehler anhäufen können. Aufgrund dieser Tatsache ergibt sich eine neue Herausforderung: Wie kann die Robustheit und Anpassungsfähigkeit des Nachahmungslernens verbessert werden? Bei der Ausführung einer neuen Aufgabe des Roboters, die nicht im Datensatz enthalten ist, wie zum Beispiel Hindernisvermeidung, Via-Point und externes Kraftfeld, kann der Roboter die Aufgabe dann ohne erneutes Lernen erfolgreich bewältigen.

In dieser Arbeit werden wir uns mit der hierarchischen Multitasking-Optimierung von redundanten Manipulatoren auf niedriger Ebene und dem Erlernen von Bewegungen auf hoher Ebene durch Nachahmung des menschlichen Verhaltens beschäftigen. Folglich ist es wichtig, eine hierarchische Optimierungsstrategie zu entwerfen, um diese Aufgaben in Bezug auf Prioritätsmetriken zu steuern. Konkret entwerfen wir das auf Gradientenabstieg basierende parallele neuronale Netz, um das hierarchische Multitasking-Optimierungsproblem zu lösen, das die verschiedenen Prioritätszielfunktionen kombiniert. Um exzellente motorische Fähigkeiten durch Klonen menschlicher Fähigkeiten zu reproduzieren, haben wir gleichzeitig eine neue, verbesserte Strategie vorgeschlagen, um die um die Robustheit und Anpassungsfähigkeit des Überwachten Lernens für neue Aufgaben zu verbessern.

Der Hauptbeitrag dieser Arbeit besteht darin, effektive und neuartige Algorithmen oder Strategien vorzuschlagen, um diese Probleme vom Bewegungslernen bis zur Multitasking-Optimierung anzugehen. Die wichtigsten Inhalte wurden auf internationalen Konferenzen oder in Fachzeitschriften veröffentlicht.

Acknowledgement

Throughout the writing of this thesis, I have received a great deal of support and assistance from TUM, and I still remembered the beginning day when I arrived in Munich on Oct. 16, 2018. First of all, I would like to express my sincerest gratitude to my supervisor Prof. Alois Knoll, who generously offered this precious research opportunity of a Ph.D. study and the open and passionate work environment. I still remembered that your congratulation email really inspired me and recognized the value of my work when my paper was reported by IEEE Spectrum. It is very lucky to study in our Chair of Robotics, Artificial Intelligence and Real-time Systems.

I would like to thank my mentor, Prof. Guang Chen. During these four years, you provided me with valuable suggestions and support for my research. I would like to thank Amy Bücherl, Ute Lomp. Whenever I encountered problems, you are so patient and always happy to help me. You are my hero! I would like to thank my external advisor Prof. Zhijun Li and Prof. Gary Yen. With your help, I really learned a lot from our regular discussions and the materials, expanded my research direction research approaches. Thanks for sharing your knowledge, experiences, and ideas.

I would like to thank my colleagues, Dr. Yinlong Liu, Mr. Wenjun Liu, Mr. Liguo Zhou, Ms. Mingyue Feng, Mr. Hu Cao, Mr. Shangding Gu, Mr. Xinyi Li, and Mr. Erdi Sayar. It is a good time that we worked together and discussed the job, the hot news, especially having lunch in Mensa, and cooking hot pot together. I would also thank my senior colleague Dr. Zhenshan Bing, Dr. Mingchuan Zhou, Dr. Xiebing Wang and Mr. Zhuangyi Jiang for their sharing of working experiences, instruction of life skills in Munich.

I would like to thank my dear friends in Munich, Ivan, Bibo, Barbara, Anne, Christian, Matthias, Leo, and Karsten. When I suffered some troubles and felt lonely, helpless and sad, I lost confidence and felt abandoned by the world, but thanks for your great help and company all along with me. It is my best time in Munich when living in MKH-UNO, and I cannot forget the memory of the morning greetings, the cooking time in the kitchen, and communication until very late. Ivan and Barbara, I owe you a promise–I will join your wedding in the future, wherever I am.

Finally, I would like to thank my mother Jianying Wang, my father Yushui Hu, my sister Liqun Hu and her husband Weiming Wu, my brother Yinggui Hu for their understanding, endless love, and support. Whatever I do, wherever I am, they always accompany and support me.

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.1.1 Why Optimization and Neural Network . . . 2

1.1.2 Why Policy Improvement in Robot Learning . . . 3

1.2 Thesis Outline and Contributions . . . 3

2 Hierarchical Optimization Control for Redundant Manipulator **7**

2.1 Optimization Control for OP-swab Robot . . . 7

2.1.1 Introduction . . . 7

2.1.2 Design of the Rigid–Flexible Coupling Robot . . . 10

2.1.3 Force Sensing . . . 12

2.1.4 Optimization Control for the 9-DOF Redundant Manipulator . . . 12

2.1.5 Experiments . . . 17

2.1.6 Summary . . . 22

2.2 Hierarchical Optimization Control for Surgical Robot . . . 22

2.2.1 Introduction . . . 23

2.2.2 Related Works . . . 24

2.2.3 Control Methodology . . . 24

2.2.4 Experimental Comparison . . . 29

2.2.5 Summary . . . 31

2.2.6 Appendix . . . 31

2.3 NRP: Hierarchical Optimization Control for Mobile Manipulator . . . 37

2.3.1 Introduction . . . 37

2.3.2 Coordinated Base-manipulator Trajectory Planning . . . 39

2.3.3 Optimization Problem . . . 41

2.3.4 Two-Timescale Neuronal Dynamics Design . . . 47

2.3.5 Experiment . . . 49

2.3.6 Summary . . . 54

2.4 Related Publication . . . 56

3 Imitation Learning for Manipulation **57**

3.1 Reinforcement Learning for Manipulation . . . 57

3.1.1 Introduction . . . 57

3.1.2 Motivation and Previous Works . . . 58

3.1.3 Methodology . . . 59

3.1.4 Experimental Demonstration . . . 63

3.1.5 Summary . . . 65

3.2 Fuzzy Adaptive Control and Imitation Learning for Real-time Obstacle Avoidance . . . 69

3.2.1 Introduction . . . 69

3.2.2 Adaptive Trajectories: Goal Adaptation and Obstacle Avoidance 70
3.2.3 Fuzzy Adaptive Control for Manipulator 75
3.2.4 Simulation 78
3.2.5 Summary 79
3.3 Related Publication 79

4 Robot Policy Improvement for Manipulation 81
4.1 Introduction 81
4.2 Related Work and Motivation 83
4.2.1 Imitation Learning from Demonstrations 83
4.2.2 Policy Improvement with Learning 83
4.3 Learning Problem 83
4.3.1 Nonlinear Dynamical System from Demonstration 83
4.3.2 Parameterized Policy Learning for the Dynamical System 85
4.4 Methodology for Policy Improvement 86
4.4.1 Natural Evolution Strategies 86
4.4.2 Fitness Shaping and Exponential Parameterization 88
4.4.3 Exponential NES Update Rule 88
4.5 Learning Variable Impedance Control of Stable Dynamical System 90
4.5.1 Shaping with Stable Exploration 91
4.5.2 Variable Impedance Control Learning Policy 93
4.6 Experimental Demonstration 95
4.6.1 Autonomous Motion Planning 97
4.6.2 Motion Learning with Obstacle Avoidance 98
4.6.3 Learning Stiffness of Variable Impedance Control 100
4.6.4 Analysis 101
4.7 Summary 102
4.8 Related Publication 102

5 Conclusion 105
5.1 Summary 105
5.2 Primary Contributions of the Thesis 105
5.3 Shortcomings 107
5.4 Future Work 107

Bibliography 109

List of Figures 121

List of Tables 125

Chapter 1

Introduction

With the increasing demand for robot applications in medical, search and rescue, industrial, and sporting domain, etc., there is a growing interest in designing the new robots and proposing a novel methodology for some complex scenarios.

Over the past decades, the development of control theory for the robot has achieved a considerable breakthrough. Indeed, the low-level controller directly produces the performance of the robot motion, because different scenarios will encounter different control problems and requirements, which raises high requirements on the control model and algorithm [1], such as position control (trajectory tracking), force control [2], position/force hybrid control [3], compliant control in human-robot interaction [4], ect.

Generally, the control issues can be divided into two categories: one is model-based which designs the controller from the exact robot model, another one is model uncertainties which are frequently related to adaptive control. In [5, 6, 7], model-based impedance control is applied to the robot in terms of the interaction with the environment or human, where the robot could behave with high compliance by changing the impedance. In [8], robust adaptive control is proposed for trajectories tracking, which can address the issues under model uncertainties, and improve the tracking accuracy.

Most of the work mentioned above is single-task, but for multi-task control, some special control strategies need to be designed. The multi-task control and optimization is a classic problem since the high degree of freedom (DOF) structure and complex control strategy of exploiting the redundant manipulator [9].

Varieties of control strategies are presented to solve the multi-task problem which considers the priorities of different tasks [10]. Generally, there are two common ways that are applied for prioritized multi-task controllers, including soft priority and hard priority. The hard priority of multi-task is also called "strict task priorities", which defines a hierarchical order of tasks: the more important tasks are completed with pre-order and the low priority task is controlled in the null space of higher priority tasks [11]. The soft priority is to combine the weight function of different tasks, where the high priority is with high weight and the low-priority is with a small weight. For more complex cases, when the priority is time-varying, the weights function should follow the robot actions [12].

Actually, the multi-task problem can be treated as optimization problem with constraints. Indeed, many kinds of researches contribute to this topic by exploring optimization solutions from different perspectives and metrics using evolutionary algorithms. In [13], Chen *et al.* developed an enhancing MOEA/D methods considering priority for multi-objective/multi-task optimization problem (MOP). In [14], Zhang *et al.* proposed a knee-based recursive evolutionary algorithm to optimize the MOP considering the preference metrics. Although these works have made great progress in multi-task problems, evolutionary algorithms are not suitable for practical applications, especially real-time manipulation of robots, due to a large number of search steps and decision variables, low computational cost, and low convergence speed.

In this thesis, we take a surgical robot as an example for multi-task control, where three tasks are simultaneously considered in the control strategy, including remote center of motion (RCM), manipulability measurement, and end effector task. In Robotics-assisted minimally invasive surgery (RAMIS) [15], the surgical tip should pass through a small incision on the belly, which produces the RCM constraint. Software implementation of RCM constraint is a promising way, not only a more economical approach but also more flexible because RCM points can be moved according to the requirements of the scenario [16]. In [17], Sandoval *et al.* proposed the null space control method for RCM and surgical task, which satisfied the accuracy of surgery. In [18], Su *et al.* provides an adaptive control method for MIS, in which an adaptive fuzzy controller is used to compensate the input due to the disturbance from human-computer interaction, thereby producing flexible and safe collaboration performance. In [19], Minelli *et al.* focused on dynamic perspective and developed a novel controller for RCM and surgical task. Nevertheless, the mentioned above works are optimization-free methods, which can not guarantee the physical limits and singularity issues, thereby causing possible damage to machines or patients.

On the other hand, for high-level motion planning, imitation learning is an effective way that can transfer human motor skills to robots and reduce the complexity of the algorithm in searching space. This is especially true when naturally acquiring new skills, as a robot must learn from the interaction with a human being or the environment with limited programming effort [20]. Traditional imitation learning is based on supervised learning, which represents skills with the dynamical model and computes the parameters of the dynamical model by regression.

In [21, 22, 23], A classic learning method-Dynamic Motion Primitives (DMP) is proposed to simulate complex behaviors. It can fit different linear or non-linear trajectories and is widely used to clone human skills in robot learning. In [24, 25], the Probabilistic movement primitives (ProMP) are introduced to represent motion trajectories, which can naturally model the features of the motion as a probability distribution. Although both of DMP and ProMP are effective to clone the complex behaviors, there are some different characteristics: DMP can adapt to a new start or goal position, and ProMP can adapt to intermediate via-points. In [26], Huang *et al.* proposed a kernel treatment-based imitation learning method called kernelized movement primitives (KMP) to encode the human skills, which is powerful in adaptability for via-points, a new start or goal tasks. In [20, 27, 28], the classical supervised learning, such as Gaussian mixture models (GMM), Gaussian Process Regression (GPR), are proposed to imitate the human behaviors with high-dimension input. However, the major drawback of these methods is that they are highly dependent on large amounts of expert data. In a real-world scenario, it is significant to improve the adaptability and robustness of imitation learning under limited data conditions.

1.1 Motivation

This section will provide the motivation of our works in terms of multi-task control and imitation learning. Specifically, the motivations are explained with two parts.

1.1.1 Why Optimization and Neural Network

For the multi-task control problem of the manipulator, we should consider the metrics of different task priorities and real-time performance. The traditional optimization-free approach is not a general solution for multi-tasks problem, because the control model and strategies are complex along with different scenarios, which means that it is impossible to find a common solution for different situations. Since the drawbacks of optimization-free approaches, we move forward to the optimization-based strategies for multi-task control problems. As the analysis knows, the evolutionary algorithms are a candidate solution for multi-task optimization, but low computational cost and convergence speed limits the online performance of robot manipulation. Therefore, we are toward convex optimization for our case considering soft priority.

To address the multi-task control problem of the surgical manipulator, we reformulate it as an optimization problem by soft priority strategy. There are two major solution methods for optimization problem-numerical optimization methods and neural network methods. Since the similar structure of biological counterpart, the neural network is a computational model that processes information in a parallel and distributed manner. The neural network-based solver has superior performance in solving optimization problems with high efficiency and high precision because it has no operation of matrix inversion, high-order non-linear computation. Compared with the numerical optimization methods, neural networks require weaker conditions, therefore, it is suitable for handling optimization problems [29].

1.1.2 Why Policy Improvement in Robot Learning

The traditional supervised learning-based imitation learning clones the human skills by teaching by demonstrations. Generally, the robot can reproduce similar behaviors as the teachers, and even predicts variability, correlations, and uncertainty for new tasks. However, it has its limits in the performance of more complex tasks with small amounts of data, owing to compounding errors. When the environment or tasks are changing, the learning approaches can not fulfill the new task without re-teaching. The traditional methods are not so adaptable and highly dependent on large amounts of data. Therefore, two factors should be considered in applications: first, it is impossible to cover the entire parameters space of the dataset; second, the efficiency of training is primarily metric in real-time performance which means the size of the dataset should be reasonable.

Considering a special case, when the external disturbance is applied to the dynamical system, the robot's motion will deviate from the nominal trajectories, so how to pull the robot to coincide with the reference trajectories without re-teaching. Therefore, we hope to design a policy that the stiffness of the controller can automatically increase along with the disturbance at this time. In this thesis, a hierarchical learning strategy is proposed to improve the robustness and adaptability of imitation learning: low-level learning comprises only those behaviors cloned with supervised learning, while high-level learning constitutes policy improvement.

1.2 Thesis Outline and Contributions

From the motivation, we present a hierarchical optimization control paradigm for the multi-task control problem of the redundant robot and also propose robust and adaptive imitation learning methods that transfer the human skills to the robot. The traditional primal-dual neural network [30] or recurrent neural network methods [31] cannot guarantee the convergence of error at the finite time, because of the optimal solution changing over time. Moreover, although the PDNN or RNN can obtain their solvability constraints at convex optimization problems, they may change drastically and become unpredictable of dynamic behaviors.

To improve the convergence speed in finite time, a varying parameters recurrent neural network (VPRNN, also called ZNN) based hierarchical control of a 7-DOF robot manipulator for robot-assisted minimally invasive surgery has been proposed, where it integrates multiple tasks based on their priority and guarantees task tracking [32], RCM, and manipulability optimization at the same time. It is a parallel processing approach with high efficiency and high precision. Its unique advantage is that it is a real-time solver without any pre-training. Its computation errors have super-exponential convergent rate and strong robustness, which is promising for the online solution of the time-varying optimization problem [33].

To address the second problem, the two-timescale recurrent neural networks optimization scheme is proposed and tested with a 9 DOFs nonholonomic mobile-based manipulator, which has a faster transient states response in the hidden layer(s). Compared with the traditional optimization solution of a redundant manipulator, infinity norm and slack variable are additionally introduced and leveraged by the optimization algorithm. The former takes into account the joint limits effectively by considering individual joint variables and the latter relaxes the equality constraint by decreasing the infeasible solution area.

To improve robustness and adaptability of imitation learning, the Gaussian mixture model-based (GMM) dynamical system is firstly formulated to encode a motion from the demonstration. We then derive the sufficient conditions of the GMM parameters that guarantee the global stability of the dynamical system from any initial state, using the Lyapunov stability theorem. Finally, a method based on exponential natural evolution strategies is proposed to optimize the parameters of the dynamical system associated with the stiffness of variable impedance control, in which the exploration noise is subject to stability conditions of the dynamical system in the exploration space, thus guaranteeing the global stability.

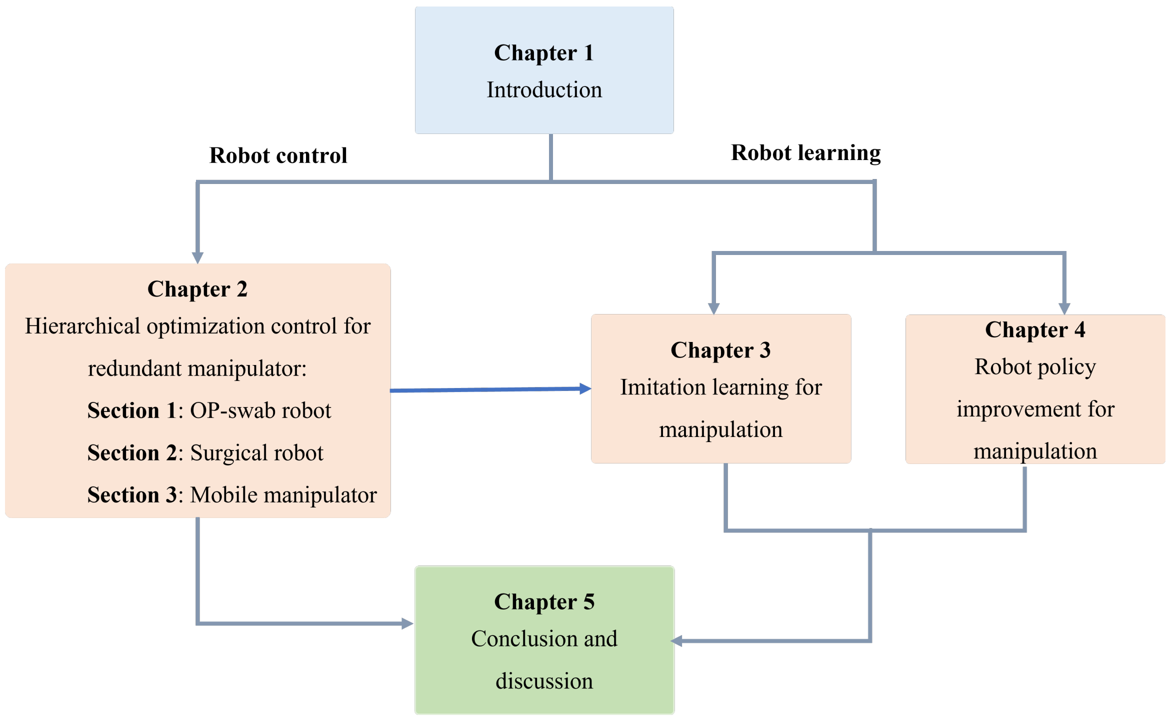

Chapter 1
Introduction
Robot control
Robot learning
Chapter 2
Hierarchical optimization control for redundant manipulator:
Section 1: OP-swab robot
Section 2: Surgical robot
Section 3: Mobile manipulator
Chapter 3
Imitation learning for manipulation
Chapter 4
Robot policy improvement for manipulation
Chapter 5
Conclusion and discussion

Figure 1.1: Thesis structure.

The outline of the thesis consists of three parts after the introduction chapter shown in Fig. 1.1. We firstly introduce the multi-task control problem and derive the optimization issues. We then proposed the neural network-based hierarchical optimization control method to address the multi-task control in terms of surgical robots and mobile manipulator. Afterward, we discuss motion generation by imitation learning. We propose policy improvement strategies (reinforcement learning and exponential natural evolution strategies) to improve the robustness and adaptability of the learning system. The main research framework of this thesis is shown in Fig. 1.2.

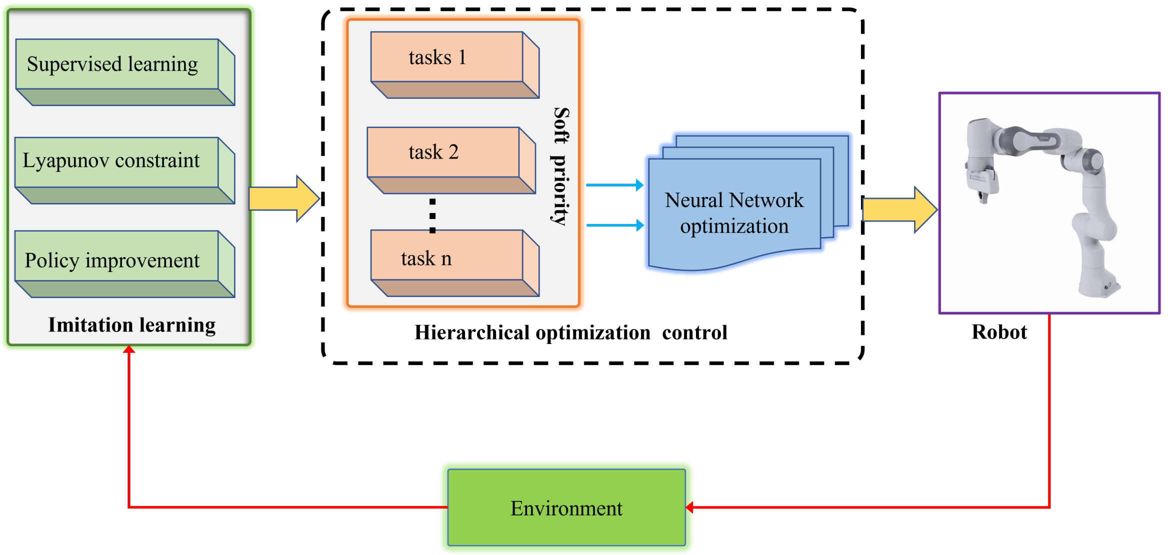


Figure 1.2: The main research framework of this thesis.

This thesis is subdivided into 3 main chapters to present our work and the contributions can be summarized as follows.

1 Chapter 2 introduces three sections of robot optimization control. The first section is a optimization control strategy for Oropharyngeal-swabs (OP-swab) robot of multi-task control problems. The second section is a hierarchical optimization strategy for multi-task control problems. The third section represents a constraint planning and optimization control scheme for a highly redundant mobile manipulator considering a complex indoor environment, where the experiments of predefined trajectories and obstacle avoidance cases using BI^2RRT^* planner are demonstrated in our team developed Neurorobotics platform and achieve a superior performance. The details of the contributions are as follows:

   Section 1 : (a) A rigid-flexible coupling robot is designed to assist COVID-19 OP-swab sampling, where the MPA for OP-swab sampling is developed to achieve flexible sampling and thereby provides compliant, safe, stable, and reliable sampling; (b) An improved motion planning method is proposed for the 9-DOF highly redundant robot-assisted OP-swab sampling system, which guarantees efficiency and accuracy, where the multi-constraint kinematics model is derived as an optimization problem; then, the VP-ZNN is applied to solve the optimization problem; (c) Substantive practical experiment testing of the developed OP-swab sampling robot system is demonstrated in the human oral cavity phantom and volunteers.

   Section 2: (a) A novel optimization scheme is derived for multi tasks constraints considering task tracking, compliance with the RCM constraint, joint angle and velocity limits, and manipulability index. Compared with the traditional single index optimization problem, the proposed novel hierarchical optimization framework of multi-tasks can further improve the robot stability, safety, and success rate of surgery; (b) A varying-parameter recurrent neural network (VPRNN) based hierarchical optimization of a 7-DOF surgical manipulator for Robot-Assisted Minimally Invasive Surgery (RAMIS), which guarantees task tracking, Remote Center of Motion (RCM), and manipulability index optimization. A theoretically grounded hierarchical optimization framework based is introduced to control multiple tasks based on their priority;

Section 3: (a)The kinematic model of a 9 DOFs mobile-based manipulator is derived, and the tracking control problem including infinity norm and slack variable metrics is formulated as a hierarchical optimization problem. (b) A two-timescale recurrent neural networks optimization scheme is introduced to address the hierarchical optimization problem with planning trajectory.

2 Chapter 3 presents two sections of imitation learning for robot manipulation. The first section designs a reinforcement learning-based manipulation skill transferring strategy for a surgical robot. The proposed method consists of two main steps: (a) We use the Gaussian mixture model and Gaussian mixture regression-based dynamic movement primitive to model the high-dimensional human-like reaching and puncture skill by human demonstrations; (b) Reinforcement learning is adopted to improve the adaptability of the varying via-RCM point tasks, which reduces the risks and cost for the practical surgical operation. The second section is model-based manipulator control, which combines the fuzzy adaptive control method with imitation learning: The learning-control strategy is proposed, where the high-level learning scheme aims at imitating the motor skill and generating the optimal trajectory for obstacle avoidance; while the lower-level control scheme focuses on the safety and stability of the robot's movement with unknown disturbances.

3 Chapter 4 proposes a policy improvement strategy to improve robustness and adaptability of imitation learning, where the low level learning is only for behaviors cloned with supervised learning, and high level learning constitutes policy improvement. The contributions of this chapter are described as follows:

(a) Exponential natural evolution strategies are proposed for learning the parameters of a policy that can improve the robustness and adaptability of the dynamical system, in which the low level learning is based solely on behavioral cloning using GMM, while the high level refers to the parameter learning of policy improvement based on considering robustness and adaptability by exponential NES; (b) exponential NES are also explored for learning the stiffness of the variable impedance control; the stiffness of the controller can be modified online according to the task's requirements; (c) the proposed method can learn the covariance matrix parameter which is used to modify the exploration noise in the parameter space; (d) offline learning and online robot experiments are conducted to demonstrate the effectiveness of the proposed scheme.

4 Chapter 5 concludes the thesis and discusses future research directions of robot control and imitation learning.

Chapter 2

Hierarchical Optimization Control for Redundant Manipulator

2.1 Optimization Control for OP-swab Robot

In this section, we will design and control a novel 9-DOFs robot to assist the COVID-19 OP-swab sampling [34]. The outbreak of novel coronavirus pneumonia (COVID-19) has caused mortality and morbidity worldwide. Oropharyngeal-swabs (OP-swab) sampling is widely used for the diagnosis of COVID-19 in the world. To avoid the clinical staff from being affected by the virus, we developed a novel 9-DOFs rigid-flexible coupling (RFC) robot to assist the COVID-19 OP-swab sampling. This robot composes of a visual system, a UR5 robot arm, a micro pneumatic actuator (MPA), and a force sensing system. The robot is expected to reduce risk and free up the clinical staff from the long-term repetitive sampling work through remote sampling. Compared with a rigid sampling robot, the developed force-sensing RFC robot can facilitate OP-swab sampling procedures in a safer and softer way. Additionally, a novel varying-parameters zeroing neural network-based optimization method is also proposed for motion planning of the 9-DOFs redundant manipulator. The developed robot system is validated by OP-swab sampling on both oral cavity phantoms and volunteers.

2.1.1 Introduction

The outbreak of novel coronavirus pneumonia (COVID-19) is affecting the entire world, which has caused a large number of deaths with an increase in the spread of COVID-19. To control the spread of COVID-19 at the early stage, oropharyngeal-swab (OP-swab) sampling is commonly adopted with respect to sample collection and specimen sources for diagnosis [35]. However, protecting the safety of medical staff during the sampling process raises a new challenge because of susceptibility to infection from person to person through respiratory droplets and contact transmission in an unprotected environment [36]. A variety of related studies have reported that respiratory droplets, feces and urine are the routes of transmission [37]. To address these issues in OP-swab sampling, robotics could play a key role in disease prevention.

Considering the high risk of infection of COVID-19 and negative tests of nasopharyngeal swabs (NP-swab) caused by irregular sampling, it is necessary to design an OP-swab sampling robot to assist healthcare staff through remote access. Robot-assisted OP-swab sampling is a promising technique because it relieves the burden from medical staff, is convenient for large-scale deployment, is cost-effective, and offers sampling standardization. As reported in [38], an OP-swab robot could speed up the sampling process especially because there is a lack of qualified healthcare workers. In [39] and [40], a semi-automatic OP-swab robot (compared to the traditional human method) was developed with a teleoperation scheme that achieved

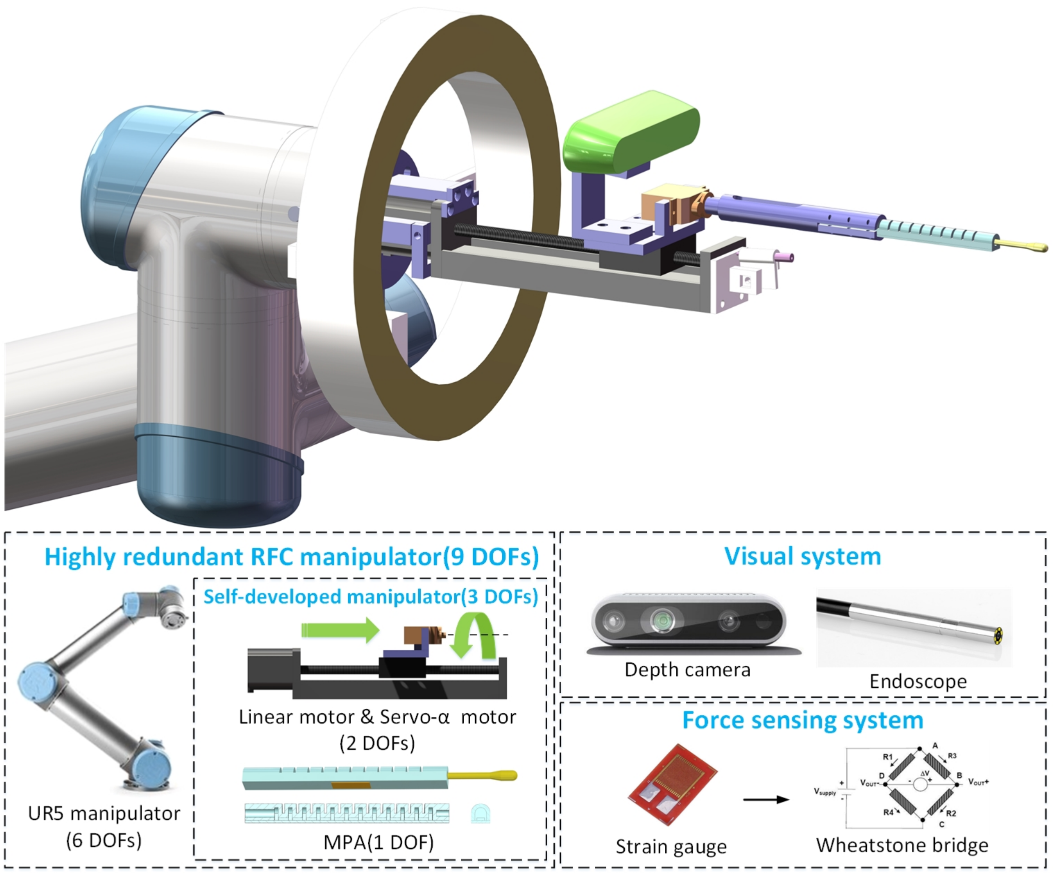


Figure 2.1: 9-degree-of-freedom (DOF) redundant rigid–flexible coupling robot.

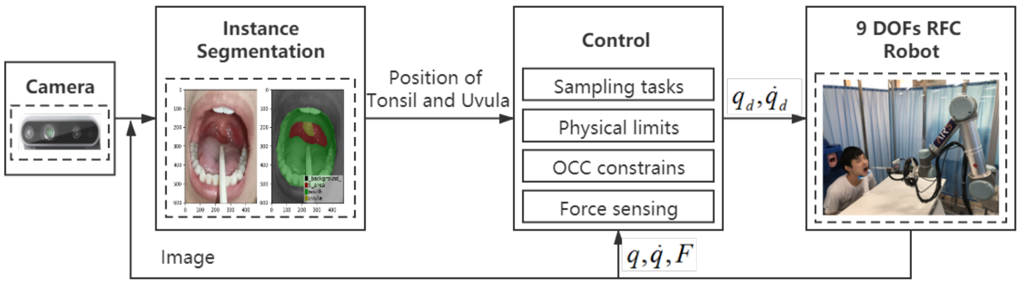


Figure 2.2: Control framework of COVID-19 OP-swab sampling with a redundant rigid–flexible coupling robot.

good performance in clinical practice with a 95% success rate. The OP-swab robot system mainly includes a snake-like manipulator, two haptic devices (Omega.3 and Omega.6) for teleoperation, an endoscope for visualization (assistance operating in the oral cavity) and a force-sensing system for safety protection. All OP-swab sampling processes were controlled by experts' teleoperation, where one haptic device (Omega.3) control the tongue depressor and another haptic device (Omega.6) operate the manipulator. Wang *et al.* [41] have designed a low-cost robot for assistance in sampling of NP-swab; a swab gripper is attached to the rotation link with its extruded active 2-degree-of-freedom (DOF) end-effector for actuating the swab and a generic 6-DOF passive arm for global positioning. In [42], the team from the University of Southern Denmark and the Lifeline Robotics company developed the first automatic swab robot for COVID-19 sampling. In [42], the robot is integrated with a UR5 manipulator, a visual system and dexterous rotatable rigid connectors assembled with OP-swab which could perform the sampling task quickly using an automated scheme.

Of note, medical security remains vital to OP-swab sampling because human throat is fragile and easily injured. Nonetheless, although many desirable results were obtained, the abovementioned end-effectors were designed on the basis of a rigid body structure, which may cause physical injuries of the oral cavity during improper operation or other medical malpractices. On the basis of our previous work [43], a novel micro-pneumatic actuator (MPA) for throat swab sampling is developed to achieve flexible collection and integrate a force sensor, which provides safe, stable, and reliable sampling experience, as shown in Fig. 2.1. Furthermore, force-sensing actuation offers compliance, which is helpful against shocks, particularly during interaction with oral cavity. Compared with the rigid body-based sampling robots [39, 41] and [42], the developed MPA is made of soft material, which is safer and lighter. To ensure operation safety, we develop a novel strain gauge sensor attached to MPA which means that the proposed MPA is safer than that in [42]. Moreover, MPA is smaller and has a 7.5 mm cross-sectional diameter, which is convenient for working in human oral cavity.

To avoid contact with the OP-swab specimen and collision with the manipulator, it is essential to consider the constraints of the oral cavity space during motion planning. Inspired by our previous work [44] on remote center of motion (RCM) techniques in minimally invasive surgical robots, the second to the last link of the OP-swab sampling robot is constrained with the oral cavity center (OCC) constraint, which guarantees the absence of collision between the inserted end-effector with the oral cavity. Although mechanical implementation is usually safer for an open minimally invasive surgery, programmable RCM is cost effective and convenient to be implemented [45]. The traditional Cartesian adaptive control in [45] was applied for surgical control with the RCM constraint but without considering the physical limits of the manipulator, such as the joint angle and velocity. For practical applications, we propose an optimization strategy for the multi-constraint problem, where the joint angle and velocity, and the OCC constraint-based kinematic model are derived as a quadratic programming problem. In this study, a comparison between the varying-parameter zeroing neural network (VP-ZNN) and the traditional gradient descent method [46] is proposed for the optimization problem that has achieved superior performance, which can converge at the finite time owing to the optimal solution changing over time.

To test the developed OP-swab sampling robot, which is shown in Fig. 2.1, the experiments are simultaneously conducted in both the human oral cavity phantom and volunteers, regarding the sampling time, operation complexity for medical staff, safety and effectiveness. The framework of COVID-19 OP-swab sampling using the RFC robot is shown in Fig. 2.2.

The section is organized as follows. Section. 2.1.2 depicts the design of the RFC robot. Section. 2.1.4 introduces the optimization problem and methodology. Experimental results are presented in Section. 2.1.5. The conclusion and future work are described in Section. 2.1.6.

2.1.2 Design of the Rigid–Flexible Coupling Robot

In this section, we will introduce the concept of robot design and the tests of the MPA and force sensing modules that would calibrated the movement and sensing ability.

Redundant Rigid–Flexible Coupling Robot

The rigid–flexible coupling robot for OP-swab sampling consists of a UR5 manipulator and a self-developed rigid–flexible coupling manipulator. The overall structure of the 9-DOF redundant rigid–flexible coupling robot is shown in Fig. 2.1. The self-developed manipulator has 3 DOFs, including a linear motor (prismatic joint), a servo–α motor (revolute joint), and an MPA that can change the offset distance of throat swab from the center.

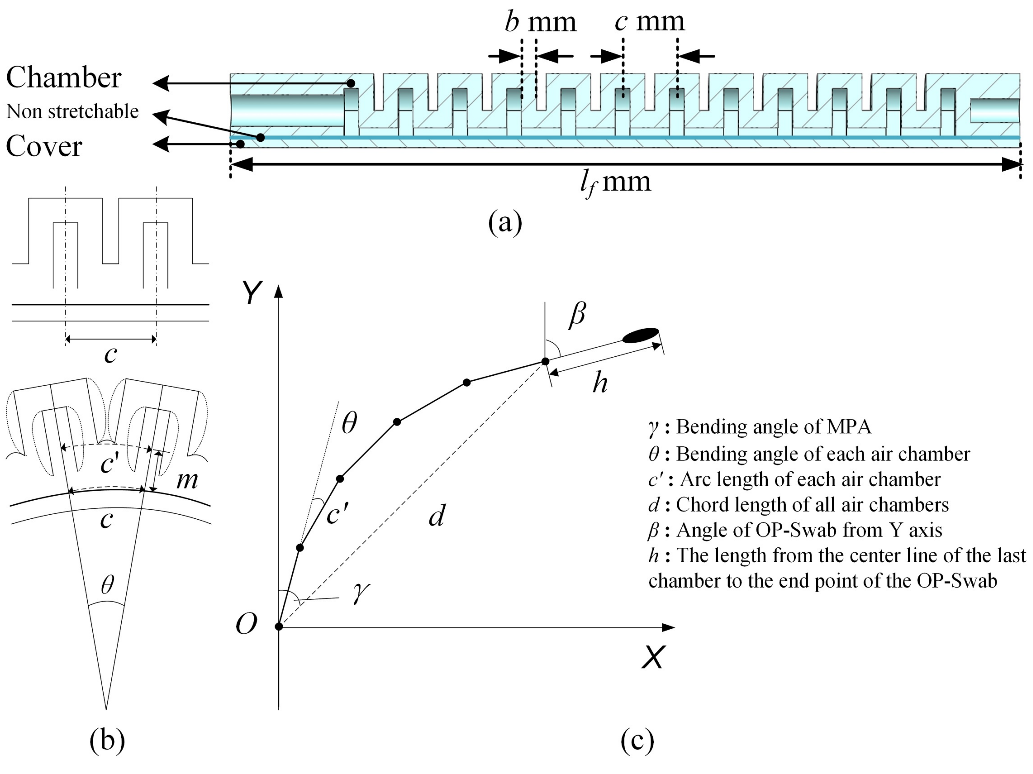

b mm
c mm
Chamber
Non stretchable
Cover
l_f mm
(a)
c
c'
m
c
θ
(b)
Y
β
h
θ
c'
d
γ
O
X
(c)
γ : Bending angle of MPA
θ : Bending angle of each air chamber
c' : Arc length of each air chamber
d : Chord length of all air chambers
β : Angle of OP-Swab from Y axis
h : The length from the center line of the last chamber to the end point of the OP-Swab

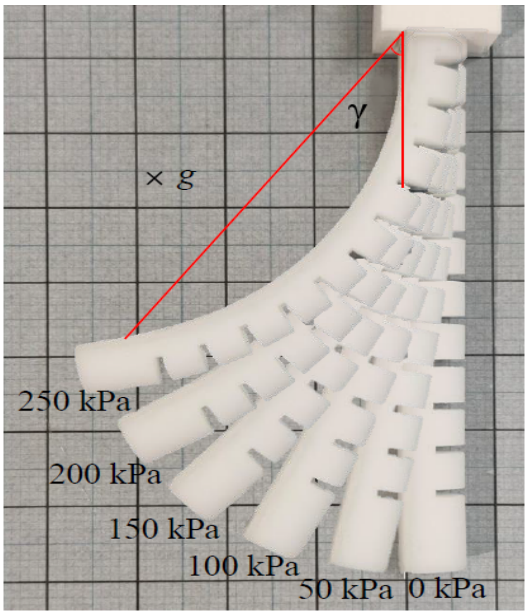
Figure 2.3: (a) Cross section of the micro-pneumatic actuator(MPA). (b) bending process of MPA. and (c) 2D modeling of MPA.

Design, Modeling, and Fabrication of MPA

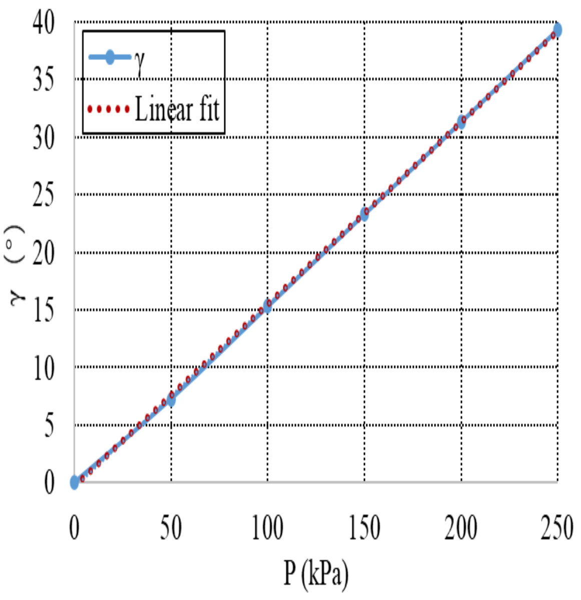
The design of MPA refers to the wrinkle shape of the elastomer robot [47]. Figure 2.3 shows that MPA consists of n effective air chambers. The sectional view of MPA is shown in Fig. 2.3(a), which contains a chamber part and a cover part. These two parts are bonded together to form a single-DOF MPA. b is the wall thickness of the air chamber; c is the distance between the air chambers, and l_f is the total length of MPA. When MPA is inflated, the air chambers will expand and repel each other, which causes MPA to bend. The bending process of MPA is shown in Fig. 2.3(b).

Assuming that all air chambers have the same bending under the same pressure, it is easy to obtain the geometric relation:

$$\theta = \beta/n = 2\gamma/n \tag{2.1}$$



(a)



(b)

Figure 2.4: Experimental results of MPA. (a) Deformation of MPA with a 50-shore hardness under different air pressures. (b) Relationship between bending angle γ and air pressure.

Owing to the small value of θ after bending, the chord length is approximately equal to the arc length at the non-tensile layer. The distance between the center line and the non-tensile layer is m, and the arc length c' is also approximately equal to the chord length; thus, c' is

$$c' = c + 2m\sin(\theta/2) \tag{2.2}$$

The overall motion model is shown in Fig. 2.3 (c), where the chord length d is

$$d = \frac{c'\sin(n\theta/2)}{\sin(\theta/2)} \tag{2.3}$$

The forward kinematics of MPA is modeled as

$$\begin{cases} x = d\sin\left(n\frac{\theta}{2}\right) + h\sin(\beta) \\ y = d\cos\left(n\frac{\theta}{2}\right) + h\cos(\beta) \end{cases} \tag{2.4}$$

According to the abovementioned information, we design MPA with 12 air chambers. However, the number of effective air chambers, i.e., the number of effective joints is $n = 11$ because only half of the head chamber and tail chamber will bend. The wall thickness b of each chamber is 1.5 mm , the width of each air chamber c is 4.5 mm and the total length l_f is 80 mm.

The chamber part and the cover part of MPA are made by 3D printing with a 50-shore hardness. The air pressure is controlled by the proportional valve ITV1050-312L. Then, a deformation experiment of MPA was conducted, and the experimental results are shown in Fig. 2.4(a), where the gravity is directly perpendicularly to the plane of the chapter. The relationship between the bending angle γ and air pressure is expressed in Fig. 2.4(b). It is observed that the bending angle γ is approximately proportional to the air pressure. After data fitting, the linear equation for the angle γ (°) and pressure P (kPa) is written as

$$\gamma = 0.160325P - 0.731885 \tag{2.5}$$

The correlation coefficient R is 0.99990 and the standard deviation S is 0.6464, which confirms that γ and P are linearly dependent. By combining equations (2.1) and (2.5), we obtain

$$\theta = 0.02915P - 0.13307 \tag{2.6}$$

Finally, the end-point position of MPA can be calculated by combining equations (2.2), (2.3), (2.4), and (2.6).

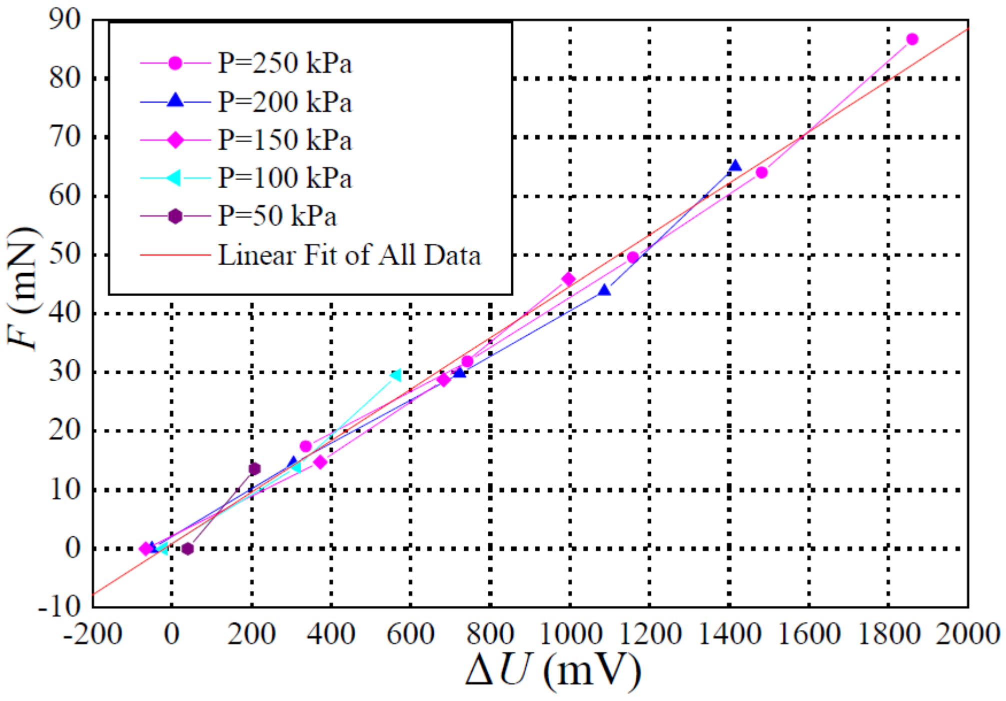


Figure 2.5: Relationship between F and ΔU.

2.1.3 Force Sensing

To detect the contact force F during OP-swab sampling, a strain gauge sensor (BF350-3AA23T0) attached to MPA is adopted. It is necessary to explore the mapping relationship between the force and voltage for force measurement calibration. Actually, the current output voltage U_{now} is related to two factors, i.e., voltage ΔU generated by the force, and voltage $U_{F=0}$ caused by the air pressure without a load. Therefore, we have $\Delta U = U_{\text{now}} - U_{F=0}$. After 20 group experiments, the relationship between no-load voltage and air pressure is obtained by linear fitting: $U_{F=0} = -10.693P + 2770.571$ (mV). Force calibration was performed using the electronic balance (CX-668). The relationship between ΔU and F is approximately linear; thus, we know that

$$F = 0.04378\Delta U + 0.88828 \tag{2.7}$$

The details are shown in Fig. 2.5. The correlation coefficient R is 0.99461 and the standard deviation S is 2.65733.

2.1.4 Optimization Control for the 9-DOF Redundant Manipulator

Manipulator Kinematics Model with OCC Constraints

The coordinate system of the 9-DOF redundant RFC manipulator is shown in Fig. 2.6. The forward kinematics model [48, 49] of the 9-DOF RFC robot can be defined as follows:

$$J\dot{q} = \dot{l}_d \tag{2.8}$$

where $J \in \mathbb{R}^{m\times n}$ denotes the Jacobian matrix. The joint angle and velocity constraints are defined as follows:

$$q_i^- \le q_i \le q_i^+ \tag{2.9}$$
$$\dot{q}_i^- \le \dot{q}_i \le \dot{q}_i^+ \tag{2.10}$$

where q_i^- and q_i^+ present the lower and upper bounds of q_i, respectively; and $\dot{q}_i^-$ and $\dot{q}_i^+$ denote the lower and upper bounds of the joint velocity $\dot{q}$, respectively.

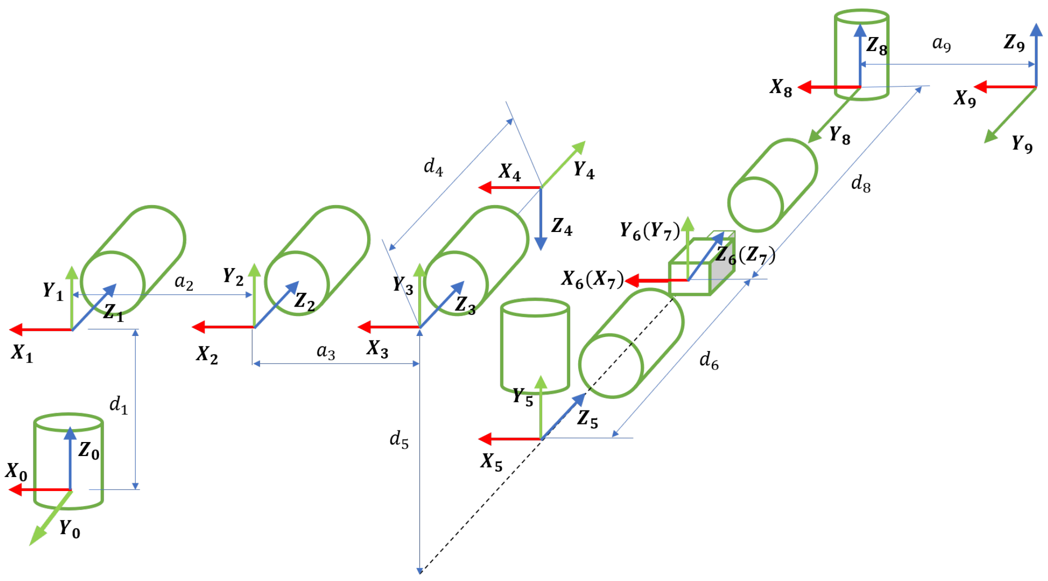


Figure 2.6: Coordinate system of the 9-DOF redundant RFC manipulator.

Actually, the constraints in (2.9) can be converted as

$$\sigma\left(q_i^- - q_i\right) \le \dot{q}_i \le \sigma\left(q_i^+ - q_i\right) \tag{2.11}$$

where σ is the positive constant. Therefore, according to (2.10) and (2.11), the joint angle and velocity constraints can be rewritten as a new constraint in the velocity level:

$$\begin{aligned} &\rho_i^- \le \dot{q}_i \le \rho_i^+ \\ &\rho_i^- = \max\left\{\dot{q}_i^-, \sigma\left(q_i^- - q_i\right)\right\} \\ &\rho_i^+ = \min\left\{\dot{q}_i^+, \sigma\left(q_i^+ - q_i\right)\right\} \end{aligned} \tag{2.12}$$

Because the RFC robot is a 9-DOF highly redundant manipulator, there are infinite solutions of $\dot{q}$ in (2.8) by the inverse kinematics. However, the convergence rate, accuracy, and computational complexity of the pseudoinverse-type solution in inverse kinematics cannot satisfy the requirements. Consequently, we need to identify an optimization solution under multiple constraints; thus, the inverse kinematics problem can be expressed as a new optimization problem. The first-priority optimization problem associated with OP-swab sampling is expressed as follows:

$$\min \quad \frac{1}{2}\dot{q}^T W \dot{q} \tag{2.13}$$

$$\text{s.t.} \quad J(q)\dot{q} = \dot{l}_d \tag{2.14}$$

$$\rho^- \leq \dot{q} \leq \rho^+ \tag{2.15}$$

where $\dot{l}_d$ represents the reference velocity associated with OP-swab sampling tasks. The weight matrix M is set as the identity matrix.

Considering the OCC constraint, the link-L_{n-1} of the RFC manipulator needs to pass through the OCC, and L_n performs the sampling tasks, which is different from the traditional RCM constraint in the last link. The geometric relationship is shown in Fig. 2.7. For the n-DOF OP-swab sampling robot, the forward kinematics mapping function of Cartesian position $l_{n-2} \in \mathbb{R}^m$ and $l_{n-1} \in \mathbb{R}^m$ can be defined as follows:

$$\begin{aligned} l_{n-2} &= f_{n-2}(q) \\ l_{n-1} &= f_{n-1}(q) \end{aligned} \tag{2.16}$$

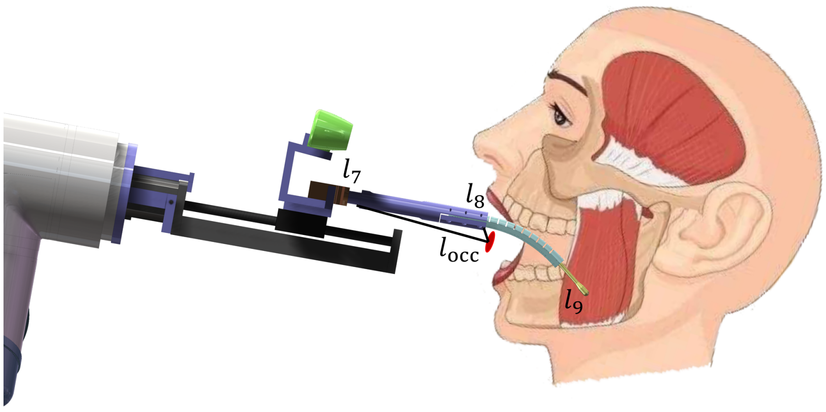


Figure 2.7: Constraint with the oral cavity center.

Unlike common RCM constraints, l_{occ} should be always on the straight line straight line between l_{n-1} and l_{n-2} (second to the last link), where l_{n-1} is the end position of the link L_{n-1} and l_{n-2} is the end position of the link L_{n-2}. During the actual OP-swab sampling, we want to keep the error of the OCC constraint E_{occ} as small as possible. Lines 1 and 2 are constructed as follows: $\overrightarrow{l_{n-2}r_{n-1}} = l_{n-1} - l_{n-2}$, $\overrightarrow{l_{n-2}l_{occ}} = l_{occ} - l_{n-2}$, respectively. Utilizing the relationship between E_{occ} and the vector projection, E_{occ} can be further written as follows:

$$E_{occ} = \frac{\overrightarrow{l_{n-2}l_{occ}} \times \overrightarrow{l_{n-2}l_{n-1}}}{L} \tag{2.17}$$

where $L = \|l_{n-2} - l_{n-1}\|$ is the length of the second to the last link. The derivative of E_{occ} in (2.17) with respect to time is reformulated as follows:

$$J_{occ}\dot{q} = \dot{E}_{occ} \tag{2.18}$$

where $J_{occ} \in \mathbb{R}^{m \times n}$ denotes the Jacobian matrix of the OCC constraint. Regarding the OCC constraint task, we want to keep the OCC error E_{occ} at the minimum value.

In this section, it is necessary to find a feasible solution for multiple constraints and guarantee that the tracking error and OCC error always remain in a small range. For COVID-19 sampling tasks, we aim to reformulate OCC (2.18) and joint physical limits (2.15) in an optimization scheme and design a method to solve the optimization problem. Consequently, by simultaneously taking OCC, end-effector task, and joint physical constraints into account, the new multi-task optimization problem can be formulated as:

$$\begin{aligned} \min \quad & \frac{1}{2}\dot{q}^T W \dot{q} \\ \text{s.t.} \quad & J\dot{q} = v_d \\ & J_{occ}\dot{q} = v_{occ} \\ & \rho^- \leq \dot{q} \leq \rho^+ \end{aligned} \tag{2.19}$$

where $v_d = \dot{l}_{nd}$ and $v_{occ} = 0$. Without the explicit expression of l_n and e_{occ}, the actual trajectory will drift and the position error cannot converge to zero from a random initial position. To overcome this issue in (2.19), a feedback item associated with position signals is integrated into v_d and v_{occ}:

$$v_d = -k_1(f_n(q) - l_{nd}) + \dot{l}_{nd} \tag{2.20}$$

$$v_{occ} = -k_2(l_{occ}) \tag{2.21}$$

Of note, we should manage the priority strategy for the multiple-task optimization problem by different weights. Therefore, the objective function is defined as

$$F(\dot{q}) = \frac{\varepsilon_0}{2}\dot{q}^T\dot{q} + \frac{\varepsilon_1}{2}||J\dot{q} - v_d||^2 + \frac{\varepsilon_2}{2}||J_{occ}\dot{q} - v_{occ}||^2 \tag{2.22}$$

The optimization problem in (2.22) can be rewritten as follows:

$$\begin{aligned} \min \quad & F(\dot{q}) \\ \text{s.t.} \quad & J\dot{q} = v_d \\ & J_{occ}\dot{q} = v_{occ} \\ & \rho^- \leq \dot{q} \leq \rho^+ \end{aligned} \tag{2.23}$$

where $\varepsilon_0 > 0$, $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ are the constants employed to prioritize different tasks.

Neural Network Design

In this section, we design the varying-parameter zeroing neural network (VP-ZNN) [50, 51] to solve the optimization problem in (2.22). The multiple tasks optimization problem in (2.22) is firstly converted into a equivalent problem, and thus the varying parameter zeroing neural network is employed to solve it.

In this section, we describe the design of the VP-ZNN, which is used to solve the optimization problem in (2.22). First, the multiple-task optimization problem in (2.22) is converted into an equivalent problem; thus, the VP-ZNN is employed to solve it.

To obtain the equivalent problem from (2.22), the Lagrange function The Lagrange function of (2.22) constraints is formulated as follows:

$$\mathcal{L}(\dot{q}, \xi_1, \xi_2) = \frac{\varepsilon_1}{2}||J\dot{q} - v_d||^2 + \frac{\varepsilon_2}{2}||J_{occ}\dot{q} - v_{occ}||^2 + \frac{\varepsilon_0}{2}\dot{q}^T\dot{q} + \xi_1^T(v_d - J\dot{q}) + \xi_2^T(v_{occ} - J_{occ}\dot{q}) \tag{2.24}$$

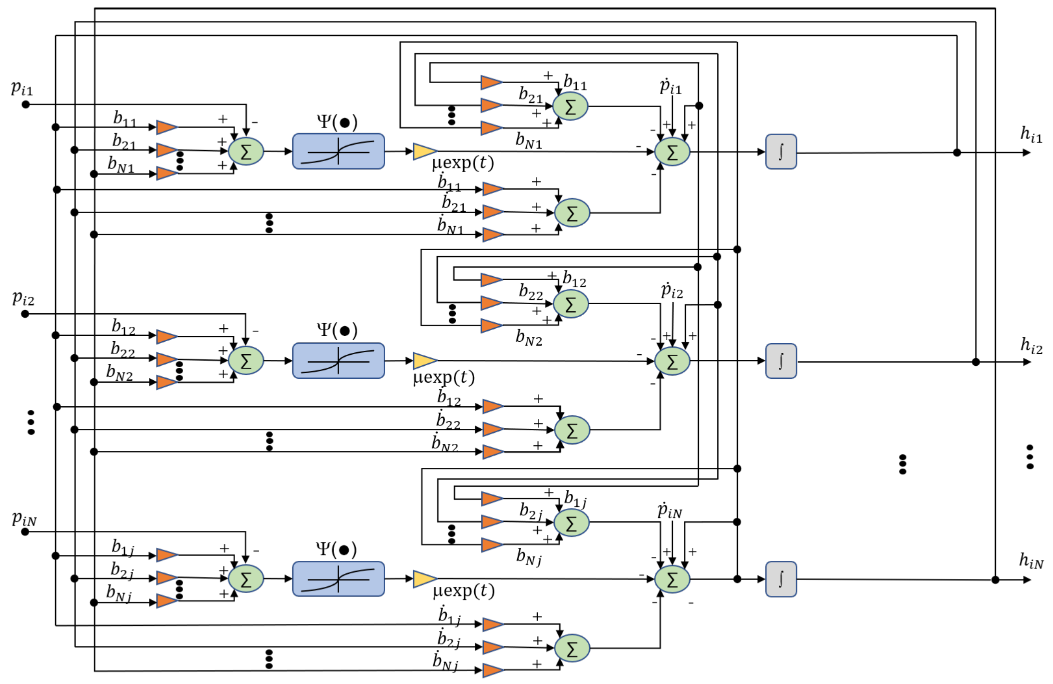


Figure 2.8: Structure of VP-ZNN.

where $\xi_1 \in \mathbb{R}^m$ and $\xi_2 \in \mathbb{R}^m$; $\nabla\mathcal{L} = \left[\frac{\partial\mathcal{L}}{\partial\dot{q}}, \frac{\partial\mathcal{L}}{\partial\xi_1}, \frac{\partial\mathcal{L}}{\partial\xi_2}\right]^T$ indicates the gradient of (2.24). As the KKT condition defined in [52], if $\nabla\mathcal{L}$ is continuous, the optimization solution satisfies the following condition:

$$\nabla\mathcal{L} = 0 \tag{2.25}$$

The state decision variable $l(t) = [\dot{q}, \xi_1, \xi_2]^T \in R^{n+2m}$. The problem in (2.25) is equivalent to the following:

$$B(t)l(t) = P(t) \tag{2.26}$$

where $B \in \mathbb{R}^{(n+2m)\times(n+2m)}$ and $P(t) \in \mathbb{R}^{(n+2m)}$. The bounds of state variable $\dot{q}^b$ are expressed as

$$\dot{q}^b = \begin{cases} \rho^+, & \dot{q} > \rho^+ \\ \rho, & \rho^- \le \dot{q} \le \rho^+ \\ \rho^-, & \dot{q} < \rho^- \end{cases}$$

Therefore, the bound of state variable $l(t)$ are defined as

$$l^b(t) = \begin{bmatrix} \dot{q}^b & \xi_1^b & \xi_2^b \end{bmatrix},\ \xi_1^b, \xi_2^b \in \mathbb{R}$$

The error model of the novel VP-ZNN is expressed as

$$e(t) = B(t)l(t) - P(t) \tag{2.27}$$

To ensure the model error convergence to zero, we define the following formulation,

$$\frac{de(t)}{dt} = -\mu\exp(t)\Psi(e(t)) \tag{2.28}$$
$$\Psi(e(t)) = \frac{(1+\exp(-\delta))(1-\exp(-\delta e_i(t)))}{(1-\exp(-\delta))(1+\exp(-\delta e_i(t)))}$$

where $\mu > 0$ is the constant that can adjust the convergence rate; $\Psi(e_i(t))$ denotes the activation function, and $\xi \geq 2$, which makes $0 \leq |e_i(t)| \leq 11$. Clearly, the error in (2.28) converges to zero with exponential convergence.

The function in (2.28) is expanded as

$$B(t)\dot{l}(t) = -\dot{B}(t)l(t) + \dot{P}(t) - \mu \exp(t)\Psi\left(l(t) - l^b(t) + B(t)l(t) - P(t)\right) \tag{2.29}$$

We further modify the VP-ZNN in (2.29) as

$$\begin{aligned}\dot{l}(t) = {} & (I - B(t))\dot{l}(t) - \dot{B}(t)l(t) + \dot{P}(t) \\ & - \mu \exp(t)\Psi\left(l(t) - l^b(t) + B(t)l(t) - P(t)\right)\end{aligned} \tag{2.30}$$

For comparison, the traditional gradient descent-based recurrent neural network is denoted as

$$\dot{l}(t) = \mu(-l(t) + P_\Omega[l(t) - (B(t)l(t) - P(t))]) \tag{2.31}$$

For the online solving process, the neural network consists of N neurons and the neural network is designed as,

$$\dot{l}_i = \sum_{j=1}^{N}(I_{ij} - B_{ij}(t))\dot{l}_j(t) - \sum_{j=1}^{N}\dot{B}_{ij}(t)l_j(t) - \mu \exp(t)\Psi\left(\sum_{j=1}^{N}B_{ij}(t)l_j(t) - P_i(t)\right) + \dot{P}_i(t) \tag{2.32}$$

The structure of varying-parameters of zeroing neural network is shown in Fig. 2.8.

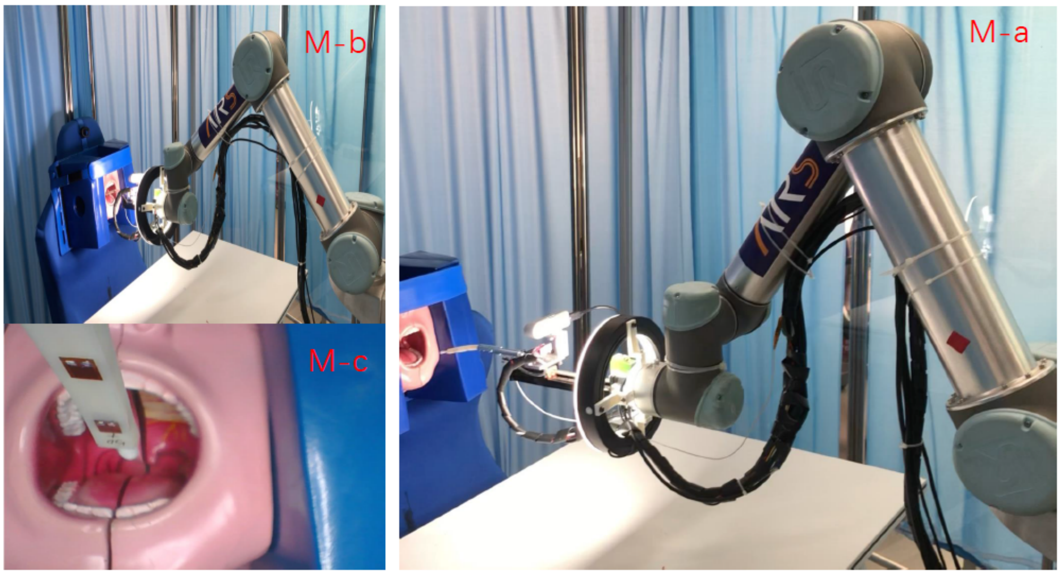


Figure 2.9: Demonstration with an oral cavity phantom. In our experiments, the 1:1 human oral cavity is tested.

2.1.5 Experiments

In this section, we present the tests with an oral cavity phantom and volunteers using the OP-swab robot system, approval from the Institutional Review Board of The Chinese University of Hong Kong, Shenzhen was obtained(IRB number CUHKSZ-D-20210002). Then, the advantages and disadvantages are summarized and discussed. The parameters of the VP-ZNN are set as: $\varepsilon_0 = 0.1$, $\varepsilon_1 = 10$, $\varepsilon_2 = 10$, $\mu = 0.01$, $k_1 = 10$, and $k_2 = 10$.

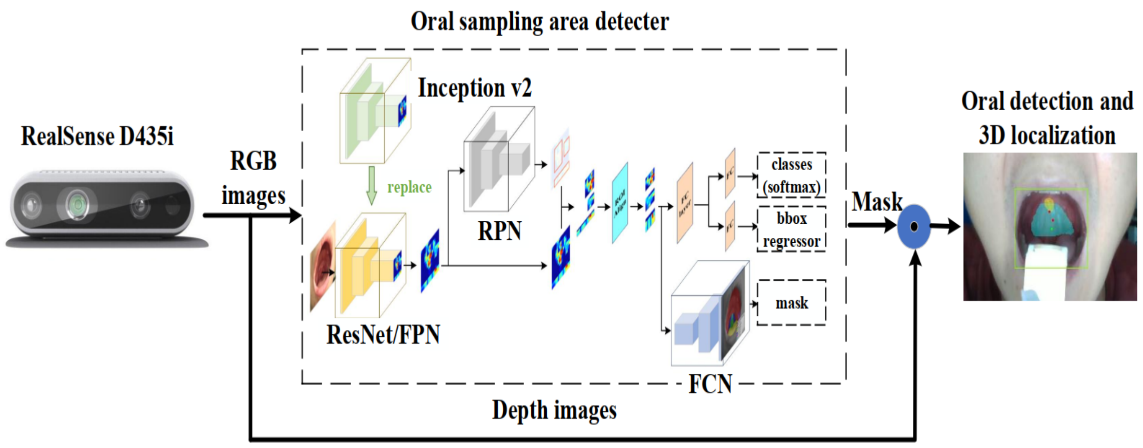


Figure 2.10: The visual system for oral detection, segmentation and localization using Mask R-CNN [53].

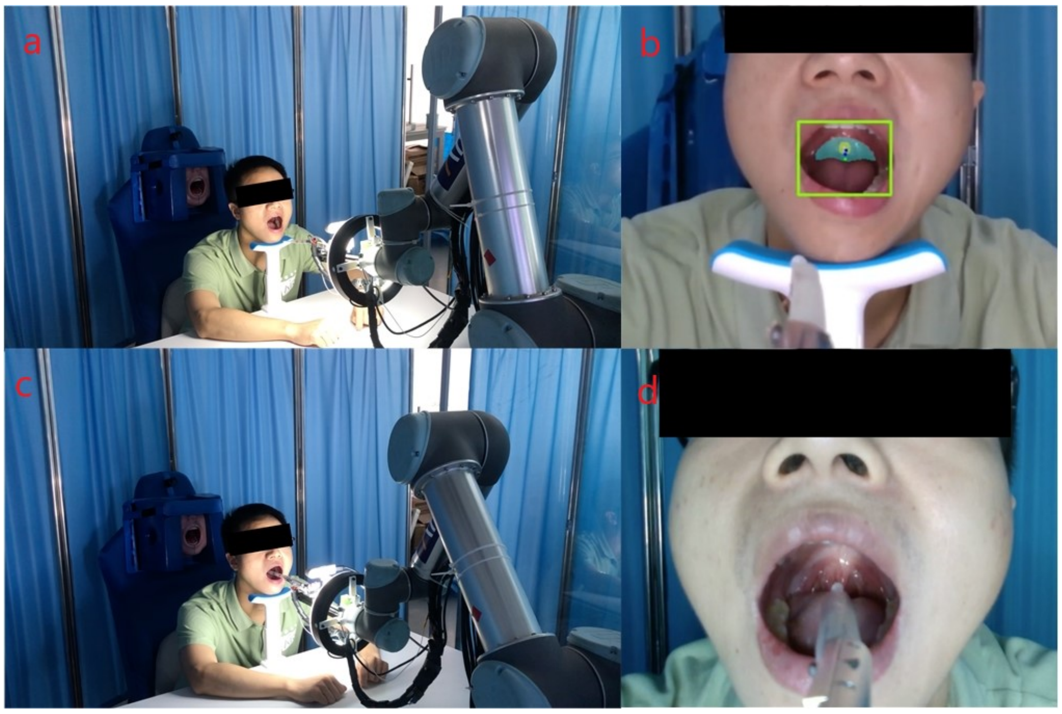
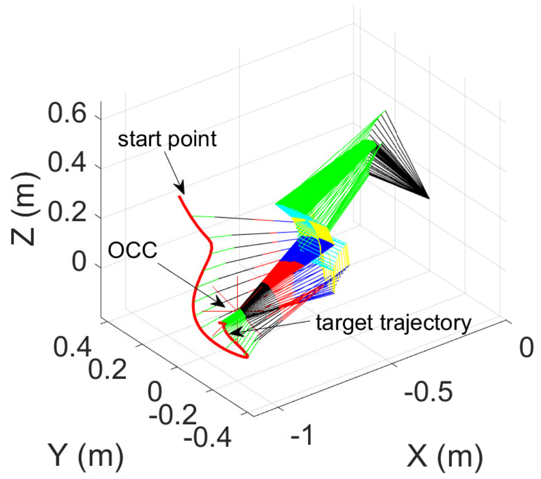
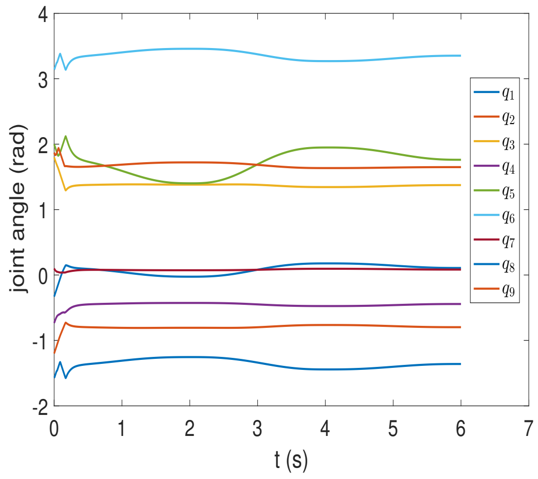


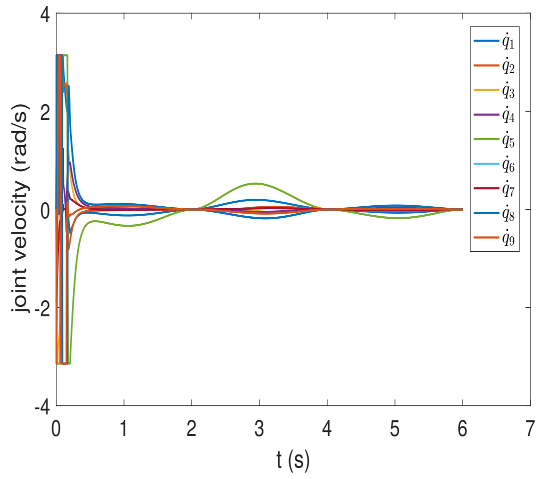
Figure 2.11: Sampling process from vision detection to sampling tasks. In our experiments, many volunteers were tested, and the recognition rate, control precision, sampling time, and sampling contact force were recorded.



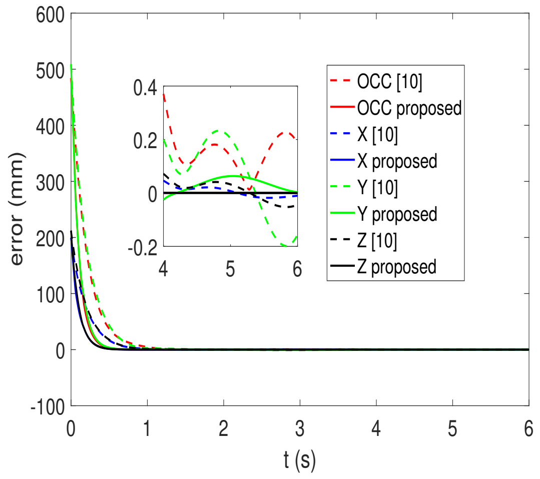
(a) Motion trajectories.



(b) Joint trajectories.



(c) Joint angle velocity.



(d) Tracking errors.

Figure 2.12: Tracking results: the manipulator track of the desired trajectories (sampling tasks) in the Cartesian space. (a) The 3D trajectories of the manipulator; (b) the joint angle trajectories; (c) the joint velocity trajectories (located in bounds); and (d) comparison experiments with [44].

Demonstration with Phantom Experiments

For safety reasons, first, we conduct the experiments with an oral cavity phantom, which will help to confirm the safety of the OP-swab robot system. The oral cavity phantom has a 1:1 size of human oral cavity, which enables natural simulation of workflow across OP-swab sampling and allows us to work extremely close to conditions in practice. The procedures of experiments are as follows:

1 The phantom oral cavity is detected and segmented with the RealSense Camera, which is configured on the RFC manipulator. Because the sampling areas are the left and right tonsils and the palate area, we recognize and locate the target position by Mask R-CNN.

2 The desired sampling trajectories are obtained by online motion planning with an oral cavity constraint, in which we locate the target position in the first step and then generate the trajectories. The desired trajectories are a piecewise straight line from the palate area to the left tonsils to the right tonsils in the Cartesian space, where the relationship between each axis and time is the minimum jerk curve.

3 The robot is driven to perform the sampling tasks. In this phase, the desired joint trajectories are obtained from optimization control methods. Moreover, multiple protection mechanisms with force sensing are activated during robot operation.

The experimental scenarios of the oral cavity phantom are shown in Fig. 2.9. We collect the dataset for the oral cavity phantom which is trained by Mask R-CNN [54]. To obtain the category and location of the oral sampling areas, an oral visual detector is trained and obtained. First, an oral cavity image dataset is created by collecting from RealSense D435i. Second, the Mask R-CNN with a Inception v4 module as its backbone is trained on the oral cavity image dataset and an oral cavity detector is obtained. Finally, oral cavity coordinates are mapped to depth images to obtain the depth values of the oral cavity. We achieved a 95% recognition success rate on the model, and the sampling time is less than 20 s. The structure of visual system is shown in Fig. 2.10.

Demonstration with Volunteers Experiments

On the basis of many trials of the OP-swab RFC robot system with the oral cavity phantom, the robustness and safety were significantly improved. Then, we conducted a number of volunteer experiments and achieved milestone significance for the future clinical application of OP-swab sampling. The procedures of experiments are the same as those described in Sec. 2.1.5. The dataset of the human oral cavity is collected by 29 volunteers and trained using Mask R-CNN.

The experimental scenarios of different subjects are shown in Fig. 2.11. The desired trajectories are the same as in Sec. 2.1.5. We design the piecewise straight-line trajectories from the palate to the right tonsils to the left tonsils in the Cartesian space, and the relationship between each axis and time is the minimum jerk curve. Figure 2.12 shows the tracking results associated with one of the trials of planning trajectories. The motion trajectories in the Cartesian space are shown in Fig. 2.12a. Figures 2.12b and 2.12c show that the joint trajectories located in the physical limits and joint velocity are smooth within the velocity limits, respectively. Figure 2.12d shows the tracking error in the Cartesian space, where all errors rapidly converge to a small value (0.02 mm). In addition, we added a comparison experiment shown in Fig. 2.12d, where the proposed method has a faster convergence rate and smaller errors than [44]. The average recognition rate, control precision, sampling time, and sampling contact force are recorded and shown in Table. 2.1. The comparative experiments were conducted to examine the effectiveness of robotic OP-swab sampling. The swab quality

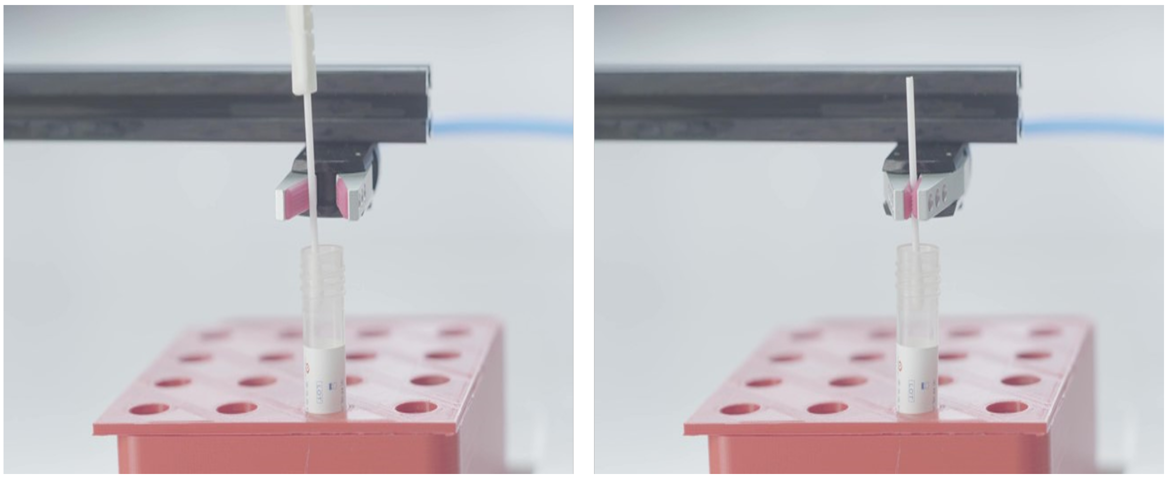


Figure 2.13: PCR-test.

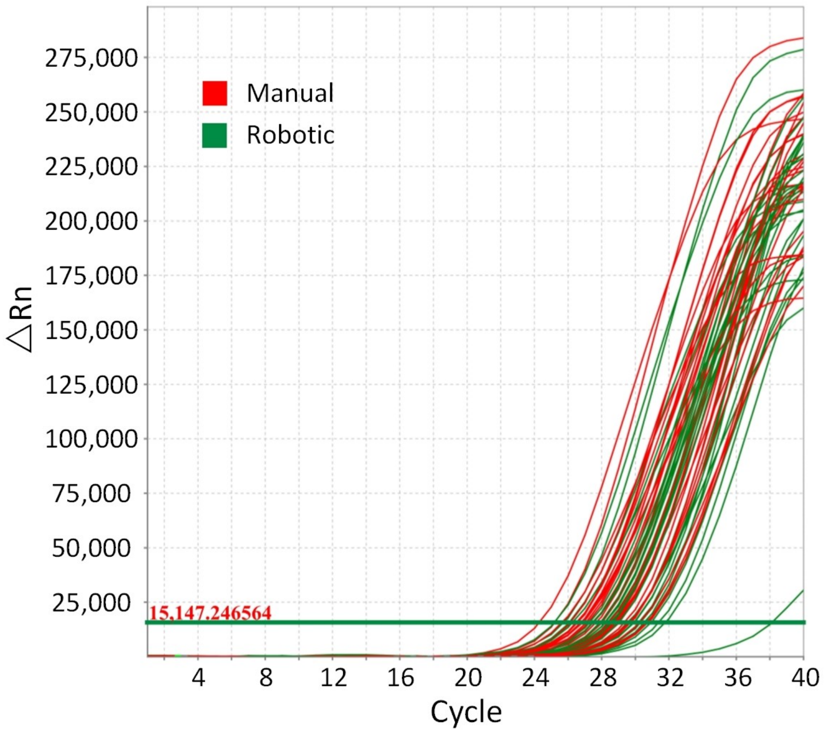


Figure 2.14: RT-PCR test results.

was verified according to the threshold cycle (Ct) value of the selected reference gene (RNase P) by the RT-PCR test [55], which is shown in Fig. 2.13.

Figure 2.14 shows the RT-PCR test results compared with those for the manual scheme. OP-swab with Ct values of ≤ 37 and > 37 are considered as qualified and unqualified samples, respectively, and the details are shown in Table 2.2. The test results show that the samples are qualified.

Table 2.1: Average sampling parameters.

Metric \ Types	Phantom	Volunteers
Recognition rate	95%	93%
Control precision	$\leq$0.2 mm	$\leq$0.2 mm
Sampling time	18 ± 2 s	20 ± 4 s
Sampling force	≈ 150 mN	≈ 150 mN

Table 2.2: Ct distribution.

Ct	(24–27)	(27–30)	(30–33)	(33–37)	>37	Qualified rate
Manual	11	16	3	0	0	100%
Robotic	5	21	3	0	1	96.67%

2.1.6 Summary

In this study, we designed a 9-DOF redundant RFC robot to assist the COVID-19 OP-swab sampling. Moreover, we formulate the sampling tasks, physical limits, and OCC constraints as a novel optimization problem. Then, a VP-ZNN is proposed to solve the multi-constraint optimization problem online. The experimental results of the oral cavity phantom and volunteer experiments demonstrate the effectiveness of the designed robot system and proposed control methods. The average sampling time on phantoms and volunteers are 18 s and 20 s respectively. To maintain the sterility of the arm and effector between patients, the MPA is designed as a disposable device with a quick connector that is easily to be replaced, where the disposable protective film is attached to the MPA. In the future, we will focus on improving the robustness of the system and move forward to clinical testing.

2.2 Hierarchical Optimization Control for Surgical Robot

In this section, we will discuss the hierarchical optimization control method with multi-task constraints for surgical robot [56][44][57]. For the time varying optimization problem, the tracking error cannot converge to zero at the finite time because of the optimal solution changing over time. This section proposes a novel varying parameter recurrent neural network (VPRNN) based hierarchical optimization of a 7-DOF surgical manipulator for Robot-Assisted Minimally Invasive Surgery (RAMIS), which guarantees task tracking, Remote Center of Motion (RCM) and manipulability index optimization. A theoretically grounded hierarchical optimization framework based is introduced to control multiple tasks based on their priority. Finally, the effectiveness of the proposed control strategy is demonstrated with both simulation and experimental results. The results show that the proposed VPRNN-based method can optimal three tasks at the same time and have better performance than previous work.

2.2.1 Introduction

Robotic technology is increasingly implemented to assist surgeon [58]. Robot-assisted surgery has several advantages such as better surgical accuracy, increased workspace, enhanced dexterity, and improved vision for surgeons [59]. Multiple tasks need to be considered during the surgical operation [60], such as the control of the surgical tip and the manipulability of the surgical manipulator, which is vital in Robot-Assisted Minimally Invasive Surgery (RAMIS). The accurate tracking control of the surgical tip is of vital importance for surgical operations [61]. Manipulability index [62, 63] of a surgical tool tip determines the maximum distance from singularities, flexible motion, and more extensive operational space of the robot manipulators. Usually, for surgical robot operation, multiple tasks are characterized by different priority levels. The multiple general tasks are listed as follows (T1–T3):

T1: The tracking control of the surgical tip must be accurate, which guarantees the success rate of surgery using the robot manipulator [64]. T2: In order to ensure the safety and rationality of the operation, the surgical tip should pass through a small incision in the abdominal wall of the patient. Kinematic constraints produced by each small incision should be respected, generally identified as the Remote Center of Motion (RCM) constraint [65]. T3: The manipulability of the robot manipulator should be enough to perform the surgical operation [66].

To effectively evaluate the multiple operational tasks on the robot manipulator, a lot of research activity has been attracted and performed in this area. Various approaches have been implemented to track the desired position and guarantee the RCM constraint at the same time using kinematic solutions [65, 67]. Similarly, Ali *et al.* [68] introduced an adaptive controller of sclera force and insertion depth to exploit the surgery maintaining the RCM constraint, as well. Except for fulfilling the RCM constraint, Jin *et al.* [69] proposed to adopt neural networks to optimize the manipulability index during the tracking without influence its accuracy. Nevertheless, few works are proposed to handle all the operational tasks at the same time.

In our previous work [70, 71], the hierarchical operational space formulation [72] is utilized to unite the three parts: the main surgical tracking task based on the Cartesian compliance control, the RCM constraint in its null space of the end-effector, and a manipulability optimization control in the null space of the robot wrist using a constrained quadratic programming (QP) [73]. Although it achieved better performance in terms of accuracy and manipulability index, the convergence rate is slow, and the tracking error is larger.

In this section, varying parameters recurrent neural network (VPRNN) based hierarchical control of a 7-DOF robot manipulator for robot-assisted minimally invasive surgery has been proposed, where it integrates multiple tasks based on their priority and guarantees task tracking [32], RCM, and manipulability optimization at the same time. The gradient descent neural network-based traditional methods cannot guarantee that the convergence of the error to 0 at the finite time because of the optimal solution changing over time. The proposed VPRNN framework can solve the time-vary QP problem with fast convergence rate performance, which is promising for online solution of the time-varying optimization problem [33]. The proposed methodology represents an advance concerning the manipulability optimization solution proposed in [70]. It utilizes a VPRNN [74] based hierarchical control to combine the multiple tasks into a single controller [75]. Furthermore, the overall convergence rate has also been improved with respect to our previous works [69, 70]. Finally, simulations and experiments applying a 7-DOF serial robot KUKA LWR4+ are performed to demonstrate the effectiveness of the proposed method.

The structure of this section is organized as follows. Section 2.2.2 describes the related works of the RA-MIS. The control method is presented in Section 2.2.3. In Section 2.2.4,

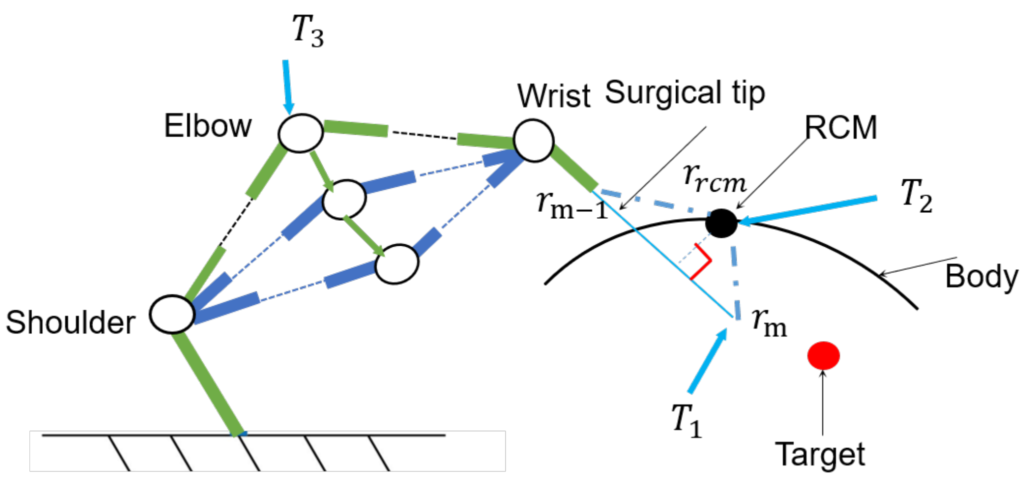


Figure 2.15: Multiple operational tasks in the Robot-assisted Minimally Invasive Surgery.

the performance of the proposed control system is illustrated by applying simulation and experiments. Conclusions are described in Section 2.2.5.

2.2.2 Related Works

Generally, there are active and passive RCM constraints. The passive constraint is designed mechanically, while the active method is well known to be accomplished by software and controller [75, 76]. Moreover, it is prevalent in non-clinical research since its low cost and flexible task space. As it is shown in Fig. 2.15, the multiple operational tasks (T1–T3) should be considered at the same time.

Sandoval *et al.* [75] proposed to exploit the redundancy to combine the T1 and the T2 at the same time. In fact, an improved dynamic control method is proposed to apply the task redundancy for the RCM constraint (T2), without the influence of the surgical operation (T1). Jin *et al.* [69] proposed to use neural networks to combine the T1 and the T3. In our previous works [70, 77], we utilized the hierarchical operational space formulation [72] to combine the three tasks T1, T2, and T3.

2.2.3 Control Methodology

This section aims at controlling the redundant manipulator to perform a surgical tracking task, while considering the RCM constraint and optimize the manipulability of the manipulator at the same time, using VPRNN.

Remote Center of Motion Constraint

For the surgical tracking tasks, the robot's tooltip should respect the RCM constraint, which is shown in Fig. 2.15. Obviously, $\boldsymbol{r}_{rcm}$ should be held in a line of $\boldsymbol{r}_m$ and $\boldsymbol{r}_{m-1}$, where $\boldsymbol{r}_m$ is the end of tooltip and $\boldsymbol{r}_{m-1}$ is the Cartesian position of joint $m-1$. Thus, the geometric relationship of RCM error model is defined as,

$$\overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{rcm}} \times \overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{m}} = \mathbf{0} \tag{2.33}$$

where $\overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{rcm}}$ represents the vector of line $\boldsymbol{r}_{m-1}\boldsymbol{r}_{rcm}$, and $\overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{m}}$ represent the vector of line $\boldsymbol{r}_{m-1}\boldsymbol{r}_{m}$, respectively. It can be seen that line $\boldsymbol{r}_{m-1}\boldsymbol{r}_{rcm}$ and line $\boldsymbol{r}_{m-1}\boldsymbol{r}_{m}$ should be parallel. Then, the RCM error constraint can be expressed as,

$$\boldsymbol{E}_{rcm}(\boldsymbol{q}) = \frac{\overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{rcm}} \times \overrightarrow{\boldsymbol{r}_{m-1}\boldsymbol{r}_{m}}}{d} \tag{2.34}$$

We define the Cartesian coordinates $\boldsymbol{r}_{m-1} = [x_{m-1}, y_{m-1}, z_{m-1}]^T$, $\boldsymbol{r}_m = [x_m, y_m, z_m]^T$ and $\boldsymbol{r}_{rcm} = [x_{rcm}, y_{rcm}, z_{rcm}]^T$. The RCM error model can be expanded as follows:

$$\boldsymbol{E}_{rcm} = \frac{1}{d}\begin{bmatrix} (y_n - y_{n-1})(z_{n-1} - z_{rcm}) - (y_{n-1} - y_{rcm})(z_n - z_{n-1}) \\ (x_{n-1} - x_{rcm})(z_n - z_{n-1}) - (x_n - x_{n1})(z_{n-1} - z_{rcm}) \\ (x_n - x_{n-1})(y_{n-1} - y_{rcm}) - (x_{n-1} - x_{rcm})(y_n - y_{n-1}) \end{bmatrix} \tag{2.35}$$

Problem formulation

The kinematic formulation of multi-DOFs redundant manipulator is,

$$\begin{aligned} \boldsymbol{X}_d &= f_1(\boldsymbol{q}) \\ \boldsymbol{E}_{rcm} &= f_2(\boldsymbol{q}) \end{aligned} \tag{2.36}$$

where $\boldsymbol{X}_d \in \mathbb{R}^n$ and $\boldsymbol{E}_{rcm} \in \mathbb{R}^n$ $(n = 3)$ are end-effector and RCM Cartesian coordinate, respectively. The relationship between the end-effector velocity $\dot{\boldsymbol{X}}$ and the joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^m (m = 7)$ is expressed as,

$$\begin{aligned} \dot{\boldsymbol{X}}_d &= \boldsymbol{J}_T\dot{\boldsymbol{q}} \\ \dot{\boldsymbol{E}}_{rcm} &= \boldsymbol{J}_{rcm}\dot{\boldsymbol{q}} \end{aligned} \tag{2.37}$$

where $\boldsymbol{J}_T \in \mathbb{R}^{n\times m}$ and $\boldsymbol{J}_{rcm} \in \mathbb{R}^{n\times m}$ denote the end-effector and RCM Jacobian matrix, respectively. As it is well-known the manipulability measure gives a scalar representation of the gain between joint velocities $\dot{\boldsymbol{q}}$ and task velocities $\dot{\boldsymbol{X}}$, and, consequently, measures the ability of the robot to move its end-effector, i.e., a configuration $\boldsymbol{q}$ for which the Jacobian is rank deficient. The manipulability measure depends on the $\boldsymbol{J}_T$ and is given by:

$$\mu = \sqrt{det\left(\boldsymbol{J}_T(\boldsymbol{q})\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{q})\right)} = \sqrt{\mu_1\mu_2...\mu_m} \tag{2.38}$$

where $\mu_i (i = 1, 2 \dots, m)$ denotes the i-th largest eigenvalue of $\boldsymbol{J}_T\boldsymbol{J}_T^T$

For a redundant manipulator with the desired workspace task $\boldsymbol{X}_d$ and a RCM constraint $\boldsymbol{E}_{rcm}$, the manipulability optimization problem can be formulated as [69]:

$$\begin{aligned} \min \quad & -\mu \qquad (2.39) \\ \text{s.t.} \quad & \boldsymbol{X}_d = f_1(q) \qquad (2.40) \\ & \boldsymbol{E}_{rcm}(\boldsymbol{q}) = f_2(q) \qquad (2.41) \\ & \boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^m \end{aligned}$$

where $f_1(\boldsymbol{q})$ and $f_2(\boldsymbol{q})$ are the forward kinematic functions of the end-effector and of the "wrist", respectively. The two equality constraints has been introduced to guarantee that the optimization of the manipulability index does not affect tracking of the desired end-effector trajectory (first equality constraint) and fulfilment of RCM constraint (second equality constraint).

The previous optimization problem is characterized by a cost function that is usually non-convex, and by nonlinear equality constraints [78], and it thus represents a challenging problem. In order to address the non-convexity of the cost function, we can reformulate the optimization problem as follows:

$$\min \quad -\dot{\boldsymbol{\mu}} \tag{2.42}$$
$$\text{s.t.} \quad \boldsymbol{J}\dot{q} = \boldsymbol{v}_d \tag{2.43}$$
$$\boldsymbol{J}_{rcm}(\boldsymbol{q}) = \boldsymbol{v}_{rcm} \tag{2.44}$$
$$\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^m$$

where $v_d = \dot{X}_d$, and $v_{rcm} = \dot{E}_{rcm}$. On the other side, a way to handle the two nonlinear equality constraints is introduced in Section 2.2.3.

Considering the safety in surgical task [79, 80], the joint velocities and joint angles cannot exceed kinematic limitations. The convex sets of admissible joint positions and velocities can be introduced as follows:

$$\Omega_{\mathbf{q}} = \{q_i \in \mathbb{R} \mid \underline{q}_i \le q_i \le \bar{q}_i,\ i = 1, 2, \ldots, m\}$$
$$\Omega_{\dot{\mathbf{q}}} = \{\dot{q}_i \in \mathbb{R} \mid \underline{\dot{q}}_i \le \dot{q}_i \le \bar{\dot{q}}_i,\ i = 1, 2, \ldots, m\}$$

where $\underline{\dot{q}}_i$ and $\bar{\dot{q}}_i$ are lower and upper bounds on joint velocities, respectively, and $\underline{q}_i$ and $\bar{q}_i$ are lower and upper bounds on joint angles, respectively.

In order to express these two constraints as a single constraint on joint positions and velocities, the constraint on joint positions can be reformulated [69] as follows:

$$\Omega_q = \{\dot{q}_i \in \mathbb{R} \mid -\alpha(q_i - \underline{q}_i) \le \dot{q}_i \le -\alpha(q_i - \bar{q}_i),\ i = 1, 2, \ldots, m\}$$

where α is a positive constant, and the two constraints can be rewritten as:

$$\Omega_{\dot{q}} = \{\dot{q}_i \in \mathbb{R} \mid \max(\underline{\dot{q}}_i, -\alpha(q_i - \underline{q}_i)) \le \dot{q}_i \le \min(\bar{\dot{q}}_i, -\alpha(q_i - \bar{q}_i)),\ i = 1, 2, \ldots, m\}$$

The derivative level manipulability can be obtained as,

$$\begin{cases} \frac{d(\mu^2/2)}{dt} = \det\left(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right) tr\left(\dot{\boldsymbol{J}}_T\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right) \\ \frac{d(\mu^2/2)}{dt} = \boldsymbol{\mu}\dot{\boldsymbol{\mu}} = \sqrt{\det\left(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)}\dot{\boldsymbol{\mu}} \end{cases} \tag{2.45}$$

From (2.45), the $\dot{\boldsymbol{\mu}}$ can be obtained,

$$\dot{\boldsymbol{\mu}} = \sqrt{\det\left(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)} tr\left(\dot{\boldsymbol{J}}_T\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right) \tag{2.46}$$

We define the $\dot{\boldsymbol{J}}_T$ as,

$$\dot{\boldsymbol{J}}_T = \sum_{i=1}^{m} \frac{\partial \boldsymbol{J}_T}{\partial q_i}\dot{q}_i = \sum_{i=1}^{m} h_i\dot{q}_i \tag{2.47}$$

Therefore,

$$tr\left(\dot{\boldsymbol{J}}_T\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right) = \sum_{i=1}^{m} \dot{q}_i \cdot tr\left(h_i\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right) \tag{2.48}$$

Considering the convenience of computing of $(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}$, we transform the variable with vectorization as,

$$\begin{aligned} tr\left(h_i\boldsymbol{J}_T^{\mathrm{T}}(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right) &= tr\left((\boldsymbol{J}_T h_i^{\mathrm{T}})^{\mathrm{T}}\left((\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right)\right) \\ &= \mathrm{vec}^{\mathrm{T}}\left(\boldsymbol{J}_T h_i^{\mathrm{T}}\right)\mathrm{vec}\left(\left((\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right)\right) \end{aligned} \tag{2.49}$$

Therefore the $\dot{\mu}$ can be further expressed as,

$$\begin{aligned}\dot{\mu} &= \mu\sum_{i=1}^{m}\dot{q}_i \text{vec}^{\mathrm{T}}\left(\boldsymbol{J}_T h_i^{\mathrm{T}}\right)\text{vec}\left((\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right)\\ &= \mu[\dot{q}_i,\dot{q}_2\ldots,\dot{q}_m][d_1,d_2\ldots,d_m]^{\mathrm{T}}\text{vec}\left((\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right)\end{aligned} \tag{2.50}$$

where $d_i = \text{vec}^{\mathrm{T}}\left(\boldsymbol{J}_T h_i^{\mathrm{T}}\right)$. Moreover, the new symbol '$\diamond$' is defined to simplify the expression in (2.50),

$$\boldsymbol{J}_T\diamondsuit h = [d_1,d_2\ldots,d_m]^{\mathrm{T}} = \left[\text{vec}^{\mathrm{T}}\left(h_1^{\mathrm{T}}\right),\text{vec}^{\mathrm{T}}\left(h_2^{\mathrm{T}}\right)\ldots,\text{vec}^{\mathrm{T}}\left(h_m^{\mathrm{T}}\right)\right]^{\mathrm{T}}\left(I_n\otimes\boldsymbol{J}_T^{\mathrm{T}}\right) \tag{2.51}$$

where the rules of Kronecker product '$\otimes$' satisfy: $\text{vec}(abc) = \left(b^{\mathrm{T}}\otimes \mathrm{a}\right)\text{vec}(c)$. We define the notation $\boldsymbol{\psi} = \text{vec}\left((\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}})^{-1}\right)$ $(\boldsymbol{\psi}\in\mathbb{R}^{n^2})$. Therefore, $\dot{\mu} = \mu\dot{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{J}_T\diamondsuit h)\boldsymbol{\psi}$, and $I_n = \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\left(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)^{-1} = \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\boldsymbol{\psi}$, $\text{vec}(I_n) = \text{vec}\left(\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\boldsymbol{\psi}\right) = \left(I_n\otimes\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\boldsymbol{\psi}$.

It should be noted that μ is nonnegative and independent of vector $\dot{\boldsymbol{q}}$ and $\boldsymbol{\psi}$, so the optimization function is equivalent to $\dot{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{J}_T\diamondsuit h)\boldsymbol{\psi}$. Therefore, the manipulability optimization problem, including all the aforementioned constraints, can be formulated as:

$$\begin{array}{cc}\min\limits_{q,\dot{q}\in\mathbb{R}^m} & -\dot{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{J}_T\diamondsuit h)\boldsymbol{\psi}\\ \text{s.t.} & \left(I_n\otimes\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\boldsymbol{\psi} = \text{vec}(I_n)\\ & \boldsymbol{J}_T\dot{\boldsymbol{q}} = \boldsymbol{v}_d\\ & \boldsymbol{J}_{rcm}\dot{\boldsymbol{q}} = \boldsymbol{v}_{rcm}\\ & \dot{\boldsymbol{q}}\in\Omega_{q,\dot{q}}\end{array} \tag{2.52}$$

Reformulation as a Constrained Quadratic Programming

There exist the joint angle drift because of the loss of explicit information on $\boldsymbol{X}_d$ and $\boldsymbol{E}_{rcm}$. Therefore, we design the feedback controller to restrict the movement of robot for end-effector and RCM velocity constraint in (2.37),

$$\begin{aligned}\boldsymbol{v}_d &= -k_1(f_1(\boldsymbol{q})-\boldsymbol{X}_d)+\dot{\boldsymbol{X}}_d\\ \boldsymbol{v}_{rcm} &= -k_2(f_2(\boldsymbol{q})-\boldsymbol{E}_{rcm})+\dot{\boldsymbol{E}}_{rcm}\end{aligned} \tag{2.53}$$

where k_1, k_2 are positive feedback gain.

Furthermore, in order to guarantee that the problem is convex and compliant with the formulation proposed in [69], the objective function including three tasks is defined as,

$$f(\dot{\boldsymbol{q}},\boldsymbol{\psi}) = -h_0\dot{\boldsymbol{q}}^{\mathrm{T}}(\boldsymbol{J}_T\diamondsuit h)\boldsymbol{\psi} + \frac{h_1}{2}\|\dot{\boldsymbol{q}}\|^2 + \frac{h_2}{2}\|\boldsymbol{J}_T\dot{\boldsymbol{q}}-\boldsymbol{v}_d\|^2 \tag{2.54}$$

$$+\frac{h_3}{2}\|\boldsymbol{J}_{rcm}\dot{\boldsymbol{q}}-\boldsymbol{v}_{rcm}\|^2 + \frac{h_4}{2}\left\|\left(I_n\otimes\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\boldsymbol{\psi}-\text{vec}(I_n)\right\|^2 \tag{2.55}$$

where h_0, h_1, h_2, h_3, and h_4 are positive constants.

The optimization problem in (2.52) can be reformulated as:

$$\begin{array}{cc}\min\limits_{q,\dot{q}\in\mathbb{R}^m} & f(\dot{\boldsymbol{q}},\boldsymbol{\psi})\\ \text{s.t.} & \left(I_n\otimes\boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\boldsymbol{\psi} = \text{vec}(I_n)\\ & \boldsymbol{J}_T\dot{\boldsymbol{q}} = \boldsymbol{v}_d\\ & \boldsymbol{J}_{rcm}\dot{\boldsymbol{q}} = \boldsymbol{v}_{rcm}\\ & \dot{\boldsymbol{q}}\in\Omega_{q,\dot{q}}\end{array} \tag{2.56}$$

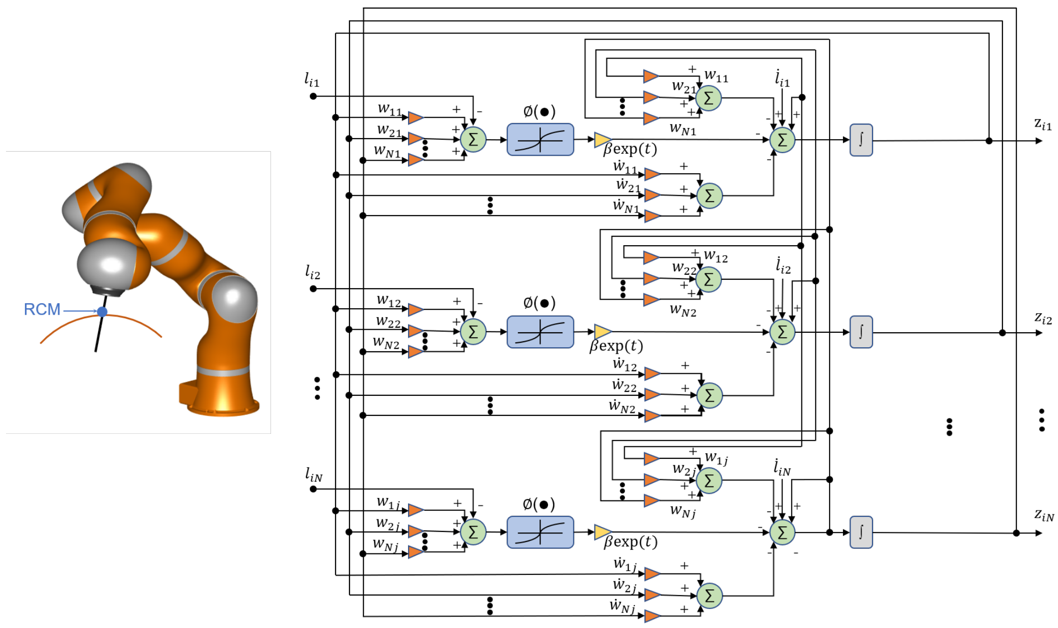


Figure 2.16: VPRNN based Hierarchical Control Framework, where $N = m + 2n^2 + 2n$. The details are shown in Sec. 2.2.3.

Varying-Parameter Hierarchical Optimization Scheme

To obtain the solution of QP problem in (2.56), the Lagrange function is defined as,

$$L(\dot{\boldsymbol{q}}, \boldsymbol{\psi}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3) = -h_0 \dot{\boldsymbol{q}}^{\mathrm{T}} (\boldsymbol{J}_T \diamondsuit h) \boldsymbol{\psi} + \frac{h_1}{2} \|\dot{\boldsymbol{q}}\|^2 + \frac{h_2}{2} \|\boldsymbol{J}_T \dot{\boldsymbol{q}} - \boldsymbol{v}_d\|^2 + \frac{h_3}{2} \|\boldsymbol{J}_{rcm} \dot{\boldsymbol{q}} - \boldsymbol{v}_{rcm}\|^2$$
$$+ \frac{h_4}{2} \left\| \left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \boldsymbol{\psi} - \mathrm{vec}(I_n) \right\|^2 + \boldsymbol{\lambda}_1^{\mathrm{T}} (\boldsymbol{v}_d - \boldsymbol{J}_T \dot{\boldsymbol{q}})$$
$$+ \boldsymbol{\lambda}_2^{\mathrm{T}} (\boldsymbol{v}_{rcm} - \boldsymbol{J}_{rcm} \dot{\boldsymbol{q}}) + \boldsymbol{\lambda}_3^{\mathrm{T}} \left(\mathrm{vec}(I_n) - \left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \boldsymbol{\psi}\right) \tag{2.57}$$

where $\boldsymbol{\lambda}_1 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_2 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_3 \in \mathbb{R}^{n^2}$

We define $\partial L = [\partial L/\partial \dot{\boldsymbol{q}}, \partial L/\partial \boldsymbol{\psi}, \partial L/\partial \boldsymbol{\lambda}_1, \partial L/\partial \boldsymbol{\lambda}_2, \partial L/\partial \boldsymbol{\lambda}_3]$.

$$\partial L =$$
$$\begin{cases} \partial L/\partial \dot{\boldsymbol{q}} = -h_0 (\boldsymbol{J}_T \diamondsuit h) \boldsymbol{\psi} + h_1 \dot{\boldsymbol{q}} + h_2 \boldsymbol{J}_T^{\mathrm{T}} (\boldsymbol{J}_T \dot{\boldsymbol{q}} - \boldsymbol{v}_d) + h_3 \boldsymbol{J}_{rcm}^{\mathrm{T}} (\boldsymbol{J}_{rcm} \dot{\boldsymbol{q}} - \boldsymbol{v}_{rcm}) - \boldsymbol{J}_T^{\mathrm{T}} \boldsymbol{\lambda}_1 - \boldsymbol{J}_{rcm}^{\mathrm{T}} \boldsymbol{\lambda}_2 \\ \partial L/\partial \boldsymbol{\psi} = -h_0 (\boldsymbol{J}_T \diamondsuit h)^{\mathrm{T}} \dot{\boldsymbol{q}} - \left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \boldsymbol{\lambda}_3 + h_4 \left(\left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \left(\left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \boldsymbol{\psi} - \mathrm{vec}(I_n)\right)\right) \\ \partial L/\partial \boldsymbol{\lambda}_1 = -(\boldsymbol{J}_T \dot{\boldsymbol{q}} - \boldsymbol{v}_d) \\ \partial L/\partial \boldsymbol{\lambda}_2 = -(\boldsymbol{J}_{rcm} \dot{\boldsymbol{q}} - \boldsymbol{v}_{rcm}) \\ \partial L/\partial \boldsymbol{\lambda}_3 = -\left(\left(I_n \otimes \boldsymbol{J}_T \boldsymbol{J}_T^{\mathrm{T}}\right) \boldsymbol{\psi} - \mathrm{vec}(I_n)\right) \end{cases} \tag{2.58}$$

If ∂L are continuous, the optimal solution satisfy $\partial L = 0$.

To solve the problem in (2.56), the VPRNN is proposed. Firstly, the decision vector $\boldsymbol{z}$ is defined as: $\boldsymbol{z} = [\dot{\boldsymbol{q}}, \boldsymbol{\psi}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3]^{\mathrm{T}}$ ($\boldsymbol{z} \in \mathbb{R}^{m+2n^2+2n}$). Then, the equation in (2.58) can be rewritten as form of a matrix,

$$\boldsymbol{W}\boldsymbol{z} = \boldsymbol{l}, \quad \boldsymbol{z} \in \Omega \tag{2.59}$$

where $\boldsymbol{W} \in \mathbb{R}^{(m+2n^2+2n)\times(m+2n^2+2n)}$, $\boldsymbol{l} \in \mathbb{R}^{m+2n^2+2n}$,

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_{11} & \boldsymbol{w}_{12} & -\boldsymbol{J}_T^{\mathrm{T}} & -\boldsymbol{J}_{rcm}^{\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{w}_{21} & \boldsymbol{w}_{22} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{w}_{25} \\ \boldsymbol{J}_T & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_{rcm} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{w}_{52} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}$$

$$\boldsymbol{l} = [\boldsymbol{l}_{11}, \boldsymbol{l}_{12}, \boldsymbol{v}_d, \boldsymbol{v}_{rcm}, \mathrm{vec}(I_n)]^{\mathrm{T}}$$
$$\boldsymbol{w}_{11} = h_1 + h_2\boldsymbol{J}_T^{\mathrm{T}}\boldsymbol{J}_T + h_3\boldsymbol{J}_{rcm}^{\mathrm{T}}\boldsymbol{J}_{rcm};$$
$$\boldsymbol{w}_{12} = -h_0(\boldsymbol{J}_T \diamondsuit h); \boldsymbol{w}_{21} = -h_0(\boldsymbol{J}_T \diamondsuit h)^{\mathrm{T}};$$
$$\boldsymbol{w}_{22} = h_4\left(I_n \otimes \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\left(I_n \otimes \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right);$$
$$\boldsymbol{w}_{25} = -\left(I_n \otimes \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right); \boldsymbol{w}_{52} = I_n \otimes \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}};$$
$$\boldsymbol{l}_{11} = h_2\boldsymbol{J}_T^{\mathrm{T}}\boldsymbol{v}_d + h_3\boldsymbol{J}_{rcm}^{\mathrm{T}}\boldsymbol{v}_{rcm};$$

$$\boldsymbol{l}_{12} = h_4\left(I_n \otimes \boldsymbol{J}_T\boldsymbol{J}_T^{\mathrm{T}}\right)\mathrm{vec}(I_n).$$

To obtain the optimization solution of (2.59), the error model of neuro-dynamics optimization is defined as,

$$\boldsymbol{\delta}(t) = \boldsymbol{W}(t)\boldsymbol{z}(t) - \boldsymbol{l}(t) \tag{2.60}$$

where $\boldsymbol{\delta}(t) \in \mathbb{R}^{m+2n^2+2n}$. To make the error model in (2.60) converge to 0, the varying parameter neuro-dynamics optimization scheme is designed as,

$$\frac{d\boldsymbol{\delta}(t)}{dt} = -\beta\exp(t)\boldsymbol{\phi}(\boldsymbol{\delta}(t)) \tag{2.61}$$

where $\beta > 0$ is the constant which can scale the convergence rate. The activation function of (2.61) is defined as,

$$\boldsymbol{\phi}(\boldsymbol{\delta}(t)) = \begin{cases} \delta_i^-, & \text{if } \delta_i(t) < \delta_i^- \\ \delta_i, & \text{if } \delta_i^- < \delta_i(t) < \delta_i^+ \\ \delta_i^+, & \text{if } \delta_i(t) > \delta_i^+ \end{cases} \tag{2.62}$$

where $\boldsymbol{\delta}_i^-$ and δ_i^+ are lower bound and upper bound of i-the element. Then, substituting (2.60) into (2.61), the extend expression of (2.61) can be rewritten as,

$$\boldsymbol{W}(t)\dot{\boldsymbol{z}}(t) = -\dot{\boldsymbol{W}}(t)\boldsymbol{z}(t) + \dot{\boldsymbol{l}}(t) - \beta\exp(t)\phi(\boldsymbol{W}(t)\boldsymbol{z}(t) - \boldsymbol{l}(t)) \tag{2.63}$$

2.2.4 Experimental Comparison

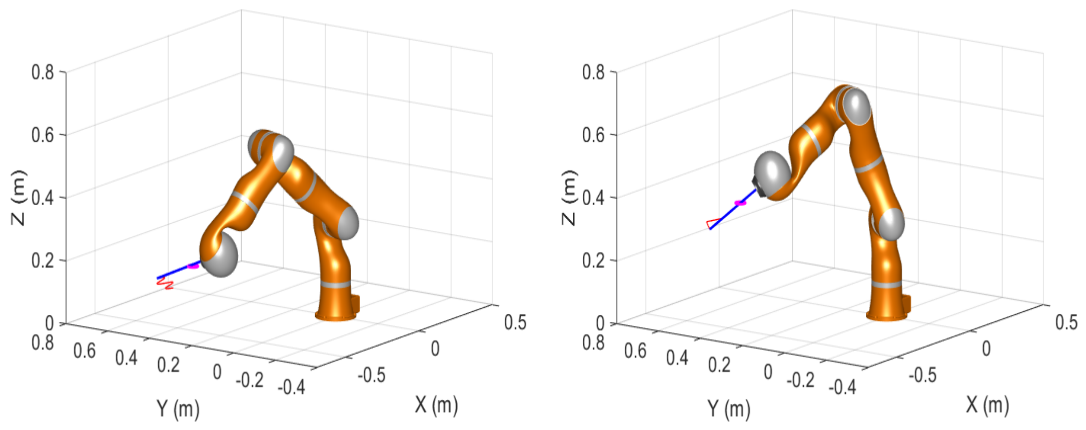
To evaluate the proposed control scheme, Simulation and experiments are carried out.The magnitude of the Cartesian position error $\boldsymbol{E}_{end}$, the RCM constraint error $||\boldsymbol{E}_{rcm}||$ and the manipulability index μ, defined in [64], are recorded for analysis. The detailed parameters of the controller are shown in Table 2.3. For DCAC and MOC, the parameters can be found in our previous works [70].

Firstly, as it is shown in Figs. 2.17a–2.17b, the demonstration using the KUKA manipulator is performed to check the feasibility of the proposed methods. Here, the sinusoid task is designed for testing.

To compare the performance of the proposed method with related works, and experimental comparisons, including RNN, Nullspace methods are performed n the same trajectory for comparison. The detailed configuration and development of the system can be found in our previous works [70]. The operative procedure is organized as follows:

Table 2.3: Experimental controller parameters

Controller	Controller parameters
RNN	$\boldsymbol{K}_X = diag[3000, 3000, 3000]$
	$\boldsymbol{D}_X = diag[30, 30, 30]$
	$\boldsymbol{K}_N = diag[800, 800, 800]$
	$\boldsymbol{D}_N = diag[10, 10, 10], \quad \lambda = 0.5$
	$h_0 = 0.01, h_1 = 0.01, h_2 = 1, h_4 = 1$
	$k_1 = 5, \beta = 10^3$
VPRNN	$\boldsymbol{K}_X = diag[3000, 3000, 3000]$
	$\boldsymbol{D}_X = diag[30, 30, 30]$
	$\boldsymbol{K}_N = diag[800, 800, 800]$
	$\boldsymbol{D}_N = diag[10, 10, 10],$
	$h_0 = 0.1, h_1 = 10, h_2 = 50, h_3 = 30$
	$h_4 = 30, k_1 = 5, k_2 = 5, \beta = 10^3$



(a) Demonstration of sine curve. **(b)** Demonstration of angle curve.

Figure 2.17: Demonstration with Kuka manipulator.

1 User 1 uses hands-on control to move the robot and pass through the RCM constraint;

2 Then, the robot autonomously tracks the set trajectory $\boldsymbol{X}_d$ to perform the surgical task, and User 2 is in charge of urgency issues in front of the visual interface.

Figure. 2.19a–2.21a shows the comparison of online performance in sine curve and angle curve tasks. Figure. 2.19b–2.21b and Fig. 2.19c–2.21c show the comparison results of tracking error and RCM error, respectively. Figure. 2.19d–2.21d show the comparison results of the manipulability index. Figure. 2.20–2.22 show the joint angles solution with VPRNN methods. From Fig. 2.19b–2.21b, it can be seen that all the errors of the end-effector are constrained in an acceptable error range within 5mm. From Fig. 2.19c–2.21c, it can be seen that the RCM error is also in an acceptable error range within 15mm.

Obviously, the proposed VPRNN-based method has an overall promising performance, including end-effector error, RCM error, manipulability index. In addition, the RNN method in [69] shows a better performance than the null space of end-effector error, RCM error, manipulability index. Therefore, we can conclude that VPRNN has the best ability to guarantee task tracking, RCM constraint, manipulability, and has the fastest convergence rate.

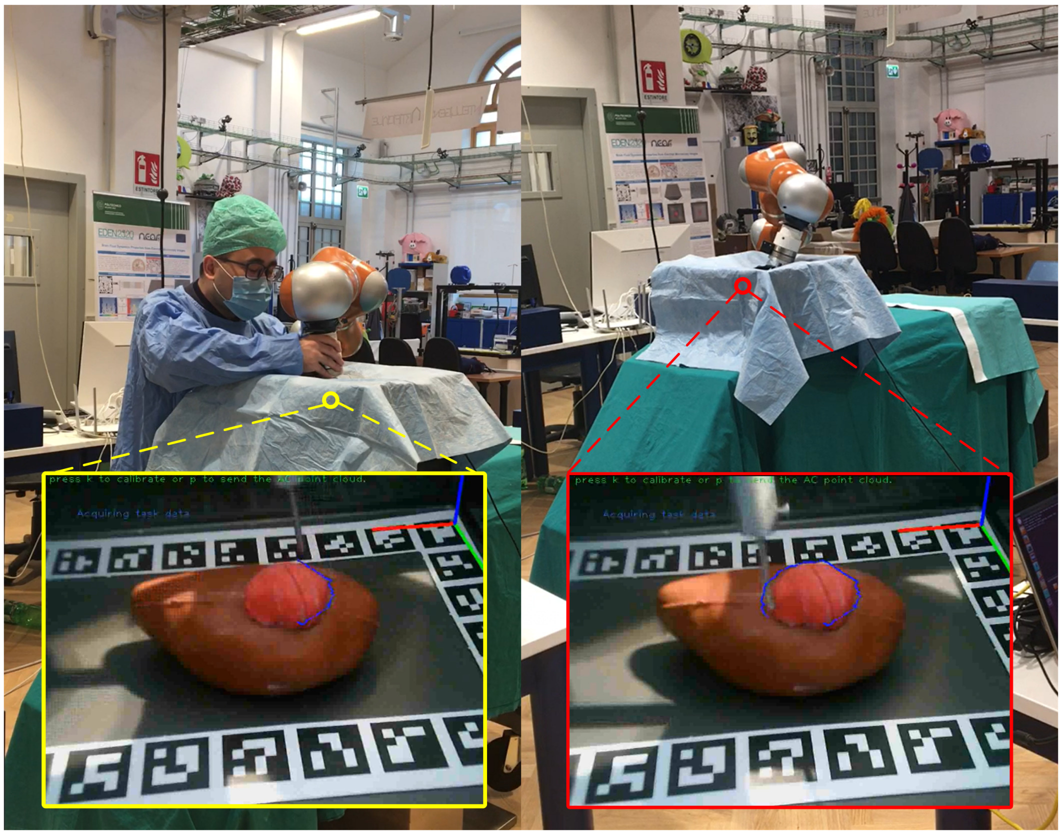


Figure 2.18: Experimental setup scene: 1) hands-on control to move the robot manipulator to pass through the RCM constraint (small incision on the patient's body); 2) autonomous tracking is activated to run the application.

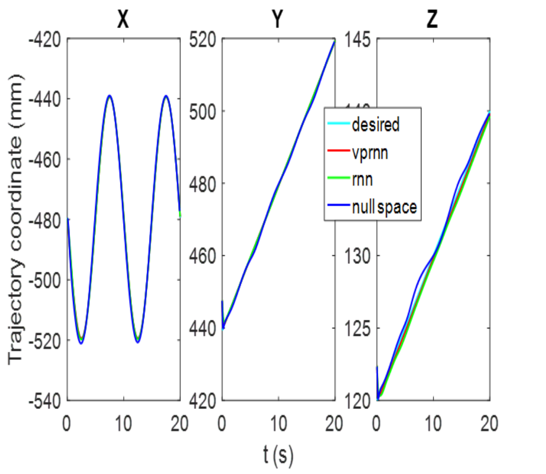
2.2.5 Summary

This section addresses varying parameters recurrent neural network (VPRNN) based on hierarchical control of a 7-DOF robot manipulator for Robot-Assisted Minimally Invasive Surgery to achieve task tracking, Remote Center of Motion (RCM) and manipulability optimization at the same time. In order to efficiently accomplish the Cartesian compliance control RCM constraint, surgical task, and manipulability optimization, a hierarchical operational space formulation is investigated. The new optimization problem is the real-time resolution for given tasks and has an excellent convergence performance even in the random initial position. Finally, in order to evaluate the accuracy of the proposed scheme, experimental evaluation has been discussed on virtual surgical tasks. Several remarks connected to [70] here detailed explaining that the recommended control scheme not only ensures the RCM constraint facing the auto-tracking phase but also develops the robot manipulability. In future works, we will attempt to the global manipulability optimization combined with the RCM constraint.

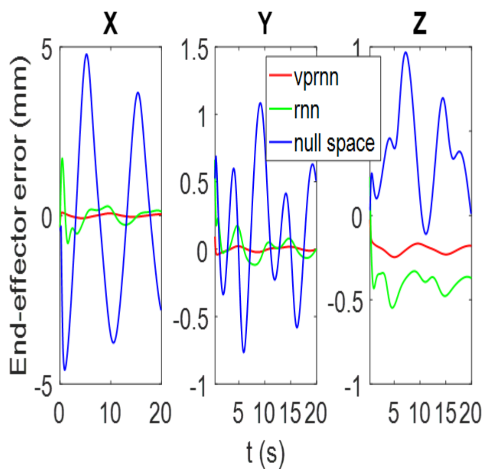
2.2.6 Appendix

Convergence Analysis

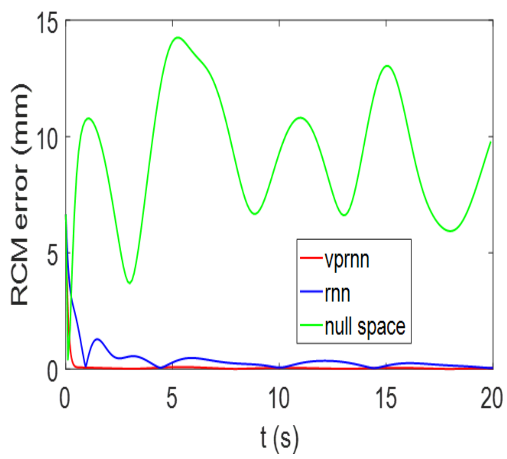
Theorem 1 *Considering the optimization problem in (2.56), if there exist the optimal solution $\boldsymbol{z}^*$ when the activation function in 2.62) is mapped to error model of varying parameter neural network, the decision variable $\boldsymbol{z}$ is defined as: $\boldsymbol{z} = [\dot{\boldsymbol{q}}, \boldsymbol{\psi}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3]^{\mathrm{T}}$ ($\boldsymbol{z} \in \mathbb{R}^{m+2n^2+2n}$) globally converges to the optimal solution $\boldsymbol{z}^* = \left[\dot{\boldsymbol{q}}^*, \boldsymbol{\psi}^*, \boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \boldsymbol{\lambda}_3^*\right]^{\mathrm{T}}$ ($\boldsymbol{z} \in \mathbb{R}^{m+2n^2+2n}$) from any initial*



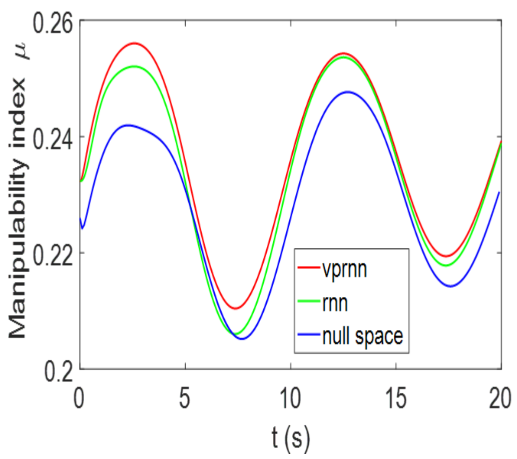
(a) Comparison results: trajectory tracking of sine curve.



(b) Comparison results: End-effector error $\boldsymbol{E}_{end}$ of sine curve.



(c) Comparison results: RCM error $\|\boldsymbol{E}_{rcm}\|$ of sine curve.



(d) Comparison results: Manipulability index μ of sine curve.

Figure 2.19: Sine curve: tracking results of end-effector's trajectory, trajectory errors, RCM errors and manipulability.

point.

Proof 1 *The candidate Lyapunov function is defined as,*

$$V(t) = \frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\delta} \tag{2.64}$$

The time derivative of $\boldsymbol{V}(t)$ *is expressed as,*

$$\dot{V}(t) = \frac{d\boldsymbol{V}(t)}{dt} = \boldsymbol{\delta}^T(t)\dot{\boldsymbol{\delta}}(t) \tag{2.65}$$

Substituting (2.62) into (2.65),

$$\dot{\boldsymbol{V}}(t) = -\beta\exp(t)\boldsymbol{\delta}^T(t)\boldsymbol{\phi}(\boldsymbol{\delta}(t)) = -\beta\exp(t)\sum_{i=1}^{N}\boldsymbol{\delta}_i(t)\boldsymbol{\phi}(\boldsymbol{\delta}_i(t)) \tag{2.66}$$

As mentioned in (2.62), the function $\boldsymbol{\phi}(\boldsymbol{\delta}(t))$ *is the monotone nondecreasing, thus we have,*

$$\delta_i(t)\boldsymbol{\phi}(\delta_i(t)) = \begin{cases} >0, & \text{if } \delta_i(t) \neq 0 \\ =0, & \text{if } \delta_i(t) = 0 \end{cases} \tag{2.67}$$

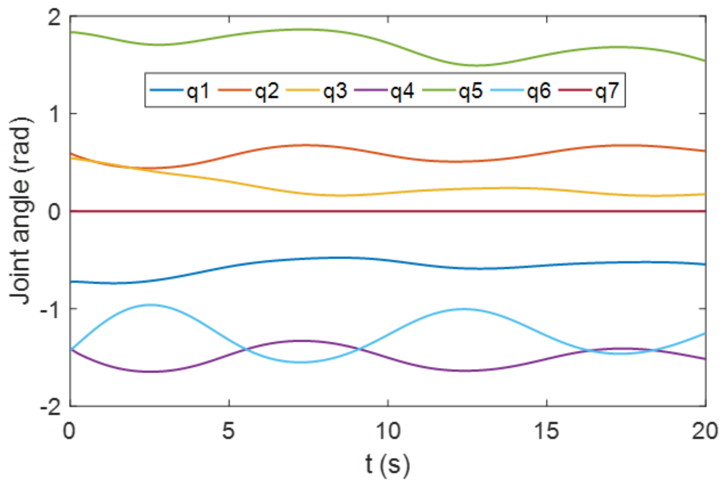


Figure 2.20: Joint angles solution using VPRNN of sine curve.

Finally, the time derivative $V(t)$ is obtained as,

$$\dot{V}(t) = \begin{cases} < 0, & \text{if } \delta_i(t) \neq 0 \\ = 0, & \text{if } \delta_i(t) = 0 \end{cases} \tag{2.68}$$

From the (2.68), it can conclude that if and only if $\delta_i(t) = 0$, $\dot{V} = 0$; otherwise $\dot{V} < 0$. The proof is finished.

Robustness Analysis

Considering the disturbance model in (2.63) as following,

$$W(t)\dot{z}(t) = -(\dot{W} + \Delta B(t))z(t) - \beta \exp(t)(W(t)z(t) - l(t)) + \dot{l}(t) + \Delta\xi(t) \tag{2.69}$$

where $\Delta B \in R^{(m+2n^2+2n)\times(m+2n^2+2n)}$ is the disturbing term of $W(t)$; $\Delta\xi \in R^{m+2n^2+2n}$ is the error from VPRNN model.

Theorem 2 *if $\|\Delta B(t)\| \leq \mu_B$, $\|\Delta\xi(t)\| \leq \mu_\xi$, $\left\|W^{-1}\right\| \leq \mu_W$, $\|l(t)\| \leq \mu_l$, and μ_B, where μ_ξ, μ_W, μ_l are all positive parameters, the error $\delta(t)$ will converge to 0 under the condition of $\beta a \exp(t) - \mu_B\mu_A > 0 (a \geq 1)$.*

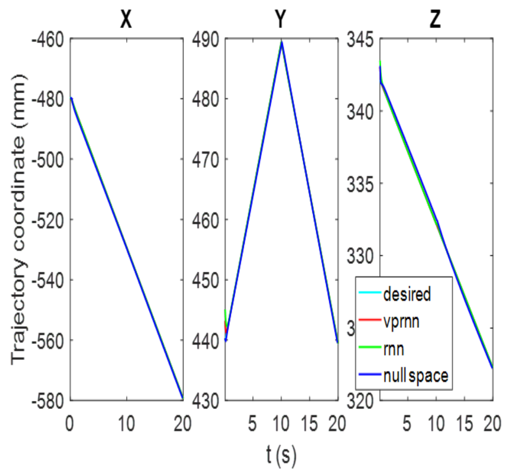
Proof 2 *Substituting the (2.60), (2.61) into (2.69), we can conclude,*

$$\dot{\delta}(t) = -\beta \exp(t)(\delta(t)) - \Delta B(t)W^{-1}\delta(t) + \Delta\xi(t) - \Delta B(t)W^{-1}l(t) \tag{2.70}$$

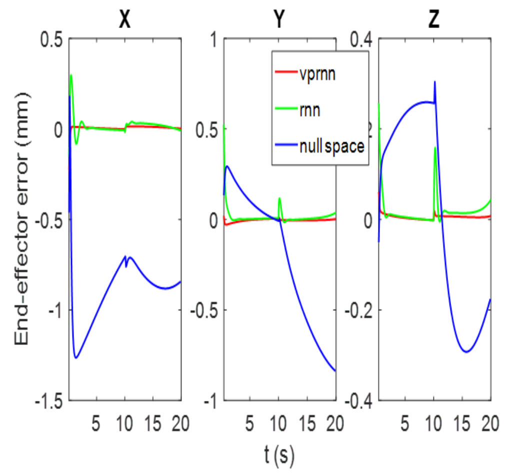
We choose a Lyapunov function as,

$$V(t) = \frac{1}{2}\delta^T(t)\delta(t) = \frac{1}{2}\sum_{j=1}^{m+2n^2+2n} \delta_i^2(t) \tag{2.71}$$

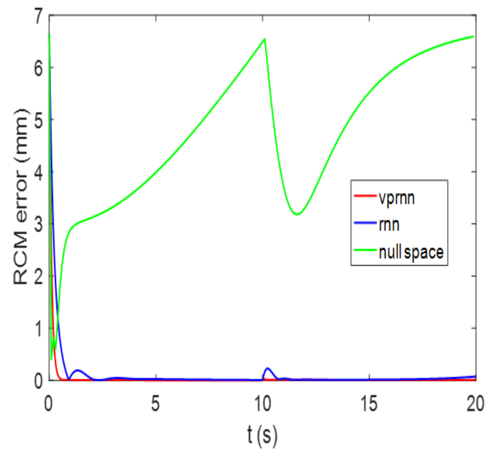
where $V(t)$ is the non-negative variable.



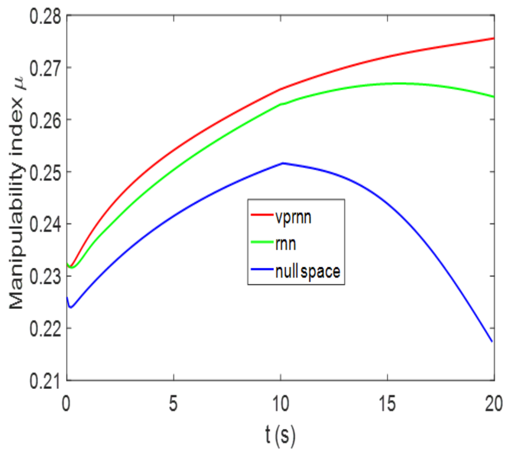
(a) Comparison results: Trajectory tracking of angle curve.



(b) Comparison results: End-effector error $\boldsymbol{E}_{end}$ of angle curve.



(c) Comparison results: RCM error $\|\boldsymbol{E}_{rcm}\|$ of angle curve.



(d) Comparison results: Manipulability index μ of angle curve.

Figure 2.21: Angle curve: tracking results of end-effector's trajectory, trajectory errors, RCM errors and manipulability.

Then, we can obtain the time derivative of $V(t)$,

$$
\begin{aligned}
& V(t) = \delta^T(t)\dot{\delta}(t) \\
& = \delta^T(t)(-\beta\exp(t)(\delta(t)) - \Delta B(t)W^{-1}\delta(t) + \Delta\xi(t) - \Delta B(t)W^{-1}l(t)) \\
& = -\beta\exp(t)\delta^T(t)(\delta(t)) + \delta^T(t)\psi(t)\delta(t) + \delta^T(t)\Delta\xi(t) + \delta^T(t)\left(-\Delta B(t)W^{-1}l(t)\right) \\
& = -\beta\exp(t)\delta^T(t)(\delta(t)) + \delta^T(t)\frac{\psi(t)+\psi^T(t)}{2}\delta(t) + \delta^T(t)\Delta\xi(t) + \delta^T(t)\left(-\Delta B(t)W^{-1}l(t)\right)
\end{aligned}
\tag{2.72}
$$

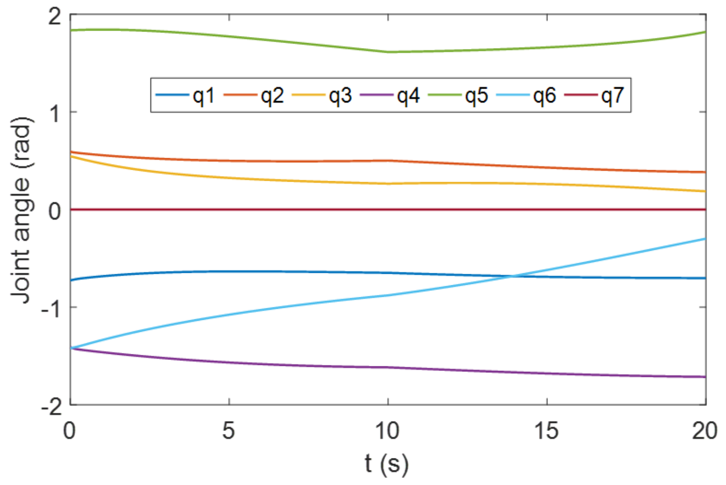


Figure 2.22: Joint angles solution using VPRNN of angle curve.

where $\psi(t) = -\Delta B(t)W^{-1}$. *Moreover,*

$$\begin{aligned}&\delta^T(t)\tfrac{\psi(t)+\psi^T(t)}{2}\delta(t)\\ &\leq \delta^T(t)\delta(t)\left|\lambda_{\max}\left(\tfrac{\psi(t)+\psi^T(t)}{2}\right)\right|\\ &\leq \delta^T(t)\delta(t)\left\|\Delta B(t)W^{-1}(t)\right\|\\ &\leq \delta^T(t)\delta(t)\mu_B\mu_W\end{aligned} \tag{2.73}$$

$$\delta^T(t)\Delta\xi(t) \leq \sum_{i=1}^{m+2n^2+2n} |\delta_i|\mu_\xi \tag{2.74}$$

$$\delta^T(t)\left(-\Delta B(t)W^{-1}l(t)\right) \leq \sum_{i=1}^{m+2n^2+2n} |\delta_i|\cdot\left\|-\Delta B(t)W^{-1}l(t)\right\| \leq \sum_{i=1}^{m+2n^2+2n} |\delta_i|\mu_B\mu_\xi\mu_l \tag{2.75}$$

Therefore, substituting (2.73)–(2.74) into (2.72),

$$\begin{aligned}\dot{V}(t) &\leq -\beta\exp(t)\delta(t)(\delta(t)) + \delta(t)\delta(t)\mu_B\mu_W + \sum_{i=1}^{m+2n^2+2n}|\delta_i|\mu_\xi + \sum_{i=1}^{m+2n^2+2n}|\delta_i|\mu_B\mu_\xi\mu_l\\ &= -\sum_{i=1}^{m+2n^2+2n}|\delta_i|\left(\beta\exp(t)P_\Omega(|\delta_i|) - \mu_B\mu_W|\delta_i| - \mu_\xi - \mu_B\mu_\xi\mu_l\right)\end{aligned} \tag{2.76}$$

We define the $\Theta_1 = \beta\exp(t)(|\delta_i|) - \mu_B\mu_W|\delta_i| - \mu_\xi - \mu_B\mu_\xi\mu_l$. *We know variable* Θ_1 *may be positive or negative.*

I *if* $\Theta_1 \geq 0$, *so* $\dot{V} \leq 0$. *It obvious that the error variable* $\delta(t)$ *converge to zero from Lyapunov theorem, and the state variable* z *converge to the optimal solution* z^*.

II *if* $\Theta_1 < 0$, *then* $\dot{V} < \delta(\delta > 0)$. *Therefore,* $\dot{V}$ *may be positive or negative.*

(1) *if* $\dot{V} \leq 0$, *we know the error variable* $\delta(t)$ *converge to zero, also the state variable* z *will converge to optimal solution* z^*.

(2) *if $\dot V > 0 (0 < \dot V < \delta)$, and consider the linear activation function $\Theta_1(|\delta(t)|) = a|\delta(t)| (a \geq 1)$, and $\beta a \exp(t) - \mu_B \mu_W > 0$, so we can obtain,*

$$\begin{aligned}\dot V &\leq - \sum_{i=1}^{m+2n^2+2n} |\delta_i| (\beta \exp(t) a |\delta_i| - \mu_B \mu_W |\delta_i| - \mu_\xi - \mu_B \mu_W \mu_l)) \\ &= -(\beta a \exp(t) - \mu_B \mu_W) \sum_{i=1}^{n+m} |\delta_i| \left(|\delta_i| - \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W} \right)\end{aligned} \tag{2.77}$$

It is easy to obtain $\beta > \frac{\mu_B \mu_W}{a}$. According to (2.77), in this case, $0 < \dot V < \delta$, so $V(t)$ increase that means $|\delta_i|$ will increase, and $\dot V$ will decrease. Therefore, $\dot V$ always exists a moment $\dot V \leq 0$, then the control system will stabilize again.

It should be noted that when $\dot v(t) = 0$, $|\delta_i| = |\delta|^+$, $|\delta|^+$ is the upper bound. We define $\Theta_2 = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W}$. If $\dot V(t) = 0$, $\sum_{i=1}^{m+2n^2+2n} |\delta_i| (|\delta_i| - \Theta_2) = 0$, and the variable $|\delta_i|$ can be seen as the input, the function $|\delta_i| (|\delta_i| - \Theta_2)$ will obtain the minimum output, if $|\delta_i| = 0.5\Theta_2$. Moreover, $|\delta_i| (|\delta_i| - \Theta_2) > 0$ when $|\delta_i| > \Theta_2$.

We know $\sum_{i=1}^{m+2n^2+2n} |\delta_i| (|\delta_i| - \Theta_2) = 0$ and the function mentioned above has the negative minimum output. We assume that $\delta_j (i = j)$ is the upper bound δ^+, so δ_j will be achieved if and only if the rest $m + 2n^2 + 2n - 1$ terms $|\delta_i| (|\delta_i| - \Theta_2)$ obtain the minimum point. Therefore,

$$\sum_{i=1}^{m+2n^2+2n} |\delta_i| (|\delta_i| - \Theta_2) = \sum_{i=1, i \neq j}^{m+2n^2+2n} |\delta_i| (|\delta_i| - \Theta_2) + \left|\delta_j\right| \left(\left|\delta_j\right| - \Theta_2 \right) \tag{2.78}$$

Then, $\Theta_2 = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W}$ is substituted in (2.78), and obtain,

$$\sum_{i=1}^{m+2n^2+2n} |\delta_i| (|\delta_i| - \Theta_2) = \left|\delta_j\right|^2 - \left|\delta_j\right| \left(\frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W} \right) - \frac{m+2n^2+2n-1}{4} \left(\frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W} \right)^2 = 0 \tag{2.79}$$

According to analysis, we can conclude the upper bound $\left|\delta_j\right|$,

$$\left|\delta_j\right| = \frac{1}{2} \left((1 + \sqrt{m + 2n^2 + 2n}) \eta \right)$$

where $\eta = \frac{\mu_\xi + \mu_B \mu_W \mu_l}{\beta a \exp(t) - \mu_B \mu_W}$, $|\delta_{end}| (end = m + 2n^2 + 2n)$ converge to zero.

2.3 NRP: Hierarchical Optimization Control for Mobile Manipulator

This section represents a constraint planning and optimization control scheme for a highly redundant mobile manipulator considering a complex indoor environment. Compared with the traditional optimization solution of a redundant manipulator, infinity norm and slack variable are additionally introduced and leveraged by the optimization algorithm. The former takes into account the joint limits effectively by considering individual joint velocities and the latter relaxes the equality constraint by decreasing the infeasible solution area. By using derived kinematic equations, the tracking control problem is expressed as an optimization problem and converted into a new quadratic programming (QP) problem. To address the optimization problem, the two-timescale recurrent neural networks optimization scheme is proposed and tested with a 9 DOFs nonholonomic mobile-based manipulator. Additionally, the BI^2RRT^* path-planning algorithm incorporates trajectory planning in the complex environment where different obstacles are positioned. To test and evaluate the proposed optimization scheme, both predefined and path-planning trajectories are tested in the Neurorobotics Platform (NRP) [1] which is open access and open source integrative simulation framework powered by Gazebo and developed by our team.

2.3.1 Introduction

Robot manipulation is receiving increasing attention in medical services, industrial production and space exploration, etc. Redundant manipulators have the extra degree of freedom (DOF) that provides a wide workspace to meet additional design objectives such as joint torque minimization [81], collision avoidance [82], joint-limit avoidance, and avoiding singularity configurations [83], etc. In practice, the redundant manipulator is in demand for remote manipulation scenarios, such as moving containers from the dock to the warehouse, which requires a larger workspace. To extend the workspace of the robot arm, mobile manipulators are more desirable because locomotion parts (tracks, wheels, legs) increase the workspace of the manipulator considerably and provide versatility, and are capable of solving a variety of tasks [84]. However, the manipulation of the mobile manipulator is more complex due to the increased DOFs. In this paper, we will study the motion planning of obstacle avoidance and optimization control for the 9-DOFs mobile manipulator.

Trajectory planning is one of the core problems in mobile robot navigation and mobile manipulator operation, aiming to provide trackable trajectories for mobile base or end effector based on existing maps. The most widely used methods are the sampling-based planning algorithm. In [85], Kavraki *et al.* proposed the Probabilistic Roadmap (PRM). The algorithm has few parameters and a simple structure, but its completely random sampling process makes most of the obtained nodes deviate from the final trajectory, leading to a significant increase in computational effort. In [86], Kuffner *et al.* proposed the rapidly-exploring random trees (RRT), a single-query algorithm that is only responsible for finding feasible trajectories, and the results are generally not optimal, sometimes very tortuous. In [87], the RRT* algorithm was proposed. This method adds steps of reselection of the parent nodes and rewiring of the tree, improving the basic RRT method to an asymptotically optimal algorithm. However, while improving the final trajectory, the planning time consumed is also growing significantly. In [88], Gammell *et al.* proposed the Informed-RRT* to restrict the sampling space to a hyper-ellipsoid region, which improves the sampling efficiency thus greatly reduces the planning time. In [89], Informed-RRT* was extended to a bidirectional variant and applied to mobile manipulation. The variant reduces the searching time for an initial feasible trajectory, thus

[1] https://neurorobotics.net

leaving more time for trajectory refinement to get better results. Another approach involves gradient-based methods. Ratliff *et al.* proposed the covariant Hamiltonian optimization algorithm [90], which uses covariant gradients and functional gradients on optimization to make many trajectory planning problems simple and trainable. However, this approach usually requires a complex parameter tuning process, and in some cases, derivative information is not available. Considering the efficiency of RRT-like algorithms and the complexity of mobile manipulator trajectory planning, we choose a method similar to [89], i.e. the Bidirectional Informed RRT* algorithm (BI^2RRT^*).

When the trajectories are defined by the planner in Cartesian space, it is important to map the control tasks from Cartesian space to joint space. Over past decades, tracking control problems have been well-studied for redundant manipulators [63, 91]. Generally, there are two common approaches to address redundant manipulators: optimization-based and optimization-free. In [92], Kumar *et al.* introduced a pseudoinverse-based RBF neural network method for trajectory tracking in Cartesian space. In [93], adaptive fuzzy sliding mode control was proposed for redundant robots achieving precise performance in the presence of uncertainties. In [94], Jiang *et al.* proposed an impedance controller for the 7-DOFs robot arm to address the problem of interaction with humans. Although these optimization-free studies have achieved good performance in some special scenarios, they are based on pseudoinverse for tracking in task space, which may suffer from repeatability, singularity, and high computation due to the pseudo-inverse of the Jacobian matrix. Furthermore, it is difficult to handle multiple constraints problems, especially inequality constraints [95].

Another approach for tracking control problems of the redundant manipulator is the optimization-based method which is inverse-free and suitable for real-time tracking tasks. In [95], Hassan *et al.* summarized the recurrent neural network (RNN) approaches for quadratic programming (QP) problem of inverse kinematics solution. In [96, 97], primal-dual neural network (PDNN) was developed for redundant robot control, and the global convergence analysis was also provided. Although the PDNN or RNN can obtain their solvability constraints at convex optimization problems, they may change drastically and dynamic behaviors may become unpredictable [98]. In [99], Zhang *et al.* introduced a novel zeroing neural network which is derived from state residual error model to address the QP problem with exponential convergence speed and high precision, but it traded running time for convergence performance which lower the real-time manipulation.

Moreover, the mentioned work aimed only at the single metric optimization of tracking accuracy. Generally, joint-velocity minimization is investigated mostly using the l_2 (Euclidean) norm which is also associated with minimum-energy [100]. One of the reasons why it is appealing to researchers is that closed-form analytical expressions of the optimization problem can be obtained [101]. However, l_2 only minimizes the sum of the squares of joint velocities and does not take into account magnitudes of individual joint velocity. Therefore, it might be undesirable for applications in which the magnitude of individual joint limits are of the essence. In this paper, an additional metric l_∞ norm minimizes the largest component of joint velocity is considered into the objective function, which is more consistent with physical limits than the l_2 norm [102].

It should be noted that the optimization problem sometimes yields infeasible solution due to strict equality constraints, thereby resulting in unsmooth and jittery movements under the requirement of high precision, which is very dangerous during interaction with the human or the environment, especially a surgical task. Motivated by [103] of slack variables, therefore, the infeasible solution area of trajectory tracking can be relaxed using *slack* variables. The slack variable is activated in the cases where the tracking errors cannot achieve the desired level, which trades accuracy for smoothness of motion within an acceptable range. We tested and simulated the proposed optimization scheme in the NRP which is an open-source platform and helps researchers to develop novel robotic control algorithms [104].

In this section, we firstly use a 9 DOFs mobile-based manipulator (Fig. 2.23) and derive the tracking control problem including multiple metrics as an optimization problem (Section 2.3.3). Besides, the infinity norm is added to the optimization problem so as to consider joint limits strictly. To be able to express the infinity norm in terms of QP, it is converted into an inequality constraint. Furthermore, slack variables are employed to avoid cases where optimization fails to satisfy the equality constraint, which may cause unsmooth and jittery movements in the end-effector. Afterward, a two-timescale recurrent neural networks (TNN) optimization scheme has been developed and kinematic equations are embedded in TNN (Section 2.3.4). Next, BI^2RRT^* path-planning algorithm is employed for trajectory planning among obstacles.

2.3.2 Coordinated Base-manipulator Trajectory Planning

This section describes the trajectory planning of our mobile manipulator, which aims to plan Cartesian trajectories of both the mobile base and the end effector. Motived by work proposed in [89], the BI^2RRT^* algorithm can address the mobile manipulator planning with respect to joint limits, avoid self-collisions as well as collisions with obstacles in the environment. For the completeness of exposition, we summarize the key points of our implementation on the Neurorobotics Platform. The algorithm pseudocode is shown in Alg. 1.

Our mobile manipulator consists of a mobile base and a Kuka robotic arm. For efficient sampling we equate the mobile base to two prismatic joints and one revolute joint, so we get a simple movement chain and its configuration can be given by:

$$\mathcal{X} = (x_m, y_m, \varphi, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7) \tag{2.80}$$

x_m, y_m, φ denote the configuration of the mobile base and θ_i stands for the joint configuration of the Kuka robot arm. In our problem the starting pose $\boldsymbol{p}_{init}$ and target pose $\boldsymbol{p}_{target}$ of the end-effector are given. We model our 9DOF model as a kinematic chain and can find the corresponding states $\mathcal{X}_{init}, \mathcal{X}_{target}$ in joint space. The basic RRT algorithm takes the initial state as the root node and constructs a rapidly-exploring random tree in the configuration space. And as improvement in the BI^2RRT^* algorithm two rapidly-exploring random trees $\mathcal{T}_i$ and $\mathcal{T}_t$ are initialized from the initial state $\mathcal{X}_{init}$ and the target state $\mathcal{X}_{target}$ respectively. According to [89], this enables to obtain the first feasible trajectory quickly, thus increasing the time of the informed trajectory refinement process. For simplicity and clarity, in the following we describe the growth process of only one random tree $\mathcal{T}_i$, but the same process occurs alternately on both trees $\mathcal{T}_i$ and $\mathcal{T}_t$.

In each iteration we get a sampling point $\mathcal{X}_{rand}$ from the configuration space. The sampling rule is that $\mathcal{X}_{rand}$ is set to any point in the space with a certain probability, or set directly to the target $\mathcal{X}_{target}$. This sampling method increases the chance of the tree growing directly towards the target. Then the $FindNearest$ function is used obtain the nearest node $\mathcal{X}_{near}$ to $\mathcal{X}_{rand}$ in $\mathcal{T}_i$ and the $NewNode$ function is used to generate a new node $\mathcal{X}_{new}$ on $\mathcal{T}_i$ at certain distance u from $\mathcal{X}_{near}$ on the line between $\mathcal{X}_{near}$ and $\mathcal{X}_{rand}$. Next step we use the $CheckCollision$ function to check whether there are obstacles between the two points $\mathcal{X}_{near}$ and $\mathcal{X}_{new}$, if not we will enter the asymptotic optimal process. Firstly in configuration space we construct a hypersphere with $\mathcal{X}_{new}$ as the center and define the tree nodes of $\mathcal{T}_i$ within the hypersphere as the set of neighbor nodes $\mathcal{X}_{neighbor}$. Then the process can be divided into two stages. The first stage is the reselection of a parent node. The trajectory cost of every neighbor node $\mathcal{X}_{neighbor}$ to the initial node $\mathcal{X}_{init}$ plus the trajectory cost of the node $\mathcal{X}_{new}$ to the neighbor node $\mathcal{X}_{neighbor}$ is calculated, the neighbor node with the smallest cost is selected as the parent

Algorithm 1: Coordinated BI^2RRT^* Planning

```
Input: $\boldsymbol{p}_{init}, \boldsymbol{p}_{target}$
Output: $\boldsymbol{Traj}_{base}, \boldsymbol{Traj}_{eef}$
$(\mathcal{X}_{init}, \mathcal{X}_{target}) \leftarrow IKNN(\boldsymbol{p}_{init}, \boldsymbol{p}_{target})$
Initialization : $(\mathcal{T}_i, \mathcal{T}_t) \leftarrow TreeInit(\mathcal{X}_{init}, \mathcal{X}_{target})$,
$TrajCost(\mathcal{T}_i) \leftarrow \infty,\ TrajCost(\mathcal{T}_t) \leftarrow \infty,\ \boldsymbol{T}_{\mathcal{T}} \leftarrow \emptyset$
while $TimeOut()$ do
    $c_{best} \leftarrow min_{\mathcal{T} \in \boldsymbol{T}_{\mathcal{T}}}\{TrajCost(\mathcal{T})\}$
    if $c_{best} < \infty$ then
        $c_{min} \leftarrow \left\| \mathcal{X}_{target} - \mathcal{X}_{init} \right\|_2$
        $\mathcal{X}_{rand} \leftarrow EllipseSample(c_{best}, c_{min}, \mathcal{X}_{init}, \mathcal{X}_{target})$
    else
        $\mathcal{X}_{rand} \leftarrow RandomSample(\mathcal{X}_{init}, \mathcal{X}_{target})$
    end
    $\mathcal{X}_{near} \leftarrow FindNearest(\mathcal{T}_i, \mathcal{X}_{rand})$
    $\mathcal{X}_{new} \leftarrow NewNode\ (\mathcal{X}_{near}, \mathcal{X}_{rand}, u)$
    if $CheckCollision(\mathcal{X}_{near}, \mathcal{X}_{new})$ then
        $\mathcal{X}_{neighbor} \leftarrow FindNeighbor(\mathcal{T}_i, \mathcal{X}_{new})$
        for $\forall\ \mathcal{X}_{neighbor} \in \mathcal{X}_{neighbor}$ do
            $\mathcal{T}_i \leftarrow ChooseParent(\mathcal{T}_i, \mathcal{X}_{neighbor}, \mathcal{X}_{new})$
            $\mathcal{T}_i \leftarrow RewireTree(\mathcal{T}_i, \mathcal{X}_{neighbor}, \mathcal{X}_{new})$
        end
    end
    $\mathcal{X}_{connect\_near} \leftarrow FindNearest(\mathcal{T}_t, \mathcal{X}_{last})$
    $\mathcal{X}_{connect\_new} \leftarrow NewNode(\mathcal{X}_{connect\_near}, \mathcal{X}_{last}, u)$
    if $CheckCollision(\mathcal{X}_{connect\_new}, \mathcal{X}_{last})$ then
        $\mathcal{T} \leftarrow MergeTrajectory(\mathcal{T}_i, \mathcal{T}_t)$
        $\boldsymbol{T}_{\mathcal{T}} \leftarrow \boldsymbol{T}_{\mathcal{T}} \cup \{\mathcal{T}\}$
    end
    $Swap\ (\mathcal{T}_i, \mathcal{T}_t)$
end
$\mathcal{T}_{final} \leftarrow UniTimeResample(\mathcal{T}, v, \alpha)$
$(\boldsymbol{Traj}_{base}, \boldsymbol{Traj}_{eef}) \leftarrow IKNN(\mathcal{T}_{final})$
```

node of $\mathcal{X}_{new}$. The second stage is rewiring of the tree. For each neighbor node $\mathcal{X}_{neighbor}$, modify its parent node to $\mathcal{X}_{new}$ and calculate the trajectory cost, perform the modification if the trajectory cost is smaller than before. After the asymptotic optimal step, we move to the bidirectional connection step. Likewise, we use the $FindNearest$ function to obtain the nearest node $\mathcal{X}_{connect\_near}$ on tree $\mathcal{T}_t$ to the last added node $\mathcal{X}_{last}$ on tree $\mathcal{T}_i$ and the $NewNode$ function is used to generate a new node $\mathcal{X}_{connect\_new}$ at certain distance u from $\mathcal{X}_{connect\_near}$ on $\mathcal{T}_t$. Then we simply use a line to connet $\mathcal{X}_{last}$ and $\mathcal{X}_{connect\_new}$ and use the $CheckCollision$ function to check whether collision exists between the two points. If not the two points are connected directly and the two tree $\mathcal{T}_i, \mathcal{T}_t$ will merge to get a feasible trajectory $\mathcal{P}$.

Once an initial feasible trajectory is found, a hyper-ellipsoid is constructed with $\mathcal{X}_{init}$, $\mathcal{X}_{target}$ as the focus points and current trajectory cost c_{best} as the long axis length. According to [88], the sum of the distances from the points outside the hyper-ellipse to the two focus points must be larger than that from the points inside the hyper-ellipse, so we can restrict the sampling space to the above constructed hyper-ellipse. This will significantly improve the sampling efficiency, especially in high-dimensional configuration space. We use the method described above to obtain the optimal trajectory asymptotically over iterations. After getting an optimal trajectory in configuration space we then resample the trajectory with a uniform time step using a higher-order spline smoother $UniTimeResample$ and perform the forward kinematics (FK) solution to get the planning trajectories cartesian space.

2.3.3 Optimization Problem

This section aims to introduce the kinematic model of the 9DOF robot model and the tracking control optimization problem considering the multiple metrics.

Kinematic Model of Mobile Based Kuka Manipulator

In this section, a 9DOF mobile-based Kuka manipulator is used and tested to prove the effectiveness of the planner and TNN optimization scheme. The position of the end-effector is defined by the position and orientation of the mobile base and manipulator. To obtain the end-effector position, we derive the kinematic equation for mobile base and Kuka manipulator and then we reformulate them as a single equation. In order to derive the kinematic model of a 9DOF, some notations are introduced in Fig.2.23 and explained as Table. 2.4.

Table 2.4: Explanation of the used notations shown in Fig.2.23

Parameters	Explanation
P_M	7DOF Kuka manipulator location on mobile platform
d	Distance from P_0 to P_M
b	Distance between the driving wheels and the axis of symmetry
r	Wheel radius
R	Distance between Q (*Instantanous Center of Rotation*) and the left driving wheel
ψ	Heading angle of the mobile platform defined from the x axis.
w	Angular velocity of mobile platform around Q
$\dot{\theta}_l$, $\dot{\theta}_r$	Angular velocities of left and right wheels, respectively

The forward kinematics of the 9DOF mobile-based manipulator are formulated as follows:

$$r(t) = f(\psi(t)) \tag{2.81}$$

where $r(t) \in \mathbb{R}^m$ is end-effector position in Cartesian space, and $\psi(t) \in \mathbb{R}^n$ is joint angles; $f(.)$ is nonlinear mapping function. The 7DOF redundant Kuka arm is placed on the mobile

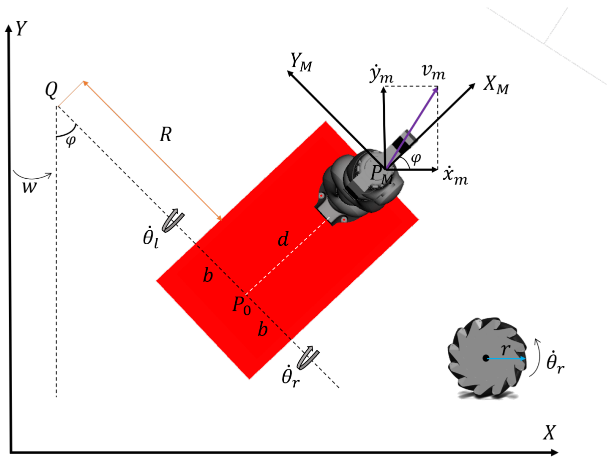


Figure 2.23: Top view of 9DOF mobile-based manipulator.

platform. For redundant manipulators, joint-space has more degree of freedom (DOF) than required for the end-effector's position i.e, $n > m$ [105]. The redundancy helps simultaneously to achieve many useful objectives, such as collision avoidance, joint-limit avoidance, singularity [83].

Fig. 2.24 describes the coordinate frames of a 9DOF mobile-based manipulator. The base frame of Kuka manipulator X_M, Y_M, and Z_M are attached to the mobile platform. The end-effector position vector with respect to the W (World) frame can be obtained with homogeneous transform matrices shown as follow:

$$^W\boldsymbol{r}_{end} = {}^WT_M{}^MT_1{}^1T_2{}^2T_3{}^3T_4{}^4T_5{}^5T_6{}^6T_7{}^7\boldsymbol{r}_{end} \tag{2.82}$$

where $^{(.)}T_{(.)} \in \mathbb{R}^{4\times4}$ and $^{(.)}\boldsymbol{r}_{(.)} \in \mathbb{R}^4$ denote the homogeneous transform matrices and homogeneous coordinate representation, respectively. $\boldsymbol{r}$ can be expressed shown as:

$$^Wr_{end} = \begin{bmatrix} ^W\boldsymbol{p} \\ 1 \end{bmatrix} = \begin{bmatrix} ^Wp_x \\ ^Wp_y \\ ^Wp_z \\ 1 \end{bmatrix} \tag{2.83}$$

The end-effector position vector can be divided into the position vector of the mobile base with respect to world frame and the position vector of end-effector with respect to M frame. Separation of the position vectors as shown in Eq. (2.84) makes it easier to analytically derive the kinematic equations of mobile part and Kuka robot.

$$^W\boldsymbol{p} = \begin{bmatrix} ^Wp_x \\ ^Wp_y \\ ^Wp_z \end{bmatrix} = \underbrace{\overbrace{\begin{bmatrix} ^Wx_c \\ ^Wy_c \\ ^Wz_c \end{bmatrix}}^{mobile\ part}}_{^W\boldsymbol{p}_M} + \overbrace{^M\boldsymbol{p}_{end}}^{kuka\ arm} \tag{2.84}$$

where $^M\boldsymbol{p}_{end}$ and $^W\boldsymbol{p}_M$ are the position vector of end-effector with respect to M frame and position vector of M with respect to W frame, respectively.

Firstly, we derive the kinematic equation of the mobile base. As shown in Fig. 2.23, the mobile platform can be considered rotating about a point Q (i.e *Instantaneous Center of*

Rotation) at any time instant t. The tangential velocity of any point on the left and right wheel can be calculated based on their distances from the axis of rotation and this velocity should be equal to the velocity about the point Q. According to the geometric relationship, we have the formulations as follow:

$$\dot{\psi} = w \tag{2.85}$$

$$Rw = r\dot{\theta}_l \tag{2.86}$$

$$(R+2b)w = r\dot{\theta}_r \tag{2.87}$$

where R is the distance between Q and the left driving wheel; w is the angular velocity of the robot; r denotes the wheel radius. The definition is shown in Table. 2.4. We assume that the mobile base is a rigid object and there is no slippage between the wheels and the floor. The tangential velocity of P_0 and P_M can be expressed below:

$$(R+b)w = (\dot{x}_0 \cos\psi + \dot{y}_0 \sin\psi) = (\dot{x}_m \cos\psi + \dot{y}_m \sin\psi) \tag{2.88}$$

Then these velocities can be expressed with respect to the angular velocities of each wheel:

$$\dot{x}_m \cos\psi + \dot{y}_m \sin\psi = r\dot{\theta}_l + bw \tag{2.89}$$

$$\dot{x}_m \cos\psi + \dot{y}_m \sin\psi = r\dot{\theta}_r - bw \tag{2.90}$$

From Eq. (2.89) and Eq. (2.90), the angular velocity w of mobile platform can be calculated below:

$$w = \frac{r}{2b}\left(\dot{\theta}_r - \dot{\theta}_l\right) \tag{2.91}$$

Furthermore, the mobile platform moves only in the direction of the axis of symmetry.

$$\dot{x}_m \cos\psi - \dot{y}_m \sin\psi + dw = 0 \tag{2.92}$$

Summation of Eq. (2.89) and Eq. (2.90) result in:

$$\dot{x}_m \cos\psi + \dot{y}_m \sin\psi = \frac{r}{2}\left(\dot{\theta}_l + \dot{\theta}_r\right) \tag{2.93}$$

Combination with Eq. (2.92) and Eq. (2.93), we can express the kinematic equation of mobile base below:

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ -\frac{dr}{2b} & \frac{dr}{2b} \end{bmatrix} \begin{bmatrix} \dot{\theta}_l \\ \dot{\theta}_r \end{bmatrix} \tag{2.94}$$

As Eq. (2.94) shows the relation between mobile platform and joint (i.e wheels) velocity, the Jacobian matrix can be directly obtained as:

$$J_m = \begin{bmatrix} \frac{r}{2}\cos\psi + \frac{dr}{2b}\sin\psi & \frac{r}{2}\cos\psi - \frac{dr}{2b}\sin\psi \\ \frac{r}{2}\sin\psi - \frac{dr}{2b}\cos\psi & \frac{r}{2}\sin\psi + \frac{dr}{2b}\cos\psi \end{bmatrix} \tag{2.95}$$

Due to nonlinearity and redundancy, It is tough to solve directly corresponding $\psi_d(t)$ for a desired end-effector position $\psi_d(t) = f^{-1}(r_d(t))$. However, mapping from the joint space to workspace in velocity level is an affine mapping and can be used to simplify the problem which can be defined as below [106]:

The time derivative of Eq. (2.84) can be expressed as:

$${}^{W}\dot{\boldsymbol{p}} = \begin{bmatrix} {}^{W}\dot{p}_x \\ {}^{W}\dot{p}_y \\ {}^{W}\dot{p}_z \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{W}\dot{x}_c \\ {}^{W}\dot{y}_c \\ {}^{W}\dot{z}_c \end{bmatrix}}_{{}^{W}\dot{\boldsymbol{p}}_M} + {}^{M}\dot{\boldsymbol{p}}_{end} \tag{2.96}$$

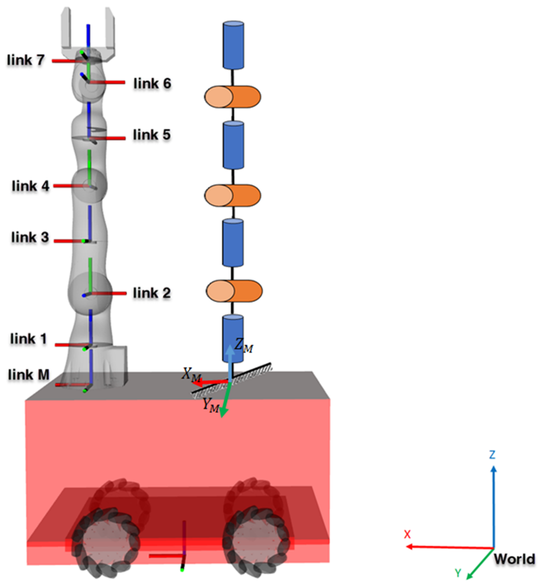


Figure 2.24: Coordinate frames of 9DOF mobile based manipulator.

where ${}^{M}\dot{\boldsymbol{p}}_{end}$ and ${}^{W}\dot{\boldsymbol{p}}_{M}$ are the velocity vector end-effector frame with respect to M frame and velocity vector M frame with respect to W frame, respectively. The Eq. (2.96) can be expressed in terms of Jacobian matrix and joint variables as:

$$ {}^{W}\dot{\boldsymbol{p}} = J_w \dot{\psi} \tag{2.97}$$

where $J_w \in \mathbb{R}^{m\times n}$ is the Jacobian matrix and $\dot{\psi} = \left[\dot{\theta}_l, \dot{\theta}_r, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\theta}_5, \dot{\theta}_6, \dot{\theta}_7\right]^T \in \mathbb{R}^n$ is a velocity vector consisting of the angular velocities of left $\dot{\theta}_l$ and right wheel $\dot{\theta}_r$ and joint velocity of the robot arm $\dot{\theta}_i$ ($i = 1,..,7$).

The 1st and 2nd column of the J_w come from the Eq. (2.95) with an augmented 0 value in the 3rd row to match the size. If we take the partial derivative of end-effector ${}^{W}\boldsymbol{p}$ with respect to its variables, then we can obtain the rest of the columns. It is worth mentioning that the heading angle ψ would be also its variable and therefore it is expected to appear in the Jacobian matrix. However, ψ is just an intermediate variable and depends on the wheel velocity $\psi = \frac{r(\theta_r - \theta_l)}{2b}$, we can combine it with the 1st and 2nd column of the J_w. Finally, the Jacobian matrix J_w is given as:

$$ J_w = \begin{bmatrix} J_{m_1} - J_{\psi} & J_{m_2} - J_{\psi} & J_{\theta_1} & \cdots & J_{\theta_7} \end{bmatrix} \in \mathbb{R}^{m\times n} \tag{2.98}$$

where J_{m_i} is the i^{th} column of J_m with augmented 0 in the 3rd row and J_h is the partial derivative of end-effector position vector ${}^{W}\boldsymbol{p}$ with respect to the variable h (i.e $J_h = \frac{\partial {}^{W}\boldsymbol{p}}{\partial h}$).

The joint limits can be defined as follow:

$$\Omega_\theta = \{\theta_i \in \mathbb{R} \mid \theta_i^- \le \theta_i \le \theta_i^+; i = l, r, 1, \ldots, n-2\}$$
$$\Omega_{\dot{\theta}} = \{\dot{\theta}_i \in \mathbb{R} \mid \dot{\theta}_i^- \le \dot{\theta}_i \le \dot{\theta}_i^+,\ i = l, r, 1, \ldots, n-2\}$$

where $\dot{\theta}_i^-$ and $\dot{\theta}_i^+$ are lower and upper bounds on joint velocities, respectively, and θ_i^- and θ_i^+ are lower and upper bounds on joint angles, respectively.
The joint position and velocity constraints can be merged and expressed as a single constraint.

The merged constraint can be reformulated as follows [69]:

$$\Omega_\theta = \{\dot{\theta}_i \in \mathbb{R} \mid \beta\left(\theta_i^- - \theta_i\right) \leq \dot{\theta}_i \leq \beta\left(\theta_i^+ - \theta_i\right), \; i = l, r, 1, \ldots, n-2\} \quad (2.99)$$

where β is a positive constant, and the two constraints can be rewritten as:

$$\Omega_{\dot{\theta}} = \{\dot{\theta}_i \in \mathbb{R} \mid \max(\dot{\theta}_i^-, \beta(\theta_i^- - \theta_i)) \leq \dot{\theta}_i \leq \min(\dot{\theta}_i^+, \beta(\theta_i^+ - \theta_i)), i = l, r, 1, \ldots, n-2\} \quad (2.100)$$

Optimization Problem Formulation

Control solutions used to be found analytically using an inverse of the Jacobian matrix for non-redundant manipulators. Since the Jacobian matrix is nonsquare for redundant manipulators i.e $n > m$, It can be calculated with a pseudo-inverse method. However, the intensive computation burden of performing continuously pseudo-inverse of the Jacobian matrix precludes this method from being ubiquitous. Moreover, joint limits might be exceeded due to lack of constraints and some configurations of the manipulator lead to the singularity of the Jacobian matrix and result in local instability. The parallel processing capability of neural networks can be employed for solving control problems in redundant manipulators. In this section, we explain our optimization problem.

The first proposed algorithm is expressed below:

$$\min \quad \frac{1}{2}\dot{\theta}^T W \dot{\theta} \quad (2.101)$$

$$\text{s.t.} \quad J_w \dot{\theta} = \dot{r}_d \quad (2.102)$$

$$\dot{\theta} \in \Omega_{\dot{\theta}}$$

where $\dot{\theta} \in \mathbb{R}^n$ is a decision variable and $W \in \mathbb{R}^{n \times n}$ symmetric weighting matrix. If W is inertia matrix, the objective function corresponds to minimizing the kinetic energy. The matrix W is set as an identity matrix, then objective function is to minimize the l_2 norm of the joint velocity $||\dot{\theta}(t)||_2^2$. The minimization of the objective function is defined as Eq. (2.101). Equation (2.102) is the equality constraints describing the relationship between end-effector and joint velocities.

Since the optimization is velocity level, the position drift should be considered because of the loss of explicit information of optimization formulation in task space. Therefore, the position feedback item are added in Eq. (2.102) with shown in equation below:

$$v_d = -k_1(r - r_d) + \dot{r}_d \quad (2.103)$$

where k_1 is a feedback coefficient value and r is the current position of the end-effector. Since multi-tasks have different priorities, they can be scaled using weight coefficients. So multi-task optimization problem can be extended as follows:

$$\min \quad \frac{c_0}{2}\dot{\theta}^T\dot{\theta} + \frac{c_1}{2}||J_w\dot{\theta} - v_d||^2 \quad (2.104)$$

$$\text{s.t.} \quad J_w\dot{\theta} = v_d \quad (2.105)$$

$$\dot{\theta} \in \Omega_{\dot{\theta}}$$

where $c_0 \geq 0$, and $c_1 \geq 0$ are used to prioritize different tasks. The optimization problem above might yield some solutions outside of the joint limits. In this case, actual values are forced to be clipped and do not satisfy the optimization equation. Since infinity-norm minimizes the absolute value of the maximum component, It finds solutions satisfying the physical limits constraints.Therefore, the infinity norm optimization task $min\ \|\dot{\theta}_\infty\|$ can be added as an additional task in the Eq. (2.104). First, let's define an auxiliary variable:

$$\varsigma = \|\dot{\theta}_\infty\| = \max\left(|\dot{\theta}_1|, |\dot{\theta}_2|, ..., |\dot{\theta}_n|\right) \tag{2.106}$$

So, the optimization problem can be defined as:

$$\begin{aligned} \min \quad & \frac{c_0}{2}\dot{\theta}^T\dot{\theta} + \frac{c_1}{2}\|J\dot{\theta} - v_d\|^2 + \frac{c_2}{2}\varsigma^2 \\ \text{s.t.} \quad & J_w\dot{\theta} = v_d \\ & \dot{\theta} \in \Omega_{\dot{\theta}} \end{aligned} \tag{2.107}$$

where $c_0 \geq 0$, $c_1 \geq 0$ and $c_2 \geq 0$ are used to prioritize different tasks. However, the objective item ς in Eq. (2.107) is not in QP form and cannot obtain the gradient easily. It can be converted into QP by expressing the infinite norm part as an inequality constraint. We can further convert it into an inequality constraint:

$$\varsigma \geq |\mathbf{I}_i^T\dot{\theta}| \tag{2.108}$$

where $\mathbf{I}_i$ denotes ith column vector of the identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$. Then the inequality expression can be written as:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{1}_{n\times1} \\ -\mathbf{I} & -\mathbf{1}_{n\times1} \end{bmatrix}\begin{bmatrix} \dot{\theta} \\ \varsigma \end{bmatrix} \leq 0 \tag{2.109}$$

where $\mathbf{1}$ denoting a vector composing of 1. Then we can formulate the optimization equations below:

$$E = \begin{bmatrix} \mathbf{I}, & -\mathbf{1}_{n\times1} \\ -\mathbf{I}, & -\mathbf{1}_{n\times1} \end{bmatrix} \in \mathbb{R}^{2n\times(n+1)} \tag{2.110}$$

$$Q = \begin{bmatrix} c_0\mathbf{I} + c_1J_w^TJ_w & \mathbf{0}_{n\times1} \\ \mathbf{0}_{1\times n} & c_2 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} \tag{2.111}$$

$$\Lambda = \begin{bmatrix} J_w & \mathbf{0}_{m\times1} \end{bmatrix} \in \mathbb{R}^{m\times(n+1)} \tag{2.112}$$

$$b = \begin{bmatrix} -c_1J_w^Tv_d \\ 0 \end{bmatrix} \in \mathbb{R}^{n+1} \tag{2.113}$$

$$x = \begin{bmatrix} \dot{\theta} \\ \varsigma \end{bmatrix} \in \mathbb{R}^{n+1} \tag{2.114}$$

$$x^- \leq x \leq x^+ \tag{2.115}$$

where the identity matrix $\mathbf{I} \in \mathbb{R}^{(k+1)\times(k+1)}$. $x^- = [\dot{\theta}^-, 0]^T$ and $x^+ = [\dot{\theta}^+, \sigma]^T$ represent the upper and lower limit of joint velocity respectively and σ is sufficiently large to replace positive infinity constant for implementation purposes.

$$\min_x \quad \frac{1}{2}x^TQx + b^Tx \tag{2.116}$$

$$\text{s.t.} \quad \Lambda x = v_d \tag{2.117}$$

$$Ex \leq 0 \tag{2.118}$$

$$x^- \leq x \leq x^+$$

which is a standard convex QP with decision variable x.

Hierarchical Quadratic Programming

One of the problems in **QP** is the equality constraint might not be satisfied in some cases and a feasible solution cannot be found. Thus, by adding a slack variable $w \in \mathbb{R}^m$, the infeasible solution area can be relaxed. For example, the robot is required to run smoothly and safely in case of violent shaking within an acceptable range of error. In this section, the relaxation variable is taken into account in the kinematic equation, which will relax the large jitters of the joint velocity during the control. A hierarchical quadratic programming model is introduced, where the first optimization problem in terms of the w and x is formulated as:

$$\min_{\omega,x} \quad \|\omega\|_2 \tag{2.119}$$

$$\text{s.t.} \quad \Lambda x + \omega = v_d \tag{2.120}$$

$$x^- \leq x \leq x^+$$

According to the solution of ω from Eq. (2.119), we substitute it into the optimization in Eq. (2.116), and obtain the second optimization problem:

$$\min_{x} \quad \frac{1}{2}x^T Q x + b^T x \tag{2.121}$$

$$\text{s.t.} \quad \Lambda x + \omega = v_d \tag{2.122}$$

$$Ex \leq 0 \tag{2.123}$$

$$x^- \leq x \leq x^+$$

When the feasible solution is not found, the equality constraint is always solvable because ω becomes a non-zero vector. If a feasible solution exists, then ω behaves like a zero vector.

2.3.4 Two-Timescale Neuronal Dynamics Design

To address the hierarchical QP problem in Eqs. (2.119–2.120) and Eqs. (2.121–2.123), the parallel processing capability of neural networks came into useful. Specifically, recurrent neural networks (RNNs) draw attention for manipulator control problems due to high parallelism, adaptivity and circuit implementability [107]. Dynamic behavior of one-time-scale RNNs may change substantially and become unpredictable for searching the global solution [98]. Therefore, the two-time-scale neurodynamic model can be used instead because it is more plausible than the one-time-scale model. Another distinction of neural networks is the number of layers. A single-layer neural network is more delicate to the local minima. Hence multilayer neural networks are used for global optimization. Given the aforementioned considerations, a two-timescale multilayer recurrent neural network(**TNN**) will be adopted in this section.

Let's express our optimization problem in Eq. (2.116) as a general nonlinear programming way below:

$$\min_{x} \quad f(x) \tag{2.124}$$

$$\text{s.t.} \quad h(x) = 0$$

$$g(x) \leq 0$$

where $f(x) = \frac{1}{2}x^T Q x + b^T x$, $h(x) = \Lambda x - v_d$ and $g(x) = Ex$.

Sequential quadratic programming (SQP) is a two-step iterative approach to nonlinear programming. The first step is to solve QP that locally approximates the nonlinear program shown as:

$$\begin{aligned}\min_{y^{(k)}} \quad & \frac{1}{2}(y^{(k)})^T \mathcal{H}(x^{(k)}) y^{(k)} + \nabla f(x^{(k)})^T y^{(k)} \\ \text{s.t.} \quad & \nabla h(x^{(k)})^T y^{(k)} + h(x^{(k)}) = 0 \\ & \nabla g(x^{(k)})^T y^{(k)} + g(x^{(k)}) \leq 0\end{aligned} \tag{2.125}$$

where $x^{(k)}$ and $y^{(k)}$ denote the decision vector and directional vector at the kth iteration, respectively; $\mathcal{H}(x) \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix for approximating Hessian of the Lagrangian of problem; $\nabla g(x) = [\nabla g_1(x), ..., \nabla g_m(x)] \in \mathbb{R}^{n\times m}$ and $\nabla h(x) = [\nabla h_1(x), ..., \nabla h_r(x)] \in \mathbb{R}^{n\times r}$.

The second step iterates decision vector via:

$$x^{(k+1)} = x^{(k)} + \alpha \boldsymbol{y}^{(k)}$$

where α and $\boldsymbol{y}^{(k)}$ are the step size and optimal directional vector of QP, respectively. Iteration of SQP stops when stopping criteria are satisfied.

The Lagrangian function of the problem in Eq. (2.124) is defined as:

$$L(x, \lambda, \eta) = f(x) + \lambda^T g(x) + \eta^T h(x)$$

where $\lambda \in \mathbb{R}^m$ and $\eta \in \mathbb{R}^r$ are dual decision vectors.

Let $L(x^*, \lambda^*, \eta^*)$ be vectors satisfying the Karush–Kuhn–Tucker (KKT) condition of the problem in Eq. 2.124 as follows:

$$\begin{aligned}\nabla L(x^*, \lambda^*, \eta^*) = 0 \quad & g(x^*) \leq 0 \quad & \lambda_i^* g_i(x^*) = 0 \\ \lambda_i^* \geq 0 \quad & i = 1, ..., m \quad & h(x^*) = 0\end{aligned}$$

Let $(x^{(k)}, y^{(k)})$ be replaced by (x, y) in QP Eq. (2.125) and (y^*, μ^*, ν^*) be vectors satisfying the KKT condition of Eq. (2.125) as follows:

$$\begin{aligned}\mathcal{H}(x)y^* + \nabla f(x) + \nabla g(x)\mu^* + \nabla h(x)\nu^* = 0 \\ \nabla g(x)^T y^* + g(x) \leq 0, \quad \mu_i^* \left[\nabla g_i(x)^T y^* + g_i(x)\right] = 0 \\ \mu_i^* \geq 0, \quad i = 1, ..., m, \quad \nabla h(x)^T y^* + h(x) = 0\end{aligned}$$

where

$$\begin{aligned}\nabla f(x) &= \mathcal{H}x + b \\ \nabla g(x) &= \Lambda \\ \nabla h(x) &= E\end{aligned}$$

The global optimum solution to the convex QP problem can be obtained when the KKT condition is satisfied.

$$\epsilon_x \frac{dx}{dt} = y \tag{2.126}$$

$$\epsilon_y \frac{dy}{dt} = F_y(x, y, \mu, \nu) \tag{2.127}$$

$$\epsilon_y \frac{d\mu}{dt} = F_\mu(x, y, \mu, \nu) \tag{2.128}$$

$$\epsilon_y \frac{d\nu}{dt} = F_\nu(x, y, \mu, \nu) \tag{2.129}$$

where ϵ_x and ϵ_y are the time constants, $x \in \mathbb{R}^n$ is an output neural state for decision vector, $y \in \mathbb{R}^n$ is a hidden neuronal state for directional vector μ and ν are hidden neuronal states for handling inequality and equality constraints, respectively. F_y, F_μ, and F_ν are vector-valued functions for solving QPs and they can be designed as:

$$F_y = -(\mathcal{H}(x)y + \nabla f(x) + \nabla g(x)\mu + \nabla h(x)\nu) \tag{2.130}$$

$$F_\mu = -\mu + \left[\mu + \nabla g(x)^T y + g(x)\right]^+ \tag{2.131}$$

$$F_\nu = \nabla h(x)^T y + h(x) \tag{2.132}$$

2.3.5 Experiment

This section consists of two parts. In the first part, a predefined trajectory is given and in the second part, a trajectory is calculated via a BI^2RRT^* path-planning algorithm [89]. The parameters of TNN are chosen as: $c_0 = 0.5$, $c_1 = 0.4$, $c_2 = 0.5$, $\epsilon_x = 0.04$, $\epsilon_y = 0.006$ and $k_1 = 10$. The initial joint angles (i.e $\theta_i = 0$, $i = l, r, 1, ..7$) are selected as 0 for all the tasks. Then, the proposed TNN-based optimization control method is used to solve the trajectory tracking problem with physical limits. We tested the designed tasks by 9-DOFs mobile manipulator using Neurorobotics platform simulator powered by Gazebo. The results as well as the simulation environment [108] are presented in this section.

Predefined Trajectory

In this section, the circular and ‘8’ shapes are selected for predefined trajectories. The circular shape trajectory is defined as:

$$x = 5sin\left(\frac{2\pi t}{T_s}\right)(m) \qquad\qquad y = -5cos\left(\frac{2\pi t}{T_s}\right)(m)$$

The circular shape trajectory in our case is given above equation and the position in z direction is chosen as $1.6m$. Initially, mobile-based manipulator is started with the configurations of $x_0 = 0$, $y_0 = -5$ and $\psi_0 = 0$.

The optimization joint position for circular shape trajectory is run in Neurorobotics platform and shown in Fig. 2.25. As the mobile platform follows the circular trajectory, the θ_l and θ_r (i.e wheel joints) are expected to be increasing. Fig.2.26a illustrates the joint position of the mobile-based manipulator. It can be seen that the left wheel velocity θ_l is lower than the right one θ_r. It is that the right wheel rotates outside and it needs to travel a slightly larger distance in the given time. The other joints $\theta_{1,..7}$ are mostly constant because z position of the desired trajectory is fixed. Joint velocities and auxiliary variables are depicted in Fig.2.26b. As the auxiliary variable of infinite norm chooses the maximum component of a vector, it is equal to the θ_r after $2.35s$. Figure 2.27a represents desired and actual trajectories of end-effector in three dimensional coordinate frames and the actual trajectory converges quickly to the desired trajectory. Fig. 2.27b delineates the tracking error of the end-effector. To demonstrate the proposed multiple metrics of infinity norm and slack variable, there are two comparison experiments are conducted. Figure 2.28a shows the comparison results TNN with TNN-infinity norm metrics of circular shape, where the infinity norm can make the trajectory smoother in the feasible area but cannot work for the infeasible area. Therefore, we need to consider the slack variable for the infeasible area, and the performance is shown in Fig. 2.28b. It can be seen that the TNN-slack outperforms TNN without the slack variable.

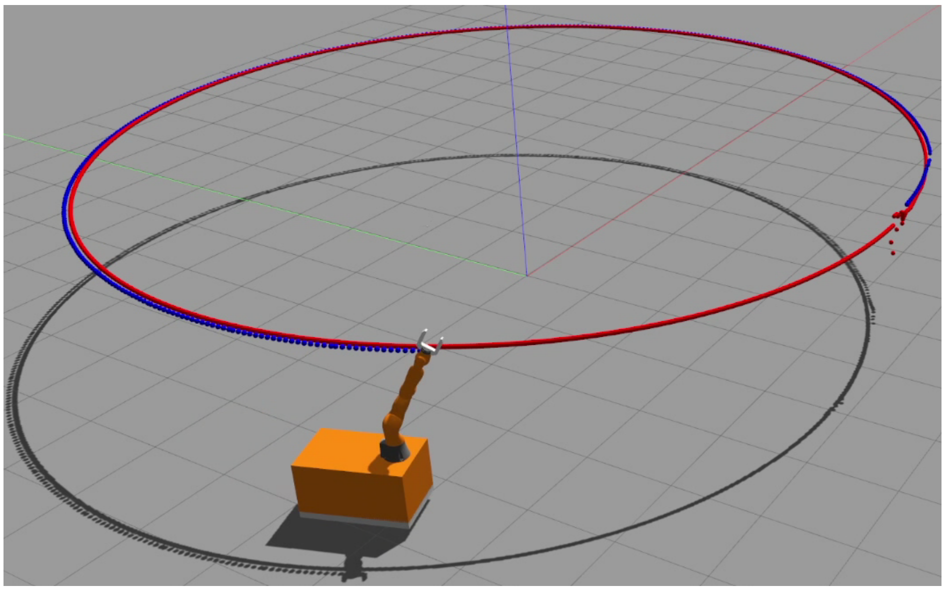
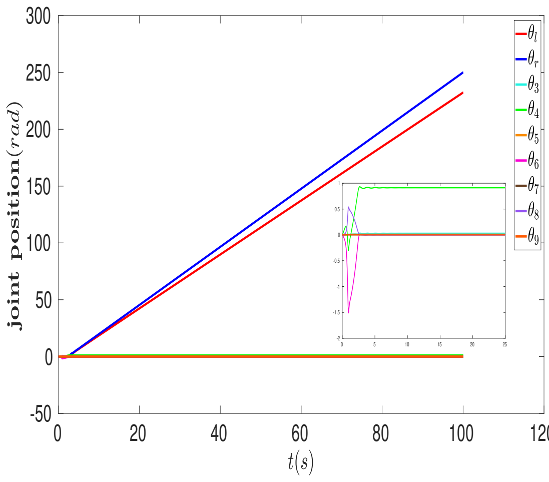
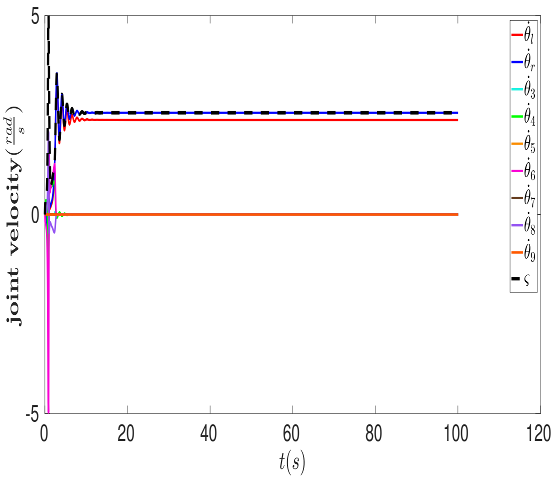


Figure 2.25: Tracking a circular predefined trajectory in simulation environment.

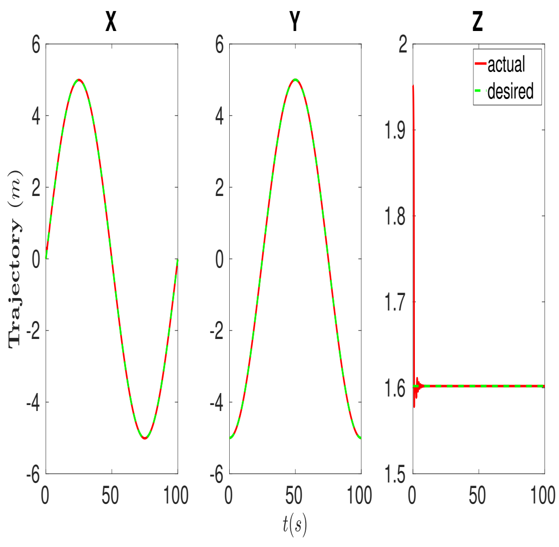


(a) Joint position of circular shape task.

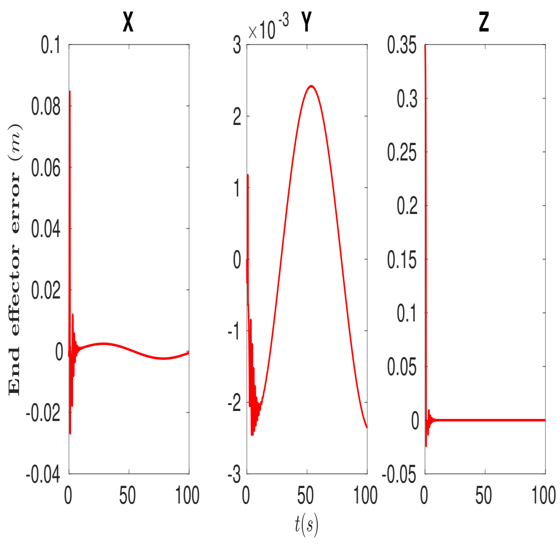


(b) state variable of circular shape task.

Figure 2.26: Circular-shape tracking results including the joint position and state variable.

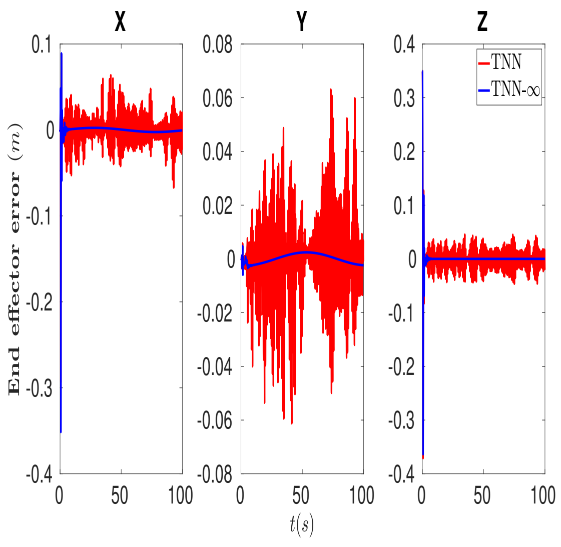


(a) 3D trajectory for circular shape.

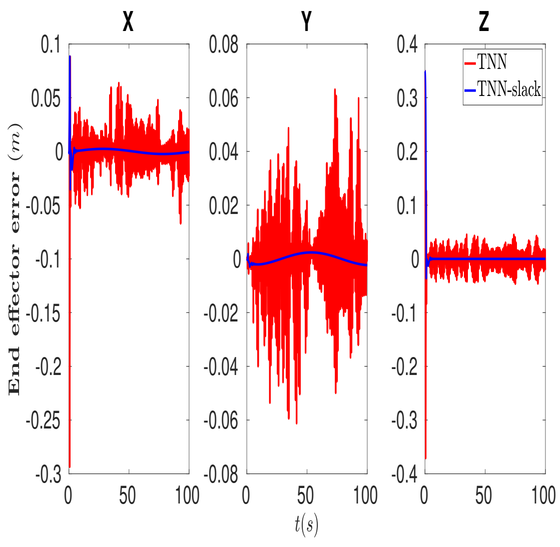


(b) End-effector errors for circular shape task.

Figure 2.27: Circular-shape tracking results including the trajectory and trajectory error.



(a) Circular task comparison: TNN vs TNN-inf.



(b) Circular task comparison: TNN vs TNN-slack.

Figure 2.28: Circular-shape comparison for different metrics: the 2.28b is about the comparison results between TNN with TNN slack variable, and 2.28a is the comparison results between TNN with TNN infinity-norm.

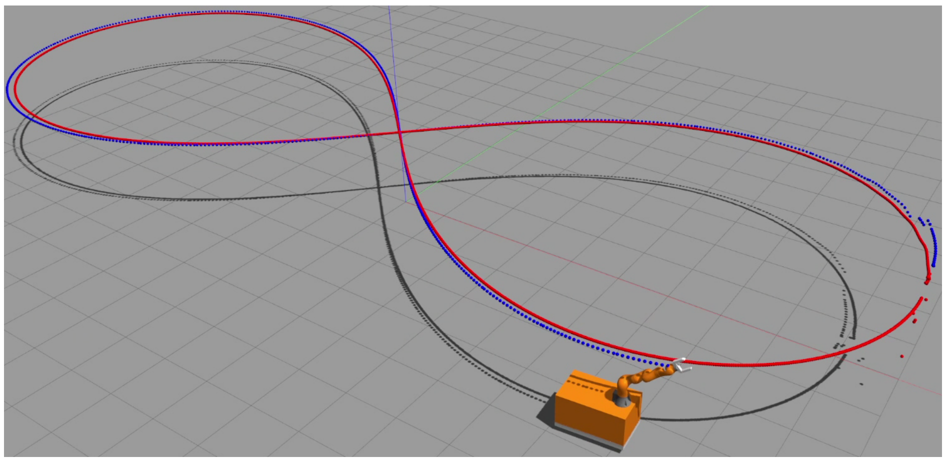
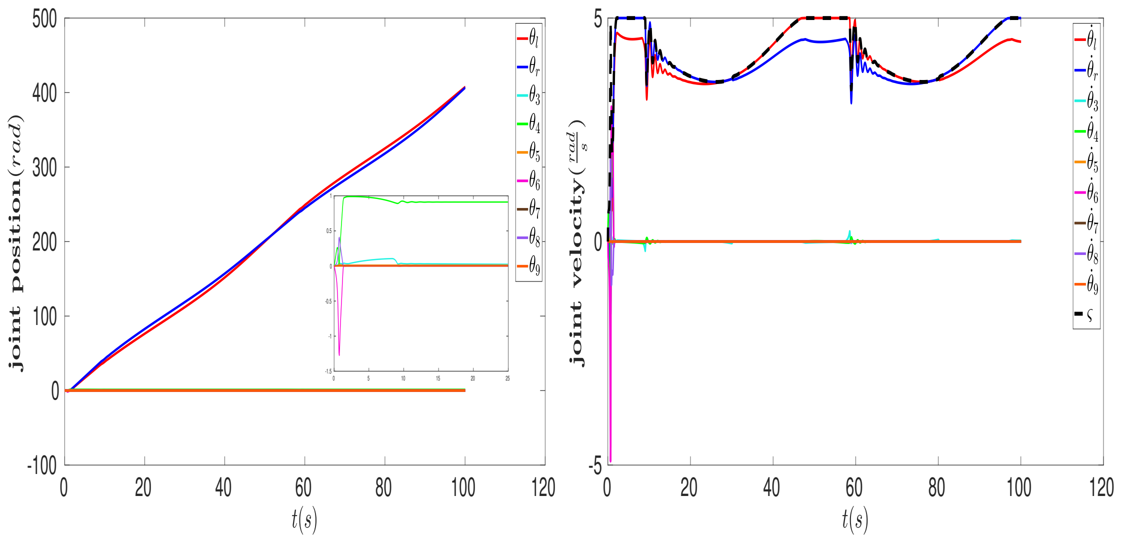
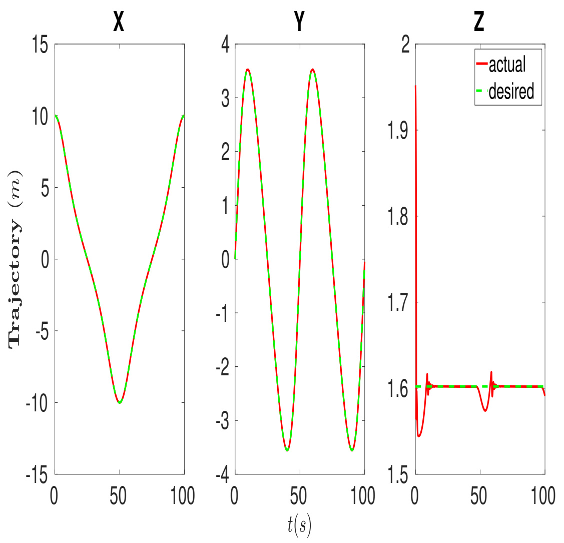


Figure 2.29: Tracking an infinite shape predefined trajectory in simulation environment.

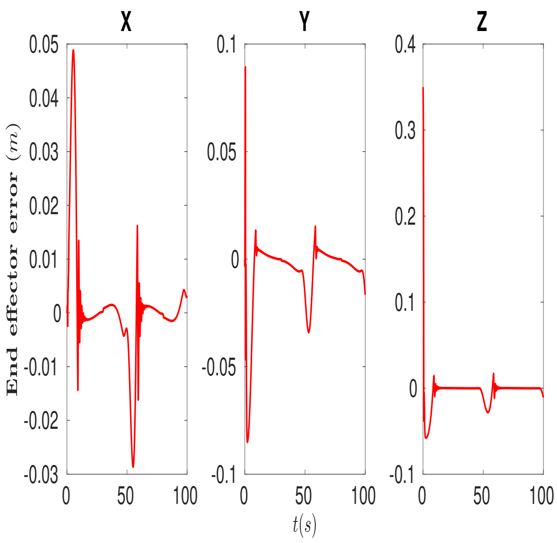


(a) Joint position of infinite shape task. **(b)** State variable of infinite shape task.

Figure 2.30: '8'-shape tracking results including the joint position and state variable.

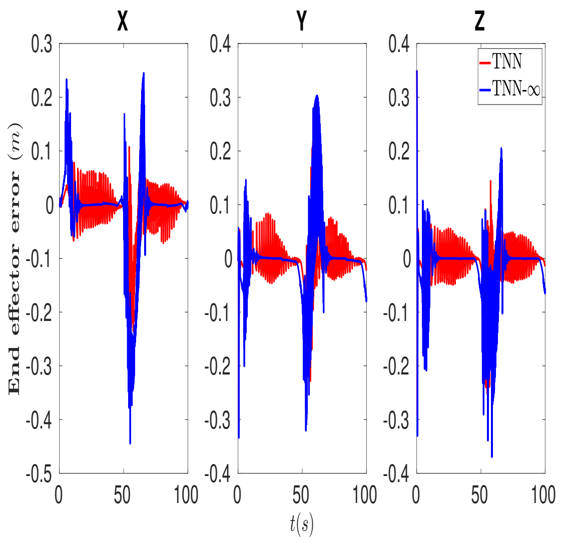


(a) 3D trajectory for circular shape.

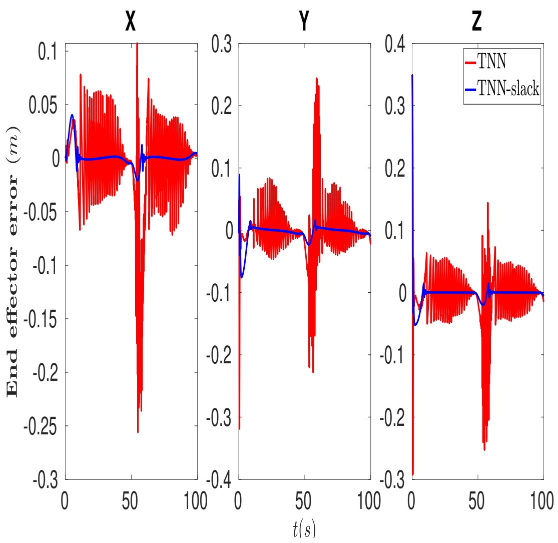


(b) End-effector errors for circular shape task.

Figure 2.31: '8'-shape tracking results including the trajectory and trajectory error.

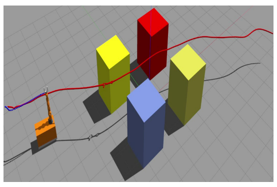


(a) '8'-shape comparison: TNN vs TNN-inf.

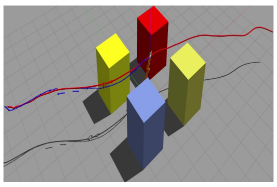


(b) '8'-shape comparison: TNN vs TNN-slack.

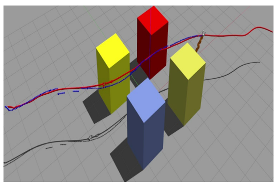
Figure 2.32: '8'-shape comparison for different metrics: the 2.32b is about the comparison results between TNN with TNN slack variable, and 2.32a is the comparison results between TNN with TNN infinity-norm.



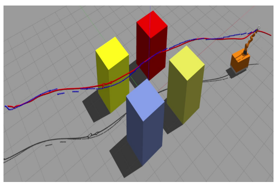
(a)



(b)



(c)



(d)

Figure 2.33: Tracking the trajectory calculated by path-planning in Gazebo simulation.

The ‘8’ shape trajectory given below equation and z direction is chosen as $1.6m$.

$$x = \frac{-10cos\left(\frac{2\pi t}{T_s}\right)}{cos\left(\frac{2\pi t}{T_s}\right)^2 - 2}(m) \qquad y = \frac{5tan\left(\frac{2\pi t}{T_s}\right)}{tan\left(\frac{2\pi t}{T_s}\right)^2 + \frac{1}{2}}(m)$$

Fig. 2.29 represents the tracking task in the simulation environment. Fig. 2.30a demonstrates the joint positions of the mobile-based manipulator. In this case, the right wheel θr rotates outside as well as inside. Therefore, the traveled distance of wheels is changing depending on the region.

Joint velocities and the auxiliary variable of the infinite norm are denoted in Fig. 2.30b. It can be inferred from the figure that given joint constraints forced to throttle back the mobile-based manipulator by limiting the $\dot{\theta}_r$ and $\dot{\theta}_l$.

From the comparison between TNN-infinity norm with TNN in the Fig. 2.32a , the auxiliary variable ς selects the maximum joint velocity successfully, and we can get a similar conclusion from the circular task. The effects of using slack variable can be seen in Fig. 2.32b. Apparently, TNN cannot satisfy the equality constraints without using the slack variable and therefore the end-effector has jittery behavior.

Optimization the Trajectory from Path Planning

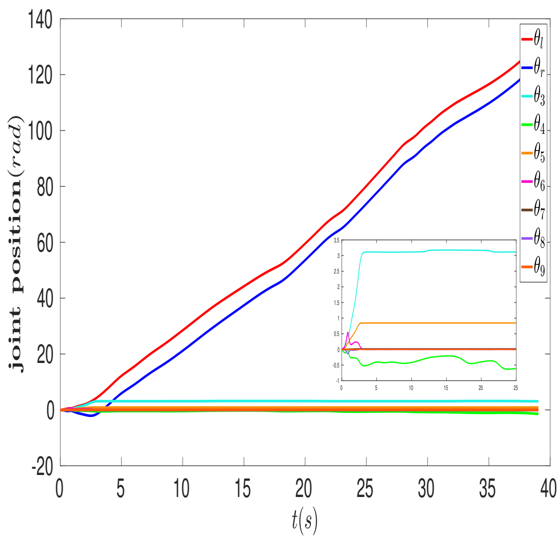
In this task, four cubes are located in the middle as obstacles in the environment. At first, the path-planning algorithm takes the start position of end-effector as $x_0 = -7.67$, $y_0 = 2.65$, $z_0 = 1.95$ and final position as $x_f = 8.64$, $y_f = -3$, $z_f = 1.2$. Then the desired trajectory is calculated based on the obstacles in the environment. Finally, the trajectory is solved by the TNN algorithm and the calculated joint positions are given to the simulator to test the task. Initially, mobile-based manipulator is started with the configurations of $x_0 = -7.67$, $y_0 = 2.65$ and $\psi_0 = 1.95$.

The different snapshots of the tested path-planning experiment are shown in Fig. 2.33. The red line indicates the desired trajectory calculated by the path-planning algorithm and the blue line is the actual trajectory followed by the end-effector of the mobile-based manipulator.

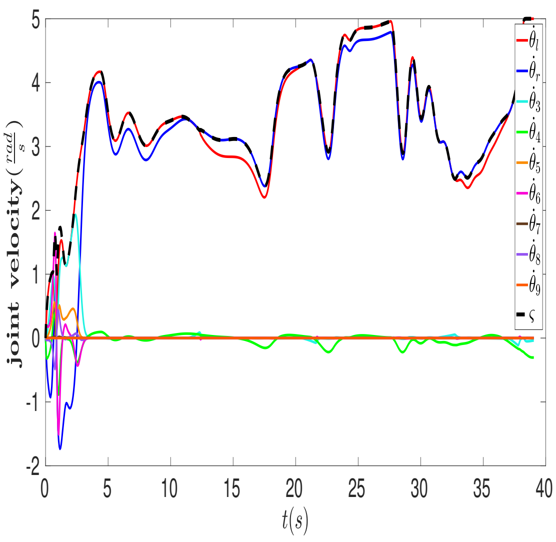
Fig. 2.34a shows the joint position together with mobile and manipulator. The small window on the figure represents only the joint positions of the Kuka arm manipulator. The joint velocities and the auxiliary variable are illustrated in Fig. 2.34b. Obviously, the proposed planning and optimization control method work efficiently for trajectory tracking in terms of obstacle avoidance.

2.3.6 Summary

In this section, the kinematic equations of the 9DOF mobile-based manipulator are initially derived. Then the optimization problem with slack variable and infinity-norm is formulated. Afterward two-timescale multi-layer recurrent neural network (TNN) is designed. Finally, the tracking control problem is tested using the 9DOF mobile-based manipulator for two different cases. In the first case, a predefined desired trajectory is given and tracked by the proposed approach. In the second case, BI^2RRT^* path planning algorithm determines the desired trajectory with obstacles. The presented results validate that the mobile-based manipulator can successfully follow the trajectory with the given joint angles from the proposed algorithm even if the joint limits restrict its mobility considerably. Slack variables and infinity norm are leveraged by optimization algorithm and compared their impacts on end-effector error in the tracking problem. It can be deduced that taking them into account can provide tracking tasks with less error as well as smooth and non-jittery motion.

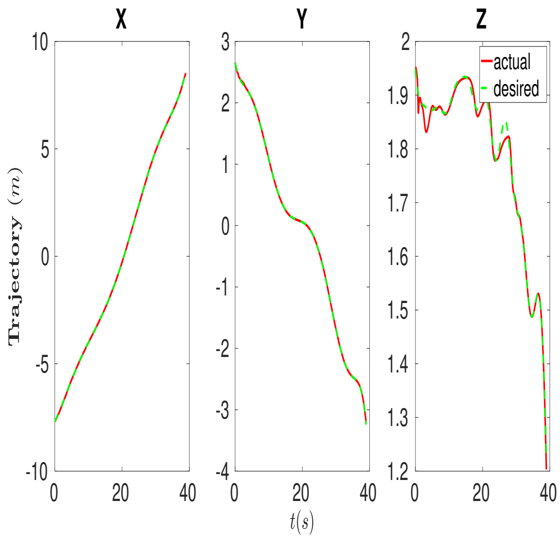


(a) Joint position of obstacle avoidance task.

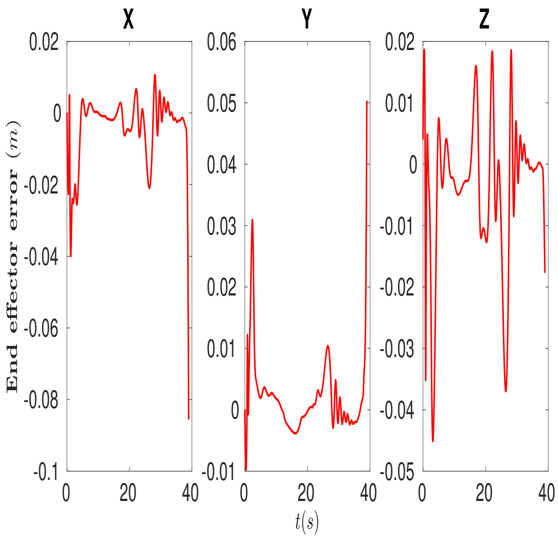


(b) State variable of obstacle avoidance task.

Figure 2.34: Obstacle avoidance task tracking results including the joint position and state variable.



(a) 3D trajectory for obstacle avoidance task.



(b) End-effector errors for obstacle avoidance task.

Figure 2.35: Obstacle avoidance task tracking results including the trajectory and trajectory error.

Future works will entail the dynamics of the mobile-based manipulator and usage of BI^2RRT^* path planning algorithm in more complex scenarios.

2.4 Related Publication

1 **Yingbai Hu**, Jian Li, Yongquan Chen, Qiwen Wang, Chuliang Chi, Heng Zhang, Qing Gao, Yuanmin Lan, Zheng Li, Zonggao Mu, Zhenglong Sun, Alois Knoll. "Design and Control of a Highly Redundant Rigid-Flexible Coupling Robot to Assist the COVID-19 Oropharyngeal-Swab Sampling". In: *IEEE Robotics and Automation Letters* (2021).

2 **Yingbai Hu**, Hang Su, Guang Chen, Giancarlo Ferrigno, Elena De Momi, Alois Knoll. "Hierarchical optimization Control of Redundant Manipulator for Robot-assisted Minimally Invasive Surgery". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2929–2934.

3 Hang Su, **Yingbai Hu***, Jiehao Li, Jing Guo, Yuan Liu, Mengyao Li, Alois Knoll, Giancarlo Ferrigno, Elena De Momi. "Improving Motion Planning for Surgical Robot with Active Constraints". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3151–3156.

4 Hang Su, **Yingbai Hu**, Hamid Reza Karimi, Alois Knoll, Giancarlo Ferrigno, Elena De Momi. "Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results". In: *Neural Networks* 131 (2020), pp. 291-299.

5 **Yingbai Hu**, Hang Su, Longbin Zhang, Miao Shu, Guang Chen, Alois Knoll. "Nonlinear Model Predictive Control for Mobile Robot Using Varying-Parameter Convergent Differential Neural Network". In: *Robotics* 8.3 (2019), pp. 64.

Chapter 3

Imitation Learning for Manipulation

3.1 Reinforcement Learning for Manipulation

In this section, we will discuss the reinforcement learning method based manipulation skill transferring technique for surgical robot [109]. The complexity of surgical operation can be released significantly if surgical robots can learn the manipulation skills by imitation from complex tasks demonstrations such as puncture, suturing, and knotting, etc.. This section proposes a reinforcement learning algorithm based manipulation skill transferring technique for robot-assisted minimally invasive surgery by teaching by demonstration. It employed Gaussian mixture model and Gaussian mixture Regression based dynamic movement primitive to model the high-dimensional human-like manipulation skill after multiple demonstrations. Furthermore, this approach fascinates the learning and trial phase performed offline, which reduces the risks and cost for the practical surgical operation. Finally, it is demonstrated by transferring manipulation skills for reaching and puncture using a KUKA LWR4+ robot in a lab setup environment. The results show the effectiveness of the proposed approach for modelling and learning of human manipulation skill.

3.1.1 Introduction

Robotic surgery has been a compelling emerging technology that holds significant promise due to the benefits it provides for surgeons, such as higher operational accuracy, extended motion range, and augmented visualization [110]. However, due to the kinematic constraints imposed by the laparoscopic surgery, i.e., which are known as the remote center of motion (RCM) constraints, the surgical operation is performed in a limited space [111]. It turns the intuitive manipulation in the conventional open surgery to time-consuming tasks [112]. The complexity of surgical operation can be released significantly if surgical robots can learn the manipulation skills by imitation from complex tasks demonstrations [113] such as puncture and knotting, etc..

Current commercial surgical robots are simply controlled by surgeons using their hands, and they involve less of autonomy on the surgical operation [114]. With the development of technology in artificial intelligence and cognition progress, increasing the autonomy of surgical robots in performing some specific complex surgical operations, like suturing or knotting, can potentially reduce the length of surgical procedures and surgeon fatigue, as well as improved accuracy [115]. Hence, the need for developing methodology and technology in surgical manipulation skill transferring reinforces.

Teaching by demonstration (TbD) has drawn extensive research attention in manipulation skill transferring from human to robot during the past decades [116, 117]. Calinon *et al.* [118, 119] investigated the methods to assign human motion skills to the robot manipulators. Dynamic movement primitive (DMP) proposed by Meier *et al.* [120] is an efficient approach to learn motor primitives for robot manipulation. In the motion modeling paradigm, each manipulation procedure features motion primitives and the corresponding goal parameters. Kormushev *et al.* [121] studied comprehend the trajectory generation for the spherical impediment by using DMP modelling combined with the synthetic capacity discipline method using Gaussian mixture Regression (GMR). For complex manipulations modelling, the reinforcement learning algorithm is capable of adapting the goal parameters with a high-dimension motion primitive [122, 123].

It is interesting to introduce GMR based motion primitives modeling strategy to learn the surgical operation skills from experienced surgeons [26]. Furthermore, it is proposed to handle the RCM constraint as sub-goal by offline tasks learning and trials using reinforcement learning. In this section, a special reinforcement learning named Policy Improvement with Path Integrals (**PI**^{2}) is proposed to optimize the path planing, which is derived from probability-based stochastic optimal control theory [122][124]. The reinforcement learning method can optimize the trajectory with disturbance by updating the parameters [125]. In addition, we can design the cost function to explore the different learning tasks even multi-task at a learning system. The **PI**^{2} is suitable for the high dimensions problem such as Cartesian space [126]. Therefore, it is convenient for robot learning problem.

It must be noticed that the proposed methodology represents an improvement with respect to the simple path planning between the start and end-point introduced in [122] [71], and combines an sub-goal task to respect the RCM constraint [64] [70] on the planned path. It means the planned path should consider not only the shape but also passing through the small incisions on the abdominal wall. Furthermore, it fascinates the learning and trial phase performed offline, which reduces the risks and cost for the practical surgical operation. Finally, experiments have been performed to demonstrate the proposed control method on a 3-D printed patient phantom using a 7-DOF robot manipulator KUKA LWR4+ .

The section structure is organized as follows. The motivation and previous works involved this section is explained in Section 3.1.2. Section 3.1.3 describes the corresponding control methodology and control framework are presented. In Section 3.1.4 the effectiveness of the proposed control scheme is demonstrated using KUKA LWR4+ robot manipulator, and conclusions are drawn in Section 3.1.5.

3.1.2 Motivation and Previous Works

In our previous works [64] [70], we inserted the surgical tool into the abdominal cavity passing through the RCM constraint shown in Fig. 3.1, by hands-on control. To reduce the complexity of operation procedures, it is interesting to utilize the TbD techniques to model and learning the surgeons' manipulation skill and transfer it to the robot, increasing the autonomy of surgical robots.

In our previous work, we utilized the reinforcement learning to learn the complex motion sequences in human-robot environment such that the robots can adapt its motions for manipulation and grasping of a mobile manipulator [71]. Nevertheless, the above algorithms considered only point to point problems, which determine the trajectory between the start and end-point of the movement, ignoring the other goals in the sub-tasks [122], such as passing through a kinematic constraint. In this section, it is suggested to include the RCM constraint as a new challenge for the manipulation skill modelling and transferring. We therefore extend the policy improvement with path integrals (**PI**^{2}) algorithm to simultaneously optimize shape and goal parameters.

3.1.3 Methodology

The control methodology here proposed aims to provide consistent and effective skill modelling and transferring techniques for robot-assisted minimally invasive surgery, learning the motion primitives of a specific task from demonstration operations operated by a surgeon, and plan the path with respect to the kinematic constraint (RCM constraint) between the start and end-point. The robot model has been discussed in [64].

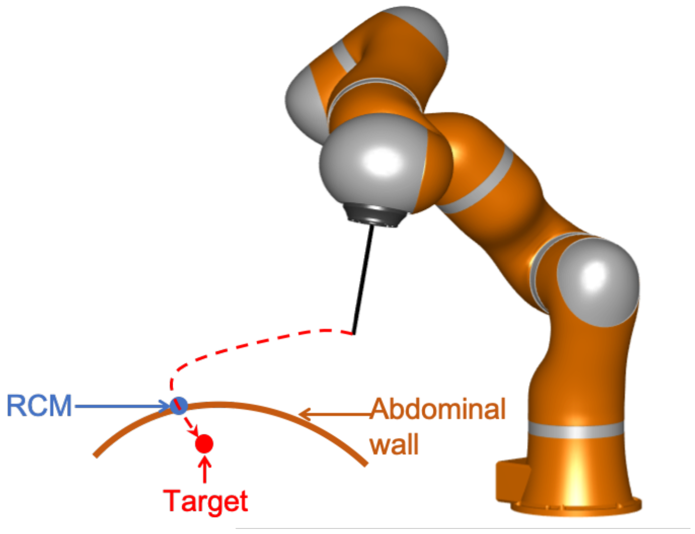


Figure 3.1: Path planning for puncture in robot-assisted surgery operation. The planned path should always pass trough the small incision on the abdonimal wall (RCM constriant).

Dynamic Movement Primitive

Given the continuous stream of movements that biological systems exhibit in their daily activities, dynamic movement primitive [71] now is a general approach in artificial and biological systems revolves around identifying movement primitives for motor control in robotics and biology. Dynamic movement primitive is represented as a set of equations, and it can model different linear or nonlinear motions which is convenient to imitate learning complex movement fusion with reinforcement learning algorithm. The dynamic movement primitive is expressed as:

$$\begin{aligned} &\ddot{X}_t = K_p\left(g - X_t\right) - K_v \dot{X}_t + F\left(s_t\right) \\ &\dot{s}_t = \alpha_s s_t \\ &F\left(s_t\right) = h_t^{\mathrm{T}}\left(s_t\right) \omega \left(g - X_0\right) \end{aligned} \tag{3.1}$$

$$h_t\left(s_t\right) = \frac{\sum_{i=1}^{N} \psi_i\left(s_t\right) s_t}{\sum_{i=1}^{N} \psi_i(s_t)_t}$$

$$\psi_i\left(s_t\right) = \exp\left(-\frac{1}{2\sigma_i}(s_t - c_i)^2\right)$$

where $\left[X_t, \dot{X}_t, \ddot{X}_t\right]$ is the Cartesian space trajectory; X_0 and g present the initial position and goal position of the attractor point in Cartesian space, respectively; K_p and K_d are the stiffness matrix, damping term of DMP in 3D Cartesian space. ω is the shape parameter of DMP; α_s is

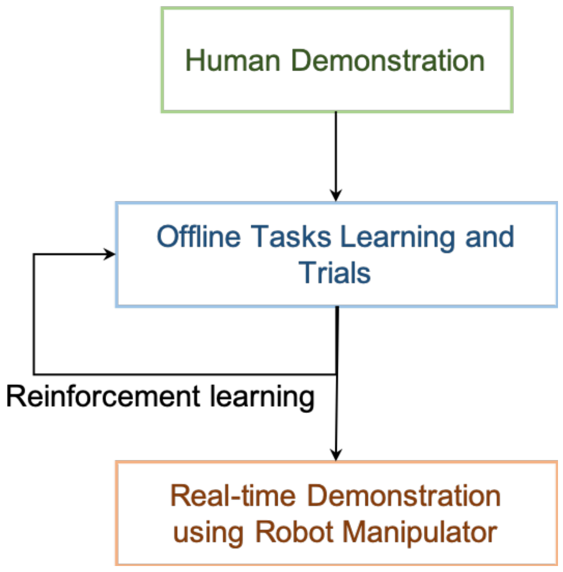

Human Demonstration
Offline Tasks Learning and Trials
Reinforcement learning
Real-time Demonstration using Robot Manipulator

Figure 3.2: Experimental procedures.

the scale parameter of Canonical system, where s_t asymptotically decays from 1 to 0; σ_i and c_i are bandwidth and center of the i-th Gaussian kernels.

It should be noted that DMP is consist of two parts including linear spring damper part $K_p(g-X_t)-K_v\dot{X}_t$ and nonlinear part $F(s_t)$ which can be applied to model the trajectories from teaching by demonstrations even the nonlinear system. Therefore, DMP is convenient to imitate the motions from human because of the feature of convergence to the attract point g.

Gaussian Mixture Model

In this part, the Gaussian Mixture Model is presented to encode the trajectories from teaching. Gaussian Mixture Model is a probability-based statistical model which can describe the probability density distribution of high-dimensional dataset by the sum of different weights of multiple Gaussian models [127]. In this section, the GMM is used to describe the position density in Cartesian space and obtain nonlinear item in in DMP by regression from each GMM. The DMP framework of multi-demonstrations is reformulated as by K component Gaussian model,

$$\ddot{X} = \sum_{k=1}^{K} h_k \left(K_k^p \left(\mu_k^X - X \right) - K_k^V \dot{X} + F \right) \tag{3.2}$$

The Cartesian space data point from demonstrations are defined as: $s_j = \left(s_{t,j}, s_{X,j}\right)(j = 1, \ldots, N)$, where N is the length of dataset. Each datapoint include the time temporal value $s_{t,j}$ and position value $s_{X,j}$. To encode the dataset of position distribution $P(s_t, s_X)$, the following GMM model is defined as

$$p\left(s_j\right) = \sum_{k=1}^{K} p(k) p\left(s_j | k\right) \tag{3.3}$$

where K is the number of the Gaussian model; $p(k)$ denotes the prior probability, and $p\left(s_j|k\right)$ is the conditional probability density function.

The manipulator works in 3-D space, so the parameters in (3.3) are denoted as:

$$\begin{aligned} p(k) &= \lambda_k \\ p\left(s_j|k\right) &= \frac{1}{\sqrt{(2\pi)^3 |\Sigma_k|}} \mathrm{e}^{\left(-\frac{1}{2}(s_j-\mu_k)^{\mathrm{T}} \Sigma_k^{-1} (s_j-\mu_k)\right)} \end{aligned} \tag{3.4}$$

We define the GMM parameters as $\Theta = \{\lambda_k, \mu_k, \Sigma_k, E_k\}$, where λ_k, Σ_k, Σ_k, E_k are prior probability, mean variable, covariance variable and cumulative posterior probability, respectively. According to Bayes theorem, the cumulative posterior probability E_k can be expressed as,

$$\begin{aligned} E_k &= \sum_{j=1}^{N} p\left(k|s_j\right) \\ p\left(k|s_j\right) &= \frac{p(k)p(s_j|k)}{\sum\limits_{m=1}^{K} p(m)p(s_j|m)} \end{aligned} \tag{3.5}$$

Then, the log-likelihood of GMM model Θ is defined,

$$\mathcal{L}_\Theta = \frac{1}{N}\sum_{j=1}^{N}\log\left(p(s_j)\right) \tag{3.6}$$

where $p\left(s_j\right) = \sum\limits_{k=1}^{K} p(k)P\left(s_j|k\right)$. To estimate GMM parameters $\Theta = \{\lambda_k, \mu_k, \Sigma_k, E_k\}$, the EM algorithm is proposed to train the model parameters, and we will obtain the model parameters after the parameters convergence. The iteration finished step is set when $\frac{\mathcal{L}_\Theta^{(t+1)}}{\mathcal{L}_\Theta^{(t)}} \leq 0.01$.

Gaussian Mixture Regression

Actually, the aim of training is to get the regression parameter F from the dataset. After the multi-demonstrations probability distributions is obtained by GMM, then the Gaussian Mixture Regression (GMM) is proposed to reconstruct the general form for the dataset.

To estimate the conditional expectation value, the observations parameters is defined as: $s = \{s_t, s_X\}$ where s_X is the spatial variable at time step s_t. So, the goal of regression is to estimate the conditional expectation of s_X when the time step s_t is fist given.

For multi-demonstrations from teaching, the GMM Θ encodes the set of trajectories from robot in Cartesian space. The k component of Gaussian mixture model is defined as,

$$\mu_k = \left\{\mu_{t,k}, \mu_{X,k}\right\} \quad , \quad \Sigma_k = \begin{pmatrix} \Sigma_{tt,k} & \Sigma_{tX,k} \\ \Sigma_{Xt,k} & \Sigma_{XX,k} \end{pmatrix} \tag{3.7}$$

where μ_k and Σ_k are mean and covariance matrix of k component GMM. When the time step s_t is given, the expected distribution $s_{X,k}$ of k-th component is expressed as,

$$\begin{aligned} p\left(s_{X,k}|s_t, k\right) &= \mathcal{N}\left(s_{X,k}; \hat{s}_{X,k}, \hat{\Sigma}_{XX,k}\right) \\ \hat{s}_{X,k} &= \mu_{X,k} + \sum_{Xt,k}\left(\Sigma_{tt,k}\right)^{-1}\left(s_t - \mu_{t,k}\right) \\ \hat{\Sigma}_{XX,k} &= \Sigma_{XX,k} - \Sigma_{Xt,k}\left(\Sigma_{tt,k}\right)^{-1}\Sigma_{tX,k} \end{aligned} \tag{3.8}$$

where $\hat{s}_{X,k}$ and $\hat{\Sigma}_{XX,k}$ are mixed from probability. According to the GMM parameters $\Theta = \{\lambda_k, \mu_k, \Sigma_k, E_k\}$, the condition probability density is obtained as,

$$p\left(s_X|s_t\right) = \sum_{k=1}^{K} h_k \mathcal{N}\left(s_X; \hat{s}_{X,k}, \hat{\Sigma}_{XX,k}\right) \tag{3.9}$$

$$h_k = \frac{p(k)p\left(s_t|k\right)}{\sum_{i=1}^{K} p(i)p\left(s_t|i\right)} = \frac{\lambda_k \mathcal{N}\left(s_t; \mu_{t,k}, \Sigma_{tt,k}\right)}{\sum_{i=1}^{K} \lambda_i \mathcal{N}\left(s_t; \mu_{t,i}, \Sigma_{tt,i}\right)}$$

From (3.8) and (3.9), the estimation of condition expectation s_X and covariance matrix are concluded as,

$$\hat{s}_X = \sum_{k=1}^{K} h_k \hat{s}_{X,k} \quad , \quad \hat{\Sigma}_{XX} = \sum_{k=1}^{K} h_k^2 \hat{\Sigma}_{XX,k} \tag{3.10}$$

Therefore, the motion $\hat{s} = \{\hat{s}_t, \hat{s}_X\}$ can be generated by estimating $\left\{\hat{s}_X, \hat{\Sigma}_{XX}\right\}$ at time step s_t.

Reinforcement Learning for Trajectories Optimization

In this section, the reinforcement learning is proposed to learning trajectories by learning the shape parameter ω with the noise added. The cost function of Reinforcement learning is defined as,

$$S(T_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2}\omega^T R\omega)dt \tag{3.11}$$

where T_i denotes the trajectory. The cost function S is consist of three parts: terminal cost ϕ_{t_N}, immediate cost r_t, and immediate control $\frac{1}{2}\omega^T R\omega$.

If the noise is added to the shape parameter of DMP ($\omega + \epsilon_t$), the trajectories would deviation from expectations trajectories. Therefore, the reinforcement learning is applied to learn the shape ω parameter from random noise, and the cost function is reformulated as,

$$S(T_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t + \frac{1}{2}(\omega + M_t\epsilon_t)^{\mathrm{T}} R(\omega + M_t\epsilon_t) \tag{3.12}$$

where M_t indicates projection matrix onto the range space of h_t which is defined as,

$$M_t = \frac{R^{-1}h_t h_t^T}{h_t^{\mathrm{T}} R^{-1} h_t}$$

The learning system is aim at minimizing the cost function S by learning the shape parameter ω.

According to the stochastic optimal control theory [122], the path integral of cost function is defined as,

$$\delta\omega_t = \int P(T_i) M_t dT_i \tag{3.13}$$

where $p(T_i)$ is the probability of trajectory T_i and it is expressed as,

$$P(T_i) = \frac{\exp\left(-\frac{1}{\gamma}S(T_i)\right)}{\int \exp\left(-\frac{1}{\gamma}S(T_i)\right) dT_i} \tag{3.14}$$

Then, the change of $\delta\omega$ can be concluded as,

$$[\delta\omega]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}\left[\delta\omega_{t_i}\right]_j}{\sum_{i=0}^{N-1} w_{j,t_i}(N-i)} \tag{3.15}$$

where $[\delta\omega]_j$ denotes the j-th element of shape parameter ω. Finally, the new parameters can be obtained,

$$\omega^{new} = \omega + \delta\omega \tag{3.16}$$

The update rule of reinforcement learning is shown in **Algorithm. 2**.

Algorithm 2: Reinforcement Learning Update Rule

```
Initialization:
$r_t$, $\phi_{t_N}$, $f(s_t)$, $\omega_0$, initial state $X_0$.
while stop after cost convergence do
    1. for $n$=1 $\cdots$ $N_k$ do
        • sample from dataset with random noise.
        $\epsilon_{t,n} \sim \mathcal{N}(0, \sigma^2\mathbf{\Sigma})$
        • compute the cost and probability.
        $S_{T_i,n} = S(\omega + \epsilon_{t,n})$
        $P_{T_i,n} = \frac{\exp\left(-\frac{1}{\gamma}S_{T_i,n}\right)}{\int \exp\left(-\frac{1}{\gamma}S_{T_i,n}\right)dT_i}$
    end
    2. Update mean:
    $\delta\omega_t = \sum_{n=1}^{N_k} P_{T_i,n}\epsilon_{t,n}.$
    $[\delta\omega]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}\left[\delta\omega_{t_i}\right]_j}{\sum_{i=0}^{N-1}w_{j,t_i}(N-i)}$
    3. Update parameters::
    $\omega^{new} = \omega + \delta\omega$
end
```

3.1.4 Experimental Demonstration

The procedure of the demonstration divided into three phases, shown in Fig. 3.2 , including demonstration phase, reinforcement learning based offline tasks learning and trials, and reinforcement learning based real-time demonstration.

Demonstration Phase

A 3-D printed patient phantom served as the abdomen cavity are used for demonstration, shown in Fig. 3.3. The surgeon uses hands-on control to insert the surgical tool into the surgical cavity with a repetition of 7 times from different initial point.

Then, dynamic movement primitive is applied to encode the multi-demonstrations from human teaching. The regression results are shown in Figs. 3.4-3.5b. Fig. 3.4 shows that all the reproductions can pass through the original RCM and converge to the goal point even from random initial position.

Offline Tasks Learning and Trials

Considering the disturbances (goal g and shape ω noise) in the environment, reinforcement learning is applied to optimize the trajectory. The immediate cost is designed as,

$$r_t = \eta_1 \|d_X\| + \eta_2 \left\|\ddot{X}_t\right\|$$

where d_X is the distance from the end-effector to the line connecting start and end-point of the movement. The learning results are shown in Fig. 3.6a. As the number of iterations increases, all the samples gradually converge to the original trajectory with random noise which proves the effectiveness of the proposed methods. Furthermore, RCM constraint is usually not fixed according the actual surgical scenario. Therefore, we hope the manipulator also can learn how to pass through the new RCM for the same tasks without human demonstrations. Hence,

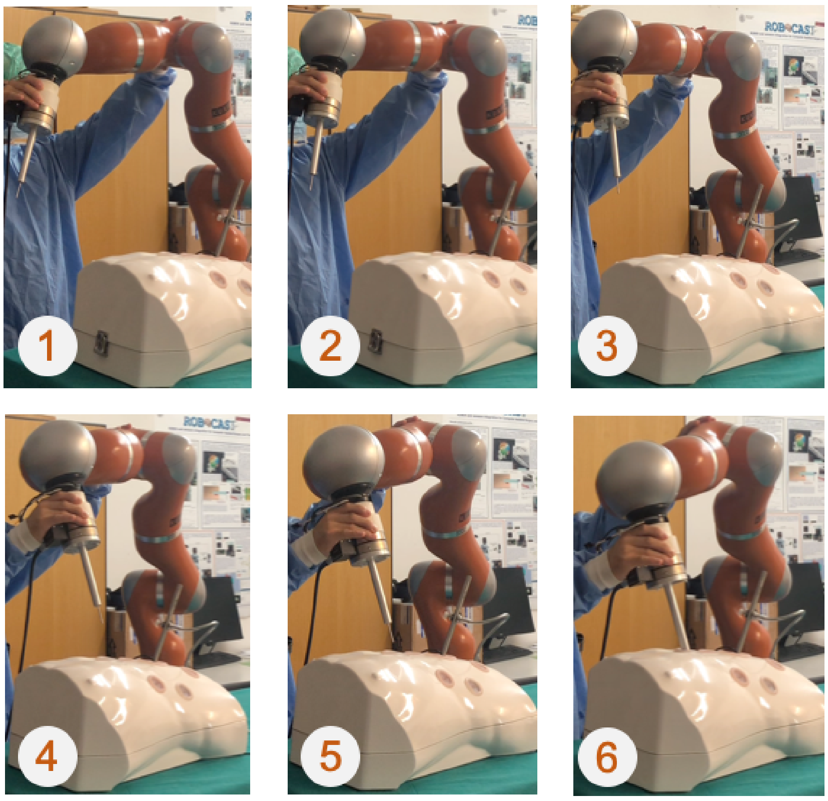


Figure 3.3: Demonstration of puncture procedure. The numbers (1-6) indicate the puncture procedure by hands-on demonstration. The 1st picture shows the start point of the tracking tasks, and the 6th picture represents the corresponding end point. The "surgeon" use hands to hold on end-effector and insert the tool tip into the abdominal cavity.

the demand to respect the new RCM constraint is treated as sub-goal. Reinforcement learning based offline training and trials are implemented to enable the manipulator to pass through the new rcm constraint without clinical trial. The significance of offline training and trials is to reduce machine wear and avoid dangers in actual experiments. In this section, simulation guides the actual experiment which means the learning process is running in simulation and the learning results is directly used in actual experiments.

The learning and trials results with new RCM constraint are shown in Figs. 3.6b-3.7a, which prove that the reinforcement learning enhanced **PI**^{2} can adapt the motion skill to meet the new task requirement without re-teaching.

It should be mentioned that in the new RCM task, the immediate cost is re-designed as,

$$r_t = \eta_1 \|d_X\| + \eta_2 \left\|\ddot{X}_t\right\| + \eta_3 \left\|X_t - P_{RCM}^{new}\right\|$$

P_{RCM} denotes the manipulator passing through RCM which is set by user. The parameters $K_p = diag[1,1,1,1,1,1,1]$, $K_v = \sqrt{2K_p}$; $\alpha_s = 0.01$; the components of GMM is set $K = 10$; $\eta_1 = 10^6$, $\eta_2 = 10^3$, $\eta_3 = 10^{10}$. $P_{RCM} = [-0.438; 0.4349; 0.2429]m$, the new $P_{RCM}^{new} = [-0.345; 0.4342; 0.2521]m$.

Real-time Demonstration with Robot Manipulator

After the offline learning process, the learning results will be applied to demonstrate in actual experiment. The robot perform the insertion of the surgical tool into the abdominal cavity autonomously. Fig. 3.8 shows one of the demonstrated experiment using KUKA LWR4+ robot manipulator and Fig. 3.9 shows its performed trajectory.

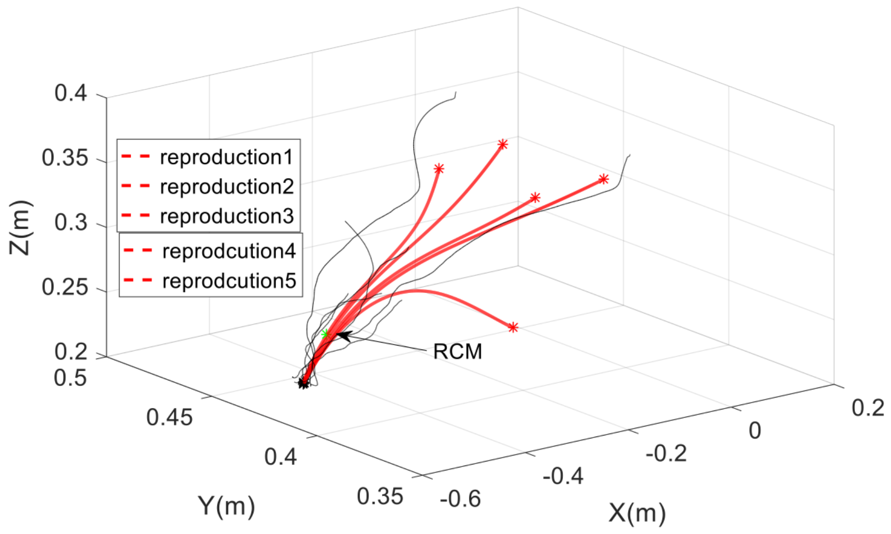
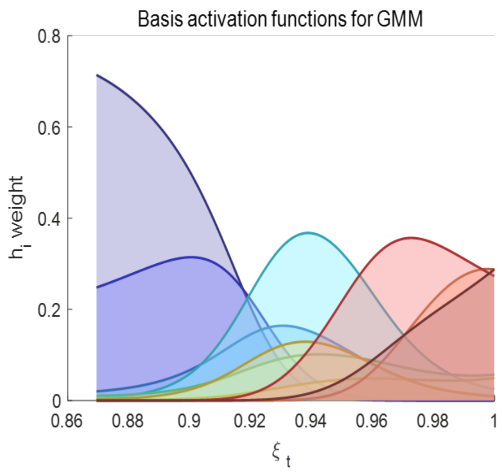


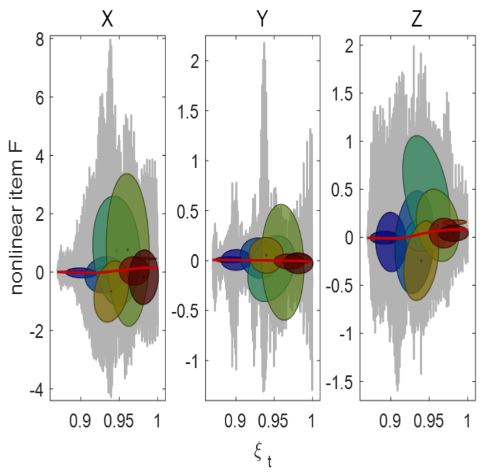
Figure 3.4: Reproduction from random initial position via RCM tasks by multi-demonstrations using GMM-GMR. The black curves denote the multi-demonstrations.

3.1.5 Summary

This section proposes a reinforcement learning algorithm based manipulation skill transferring technique for robot-assisted Minimally Invasive Surgery by Teaching by Demonstration. This approach fascinates the learning and trial phase performed offline, which reduces the risks and cost for the practical surgical operation. Reinforce learning is adopted to model the manipulation skill with trials offline until it can handle the varying kinematic constraints. The results have demonstrated the effectiveness of the proposed approach for modelling and learning of human manipulation skill. However, this work considers only kinematic constraints, ignoring the force from physical interaction on the abdominal wall. Future works will involve physical interaction analysis.

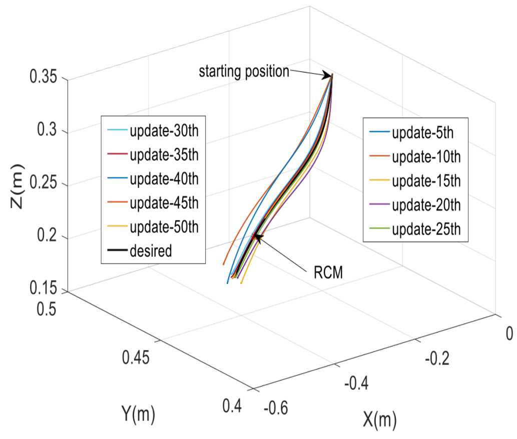


(a) Basic function of GMM.

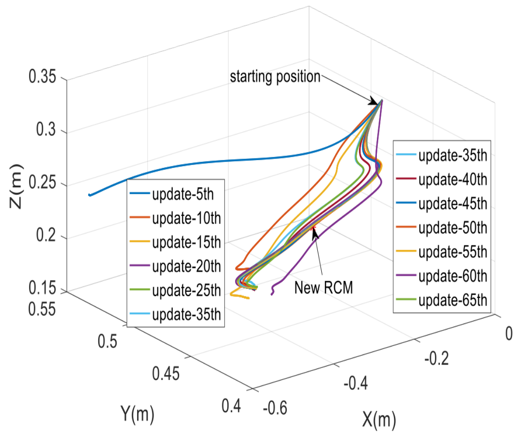


(b) Regression result of nonlinear term of motion primitives using GMM-GMR from multi-demonstrations.

Figure 3.5: Learning with GMM.

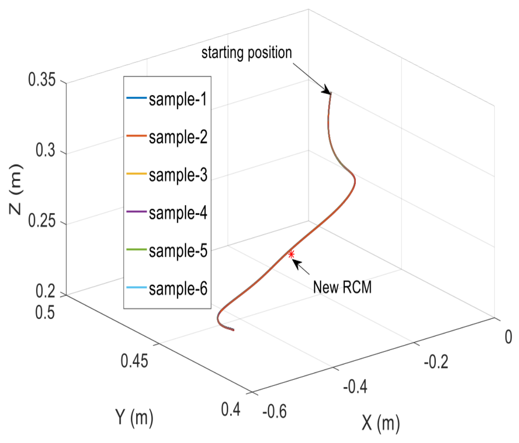


(a) The learning process of trajectory via original RCM point.

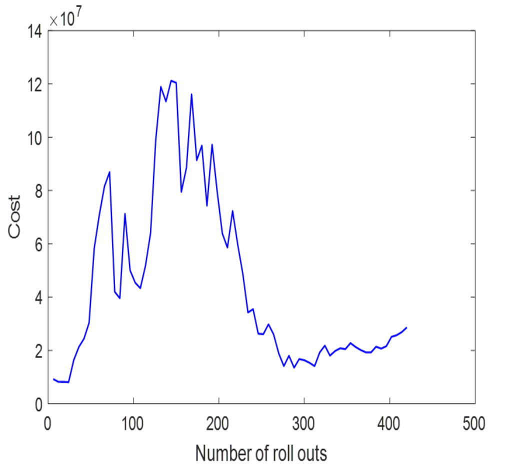


(b) The learning process of trajectory via new RCM point.

Figure 3.6: Reinforcement learning process for via-point task.



(a) The last learning results of all samples via new RCM point.



(b) Cost function by iteration of new RCM tasks.

Figure 3.7: Learning results with Reinforcement learning.

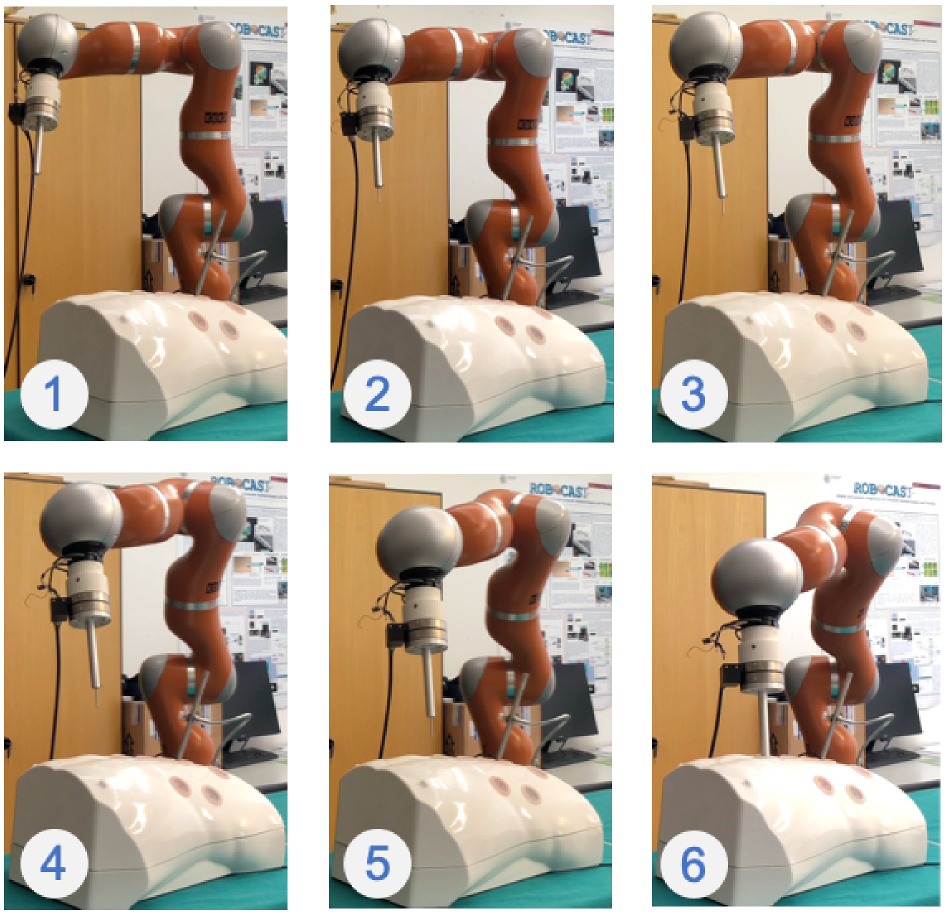


Figure 3.8: Demonstration of autonomous puncture using reinforcement learning. The numbers (1-6) indicate the puncture procedure.

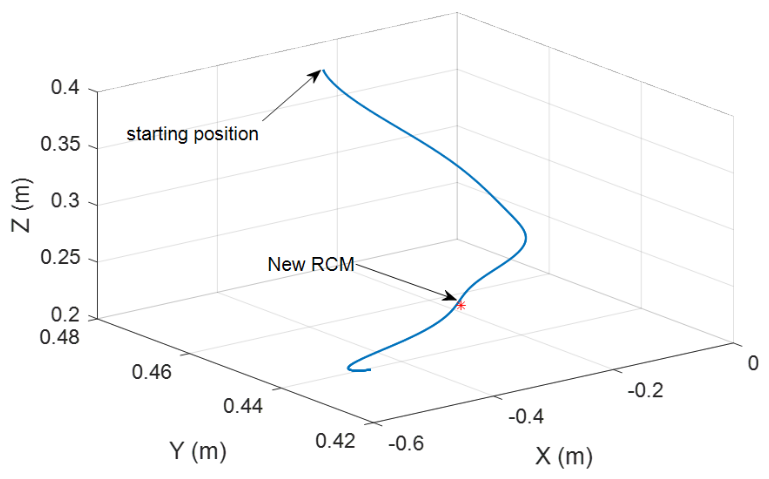


Figure 3.9: Demonstrated trajectory curve.

3.2 Fuzzy Adaptive Control and Imitation Learning for Real-time Obstacle Avoidance

In this section, we proposed an improved Dynamic Movement Primitives (DMPs) for real-time obstacle avoidance of motion planning, and also propose the fuzzy adaptive control for robot dynamics control with the given planning trajectories in Cartesian space [128]. DMPs framework is a powerful approach to imitate motor skills, which has outstanding characteristics, such as convergence to the goal position and good imitation performance. Considering complex motion scenes of manipulators, such as changing the goal position or adding obstacles, the original DMPs framework is not sufficient for the requirements. In this section, we propose a learning control-based hierarchical control strategy to adapt to new goal positions and avoid obstacles: the high-level learning scheme is targeted at imitating the motor skill and generating the optimization trajectory for obstacle avoidance; the lower-level control scheme focuses on the safety and stability of the robot's movement with unknown disturbances. Firstly, the enhanced DMPs framework is presented to imitate the trajectory from human demonstrations, where the novel DMPs can adapt to new goal position with the changing goal, and avoid single or multiple obstacles. Then, the fuzzy adaptive control method is employed to control redundant manipulators, where the fuzzy logic system (FLS) is incorporated to approximate an unknown nonlinear function term of the unknown disturbance. Finally, the effectiveness of the proposed learning-control strategy is demonstrated with simulation results. The results show that the developed hierarchical strategy has good performance for new goal adaptation and obstacle avoidance.

3.2.1 Introduction

Manipulator autonomous obstacle avoidance is a challenging topic prominent in the past decade. Generally, the main issue is how to plan a smooth trajectory of the end-effector which adapts as the task changes during the presence of many obstacles. In [129], the continuous genetic algorithm (CGA) is proposed for trajectory planning of redundant manipulators, where CGA can solve the singularity problem. In [130], the authors present an reinforcement learning-based double neural networks algorithm to manage the obstacle avoidance problem, where the reinforcement learning method is used to compute the optimization path. In [131], the particle swarm optimization (PSO) is employed for motion planning with multi-obstacle constraints, while PSO is used to obtain the optimal trajectory with the condition constraints.

However, the methods mentioned above only consider fixed goal positions, and cannot solve the problem of changing goal positions. We want to design a powerful method to solve more complex problems, such as adaption to new goal positions with multi-obstacle constraints. Imitation learning has a pivotal role in robotic technology and can learn new motor skills from human demonstrations and interactions with the environment. In robotic learning, a complex movement can be deemed as a combination of simple movements that can be easily encoded by learning methods. Dynamical Movement Primitives (DMPs) framework is a well-established method that can model nonlinear motions [77, 132], and provide powerful learning mechanisms, such as combining with machine learning methods, e.g. reinforcement learning.

In [133], dual-arm robots based on DMPs cooperate to cut a vegetable by human demonstrations. In [71], reinforcement learning fusion with DMPs is employed to learn the grasping actions with a number of iterations in the external disturbance environment. In this section, DMPs are applied to learning human movements for a mobile robot. The learning scheme is a novel human-robot interaction method that can imitate the motion from human demonstrations. Therefore, the mobile robot is able to reproduce the demonstrations by DMPs, which lead to the reduction of complexity for motion planning tasks. In [134], DMPs are utilized to learn the robot's joint trajectory fusion with sensor signals. However, the original DMPs framework has limitations that cannot solve the obstacle avoidance problem, and its adaptability is not sufficient for changing goal positions [135, 136]. In this section, we modify the original DMPs method and apply it to the manipulator movement learning with multi-obstacle constraints. Compared with the original DMPs framework, the enhanced DMPs are more powerful and can be adapted to changing goal positions with more natural trajectories.

In addition, for robotic control, various methods are applied to dynamic systems. In [97, 137], the global asymptotic PID is proposed for a robot system with input constraints. In [138], the authors explore the adaptive control system to solve the input saturation problem. In [138, 139], the closed-loop based adaptive control algorithm is applied to a exoskeleton robot. However, the robot system is highly nonlinear and the disturbance needs to be compensated. Inspired by [49, 140, 141], the fuzzy adaptive control strategy is presented to control the manipulator to compensate for the external disturbance. Overall, the learning-control system is proposed, where the high-level learning scheme is targeted at imitating the motor skill and generating the optimal trajectory for obstacle avoidance; while the lower-level control scheme focuses on the safety and stability of the robot's movement with unknown disturbances.

3.2.2 Adaptive Trajectories: Goal Adaptation and Obstacle Avoidance

Dynamical Movement Primitives

DMPs framework is a dynamic system that can be applied to imitation learning, even combined with machine learning methods for motor skills learning. The dynamic system consists of a linear spring-damper and a nonlinear item, which are defined as follows:

$$\ddot{X} = K_p\left(X_g - X\right) - K_d\dot{X} + \left(X_g - X_0\right) f\left(s, \rho\right) \tag{3.17}$$

$$f\left(s, \rho\right) = \frac{\sum_{j=1}^{N} \varphi_j\left(s\right)\rho_j}{\sum_{j=1}^{N} \varphi_j\left(s\right)} s \tag{3.18}$$

$$\dot{s} = -\lambda s \tag{3.19}$$

$$\varphi_j\left(s\right) = \exp\left(-\frac{1}{2d_j}\left(s - c_j\right)^2\right) \tag{3.20}$$

Equation (3.17) describes the transformation system, where $\left\{\ddot{X}, \dot{X}, X\right\}$ denote the robot's position, velocity and acceleration in Cartesian space; K_p and K_d are the corresponding stiffness matrix and damping matrix, respectively; X_0 and X_g are the initial and target values of the position; $f\left(s, \rho\right)$ is the nonlinear forcing item. Equation (3.19) is the canonical system acting as the decay factor, and $\lambda > 0$ is the constant parameter; d_j and c_j in equation (3.20) are the bandwidth and center of the Gaussian kernel function. Obviously, when $f\left(s, \rho\right) = 0$, the dynamic system becomes linear.

Enhanced DMPs for Goal Adaptation

The traditional framework of DMPs cannot guarantee the generation of a smooth trajectory when the start or target positions are altered. In addition, it cannot ensure the adaption of the robot to a new goal position. Hence, we introduce the novel framework of a nonlinear dynamic system, which is an extended form of the traditional DMPs.

Firstly, [142] the canonical system in equation (3.19) is modified as,

$$\dot{s} = \begin{cases} -\lambda_1, & s \geq 0 \\ 0, & s < 0 \end{cases} \tag{3.21}$$

$$w = a + 1 \tag{3.22}$$

where $\lambda_1 > 0$ is the decay rate of the canonical system. The nonlinear forcing item $f(s, \rho)$ depends only on s, which acts as a global clock for the entire system. Secondly, the transformation system in (3.17) is reformulated as,

$$\ddot{X} = K_p\left(X_g - X\right) - K_d\dot{X} - K_p\left(X_g - X_0\right)s + K_p f(s, \rho) \tag{3.23}$$

Apart from the modified canonical system (3.21) and transformation system (3.23), the function $f(s, \rho)$ is identical to (3.18).

The novel transformation system, (3.23), is a forward procedure. To model the motion from human demonstrations, we need to obtain the shape parameter ρ in a backward procedure using regression methods.

$$f_{demo} = \frac{1}{K_p}\ddot{X} - \frac{1}{K_p}\left[K_p\left(X_g - X\right) - K_d\dot{X} - K_p\left(X_g - X_0\right)s\right] \tag{3.24}$$

Finally, to obtain the shape parameter ρ from the dataset, the Locally Weighted Regression (LWR) method is used to learn ρ by regression. The objective function is defined as

$$\Phi_j = \sum_{t=1}^{N} \varphi_j(t) \left\| f_{demo} - \left(\rho_j s + b_j\right) \right\|^2 \tag{3.25}$$

The solution of regression is calculated as,

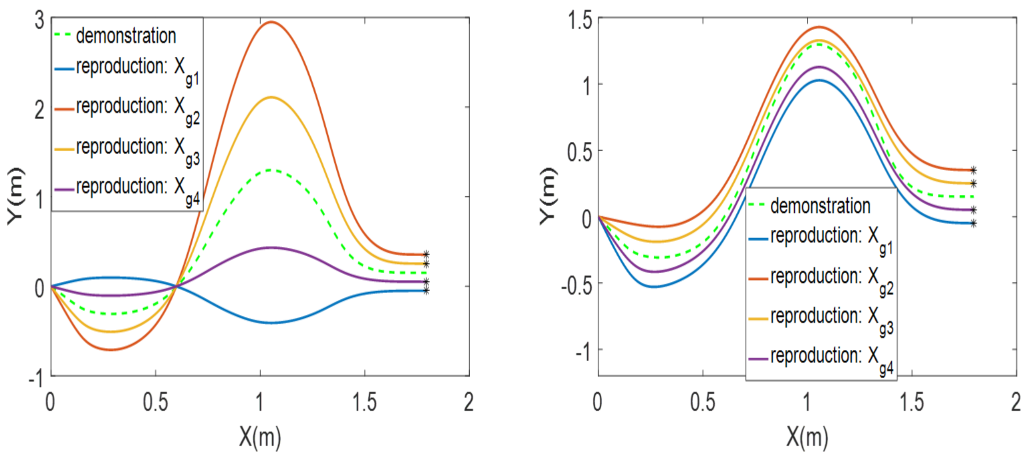
$$\begin{bmatrix} \rho_j \\ b_j \end{bmatrix} = \begin{bmatrix} R_{s^2} & R_s \\ R_s & R_\rho \end{bmatrix}^{-1} \begin{bmatrix} R_{sy} \\ R_y \end{bmatrix} \tag{3.26}$$

$$\begin{aligned} R_\rho &= \sum_{t=1}^{N} \varphi_j(t) \\ R_s &= \sum_{t=1}^{N} \varphi_j(t) s(t) \\ R_{s^2} &= \sum_{t=1}^{N} \varphi_j(t) s^2(t) \\ R_{sy} &= \sum_{t=1}^{N} \varphi_j(t) s(t) f_{demo} \end{aligned}$$

To test the proposed methods, we change the target position and compare the original DMPs with our enhanced DMPs.

The comparison results of the original DMPs and our enhanced DMPs for target position adaptation are shown in Fig. 3.10. In Fig. 3.10a, the original DMPs model the trajectory and converge to a new target position, but the shape of the motion is adjusted overshoot.

In Fig. 3.10b, it is suitable for natural movements with changing target positions. Obviously, the enhanced DMPs can improve movement adaptation.



(a) Learning with the original DMPs.
(b) Learning with the enhanced DMPs.

Figure 3.10: The original DMPs vs the enhanced DMPs for changing targets.

Obstacle Avoidance

The modified DMPs in (3.23) can adapt to the new target position but are still unable to deal with complex requirements, such as autonomous obstacle avoidance. Hence, in this part, we further extend the novel DMPs in (3.23) as a new formulation that can avoid a single obstacle and even many obstacles.

The new formulation of DMPs is reformulated as

$$\ddot{X} = K_p\left(X_g - X\right) - K_d\dot{X} - K_p\left(X_g - X_0\right)s + K_p f\left(s,\rho\right) + P\left(X,\dot{X}\right) \tag{3.27}$$

For the single-obstacle scene, the relationship between steering angle θ and velocity $\dot{\theta}$ can be defined as,

$$\dot{\theta} = \mu\theta\exp\left(-\alpha\left|\theta\right|\right) \tag{3.28}$$

where $\mu > 0$ and $\alpha > 0$ are constants; $\dot{\theta}$ is the steering angle velocity.

$$\ddot{X} = T_R\dot{X}\dot{\theta} \tag{3.29}$$

where T_R is the rotation matrix of axis $L = (M - X) \times \dot{X}$ with rotation angle of $\pi/2$. $M = [M_x, M_y, M_z]$ is the position of the obstacle, X is the position of robot's end-effector, while $\dot{X}$ and $\ddot{X}$ correspond to its velocity and acceleration, respectively.

$$P\left(X,\dot{X}\right) = \mu T_R\dot{X}\theta\exp\left(-\alpha\theta\right) \tag{3.30}$$

$$\theta = \cos^{-1}\left(\frac{(M-X)^T\dot{X}}{(|M-X|\left|\dot{X}\right|)}\right)$$

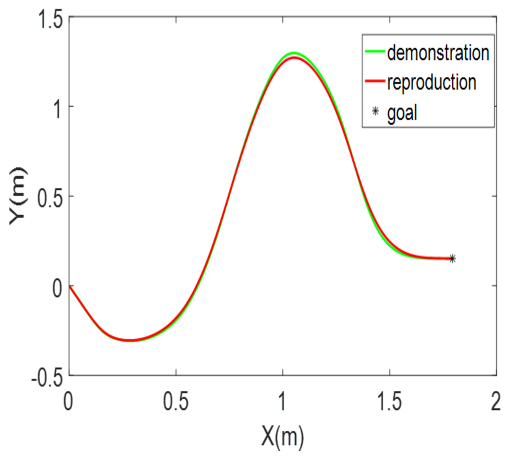
So, the modified DMPs equation in (3.27) can be rewritten as,

$$\ddot{X} = K_p\left(X_g - X\right) - K_d\dot{X} - K_p\left(X_g - X_0\right)s + K_p f\left(s,\rho\right) + \mu T_R\dot{X}\theta\exp\left(-\alpha\theta\right) \tag{3.31}$$

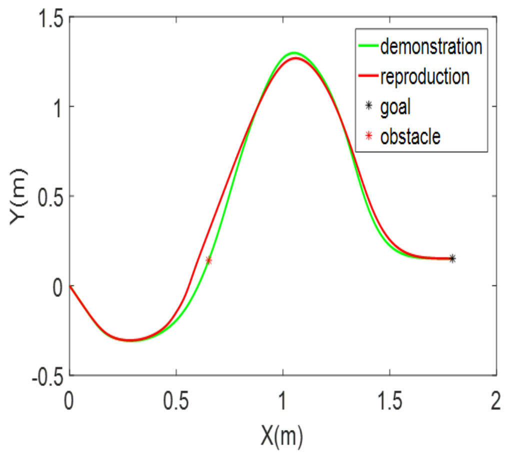
where the state variable $[X,\dot{X}] = [X_g, 0]$ is the stationary point, to which all the states converge from random initial states.

Proof 3 *To prove global convergence of the modified DMPs in (3.31), the Lypunov function is set as,*

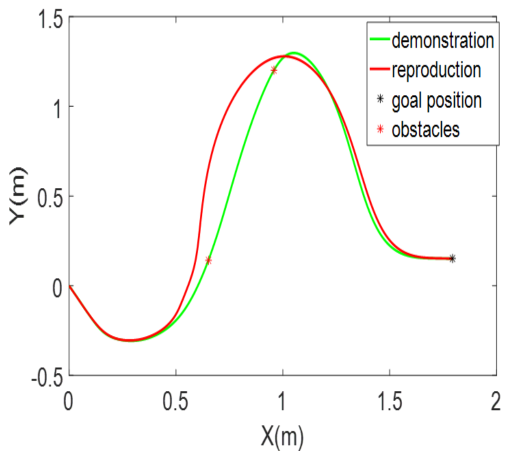
$$V\left(X,\dot{X}\right) = \frac{1}{2}\left(X_g - X\right)^T K_p\left(X_g - X\right) + \frac{1}{2}\dot{X}^T\dot{X} \tag{3.32}$$



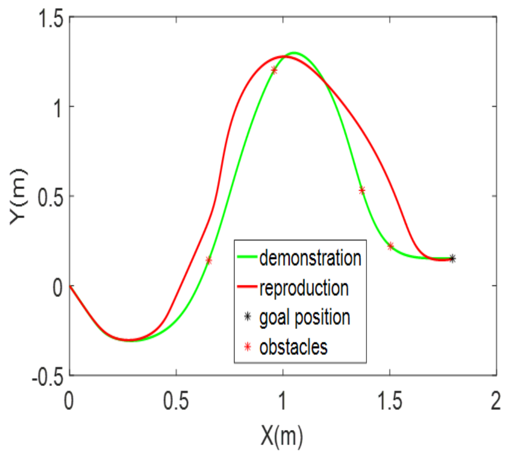
(a) Learning without obstacles.



(b) Learning with a single obstacle.

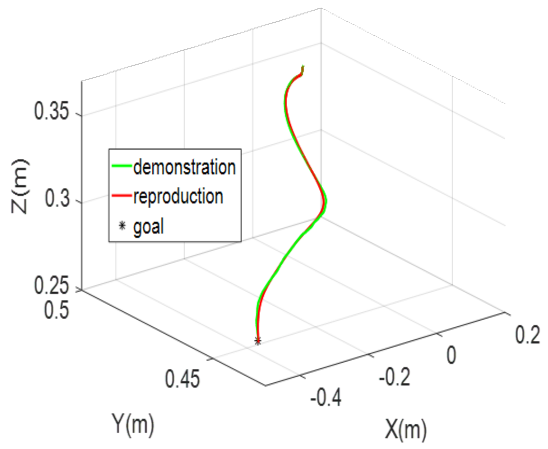


(c) Learning with many obstacles.

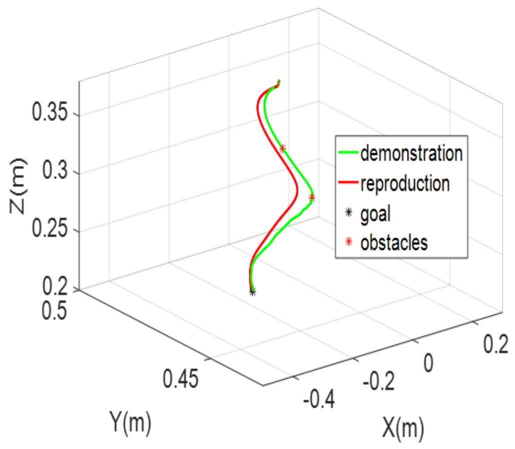


(d) Learning with more obstacles.

Figure 3.11: Imitation learning in 2D space for obstacle avoidance.

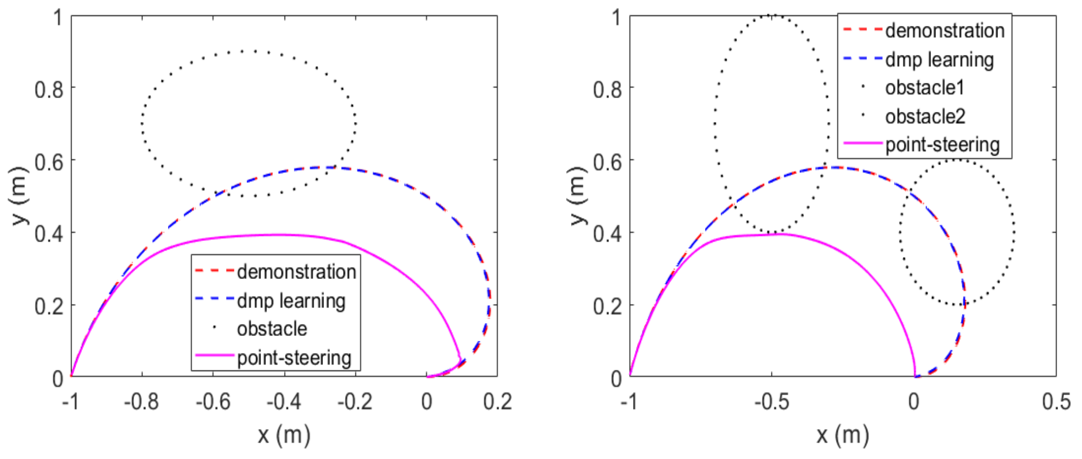


(a) Learning without obstacles.



(b) Learning with more obstacles.

Figure 3.12: Imitation learning in 3D space for obstacle avoidance.



(a) Learning for single obstacles. **(b)** Learning with two obstacles.

Figure 3.13: Imitation learning in 2D space for obstacle avoidance in complex scenario.

Then, the time derivative of $V(X,\dot{X})$ is obtained as

$$\begin{aligned}\dot{V} &= \nabla_X V^T \dot{X} + \nabla_{\dot{X}} V^T \ddot{X}\\ &= -(X_g - X)^T K_p \dot{X} + \dot{X}^T \ddot{X}\\ &= -\dot{X}^T K_p (X_g - X) + \dot{X}^T K_p (X_g - X) - \dot{X}^T K_d \dot{X}\\ &\quad - K_p \dot{X}^T (X_g - X) s + \dot{X}^T K_p f(s,\rho) + \mu \dot{X}^T T_R \dot{X} \theta \exp(-\alpha\theta)\\ &= -\dot{X}^T K_d \dot{X} - \underbrace{K_p \dot{X}^T (X_g - X) s}_{C_1} + \underbrace{\dot{X}^T K_p f(s,\rho)}_{C_2} + \underbrace{\mu \dot{X}^T T_R \dot{X} \theta \exp(-\alpha\theta)}_{C_3} \end{aligned} \tag{3.33}$$

Obviously, if $t \to \infty$, $X \to X_g$ and $s \to 0$, then $C_1 \to 0$; again when $f(s,\rho) \to 0$, then $C_2 \to 0$; since T_R is the rotation matrix with rotation angle of $\pi/2$, then $\dot{X}^T T_R \dot{X} = 0$, so $C_3 \to 0$. Therefore, we can conclude

$$\dot{V} \to -\dot{X}^T K_d \dot{X} \le 0$$

The proof is finished.

However, we hope that the function in (3.30) is more powerful i.e. has better obstacle avoidance and is adaptive to a more complex environment. If there exist more obstacles, the added item in (3.30) needs to be further modified. The details of the novel modified item can be reformulated as,

$$P(X,\dot{X}) = \mu \sum_{i=1}^{O} T_{R_i} \dot{X} \theta_i \exp(-\alpha\theta_i) \tag{3.34}$$

$$\theta_i = \cos^{-1} \frac{\left((M_i - X)^T \dot{X}\right)}{|M_i - X| \left|\dot{X}\right|} \tag{3.35}$$

where O denotes the number of obstacles.

To test the obstacle avoidance performance of the DMPs, we randomly put several obstacles on the trajectory after the demonstration. From Fig. (3.11a)–Fig. (3.13b), it can be seen that the modified DMPs can avoid the single obstacle and even multiple obstacles well.

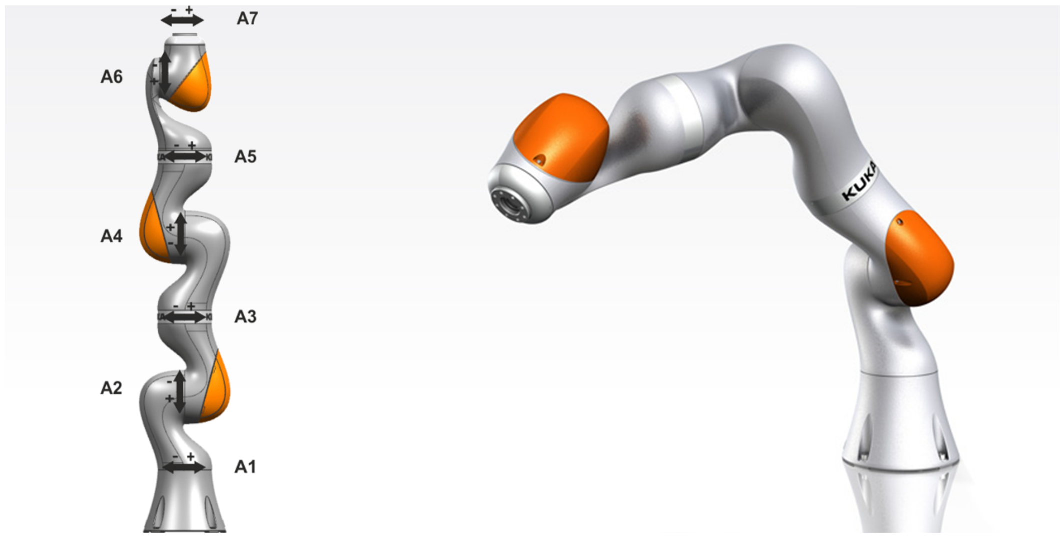


Figure 3.14: 7-DOFs kuka robot's model.

3.2.3 Fuzzy Adaptive Control for Manipulator

Dynamic System of Manipulator

Considering the dynamic model of n-DOFs (n=7) Kuka manipulator shown in Fig. 3.14,

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau_c + \tau_e \tag{3.36}$$

where $q \in R^{n\times n}$ and $\dot{q} \in R^{n\times n}$ are the joint position vector and velocity vector, respectively; $B(q) \in R^{n\times n}$ denotes the inertia matrix; $C(q,\dot{q}) \in R^{n\times n}$ is the Coriolis and Centrifugal effects; $G(q) \in R^n$ represents the gravity matrix; τ_c is the joint torque variable, and τ_e is the torque of external unknown disturbances.

The forward kinematics model of a redundant manipulator is formulated as,

$$\dot{X} = J(q)\dot{q} \tag{3.37}$$

where $J(q)$ is the Jacobian matrix of the redundant manipulator.

Property 1 *The positive-definite matrix $B^{-1}(q)$ exists, and bounded that satisfies the condition: $\left\|B^{-1}(q)\right\| \leq \varepsilon_B(\varepsilon_B > 0)$.*

Property 2 *The matrix $\dot{B}(q) - 2C(q,\dot{q})$ is skew-symmetric.*

Then, substituting (3.36) into the time derivative of (3.37) and multiplying $J^{-T}BJ^{-1}$,

$$J^{-T}BJ^{-1}\ddot{X} = J^{-T}(\tau + \tau_e - (C\dot{q} + G)) + J^{-T}BJ^{-1}\dot{J}\dot{q} \tag{3.38}$$

Hence, the nonlinear dynamic model of the manipulator in Cartesian space is expressed as,

$$B_x(X)\ddot{X} + H_x\left(X,\dot{X}\right) - \mathcal{F}_e = \mathcal{F} \tag{3.39}$$

where $X \in \mathbb{R}^m$, $\dot{X} \in \mathbb{R}^m$ and $\ddot{X} \in \mathbb{R}^m$ denote the Coordinate trajectory, velocity and acceleration in Cartesian space, respectively;

$$\begin{aligned}
&B_x(X) = J^{-T}BJ^{-1} \\
&H_x\left(X,\dot{X}\right) = J^{-T}\left[C(q,\dot{q})J^{-1}\dot{X} + G(q) - BJ^{-1}\dot{J}J^{-1}\dot{X}\right] \\
&\mathcal{F}_e = J^{-T}\tau_e
\end{aligned}$$

where matrix B_x represents the inertia matrix in task space; vector $H_x\left(X,\dot{X}\right)$ includes the gravitational, centrifugal and Coriolis torques in task space; $\mathcal{F}$ and $\mathcal{F}_e$ denote the the input signal and the external disturbance in Cartesian space, respectively.

Assumption 1 *The B_x^{-1} is assumed to be bounded with $\left\|B_x^{-1}\right\| \le \alpha_B$;*

For further convenience, the dynamics model of Cartesian space in (3.39) can be reformulated as a standard nonlinear affine system in the following form by taking $z_1 = X \in \mathbb{R}^m$ and $z_2 = \dot{X} \in \mathbb{R}^m$,

$$\dot{z}_1 = z_2 \tag{3.40}$$

$$\dot{z}_2 = {B_x}^{-1}\left(\mathcal{F} - H_x + \mathcal{F}_e\right) \tag{3.41}$$

The equation (3.41) can be rewritten as follows:

$$\dot{z}_2 = -{B_x}^{-1}H_x + {B_x}^{-1}(\mathcal{F} + \delta) \tag{3.42}$$

where $\delta = \mathcal{F}_e$; $\left\|B_x^{-1}\right\| \le \alpha_B$ and $\|\delta_i\| \le \xi_i$. To describe the robot system conventionally, the dynamic system in (3.36) should be reformulated in a standard nonlinear control affine form. We define,

$$g(z_1) = {B_x}^{-1} \tag{3.43}$$

$$f(z_1, z_2) = -{B_x}^{-1}H_x \tag{3.44}$$

$$u = \mathcal{F} \tag{3.45}$$

Therefore, the dynamic system in (3.36) can be rewritten as,

$$\dot{z}_1 = z_2 \tag{3.46}$$

$$\dot{z}_2 = f\left(z_1, z_2\right) + g\left(z_1\right)(u + \delta) \tag{3.47}$$

Fuzzy Approximation system

It is difficult to obtain the accurate dynamic model of the manipulator due to unknown disturbances. In this section, a fuzzy logic system is employed for unknown items. Firstly, the fuzzy rules are given as,

$$\Lambda_i : \text{if } \ Z_1 \text{ is } K_1^i \text{ and } \ldots \text{and } Z_l \ \text{ is } K_l^i, \text{ then } h \text{ is } \ h_i \tag{3.48}$$

i where h_i is the i-item fuzzy rule.

$$h(Z) = \frac{\sum\limits_{l=1}^{m} h^l\left(\prod_{j=1}^{n}\varphi\left(Z_j\right)\right)}{\sum\limits_{l=1}^{m}\left(\prod_{j=1}^{n}\varphi\left(Z_j\right)\right)} = \Theta^T S(Z) \tag{3.49}$$

where m is the number of fuzzy rules; $Z = [Z_1, Z_2, \ldots, Z_n] \in R^n$ is the input variable; $S(Z)$ is the known fuzzy basis function; Θ denotes the adaptable weight parameters.

Therefore, the approximate function $h(Z)$ can be defined as,

$$\tilde{h}(Z) = \tilde{\Theta}^T S(Z) \tag{3.50}$$

where $Z \in R^n$, $\tilde{\Theta} \in \Lambda_m$, $S(Z) \in R^m$.

The objective function of optimization parameters Θ^* is defined as,

$$\Theta^* = \arg\min\left\{\sup\left|h(Z) - \tilde{h}\left(Z|\tilde{\Theta}\right)\right|\right\} \tag{3.51}$$

$$M_\Theta = \left\{\tilde{\Theta}\left|\left\|\tilde{\Theta}\right\| \le M_\Theta\right.\right\} \tag{3.52}$$

where the parameter M_Θ corresponds to the bounds of $\tilde{\Theta}$ which are defined by the user.

According to the optimization result in (3.51), the unknown function $h(Z)$ can be rewritten as,

$$h(Z) = \Theta^{*T} S(Z) + \varepsilon \tag{3.53}$$

where ε presents the minimum approximation error. Actually, the minimum approximation error ε has an upper bound $\varepsilon^+ > 0$ according to the theory analysis of the fuzzy logical system,

$$|\varepsilon| \le \varepsilon^+ \tag{3.54}$$

For the robot system,

$$\Theta^* = \lambda\mathcal{E}\varpi S(Z) \tag{3.55}$$

where $\lambda \in R^{n\times n}$ is the coefficient diagonal matrix which is used to scale the updating rate; $\mathcal{E} = [X_r - X(q)]$ is the position error in Cartesian space; $\varpi \in R^2$ is the last column of a symmetric positive definite matrix according to Lyapunov theory.

The force approximation of the unknown disturbance using the fuzzy system can be defined as,

$$\mathcal{F}_h = -J^T\Theta S(Z) \tag{3.56}$$

where τ_h is used to compensate for the effects of disturbances. Finally, the input torque can be reformulated as,

$$\mathcal{F}_d = \mathcal{F}_c + \mathcal{F}_h \tag{3.57}$$

Sliding Mode Control

In this part, sliding mode control is applied to the robot system and has better performance for modeling errors and anti-disturbance. The disturbance $\delta \in R^n$ is considered in the state space form, where δ is bounded such that $|\delta_i| \le \xi_i(\xi_i > 0)$.

Thus the equations in (3.46)–(3.47) can be reformulated as,

$$\dot{z}_1 = z_2 \tag{3.58}$$

$$\dot{z}_2 = f(z_1, z_2) + g(z_1)(u + \delta) \tag{3.59}$$

The tracking error of the end-effector is defined as: $e_c = z_1 - z_{1d}$. The sliding mode surface is defined as,

$$S = \dot{e}_c + \alpha e_c \tag{3.60}$$

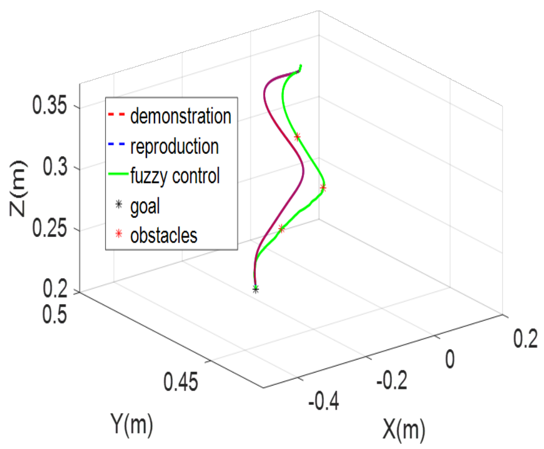
where $\alpha = diag(\alpha_1, \alpha_2, \ldots, \alpha_n)$, and $\alpha_i > 0$.

According to sliding mode theory, we aim to design an input signal u to ensure that $S = 0$. Firstly, the Lyapunov function is set as,

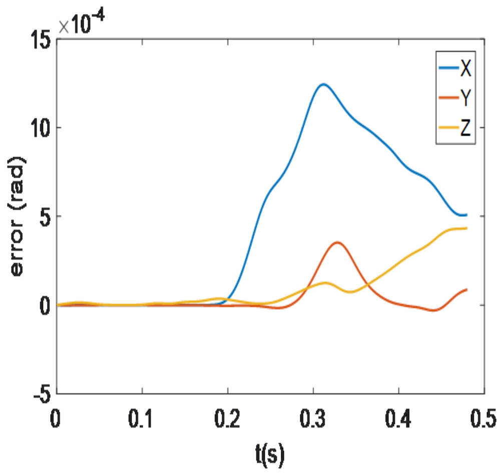
$$L = \frac{1}{2}S^T S \tag{3.61}$$

So, the time derivative of L can be concluded as,

$$\begin{aligned}\dot{L} &= S^T\dot{S}\\ &= S^T(\ddot{e}_c + \alpha\dot{e}_c)\\ &= S^T(f(z_1, z_2) + g(z_1)u + g(z_1)\delta - \ddot{z}_{1d} + \alpha(z_2 - \dot{z}_{1d}))\end{aligned} \tag{3.62}$$



(a) Tracking results in Cartesian space.



(b) Tracking errors.

Figure 3.15: Tracking results in Cartesian space using fuzzy adaptive control.

To ensure the stability of the control system, according to the equation of $\dot{L}$, the actual control law is defined as,

$$u = g(z_1)^{-1}\left(-f\left(z_1, z_2\right) + \ddot{z}_{1d} - \alpha\left(z_2 - \dot{z}_{1d}\right) - sgn\left(S\right)\right) \tag{3.63}$$

where $= diag\left\{k_1, k_2, \ldots, k_n\right\}$, and $k_i > 0$ are constants. Substituting (3.63) into (3.62), the time derivative of L is obtained as,

$$\begin{aligned}\dot{L} &= S^T\left(g\left(z_1\right)\delta - sgn\left(S\right)\right)\\ &= -\sum_{i=1}^{n} k_i\left|S_i\right| + S^T B_x^{-1}\delta\\ &\leq -\sum_{i=1}^{n}\left|S_i\right|\left(k_i - \alpha_B \xi_i\right)\end{aligned} \tag{3.64}$$

If the gain parameters satisfy that $k_i > \alpha_B \xi_i$, then $\dot{L} < 0$ which can conclude that the system is asymptotically stable. Finally, the control law is presented as,

$$u = H_x\left(X, \dot{X}\right) + B_x\left(\ddot{X}_d - \alpha\left(z_2 - \dot{X}_d\right) - sgn\left(S\right)\right) \tag{3.65}$$

3.2.4 Simulation

In our simulation scenarios, some obstacles are randomly placed in the environment. It is a challenging scene that lets the robot achieve new skills of adaption to the new target and obstacle avoidance. Firstly, the motor skills are encoded using the enhanced DMPs by human demonstrations, where a human holds the robot to teach the skills by demonstrations. After the collection of the dataset, the regression procedure is conducted using LWR, thus the shape parameter ρ is obtained, and then the imitation results are achieved via the forward DMPs. Finally, the sliding mode control-based fuzzy adaptive control is applied to the 7-DOFs Kuka model, which shows good performance in terms of trajectory tracking.

The tracking results using fuzzy adaptive control are shown in Fig. 3.15. Obviously, we can see that the real trajectory can successfully avoid obstacles. From Figs. 3.15a–3.15b, the actual trajectory almost coincides with the learned trajectory, and the tracking errors of the X-Y-Z coordinates gradually converge to zero. Therefore, the proposed learning-control strategy can not only learn the skills, adapt new goal positions and avoid obstacle, but also tracking the desired trajectory in Cartesian space.

3.2.5 Summary

In this section, we proposed a learning control-based hierarchical control strategy to adapt to new goal position and avoid obstacles: the high-level learning scheme is targeted at imitating the motor skill and generating the optimization trajectory for obstacle avoidance; the lower-level control scheme focuses on the safety and stability of robot's movement with unknown disturbances.

For the learning system, the modified DMPs have better performance regarding movement adaptation than the original DMPs. In addition, we further modify the DMPs to avoid a single obstacle and even multiple obstacles. To ensure the safety and stability of the robot, a sliding mode control-based fuzzy adaptive controller is proposed. From the learning-control results, we can conclude that the proposed method can stably and safely avoid multiple obstacles. In further work, we will test the proposed method with physical experiments, even more complex scenarios.

3.3 Related Publication

1 Hang Su, **Yingbai Hu***, Zhijun Li, Alois Knoll, Giancarlo Ferrigno, Elena De Momi. "Reinforcement Learning Based Manipulation Skill Transferring for Robot-assisted Minimally Invasive Surgery". In: *2020 International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2203–2208.

2 **Yingbai Hu**, Guang Chen, Longbin Zhang, Hang Su, Mengyao Li, Yunus Schmirander, Hu Cao, Alois Knoll. "Fuzzy Adaptive Control-based Real-time Obstacle Avoidance under Uncertain Perturbations". In: *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. 2020, pp. 50-–55

Chapter 4

Robot Policy Improvement for Manipulation

In this chapter, we will discuss how to improve robustness and adaptability of imitation learning using policy improvement approach. Robot learning through kinesthetic teaching is a promising way of cloning human behaviors, but it has its limits in the performance of complex tasks with small amounts of data, owing to compounding errors. In order to improve the robustness and adaptability of imitation learning, in this chapter, a hierarchical learning strategy is proposed: low level learning comprises only those behaviors cloned with supervised learning, while high level learning constitutes policy improvement. Firstly, the Gaussian mixture model-based (GMM) dynamical system is formulated to encode a motion from demonstration. We then derive the sufficient conditions of the GMM parameters that guarantee the global stability of the dynamical system from any initial state, using the Lyapunov stability theorem. Generally, imitation learning should reason about the motion well into the future for a wide range of tasks; it is significant importance for improving the adaptability of the learning method by policy improvement. Finally, a method based on exponential natural evolution strategies is proposed to optimize the parameters of dynamical system associated with the stiffness of variable impedance control, in which the exploration noise is subject to stability conditions of the dynamical system in the exploration space, thus guaranteeing the global stability. Empirical evaluations are conducted on manipulators for different scenarios, including motion planning with obstacle avoidance, and stiffness learning.

4.1 Introduction

Ever since the onset of pioneering research into robot learning methods of learning by demonstration have attracted much attention. Robot learning can facilitate applications in industry, manufacturing area, and healthcare, etc., because it directly clones motor skills by extracting task-relevant information that can be transferred to the robot [143] [144]. Generally, traditional imitation methods use supervised learning to obtain the regression parameters by modeling dynamic motion primitives (DMP) [21] and Gaussian mixture models (GMM) [27]. However, a major drawback of these methods is that they are not so adaptable and highly dependent on large amounts of data. Therefore, they tend to be restricted to real-world robotic applications. In a real-world scenario, it is of significant importance to design a more efficient learning policy based on the finite expert data.

Imitation learning has two central properties: policy–robustness and adaptability. Among the many research papers contributing to this issue, some focus on parameter learning to improve robustness and adaptability. In [145], Billard *et al.* present a Gaussian model-based stable estimator called SEDS for learning the parameters of the dynamical system, which can ensure global stability at the goal point. In [146], the authors extend the dynamical system to cover various regression models and proposed a learning strategy for optimizing the valid Lyapunov function [147], which displays strong robustness in terms of disturbance from a random initial state. In [148, 149], employ various modification tricks to improve the metrics of a dynamical system subjected to global stability, for example, improving the learning speed using an extreme learning machine or improving accuracy means of manifold immersion and submersion. Although the above works have made some progress in improving the robustness of imitation learning, they suffer from a limited adaptability. Specifically, we want the robot might be perform specific special tasks that are not covered by in expert data. For example, a robot is expected to perform obstacle avoidance tasks taking the shortest path, in which obstacles are configured along expert trajectories.

One possible solution combines policy-based high level learning strategies with imitation learning [150]. Several learning strategies have been investigated for improving adaptability for different scenarios. In [122, 151], the path integral policy improvement (**PI**^{2}) algorithm was tailored to learn the parameters of robot trajectories, in which reinforcement learning could adapt to learning tasks, such as via-points or obstacle avoidance tasks. Indeed, these policy improvement methods were explored with the aim of optimizing the parameters with the feedback of a cost function set by users, in which the update rule was in the form of a probability-weighted average [152]. In addition to the aforementioned work, reinforcement learning could also be used to acquire the parameters of variable impedance control [153] [154] [155], with the robot's stiffness being changed according to the task requirements. In [156], reinforcement learning was proposed for learning the impedance parameters, in which the stiffness of the impedance controller was modeled as a dynamical equation and updated in accordance with rewards from the cost function. Similarly, in [132], a covariance matrix adaptation evolution strategy (CMA-ES) is proposed for learning the variable impedance control of robotic grasping, which provides a theoretical rule for assigning the highest weight to the best population for parameter updating.

Inspired by these works, we propose the exponential natural evolution strategies (NES) algorithm to learn the stiffness of a stable nonlinear dynamical system. This combines the two properties of robustness and adaptability from kinesthetic teaching. As reported in [157], the NES provides a principled way of formulating the optimization problem based on the natural gradient. The proposed exponential NES is a more efficient learning algorithm than the previously mentioned CMA-ES method, because it obtains all parameter updating for covariance matrix adaptation from a single principle. Moreover, unlike the original NES, the parameter updating depends on the natural coordinate, which can reduce the computation of the inverse Fisher information matrix in NES [158].

In this chapter, we incorporate algorithms of exponential NES with a stable dynamical system into the imitation learning framework. First, the motions of the robot are encoded using the GMM from kinesthetic teaching, which is a normal behavioral cloning process. Then, the GMM-based stable dynamical system is derived from the Lyapunov stability theorem such that the optimized parameters of the dynamical system can be obtained from the stable condition. This process can improve the robustness of dynamical system. Finally, the exponential NES are explored for learning the parameters of the policy, which can further improve the stability and adaptability of the dynamical system for different tasks.

The rest of this chapter is organized as follows. Section 4.2 presents the work related to imitation learning and policy improvement-based learning. Section 4.3 formulates the nonlinear dynamical system-based learning problem and the parameterized policy from kinesthetic teaching. Section 4.4 introduces the exponential natural evolution strategies algorithm for parameter learning. In Section 4.5, we present the policy for learning the stiffness of variable impedance control. The simulation and experiment are conducted and discussed in Section VI. The conclusion is presented in Section 4.7.

4.2 Related Work and Motivation

In this section, we will briefly discuss the motivation of the study and the related work in the areas of imitation learning and policy improvement in learning.

4.2.1 Imitation Learning from Demonstrations

Learning from demonstrations is a popular approach in robot motion imitation. Several classical supervised learning methods have been widely applied to behavioral cloning, such as Gaussian mixture regression (GMR) [159], Gaussian process regression (GPR) [28], and hidden Markov models (HMM) [160]. However, these statistical approaches found locally optimal parameters by maximizing the likelihood such that failed to ensure global stability because they were not the theoretical solution to ensure the stability of the dynamical system. Dynamical system-based learning methods were therefore presented to ensure global stability, such as in DMP [23] and SEDS [145]. In particular, the time-invariant dynamical system in [145] and the constraints of stability condition according to Lyapunov stability are taken into consideration in the context of learning.

4.2.2 Policy Improvement with Learning

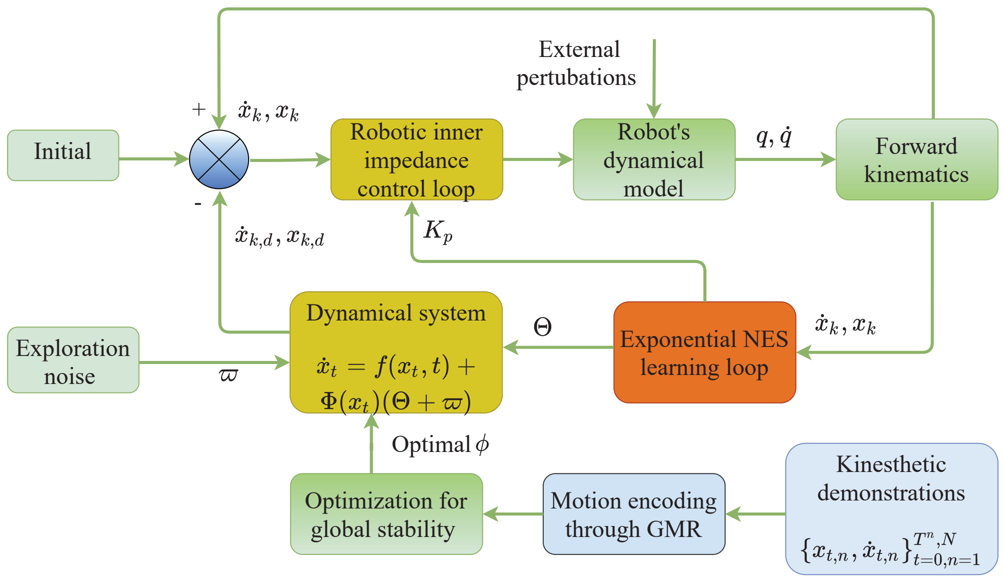
On the other hand, as regards adaptability, we focus on parameter learning to specify the task using policy improvement methods. The evolution strategies algorithm was one candidate solution for parameter learning, for example, CMA-ES and NES. In contrast to $\mathbf{PI}^2$ [71], the CMA-ES algorithm can modify the exploration noise during learning by updating its covariance matrix [132]. However, CMA-ES is not able to measure the proximity between the current policy and the updated policy on the basis of distribution in the learning process. We therefore consider exponential natural evolution strategies for this purpose. Generally, the NES learns the parameters of policy by following the natural gradient towards higher expected fitness [161]. Indeed, the traditional NES method involved computing the inverse Fisher information matrix, which can affect the efficiency of the learning process. The proposed exponential NES is a more efficient learning algorithm, because all the parameters updating for covariance matrix adaptation are obtained from a single principle [162].

4.3 Learning Problem

In this section, we will introduce the first learning problem, using GMR to model skills by kinesthetic teaching in Section 4.3.1. To adapt to different tasks in imitation learning, we formulate the second problem for learning the parameters the dynamical system in Section 4.3.2. The control flow framework is shown in Fig. 4.1.

4.3.1 Nonlinear Dynamical System from Demonstration

To imitate human skills, a date-driven kinesthetic teaching is explored for learning motor skills, needed by the robot to perform a repetitive task multiple times. The robot's motion can be encoded by learning methods, such as linear regression, support vector regression,



External pertubations
Initial
+
$\dot{x}_k, x_k$
Robotic inner impedance control loop
Robot's dynamical model
$q, \dot{q}$
Forward kinematics
-
$\dot{x}_{k,d}, x_{k,d}$
K_p
Dynamical system
$\dot{x}_t = f(x_t, t) + \Phi(x_t)(\Theta + \varpi)$
Θ
Exponential NES learning loop
$\dot{x}_k, x_k$
Exploration noise
ϖ
Optimal ϕ
Optimization for global stability
Motion encoding through GMR
Kinesthetic demonstrations
$\{x_{t,n}, \dot{x}_{t,n}\}_{t=0,n=1}^{T^n,N}$

Figure 4.1: The control flow of the learning-control framework using the proposed methods to derive a stable and robust control policy for robot system. Θ denotes the policy parameter, K_p is the stiffness of the impedance controller, $\dot{q}, q$ are the joint angle and velocity, and $\dot{x}, x$ are the state variable dynamical system.

and Gaussian mixture regression. Here, we formulate the robot's motions as an autonomous dynamical system with the state variable x and the output variable $\dot{x}$ as:

$$\dot{x} = \hat{f}(x) \tag{4.1}$$

where x is the end-effector's trajectory in Cartesian space, and $\dot{x}$ is the velocity; $\hat{f}(x)$ is the nonlinear continuous and continuously differentiable function.

The data points from demonstrations are defined as: $\left(x_{t,n}, \dot{x}_{t,n}\right)(t = 0, \ldots, T)$, where T is the time of the goal point and $N(n = 1, \cdots, N)$ is the number of samples. Each datapoint includes a position value $x_{t,n}$ and a velocity value $x_{t,n}$. To encode the dataset of position distribution $p(x_{t,n}, \dot{x}_{t,n})$, the following GMM model is defined as:

$$p\left(x_{t,n}, \dot{x}_{t,n}; \phi\right) = \sum_{k=1}^{K} p(k) p\left(x_{t,n}, \dot{x}_{t,n} | k\right) \tag{4.2}$$

where K is the number of the Gaussian model; $p(k)$ denotes the prior probability, and $p\left(x_{t,n}, \dot{x}_{t,n} | k\right)$ is the conditional probability density function; $\phi_k = \{\lambda_k, \mu_k, \sum_k\}$, where λ_k, μ_k, $\sum_k$ are prior probability, mean variable, and covariance variable, respectively. The parameters are defined as $\phi = \{\phi_1, \cdots, \phi_K\}$, and the details are expressed as:

$$\begin{aligned} &p(k) = \lambda_k \\ &p\left(x_{t,n}, \dot{x}_{t,n} | k\right) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \cdot \exp\left(-\tfrac{1}{2}([x_{t,n}, \dot{x}_{t,n}] - \mu_k)^{\mathrm{T}} \Sigma_k^{-1}([x_{t,n}, \dot{x}_{t,n}] - \Sigma_k)\right) \end{aligned}$$

For multiple demonstrations from kinesthetic teaching, the GMM encodes the set of trajectories of the robot in Cartesian space. The k component of the Gaussian mixture model is defined as:

$$\mu_k = \left\{ \begin{array}{c} \mu_k^x \\ \mu_k^{\dot{x}} \end{array} \right\} \quad , \quad \Sigma_k = \left(\begin{array}{cc} \Sigma_k^x & \Sigma_k^{x\dot{x}} \\ \Sigma_k^{\dot{x}x} & \Sigma_k^{\dot{x}} \end{array} \right) \tag{4.3}$$

where μ_k and Σ_k are the mean and covariance matrices of the k component GMM. Therefore, the posterior mean estimate is deduced by GMR as:

$$\dot{x} = \sum_{k=1}^{K} \frac{p(k)p(x|k)}{\sum\limits_{i=1}^{K} p(i)p(x|i)} \left(\mu_k^{\dot{x}} + \Sigma_k^{\dot{x}x}\left(\Sigma_k^{x}\right)^{-1}\left(x - \mu_k^{x}\right)\right) \tag{4.4}$$

To construct the form of the state equation as (4.1), the function (4.4) can be further written as:

$$\dot{x} = \hat{f}(x) = \sum_{k=1}^{K} \omega_k(x)(\Lambda_k x + d_k) \tag{4.5}$$

where:

$$\omega_k = \frac{p(k)p(x|k)}{\sum\limits_{i=1}^{K} p(i)p(x|i)} \tag{4.6}$$

$$\Lambda_k = \Sigma_k^{\dot{x}x}\left(\Sigma_k^{x}\right)^{-1} \tag{4.7}$$

$$d_k = \mu_k^{\dot{x}} - \Lambda_k \mu_k^{x} \tag{4.8}$$

It is clear that, Eq. (4.5) is a nonlinear dynamical system with nonlinear weighting terms ω_k, which can represent a wide variety of motions.

4.3.2 Parameterized Policy Learning for the Dynamical System

The dynamical system-based control law method is highly robust to motor skills learning, because it is a time-variant dynamical system and globally converges to the goal points [145]. Despite its robustness and adaptability, it requires appropriate placement of the Gaussian in the state space and is not able to perform the via-point tasks. To improve the policy parameters from the demonstrations, the evolution strategies learning algorithm is applied to the dynamical system in (4.5). In this section, a parameterized control policy for the dynamical system in (4.5) is defined as:

$$\dot{x}_t = f(x_t, t) + \Phi(x_t) \cdot (\Theta + \varpi_t) \tag{4.9}$$

where Θ is the learning parameter; $\Phi(x_t)$ denotes the control matrix; $\varpi_t \sim \mathcal{N}(0, \Sigma)$ denotes the Gaussian exploration noise.

Considering the special case of a 3-dimensional dynamical system to model the policy parameter, the dynamical system of Gaussian mixture regression in (4.5) is reformulated as:

$$\Phi(x_t) = [\Phi_1(x_t), \Phi_2(x_t), \ldots,\ \Phi_K(x_t)] \tag{4.10}$$

$$\Phi_k(x_t) = \omega_k \begin{bmatrix} x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1^t & x_2^t & x_3^t & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

$$\Theta = [\Theta_1, \Theta_2, \ldots, \Theta_K]^T \tag{4.12}$$

$$\Theta_k = \left[\Lambda_k^{1,1}, \Lambda_k^{1,2}, \Lambda_k^{1,3}, \Lambda_k^{2,1}, \Lambda_k^{2,2}, \Lambda_k^{2,3}, \Lambda_k^{3,1}, \Lambda_k^{3,2}, \Lambda_k^{3,3}, d_k^1, d_k^2, d_k^3\right] \tag{4.13}$$

From the definition in (4.10)–(4.13), the learning parameter Θ includes the matrix Λ_k and d_k of the Gaussian mixture regression, which is given in the dynamical system in (4.5). It should be noted that the parameters μ_x^k and Σ_k^x are the components of weighted ω_k, which shows the nonlinear mapping relationship. To avoid nonlinear factors in the learning policy, the policy parameters only have $\mu_k^{\dot{x}}$ and $\Sigma_k^{\dot{x},x}$. Therefore, the learning goal is to find the optimization solution of the mean of the velocity $\mu_k^{\dot{x}}$ and covariance between position and velocity $\Sigma_k^{\dot{x},x}$ in the Gaussian mixture regression. The mean and covariance of position are set as fixed parameters. The control matrix $\Phi(x_t)$ contains the nonlinear features of the policy. The features consist of the Gaussian basis functions multiplied by the input variables. The Gaussian basis functions are positioned in the input space during learning.

Generally, the policy parameters Θ are updated such that the cost function of motion trajectories $\left[x_{t_j}, x_{t_{j+1}} \dots, x_{t_N}\right]_\Theta$ from the start time t_i can be minimized:

$$S(\Theta) = \phi\left(x_{t_N}\right) + \int_{t_i}^{t_N} \left(r_{t_i} + \frac{1}{2}\rho_t^T(\Theta) R \rho_t(\Theta)\right) \tag{4.14}$$

where $\phi\left(x_{t_N}\right)$ is the terminal cost; r_{t_i} is the immediate cost; $\rho_t(\Theta) = \Phi(x_t)(\Theta + \varpi_t)$ denotes the control cost and R is the positive constant weight matrix.

In the learning process, the evolution strategies learning method will be explored to minimize the control cost $\frac{1}{2}\rho_t^T(\Theta)R\rho_t(\Theta))$. Once optimum control has been obtained, the update item $\delta\Theta_{t_i}$ can be computed at each time step. To obtain a single update vector $\delta\Theta$, a time averaging method can then be used. The details of the update rule for the learning process will be presented in Section. 4.4.

4.4 Methodology for Policy Improvement

In this section, the natural evolution strategies algorithm is introduced for learning the parameters Θ in (4.12). The natural gradient of expected fitness is firstly deduced to update the policy parameters, and the updating rule of exponential parameterization is then presented.

4.4.1 Natural Evolution Strategies

Natural evolution strategies are based on search gradients for updating the policy parameters. The sampled gradient of the expected fitness function is treated as a search gradient. In this chapter, we aim to minimize the fitness/cost function, which is defined in (4.14). The search distribution is mainly considered as the multinormal Gaussian distribution with a full covariance matrix.

First, we define the mean and covariance of the search distribution as μ_Θ and Σ_Θ and the learning variable $\Theta = \{\mu_\Theta, \Sigma_\Theta\}$. To simplify the distribution expression, we define $CC^T = \Sigma_\Theta$, and then $\xi = \mu_\Theta + Cs$, where variable s is the standard normal distribution $s \sim \mathcal{N}(0, I)$. The sample ξ is then: $x \sim \mathcal{N}(\mu_\Theta, \Sigma_\Theta)$. The search distribution of expected fitness $\bar{S} = -S$ is expressed as:

$$\mathbb{J}(\Theta) = \int \bar{S}(\xi) p(\xi|\Theta) d\xi \tag{4.15}$$

$$p(\xi|\Theta) = \frac{1}{(2\pi)^{m/2}\det(C)} \exp\left(-\frac{1}{2}\left\|C^{-1}(\xi - \mu_\Theta)\right\|^2\right)$$

The gradient of $\mathbb{J}(\Theta)$ is obtained according to the log likelihood trick:

$$\begin{aligned}\nabla_\Theta \mathbb{J}(\Theta) &= \nabla_\Theta \int \bar{S}(\xi)\, p(\xi|\Theta)\, d\xi \\ &= \int \bar{S}(\xi)\, \nabla_\Theta p(\xi|\Theta)\, d\xi \\ &= \int \bar{S}(\xi)\, \nabla_\Theta p(\xi|\Theta)\, \frac{p(\xi|\Theta)}{p(\xi|\Theta)} d\xi \\ &= \int \bar{S}(\xi)\, \nabla_\Theta \log p(\xi|\Theta)\, p(\xi|\Theta)\, d\xi \\ &= \mathbb{E}\left[\bar{S}(\xi)\, \nabla_\Theta \log p(\xi|\Theta)\right] \qquad (4.16)\end{aligned}$$

According to the Monte Carlo estimation [163], the function in (4.16) can be approximated as:

$$\nabla_\Theta \mathbb{J}(\Theta) \approx \frac{1}{l} \sum_{i=1}^{l} \bar{S}(\xi_i)\, \nabla_\Theta \log p(\xi_i|\Theta) \qquad (4.17)$$

where the parameter l represents the population size. The gradient $\nabla_\Theta \mathbb{J}(\Theta)$ offers a search direction in parameter space. Finally, the update rule is written as:

$$\Theta^{new} = \Theta + \eta \nabla_\Theta \mathbb{J}(\Theta) \qquad (4.18)$$

However, the traditional stochastic search gradient method in (4.18) is difficult to precisely determine quadratic optimum. The natural gradient-based method is a good solution and it helps mitigate the slow convergence of the plain gradient in optimization landscapes with ridges and plateaus. Actually, natural gradient is based on Riemannian geometry, and it learns the information from the manifold of probability distributions. The traditional gradient $\nabla_\Theta \mathbb{J}$ directly follows the steepest descent in the parameter space Θ in the distribution. For the maximum process of $\mathbb{J}$, it will generate a new distribution associated with updating the parameters from the hypersphere of radius c and center Θ, and thereby computing the Euclidean distance of two distributions. However, it creates a new problem that the updating relies on the particular parameterization of the distribution, where the gradients and updates follow the change in parameterization.

The natural gradient algorithm computes the natural distance $D(\Theta||\Theta + \delta\Theta)$ between $P(\xi|\Theta)$ and $P(\xi|\Theta + \delta\Theta)$ using the Kullback-Leibler (KL) divergence. The natural gradient of $\mathbb{J}$ can thus be reformulated as an optimization problem with a KL divergence constraint:

$$\max \;\; \mathbb{J}(\Theta + \delta\Theta) \approx \mathbb{J}(\Theta) + (\delta\Theta)^T \nabla_\Theta \mathbb{J} \qquad (4.19)$$

$$s.t. \;\; KL(\Theta + \delta\Theta||\Theta) = c \qquad (4.20)$$

where (4.19) is the Taylor expansion and (4.20) is the constraint of Kullback-Leibler divergence approximated [164]; c is the a small increment size. Since the KL divergence can be approximated with second order Taylor expands using Fisher Information Matrix, the function (4.20) is reformulated as:

$$\begin{aligned}&KL(\Theta||\Theta + \delta\Theta) \\ &\approx KL(\Theta||\Theta) + (\nabla_{\Theta+\delta\Theta} KL(\Theta||\Theta + \delta\Theta))^T \delta\Theta + \frac{1}{2}(\delta\Theta)^T F \delta\Theta \\ &= KL(\Theta||\Theta) - \mathbb{E}[\nabla_\Theta \log p(\xi|\Theta)]^T \delta\Theta + \frac{1}{2}(\delta\Theta)^T F \delta\Theta \\ &\approx \frac{1}{2}(\delta\Theta)^T F \delta\Theta \qquad (4.21)\end{aligned}$$

with

$$\begin{aligned} F &= \int p(\xi|\Theta)\nabla_\Theta \log p(\xi|\Theta)(\nabla_\Theta \log p(\xi|\Theta))^T d\xi \\ &= \mathbb{E}\left[\nabla_\Theta \log p(\xi|\Theta)(\nabla_\Theta \log p(\xi|\Theta))^T\right] \end{aligned} \tag{4.22}$$

The Lagrange function of optimization in (4.19) and (4.20) are written as:

$$\mathcal{L}(\delta\Theta, \beta) = \mathbb{J}(\Theta) + \nabla_\Theta \mathbb{J}^T(\Theta)\,\delta\Theta + \beta\left(\frac{1}{2}(c - \delta\Theta)^T F \delta\Theta\right) \tag{4.23}$$

where F denotes the Fisher information matrix; β is the Lagrange multiplier. From the saddle-point theorem, the optimal solution $\delta\Theta$ satisfies the following condition:

$$\frac{\partial \mathcal{L}}{\partial(\delta\Theta)} = \nabla\mathbb{J}(\Theta) - \beta F(\delta\Theta) = 0 \tag{4.24}$$

Then, the optimal solution is obtained as:

$$\delta\Theta = \beta^{-1}F^{-1}\nabla\mathbb{J}(\Theta) \tag{4.25}$$

Finally, when the Lagrange multiplier $\beta > 0$ is given, the direction of the natural gradient is written as:

$$\tilde{\nabla}_\Theta \mathbb{J}(\Theta) = F^{-1}\nabla\mathbb{J}(\Theta) \tag{4.26}$$

4.4.2 Fitness Shaping and Exponential Parameterization

Natural evolution strategies adopts rank-based fitness functions (object functions) to keep the method invariant with monotonically increasing transformations of the fitness function. Here, we define the utility-weighted values $\upsilon = [\upsilon_1, \cdots, \upsilon_l]$ with sort ascending to transform the fitness of the population. Invariance against a large set of transformations of the fitness function and/or the underlying search space is a desirable property of evolution strategies. Rank-based fitness shaping makes the algorithm invariant under monotonic transformations of the fitness function, and the natural gradient is invariant under linear transformations of the search space [162]. Therefore, when the same linear transformation is applied to the search space and the initial search distribution, the natural gradient will also be transformed.

Firstly, the population is sorted in descending order, which means that ξ_1 is the best and ξ_l is the worst individual. Then, we replace the fitness function as the utility values, and thus the estimation equation in (4.17) can be reformulated as,

$$\nabla_\Theta \mathbb{J}(\Theta) = \sum_{i=1}^{l} \upsilon_i \nabla_\Theta \log p(\xi_i|\Theta) \tag{4.27}$$

4.4.3 Exponential NES Update Rule

In traditional covariance matrix adaptation evolution strategies, the policy parameters follow the gradient step of covariance matrix $\delta\Sigma_\Theta$, whereby thereby the new covariance matrix $\Sigma_\Theta + \delta\Sigma_\Theta$ should hold the positive definite matrix property. However, since the gradient $\delta\Sigma_\Theta$ can be any symmetric matrix, we cannot guarantee this property. To address this issue, the covariance matrix is represented with an exponential map for symmetric matrices. We first give the exponential map function as:

$$\mathcal{S}_m := \{M \in \mathbb{R}^{m\times m} | M^T = M\} \tag{4.28}$$

and

$$\mathcal{P}_m := \{M \in \mathcal{S}_m | u^T M u > 0 \; for \; all \; u \in \mathbb{R}^m \backslash \{0\}\} \tag{4.29}$$

where $\mathcal{S}_m$ is the symmetric vector space; $\mathcal{P}_m$ is the manifold of a symmetric positive definite matrix. The exponential map is defined as:

$$\exp : \mathcal{S}_m \to \mathcal{P}_m, \quad M \to \sum_{n=0}^{\infty} \frac{M^n}{n!} \tag{4.30}$$

The map is a diffeomorphism, and 'exp' and its inverse operation $log : \mathcal{P}_m \to \mathcal{S}_m$ are continuous. The covariance matrix Σ_Θ can be represented as $\exp(\zeta)$ by exp map $\mathcal{P}_m \to \mathcal{S}_m$. The properties of the gradient update are given in Remark. 1.

Remark 1 *The updating of* exp *map has the following properties:* $\Sigma_\Theta + \delta\Sigma_\Theta$ *is the valid covariance matrix due to the vector space updating of* $\mathcal{S}_m$*; the gradient step is invariant with respect to linear transformation, due to the updating of the* ζ *of* exp *operation.*

To reduce the burden of computation of the Fish matrix, we do not directly compute the global coordinate $\Sigma_\Theta = \exp(\zeta)$, and replace it with a linear transformation to another coordinate system that the current search distribution is the standard normal distribution with zero mean and unit covariance. We firstly define the current search distribution as $(\mu_\Theta, C) \in \mathbb{R}^m \times \mathcal{P}_m$ and satisfy $CC^T = \Sigma_\Theta$. In tangent space, we define the updated search distribution as:

$$(\delta, M) \mapsto \left(\mu_\Theta^{new}, C^{new}\right) = \left(\mu_\Theta + C\delta, C\exp\left(\frac{1}{2}M\right)\right) \tag{4.31}$$

The coordinate frame is natural, because the Fisher matrix with respect to an orthonormal basis of (δ, M) is the identity matrix. Hence, the distribution $\mathcal{N}\left(\mu_\Theta, CC^T\right)$ is converted into $(\delta, M) = (0, 0)$. The tricks of obtaining updates in the natural coordinate frame is an option of exponential parameterization, which is used to keep the algorithm invariant under the linear transformation in the searching space. The log density mapping is written in new coordinate system:

$$\log(p(\xi|\delta, M)) = -\frac{m}{2}\log(2\pi) - tr(C) - \frac{1}{2}\left\|\exp(-1/2M)C^{-1}(\xi - \mu_\Theta)\right\|^2 \tag{4.32}$$

Considering the population $\xi_i = \mu_\Theta + C \cdot s_i$ where $s_i \sim \mathcal{N}(0, I)$, the gradient is formulated as:

$$\begin{aligned}
\nabla_\delta \mathbb{J} &= \sum_{i=1}^{l} \upsilon_i \cdot \nabla_\delta|_{\delta=0} \log(p(\xi_i|M = 0, \delta)) \\
&= \sum_{i=1}^{l} \upsilon_i \cdot \nabla_\delta|_{\delta=0} \left[-\frac{1}{2}\left\|C^{-1} \cdot (\xi_i - (\mu_\Theta + \delta))\right\|^2\right] \\
&= \sum_{i=1}^{l} \upsilon_i \cdot \nabla_\delta|_{\delta=0} C^{-1} \cdot (\xi_i - (\mu_\Theta + \delta)) \\
&= \sum_{i=1}^{l} \upsilon_i \cdot s_i
\end{aligned} \tag{4.33}$$

The gradient of M is:

$$\begin{aligned}
\nabla_M \mathbb{J} &= \sum_{i=1}^{l} \upsilon_i \cdot \nabla_M|_{M=0} \log(p(\xi_i|\delta=0,M)) \\
&= \sum_{i=1}^{l} \upsilon_i \cdot \nabla_M|_{M=0} \left[-\mathrm{tr}(C) - \frac{1}{2}\left\|\exp(1/2M)\, C^{-1}\cdot(\xi_i-\mu_\Theta)\right\|^2\right] \\
&= \sum_{i=1}^{l} \upsilon_i \cdot \left[-I - \left(C^{-1}\cdot(\xi_i-\mu_\Theta)\right)\cdot(-1/2I)\cdot\left(C^{-1}\cdot(\xi_i-\mu_\Theta)\right)^T\right] \\
&= \frac{1}{2}\sum_{i=1}^{l} \upsilon_i \cdot \left(s_i s_i^T - I\right)
\end{aligned} \tag{4.34}$$

From the gradient update of $\nabla_\delta \mathbb{J}$ in (4.33), it can be seen that μ_Θ depicts the center updating of search distribution. Similarly, σ describes the updating of step size of $\nabla\sigma\mathbb{J}$ as:

$$\nabla\sigma\mathbb{J} = \frac{\mathrm{tr}(\nabla_M \mathbb{J})}{m} \tag{4.35}$$

$$\nabla_B \mathbb{J} = \nabla_M \mathbb{J} - \nabla_\sigma \mathbb{J} \tag{4.36}$$

Finally, the updating rule with learning rates is given as:

$$\begin{aligned}
\mu_\Theta^{new} &= \mu_\Theta + \eta_\mu \cdot \nabla_\delta \mathbb{J} \\
&= \mu_\Theta + \eta_\mu \cdot \sum_{i=1}^{l} \upsilon_i s_i
\end{aligned} \tag{4.37}$$

$$\begin{aligned}
\sigma^{new} &= \sigma \cdot \exp(\eta_\delta/2 \cdot \nabla_\sigma \mathbb{J}) \\
&= \sigma \cdot \exp\left(\frac{\eta_\delta}{2} \cdot tr\left(\sum_{i=1}^{l} \upsilon_i(s_i s_i^T - I)\right)\Big/ m\right)
\end{aligned} \tag{4.38}$$

$$\begin{aligned}
B^{new} &= B \cdot \exp\left(\frac{\eta_B}{2} \cdot \nabla_B \mathbb{J}\right) \\
&= B \cdot \exp\left(\frac{\eta_B}{2} \cdot \left(\sum_{i=1}^{l} \upsilon_i(s_i s_i^T - I) - \frac{1}{m}\mathrm{tr}\left(\sum_{i=1}^{l} \upsilon_i(s_i s_i^T - I)\right)\cdot I\right)\right)
\end{aligned} \tag{4.39}$$

The update rule of the exponential natural evolution strategies algorithm is given in Algorithm. 3.

4.5 Learning Variable Impedance Control of Stable Dynamical System

In this section, we derive the stability conditions of the dynamical system using the exponential NES to learn the stiffness of variable impedance control. To guarantee global stability, the exploration noise of exponential NES is subjected to the stability conditions of the dynamical system.

Algorithm 3: Update Rule of Exponential NES

```
Input: $\eta_\sigma$, $\eta_\delta$, $\eta_B$, $m$, $\mu_\Theta^{init}$, $\Sigma_\Theta^{init} = C^T C$
$\sigma \leftarrow \sqrt[m]{|\det(C)|}$
$B \leftarrow C/\sigma$
while until stopping criterion do
    for $i$=1 $\cdots$ $l$ do
        Sampling: $s_i \sim \mathcal{N}(0,I)$;
        $\xi_i \leftarrow \mu_\Theta + \sigma B^T s_i$;
        Evaluation the cost function $S(\xi_i)$;
    end
    Sort $\{(s_i, \xi_i)\}$ with respect to $\bar{S}(\xi_i)$;
    Compute gradients:
    $\nabla_\delta \mathbb{J} \leftarrow \sum_{i=1}^{l} v_i \cdot s_i$;
    $\nabla_M \mathbb{J} \leftarrow \sum_{i=1}^{l} v_i \cdot \left(s_i s_i^T - I\right)$;
    $\nabla_\sigma \mathbb{J} \leftarrow \mathrm{tr}(\nabla_M \mathbb{J})/m$;
    $\nabla_B \mathbb{J} \leftarrow \nabla_M \mathbb{J} - \nabla_\sigma \mathbb{J} \cdot I$;
    Update mean: $\mu_\Theta \leftarrow \mu_\Theta + \eta_\delta \cdot \sigma B \cdot \nabla_\delta \mathbb{J}$;
    Update covariance matrix:
    $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma/2 \cdot \nabla_\sigma \mathbb{J})$
    $B \leftarrow B \cdot \exp(\eta_B/2 \cdot \nabla_B \mathbb{J})$
end
```

4.5.1 Shaping with Stable Exploration

For imitation learning, the dynamical system in (4.5) can generate the real trajectories according to the reference skills modeled by GMR. Generally, an unstable dynamical model will cause unexpected motion, such as deviation from the goal position. It is therefore necessary to determine the optimal parameters to regulate the dynamical system.

Theorem 3 *If the state trajectories are generated according to the dynamical system in (4.5), the dynamical model is globally asymptotically stable at the goal point x_g under the sufficient conditions:*

$$\begin{cases} d_k = -\Lambda_k x_g \\ \Lambda_k + (\Lambda_k)^T \prec 0, \ \forall k = 1, \cdots, K \end{cases} \tag{4.40}$$

where $\Lambda_k + (\Lambda_k)^T \prec 0$ denotes the negative definite matrix.

Proof 4 *To obtain the stability conditions of the dynamical system in (4.5), we firstly define a candidate Lyapunov function as:*

$$V(x) = \frac{1}{2}\left(x - x_g\right)^T \left(x - x_g\right) \tag{4.41}$$

According to the Lyapunov stability theorem of function in (4.41), we should set the parameters that satisfy the following conditions:

$$\begin{cases} V(x) > 0, \forall x \in \mathbb{R}^d \backslash \left\{x_g\right\} \\ \dot{V}(x) < 0, \forall x \in \mathbb{R}^d \backslash \left\{x_g\right\} \\ V\left(x_g\right) = 0 \ \& \ \dot{V}\left(x_g\right) = 0 \end{cases} \tag{4.42}$$

Obviously, $V(x) > 0$ except for goal position $x = x_g$. The time derivative of $V(x)$ is written as:

$$
\begin{aligned}
\frac{d(V)}{dt} &= \frac{dV}{dx}\frac{dx}{dt} \\
&= \frac{1}{2}\frac{d}{dx}\left(\left(x-x_g\right)^T\left(x-x_g\right)\right)\dot{x} \\
&= \left(x-x_g\right)^T\dot{x} \\
&= \left(x-x_g\right)^T\cdot\hat{f}(x) \\
&= \left(x-x_g\right)^T\sum_{k=1}^{K}\omega_k(x)(\Lambda_k x + d_k) \\
&= \left(x-x_g\right)^T\sum_{k=1}^{K}\omega_k(x)\left(\Lambda_k(x-x_g)+\Lambda_k x_g + d_k\right) \\
&= \sum_{k=1}^{K}\omega_k(x)\left(x-x_g\right)^T\Lambda_k\left(x-x_g\right)
\end{aligned}
$$

Since the parameter ω_k is positive, the condition $\dot{V}(x) < 0$ should be constrained with the condition $\Lambda_k + (\Lambda_k)^T \prec 0$. It is then easy to conclude that the condition $V\left(x_g\right) = 0$ & $\dot{V}\left(x_g\right) = 0$ is satisfied. The proof of (4.40) is thus concluded.

We still need to obtain the parameters $\phi = \{\phi_1, \cdots, \phi_K\}$ with the item $\phi_k = \{\lambda_k, \mu_k, \Sigma_k\}$ in (4.5). Common methods use the expectation maximization (EM) algorithm to determine the optimum parameters of the GMM. However, they cannot ensure global stability at goal position, because they do not consider the constraint in (4.40) in the optimization.

The item ϕ_k can be rewritten as: $\phi_k = \left\{\lambda_k, \mu_k^x, \mu_k^{\dot{x}}, \Sigma_k^x, \Sigma_k^{\dot{x}x}\right\}$. Here, the mean square error-based objective function of the optimization problem with constraints is defined as:

$$
\min \mathcal{F}(\phi) = \tfrac{1}{2T}\sum_{n=1}^{N}\sum_{t=0}^{T}\left(\hat{\dot{x}}_{t,n} - \dot{x}_{t,n}\right)^T\left(\hat{\dot{x}}_{t,n} - \dot{x}_{t,n}\right)
$$

$$
s.t.\begin{cases}
d_k = -\Lambda_k x_g \\
\Lambda_k + (\Lambda_k)^T \prec 0 \\
\Sigma_k \succ 0 \\
\sum_{k=1}^{K}\lambda_k = 1, \lambda_k \in (0,1)
\end{cases} \tag{4.43}
$$

To obtain the optimization results, we should firstly derive the partial derivative of priors $\frac{\partial \mathcal{F}}{\partial \lambda_k}$, mean $\frac{\partial \mathcal{F}}{\partial \mu_{x_i}^k}$, $\frac{\partial \mathcal{F}}{\partial \mu_{\dot{x}_i}^k}$, and covariance $\frac{\partial \mathcal{F}}{\partial \Sigma_{\dot{x}_i}^k}$, $\frac{\partial \mathcal{F}}{\partial \Sigma_{\dot{x}x_i}^k}$.

It should be noted that the parameter is added to the exploration noise in (4.9), which may become unstable due to random noise. To maintain the stability of the dynamical system in the learning process, we need to shape the exploration noise. The noise can be separated into ϖ_k^Λ and ϖ_k^d , which are added to parameters Λ and d, respectively, resulting in $d_{k,a} = d_k + \varpi_{k,a}^d$ and $\Lambda_{k,a} = \Lambda_k + \varpi_{k,a}^\Lambda$. The stable conditions of the dynamical system can then be rewritten as:

$$
d_{k,a} = -\Lambda_{k,a} x_g \tag{4.44}
$$

$$
\frac{\Lambda_{k,a} + \left(\Lambda_{k,a}\right)^T}{2} \prec 0 \tag{4.45}
$$

$$
\forall k = 1, \ldots K, \;\; a = 1, \ldots N_a
$$

The Λ_k matrix can be written as,

$$\Lambda_k = \underbrace{\frac{\Lambda_k + (\Lambda_k)^T}{2}}_{(a)} + \underbrace{\frac{\Lambda_k - (\Lambda_k)^T}{2}}_{(b)} \tag{4.46}$$

Actually, item (a) in Eq. (4.46) is the symmetric component and (b) is the skew-symmetric matrix, and it can be easily determined that quadratic form of (b) is 0. Hence, we just need to keep the negative definite of (a). Similarly, the noise of $\varpi_{\Lambda}^{k,a}$ can be constructed as the sum of the skew-symmetric and the symmetric noise matrix,

$$\varpi_{k,a}^{\Lambda} = \varpi_{k,a}^{skew} + \varpi_{k,a}^{sym} \tag{4.47}$$

It follows that we only need to design the symmetric matrix $\varpi_{k,a}^{sym}$ in the learning process. We sample the Gaussian noise, and obtain the symmetric matrix $\left(\varpi_{k,a}^{sym}\right)'$ and construct the matrix $\frac{\Lambda_k+(\Lambda_k)^T}{2} + \eta_{\varpi}\left(\varpi_{k,a}^{sym}\right)'$, where the parameter η_{ϖ} is decreasing from 1 to 0 until it is negative definite. For condition (4.44), the exploration noise should satisfy the following:

$$\varpi_{k,a}^{d} = \varpi_{k,a}^{\Lambda} x_g \tag{4.48}$$

After obtaining the optimal parameters from (4.43), the parameters can always preserve the conditions (4.44) and (4.45) when the noise is constrained.

4.5.2 Variable Impedance Control Learning Policy

For the dynamical system, the variable impedance controller is written as,

$$U = -K_P(x - x_d) - K_D(\dot{x} - \dot{x}_d) + U_f \tag{4.49}$$

$$K_D = \mathrm{diag}\left(2\sqrt{K_P}\right) \tag{4.50}$$

where U_f is the feedback feedforward signal; K_P, K_D are the positive definite matrix which represents the stiffness and damping matrices, respectively; U is the control input. For Cartesian space tracking tasks, the x denotes the trajectory in task space, giving us the following relationship:

$$K_{P,q} = \mathcal{J}^T K_{P,C}\mathcal{J},\ K_{D,q} = \mathcal{J}^T K_{D,C}\mathcal{J} \tag{4.51}$$

where $\mathcal{J}$ is the Jacobian matrix of the robot; $K_{P,q}$ and $K_{P,C}$ denote the joint space and the Cartesian space stiffness matrices, respectively. Similarly, $K_{D,q}$ and $K_{D,C}$ denote the damping matrices in joint space and Cartesian space, respectively.

According to the relationship in (4.50), we just need to learn the stiffness parameters. The parameterized policy of stiffness learning associated with a 3-D dynamical system can be reformulated as,

$$\Phi(x_t) = [\Phi_1(x_t), \Phi_2(x_t), \ldots, \Phi_K(x_t)] \tag{4.52}$$

$$\Phi_k(x_t) = \omega_k \begin{bmatrix} x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1^t & x_2^t & x_3^t & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.53}$$

$$\Theta = [\Theta_1, \ldots, \Theta_k, \ldots, \Theta_K]^T \tag{4.54}$$

$$\Theta_k = \left[\Lambda_k^{1,1}, \Lambda_k^{1,2}, \Lambda_k^{1,3}, \Lambda_k^{2,1}, \Lambda_k^{2,2}, \Lambda_k^{2,3}, \Lambda_k^{3,1}, \Lambda_k^{3,2}, \Lambda_k^{3,3}, \Lambda_k^{4,1}, \Lambda_k^{4,2}, \Lambda_k^{4,3}, d_k^1, d_k^2, d_k^3, d_k^4\right] \tag{4.55}$$

where $\Lambda_k^{4,1}$, $\Lambda_k^{4,2}$, $\Lambda_k^{4,3}$, and d_k^4 coincide with the stiffness parameters.

To fit the stiffness parameters into the learning process, we encode the K_P with the auxiliary dynamical system. Motivated by [152], the dynamical system of stiffness K_P is defined as:

$$\dot{K}_{P,j} = \alpha_{k_p}\left(\left(g_x^j\right)^T\left(\Theta^{K_{p,j}} + \varpi^{K_{p,j}}\right) - K_{P,j}\right) \tag{4.56}$$

$$\Theta^{K_{P,j}} = \left[\Lambda_j^{4,1}, \Lambda_j^{4,2}, \Lambda_j^{4,3}, d_j^4\right]^T \tag{4.57}$$

$$g_x^j = \omega_k^j \cdot \left[x_1^d, x_2^d, x_3^d, 1\right]^T \tag{4.58}$$

where j is the index of the case of 3-D dimension space; $\varpi^{K_{p,j}}$ is the noise added to the dynamical equation of stiffness; the parameter α_{k_p} is a large positive constant scalar so that causes the time derivative of K_P to converge to zero quickly, thus enabling the (4.56) to be rewritten as:

$$K_{P,j} = \left(g_x^j\right)^T\left(\Theta^{K_{p,j}} + \varpi^{K_{p,j}}\right) \tag{4.59}$$

Therefore, the required stiffness can be obtained by learning the parameter $\Theta^{K_P} = \left[\Theta^{K_{P,1}}, \Theta^{K_{P,2}}, \ldots\right]$. The learning scheme is shown in Fig. 4.2.

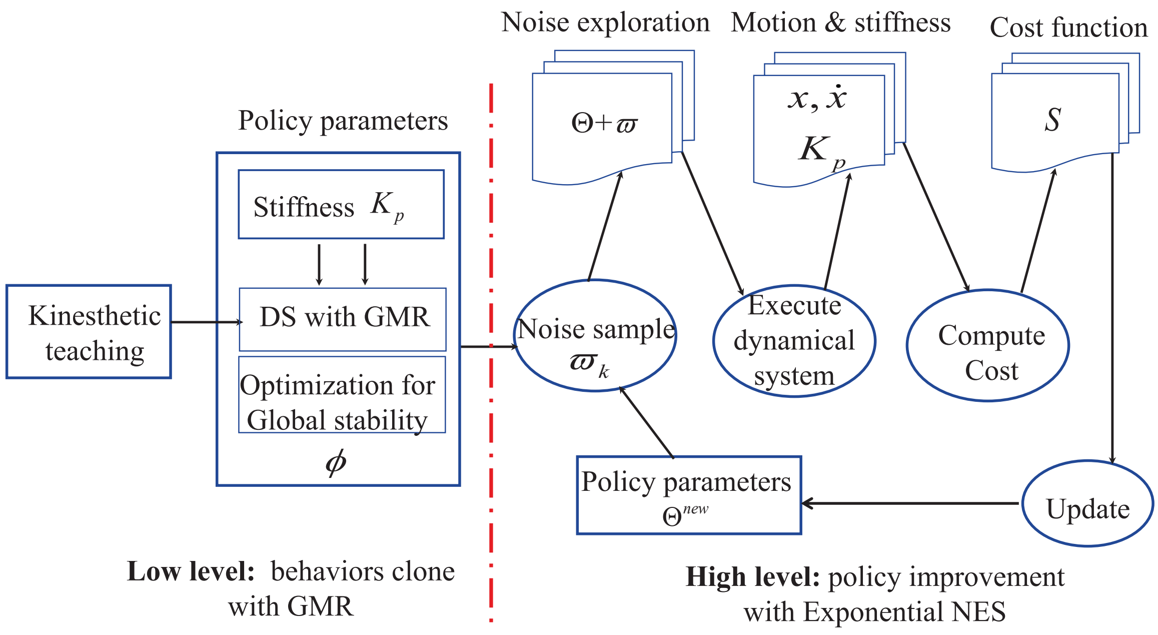


Figure 4.2: The learning process of the dynamical system and stiffness using exponential NES.

Finally, for learning stiffness tasks, the details of the cost function in (4.14) are given as:

$$\begin{aligned}
\phi\left(x_{t_N}\right) &= \left\|x_g - x_{t_N}\right\|^2 + \sum_{\substack{n=1 \\ j=1:t_N}}^{N_{via}} \min\left\|x_n^{via} - x_{t_j}\right\|^2 \\
r_{t_j} &= \alpha_1\left\|\ddot{x}_{t_j}\right\| + \alpha_2\left\|K_{P,t_j}\right\| \\
&= \alpha_1\left\|\dot{x}_{t_j} - \dot{x}_{t_{j-1}}\right\| + \alpha_2\left\|K_{P,t_j}\right\| \\
R &= \alpha_3 I
\end{aligned} \tag{4.60}$$

where α_1, α_2 and α_3 are the weights of the various components of the cost function, which is designed for users according to the tasks; N_{via} is the the number of via-points, i.e., the set points through which the dynamical system should pass in the learning tasks. In the special case of trajectory learning, we want the robot to go through points that did not appear in the primary experience. Finally, the details of the natural evolution strategies learning dynamical system is given as Algorithm. 4.

Algorithm 4: Learning with Exponential NES

```
Input: Dataset $\{x_{t,n}, \dot{x}_{t,n}\}$, $\eta_\sigma$, $\eta_\delta$, $\eta_B$, $m$, $\mu_\Theta^{init}$, $\Sigma_\Theta^{init} = C^T C$
$\sigma \leftarrow \sqrt[m]{|\det(C)|}$
$B \leftarrow C/\sigma$
Low-level Learning
    Regression with GMR from human demonstration and get parameter
      $\phi_k = \{\lambda_k, \mu_k, \sum_k\}$ ;
    Obtain the optimal parameter $\phi = \{\phi_1, \cdots, \phi_K\}$ by optimization in (4.43);
end;
High-level Learning
    while until stopping criterion do
        for $i$=1 $\cdots$ $l$ do
            Sampling: $\varpi_i \sim \mathcal{N}(0, I)$; /* To guarantee the stability of
               dynamical system, we shape the noise as in (4.47)
               (4.48)                                                  */
            $\xi_i \leftarrow \mu_\Theta + \sigma B^T \varpi_i$;
            Evaluation of the cost function $S(\xi_i)$;
        end
        Sort $\{\varpi_i, \xi_i\}$ with respect to $\bar{S}(x_i)$;
        Compute gradients:
        $\nabla_\delta \mathbb{J} \leftarrow \sum_{i=1}^{l} \upsilon_i \cdot \varpi_i$;
        $\nabla_M \mathbb{J} \leftarrow \sum_{i=1}^{l} \upsilon_i \cdot (\varpi_i \varpi_i^T - I)$;
        $\nabla_\sigma \mathbb{J} \leftarrow \mathrm{tr}(\nabla_M \mathbb{J})/m$;
        $\nabla_B \mathbb{J} \leftarrow \nabla_M \mathbb{J} - \nabla_\sigma \mathbb{J} \cdot I$;
        Update mean: $\mu_\Theta \leftarrow \mu_\Theta + \eta_\delta \cdot \sigma B \cdot \nabla_\delta \mathbb{J}$;
        Update covariance matrix:
        $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma/2 \cdot \nabla_\sigma \mathbb{J})$;
        $B \leftarrow B \cdot \exp(\eta_B/2 \cdot \nabla_B \mathbb{J})$;
    end
end;
```

4.6 Experimental Demonstration

In the following, we describe the application of the proposed algorithm in imitation learning. There are three scenarios in which we demonstrate our approach. The first case is autonomous motion planning after learning, the second case is motion learning with an obstacle, and the third case is stiffness learning of a variable impedance controller.

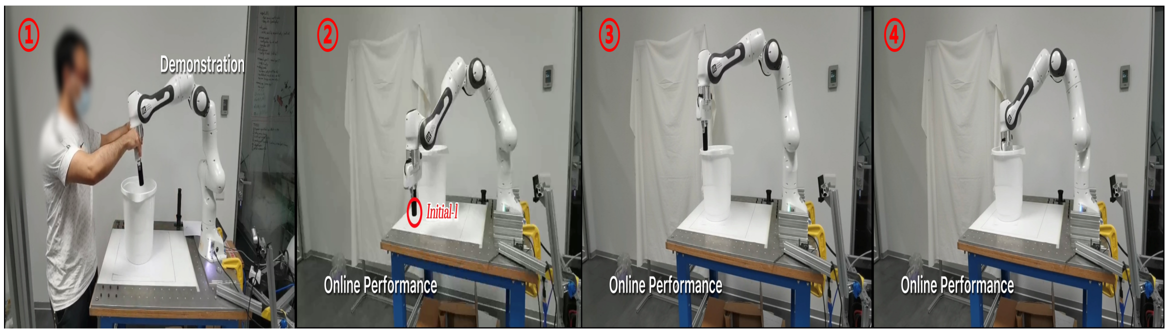


Figure 4.3: Physical experiments are conducted using Franka Emika robot, where ① is the expert data from human demonstrations; ② ③ ④ are the different running phases at the final policy.

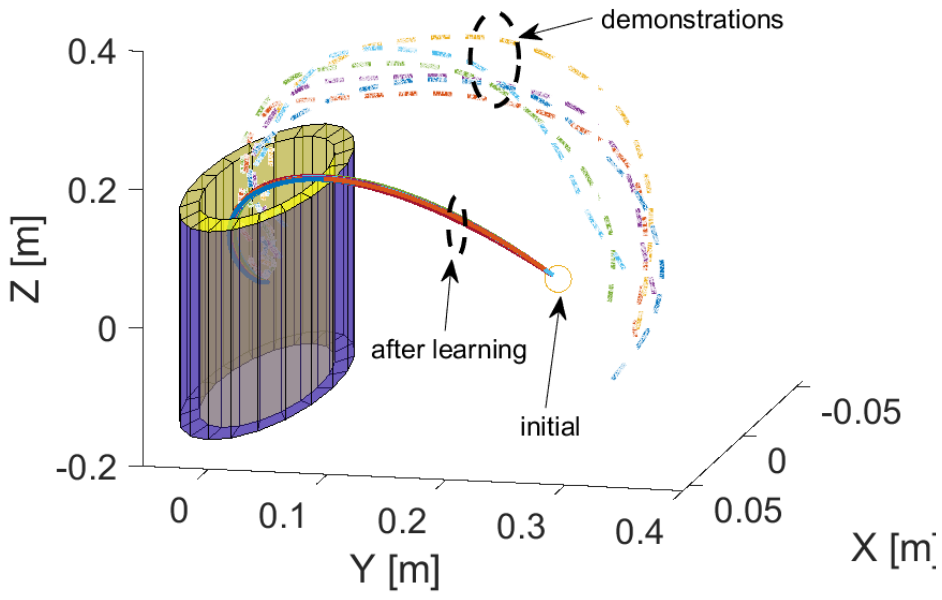
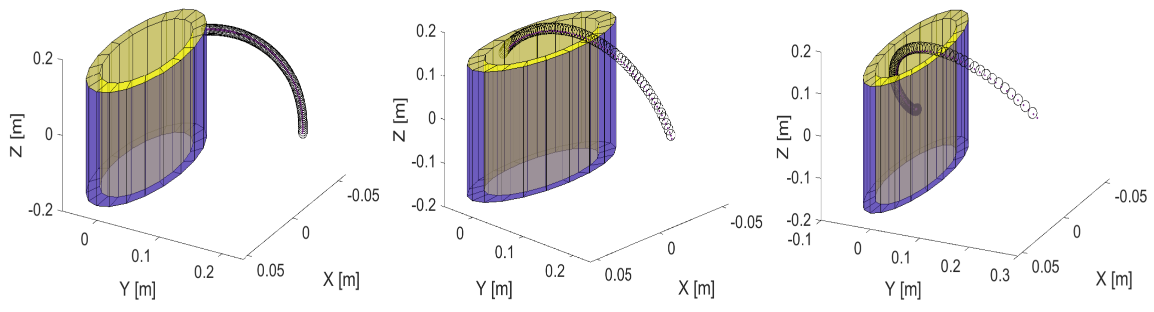


Figure 4.4: The robot is controlled to repeat the motion 6 times from human demonstration. The initial position of the reproduction is selected randomly and learned by policy improvement.

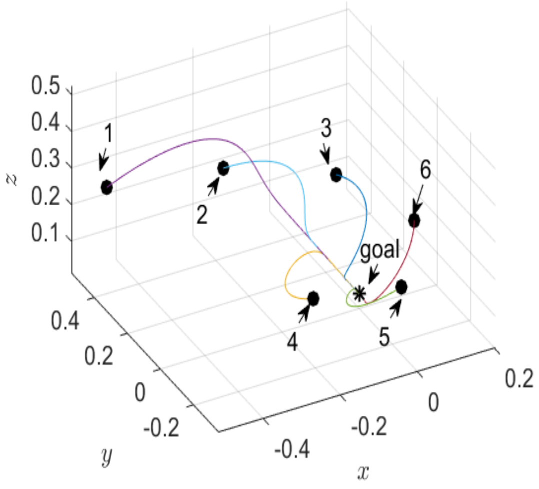


(a) 1 roll-outs **(b)** 300 roll-outs **(c)** 2000 roll-outs

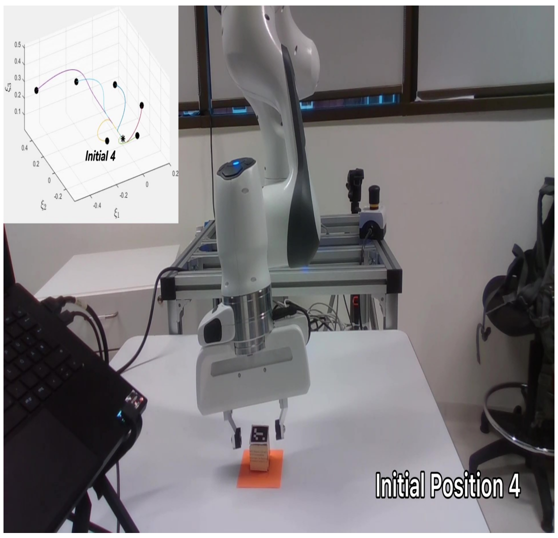
Figure 4.5: The learning process of the dynamical system for obstacle avoidance. The first roll-outs collide with the bucket, the 300 roll-outs reach into the bucket but still collide with bucket, and the 2000 roll-outs reach the goal without collision and obtain the shortest possible path under the condition of (4.43).

4.6.1 Autonomous Motion Planning

In this task, the robot is performed to grasp the object in terms of the new initial position and new goal position. Figure. 4.6 depicts the learning results of the dynamical system. Figure. 4.6a– 4.6b show the robot can converge to goal position from any initial position, which demonstrates the global convergence properties. Figure. 4.7– 4.8 show the robot can adapt the new goal position from the new initial position.

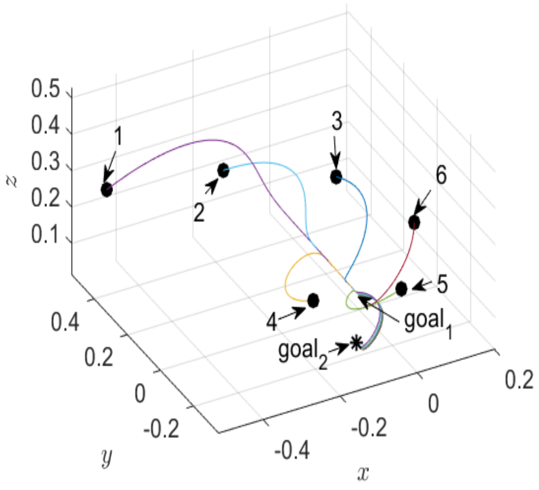


(a) The learning process of dynamical system and stiffness using exponential NES.

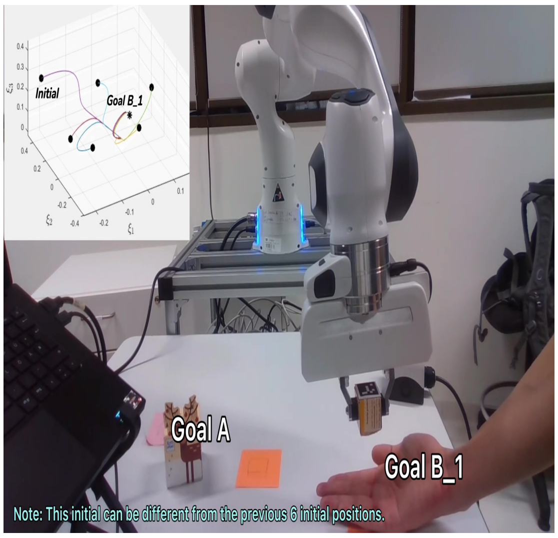


(b) Velocity of 10 samples at final policy.

Figure 4.6: Autonomous motion planning from any initial position to goal position.

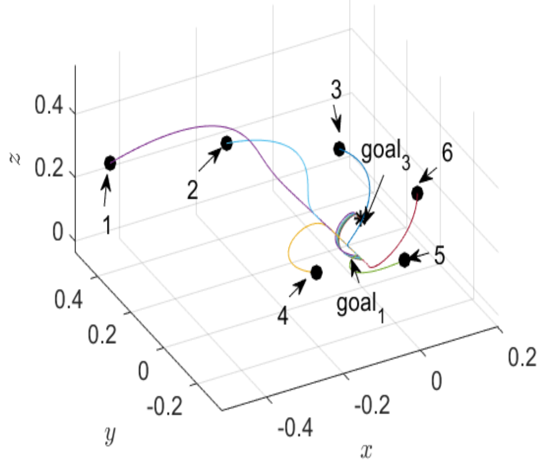


(a) The learning process of dynamical system and stiffness using exponential NES.

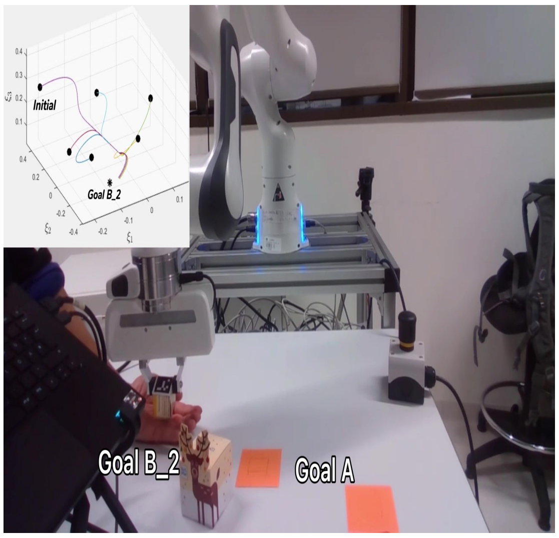

Initial
Goal B_1
Goal A
Goal B_1
Note: This initial can be different from the previous 6 initial positions.

(b) Velocity of 10 samples at final policy.

Figure 4.7: Autonomous motion planning from $goal_1$ position to to $goal_2$ position.



(a) The learning process of dynamical system and stiffness using exponential NES.



(b) Velocity of 10 samples at final policy.

Figure 4.8: Autonomous motion planning from $goal_1$ position to to $goal_3$ position.

4.6.2 Motion Learning with Obstacle Avoidance

The robot is first driven to perform the sampling tasks by human demonstration, and executes the final learning results online as shown in Fig. 4.3. In our case, the robot executes a reaching motion into a bucket and collects the data from five repetitions, as shown in Fig. 4.4. Then, the first learning strategy associated with the GMR-based nonlinear dynamical system is used to clone the motor skills. To test adaptability and stability, we randomly set the initial position and reproduce the motor behavior with the condition constraints of the nonlinear dynamical system. The cost function is detailed:

$$\begin{aligned}\phi\left(x_{t_N}\right) &= \left\|x_g - x_{t_N}\right\|^2 \\ r_{t_j} &= \alpha_1\left\|\dot{x}_{t_j} - \dot{x}_{t_{j-1}}\right\| + \alpha_2\left\|\dot{x}_{t_j}\right\| + \alpha_3 * I * \frac{1}{2}\rho_t^T(\Theta)R\rho_t(\Theta) \\ \alpha_1 &= 0.00001, \alpha_2 = 0.1,\ \alpha_3 = 0.001\end{aligned} \tag{4.61}$$

The iteration number is 200, and the number of roll-outs at each iteration is 10. When the robot collides with the bucket, the motion will stop and step into the next roll-out. The learning process is shown in Fig. 4.5. It can be seen that the robot reaches the goal position without collision and obtains the shortest possible path under the condition constraints in (4.43).

Figure. 4.9 depicts the learning results of the cost value, with all items, including total cost, via-goal cost, acceleration cost, control cost of parameters Θ, and convergence cost of velocity decreasing rapidly and achieving the optimization results. Figure. 4.10b show the velocity trajectories at the final policy. Moreover, due to the dynamical system constraints, the motion can globally converge to the goal position. Consequently, the learning strategy can limit the state of the dynamical system under the condition constraints, which is important in a real system considering the safe state. It should be noted that the learning method can modify the exploration noise by updating the covariance matrix, which is a convenient way of shaping the noise in a large exploration space.

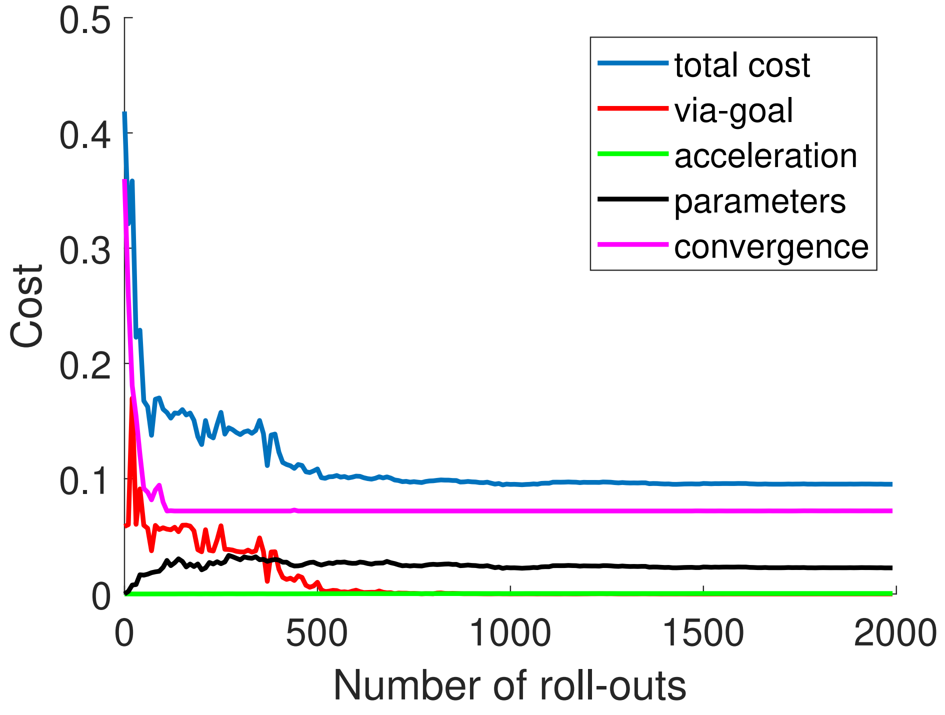
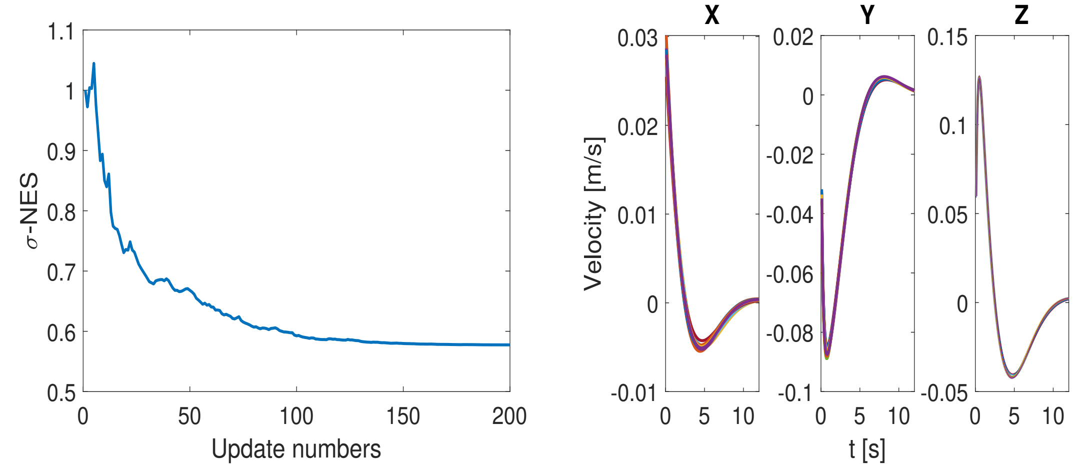


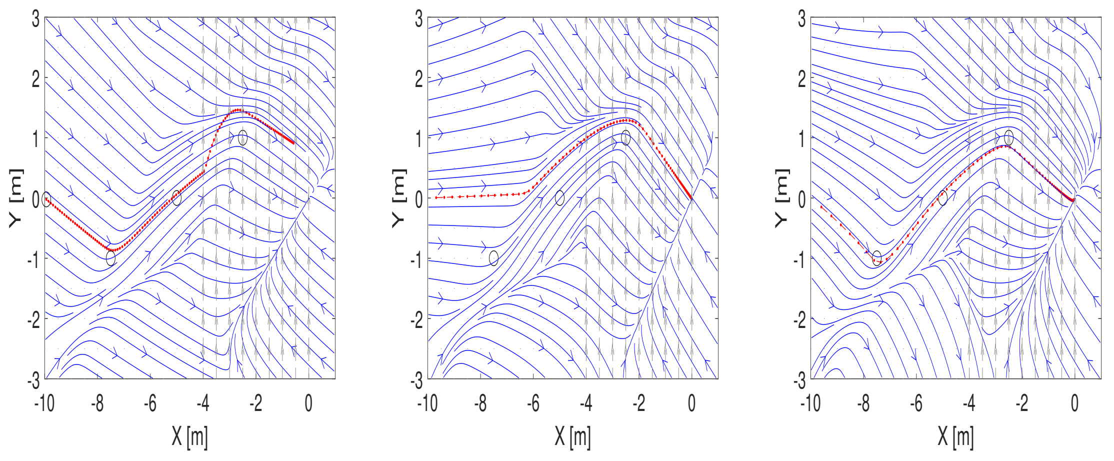
Figure 4.9: Cost value along the update numbers in the learning process.



(a) Obstacle avoidance case: learning process of σ along the update numbers.

(b) Velocity of 10 samples at final policy.

Figure 4.10: Policy learning for obstacle avoidance case using exponential NES.



(a) 1 roll-outs **(b)** 200 roll-outs **(c)** 1000 roll-outs.

Figure 4.11: The learning process of the dynamical system for stiffness learning. The first roll-outs deviate from the original trajectories with the disturbance, while the 200 roll-outs gradually converge to the original trajectories along with the iterations but still with a high error rate, and 1000 roll-outs coincide with the original trajectories and adaptation towards via -points under the condition of (4.43).

4.6.3 Learning Stiffness of Variable Impedance Control

Here, the robot should pass through the via-points, and the variable impedance controller is explored to control the robot with the external disturbance. The external force field is set as $F_{ext} = [0, 8]^T$, which means the force is only applied to the y-axis. The cost function is detailed:

$$\begin{aligned} \phi\left(x_{t_N}\right) &= \left\|x_g - x_{t_N}\right\|^2 + \sum_{\substack{n=1 \\ j=1:t_N}}^{N_{via}} \min \left\|x_n^{via} - x_{t_j}\right\|^2 \\ r_{t_j} &= \alpha_1 \left\|\dot{x}_{t_j} - \dot{x}_{t_{j-1}}\right\| + \alpha_2 \left\|K_p\right\| + \alpha_3 * I * \frac{1}{2}\rho_t^T(\Theta) R \rho_t(\Theta) \\ \alpha_1 &= 0.001, \alpha_2 = 0.0002, \alpha_3 = 0.000001 \end{aligned} \tag{4.62}$$

The learning process is shown in Fig. 4.11, where the trajectory deviates from the original trajectory in the first roll-outs due to the force field (gray dotted line), and gradually converges to the original trajectory, passing pass through the desired via-points (4 via-points and goal point) along with the iterations after 200 roll-outs, and slightly deviating from the original trajectory and passing more precisely through the via-points after 1000 roll-outs. Figure 4.12 shows the learning process cost value, where all cost items, including total cost, via-points cost, acceleration cost, parameters cost, and stiffness cost converge and decrease significantly. Specifically, the rapid decrease in via-points cost, convergence cost, and total cost means that in terms of via-points and accuracy, the policy works. Figure. 4.13a also converged reasonably quickly and reached a steady-state after 80 iterations. Clearly, the performance and convergence behavior of the policy suggest that the proposed policy is a good match for our case.

Figures. 4.13b and 4.14a show the position and velocity trajectories at the last iteration, respectively. It can be seen that the trajectories pass through the via-points at the final policy with external disturbance, and globally converge to goal position under the constraints of the dynamical system (shown in streamlines). The results of stiffness learning are shown in

Fig. 4.14b, with the stiffness level first increasing to counter external perturbation, and then decreasing when it is close to the goal position.

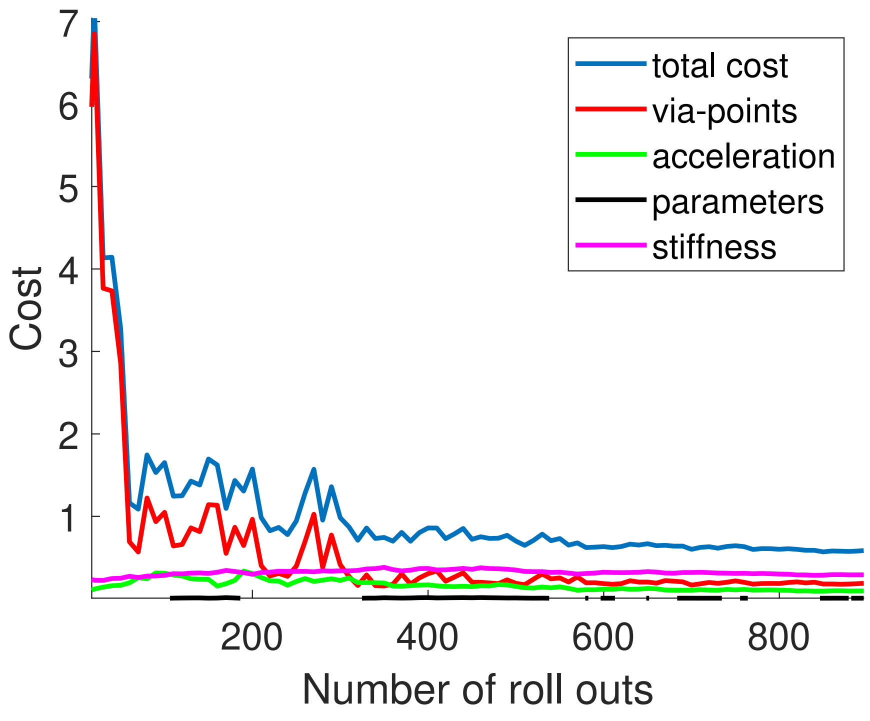
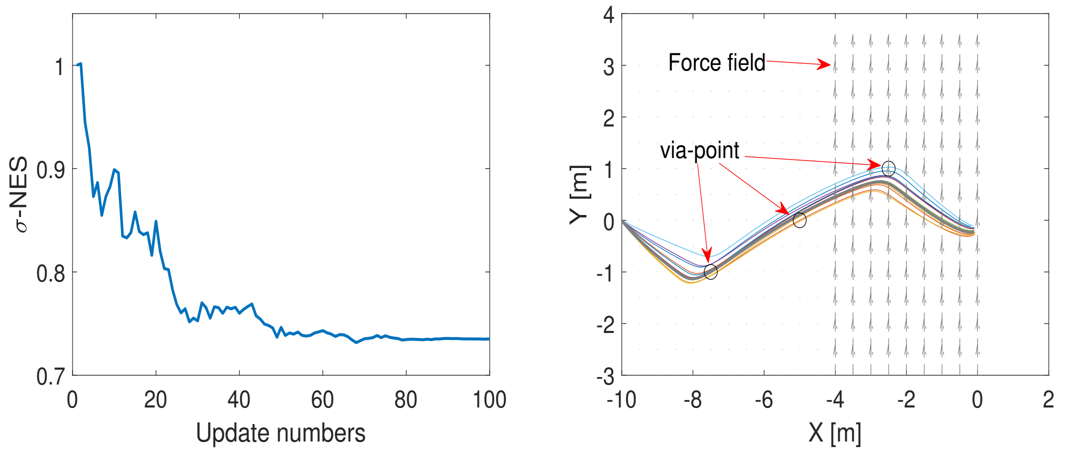


Figure 4.12: The learning process of the dynamical system and stiffness using exponential NES.



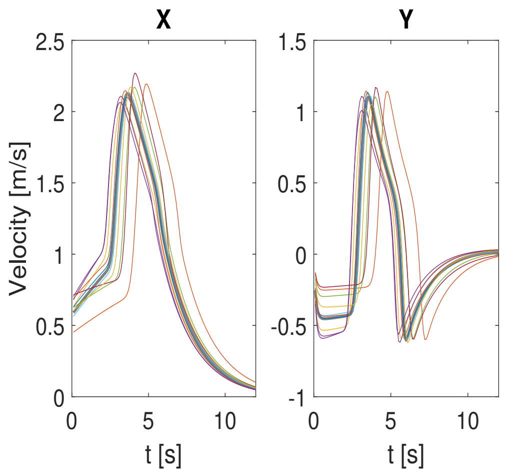
(a) Disturbance case: learning process of σ along the update numbers.

(b) The learning results of trajectory using exponential NES.

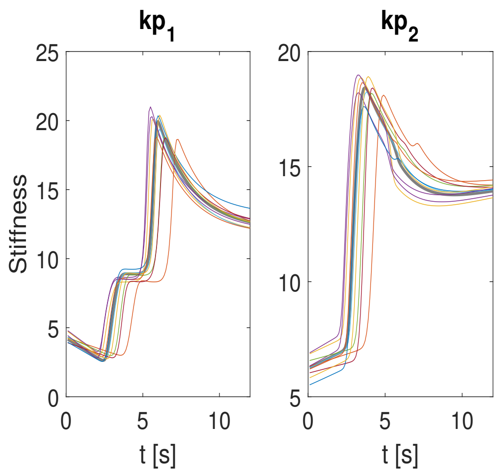
Figure 4.13: Policy learning for disturbance case using exponential NES.

4.6.4 Analysis

With imitation learning, classical approaches based on behavioral cloning can imitate the motor skills, but lack adaptability and robustness. Based on 4.6.2, the robot can avoid the



(a) Velocity of 10 samples at final policy.



(b) The learning process of dynamical system and stiffness using exponential NES.

Figure 4.14: Learning results at final policy using exponential NES.

obstacle and follow the streamlines to match the expert trajectories of the dynamical system. Based on 4.6.3, we design the new cost function to counter the external disturbance and learn the variable impedance control. The experiment indicates that the proposed learning methods can improve the adaptability and robustness of the dynamical system. It is interesting to note that the cost function can be adjusted by users according to the task requirements. This means that the proposed method can be applied much more broadly than those presented in this chapter, and only requires the design of a reasonable cost function. For instance, it will be interesting to explore a method for learning puncturing skills in surgery.

In addition, although exploration of a large policy space will benefit the learning performance in the virtual environment, the system may be unstable or become a singularity, leading to security issues. Indeed, the constraints of the dynamical system address the problem: the robot's trajectories follow the streamline, running in a safe state when exploring policy space.

4.7 Summary

This chapter focuses on improving the adaptability and robustness of robot learning. We propose a policy improvement-based hierarchical learning strategy to imitate and motor skills from human demonstration. The low-level learning method only focuses on behavioral cloning, while the high-level one aims to enhance adaptability and robustness through policy improvement. To obtain the optimal policy parameters, the exponential natural evolution strategies method is presented for learning the parameters of the dynamical system. In experiment section, we design two scenarios which are not covered by expert data, with which to demonstrate the proposed methods. Our experiments indicate that our approach can successfully avoid obstacles and counter disturbances through learning stiffness.

4.8 Related Publication

1 **Yingbai Hu**, Guang Chen, Zhijun Li, Alois Knoll. "Robot Policy Improvement With Natural Evolution Strategies for Stable Nonlinear Dynamical System". In: *IEEE Transactions*

on Cybernetics (2022).

Chapter 5

Conclusion

This chapter provides a summarization of the proposed work, and discusses the main contributions, and analyzes the pros and cons of this thesis. Meanwhile, it also gives the future research direction of robot control using optimization strategies and imitation learning of motion generation.

5.1 Summary

In this thesis, we focus on two problems of robot manipulation: low level multi-task optimization control and high level imitation learning of motion generation. The main target of the first problem is to find an efficient and general solution for multi-task control considering task priorities. The target of the second problem is to design a powerful strategy that enhances the robustness and adaptability of imitation learning for an unstructured environment.

Specifically, as cases study, the implementation consideration of multi-task control aims at OP-swab robot, surgical robot and mobile manipulator shown in chapter 2. Chapter 2 first studies the OP-swab robot case, which designs a novel OP-swab sampling robot of COVID-19 and proposes an optimization scheme with visual feedback to control two tasks: sampling task and oral center constraint. Chapter 2 also studies the multi-task control problem in the surgical robot, where three common tasks associated with the surgical task, remote center motion, and manipulability are considered. We proposed neural network-based hierarchical optimization to address the multi-task problem. Moreover, chapter 2 researches the 9-DOFs mobile manipulator from trajectory planning to optimization control, and introduces two-timescale recurrent neural networks to address the multiple metrics optimization problem considering infinity norm and slack variable, which takes into account the joint limits effectively by considering individual joint variables and relaxes the equality constraint by decreasing the infeasible solution area.

Chapter 3 and Chapter 4 focus on robot learning approaches to improve the adaptability and robustness of manipulators. Chapter 4 also tries to combine the model-based control method with the traditional imitation learning method for obstacle avoidance.

To validate the proposed control/learning strategy, simulation and physical experiments are both conducted, and the results show that our methods are competitive compared to past work.

5.2 Primary Contributions of the Thesis

This thesis studies the multi-task control problem of the redundant manipulator, and imitation learning-based motion generation that transfers human manipulation skills to the robot. The primary contributions of this thesis are:

1 **Robot design:**

In chapter 2-section 1, to avoid the clinical staff from being affected by the COVID- 19 virus, we developed a novel 9- DOFs rigid-flexible coupling (RFC) robot to assist the COVID- 19 OP-swab sampling. This robot composes of a visual system, a UR5 robot arm, a micro pneumatic actuator (MPA), and a force sensing system. The novel micro-pneumatic actuator (MPA) for throat swab sampling is developed to achieve flexible collection and integrate a force sensor. Furthermore, force-sensing actuation offers compliance, which is helpful against shocks, particularly during interaction with the oral cavity. To ensure operation safety, we develop a novel strain gauge sensor attached to MPA which means that the proposed MPA is safer. Moreover, MPA is smaller and has a 7.5 mm cross-sectional diameter, which is convenient for working in the human oral cavity. Compared with a rigid sampling robot, the developed robot can achieve flexible sampling and thereby provides a compliant, safe, stable, and reliable sampling experience.

2 **Robot control:**

(a) In chapter 2-section 1 and section 2, a novel optimization scheme is derived for multi tasks constraints considering task tracking, compliance with the RCM constraint, joint angle and velocity limits, and manipulability index. Compared with the traditional single index optimization problem, the proposed novel hierarchical optimization framework of multi-tasks can further improve the robot's stability, safety, and success rate.

(b) In chapters 2-section 1 and section 2, a varying parameter recurrent neural network (VPRNN) based hierarchical optimization of a redundant manipulator, which guarantees multi-task optimization control, such as task tracking, Remote Center of Motion (RCM), oral cavity center (OCC) constraint, and manipulability index optimization. A theoretically grounded hierarchical optimization framework based is introduced to control multiple tasks based on their priority.

(c) In chapters 2-section 3, a two-timescale recurrent neural networks (TNN) based hierarchical optimization of a redundant mobile manipulator, which has a faster transient states response in the hidden layer(s). Compared with the traditional optimization solution of a redundant manipulator, infinity norm and slack variable are additionally introduced and leveraged by the optimization algorithm. The former takes into account the joint limits effectively by considering individual joint variables and the latter relaxes the equality constraint by decreasing the infeasible solution area.

3 **Robot learning:**

(a) In chapter 3-section 1, we design a reinforcement learning-based manipulation skill transferring strategy for a surgical robot. The proposed method consists of two main steps: We use the Gaussian mixture model and Gaussian mixture regression-based dynamic movement primitives to model the high-dimensional human-like reaching and puncture skill by human demonstrations; Reinforcement learning is adopted to improve the adaptability of the varying via-RCM point tasks, which reduces the risks and cost for the practical surgical operation.

(b) In chapter 3-section 2, the learning-control strategy is proposed, where the high-level learning scheme aims at imitating the motor skill and generating the optimal trajectory for obstacle avoidance; while the lower-level adaptive control scheme focuses on the safety and stability of the robot's movement with unknown disturbances.

(c) In chapter 4, exponential natural evolution strategies are proposed for learning the parameters of a policy that can improve the robustness and adaptability of the dynamical system, in which the low level learning is based solely on behavioral cloning using GMM, while the high level refers to the parameter learning of policy improvement based on considering robustness and adaptability by exponential NES; Exponential NES are also explored for learning the stiffness of the variable impedance control; the stiffness of the controller can be modified online according to the task's requirements. The proposed method can learn the covariance matrix parameter which is used to modify the exploration noise in the parameter space.

5.3 Shortcomings

Although the proposed control and learning strategies achieved the above contributions, there exist shortcomings that should be paid attention to in future work.

First, in chapters 2, we only study the static RCM/OCC constraint, however, RCM/OCC location is not always fixed, and there exist some special scenarios where the RCM/OCC position is moving. In this case, the velocity of RCM/OCC should be considered in the optimization scheme.

Second, in chapter 3, we apply $\mathbf{PI}^2$ to find a constant input of shape parameter of DMP, and finally, find an optimal policy that the robot could adapt to the new RCM point. Indeed, it achieves the desired performance in motor learning, but the combination of the $\mathbf{PI}^2$ and DMP seems counterintuitive to the closed-loop control strategy of original $\mathbf{PI}^2$.

Third, in chapter 4, we propose the policy improvement method to explore the optimal parameters of GMM. To guarantee the stability of the dynamical system, the exploration of state space is constrained using the Lyapunov stability theorem. It is meaningful for hardware system applications involving safe operation. However, the strategy will slow down the learning and ignore some searching regions. Therefore, this strategy is not always suitable.

5.4 Future Work

1 **Design new neural network**

For the optimization control problem, we should consider the running time, convergence rate and errors, simultaneously. As aforementioned analyzed in Introduction and Shortcomings, our previous work studies different aspects: the recurrent neural network in [57] focuses on more running time, and varying-parameters zeroing neural network in [56] pays more attention to convergence speed. Indeed, they achieved superior performance in running time or convergence speed. However, in some certain cases, we should balance the convergence rate and running time for the optimization control problem.

In future work, we will design a new neural network model which has a different structure with two-scale parameters. We hope that it can solve the global optimization problem with the convex or nonconvex objective function. Notably, the proposed method has an overall promising performance in running time, convergence speed, and tracking errors.

2 **Develop new reinforcement learning algorithm**

As reported in Section. 5.3, the combination of the $\mathbf{PI}^2$ and DMP seems counterintuitive to the closed-loop control strategy of original $\mathbf{PI}^2$. Additionally, the proposed exponential NES in chapter 5 is a family of numerical optimization methods, which belongs to open-loop learning algorithms. As compared with open-loop learning algorithms, the full closed-loop learning method can correct the exploration noise and safely take the paths that might be risky in open-loop strategies, thereby achieving lower expected costs.

In future work, according to the previous work of **PI**^{2} and exponential NES, we will try to develop a novel full closed-loop learning reinforcement learning method for policy learning, which has a similar structure to **PI**^{2} and exponential NES, but it is more efficient in learning.

3 **Reinforcement learning in imitation learning**

In this thesis, we used natural evolution strategies to optimize the dynamical system of GMM and achieved contributions in imitation learning with high robustness and adaptability. Exactly, it colones motor skills with the natural way like experts, but this policy relies only on the GMM algorithm. The traditional supervised learning methods have limitations guaranteeing adaptability and global stability. Therefore, it is significant to extend the idea to different supervised learning models.

In future work, we will explore different supervised learning for behavior cloned, such as extreme learning machines (ELM), and develop a new policy for supervised learning. Similar to chapter 5, we apply a reinforcement learning algorithm to optimize the policy which can go beyond previous purely supervised learning in terms of global stability and adaptability.

4 **Biomechanical interaction in medical robot**

In future work, we will explore the deep neural network to approximate the biomechanical mathematical model of the organ model, and then calculate the stress values associated with each position of the liver tissue during the robotic resection surgery.

The general way for biomechanical models simulation is finite element analysis, but the computation speed is slow. To address this issue, the deep neural network method is used to train the biomechanical model and predict the offset of biological tissue stress.

In addition, to facilitate the application in the real world, a 3D liver model with a biomechanical mathematical model in simulation open framework architecture (SOFA) [165] physics simulator is applied for training data samplings, such as point cloud data of current organs and tissues, force vector, and offset data for each point of the stressed tissue. By designing a novel deep neural network model, we can speed up the computational process by parallel operation.

Bibliography

[1] H. Sage, M. De Mathelin, and E. Ostertag. “Robust control of robot manipulators: a survey”. In: *International Journal of control* 72.16 (1999), pp. 1498–1522.

[2] T. Yoshikawa. “Force control of robot manipulators”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 220–226.

[3] M. H. Raibert and J. J. Craig. “Hybrid position/force control of manipulators”. In: (1981).

[4] S. G. Khan, G. Herrmann, M. Al Grafi, T. Pipe, and C. Melhuish. “Compliance control and human–robot interaction: Part 1—survey”. In: *International journal of humanoid robotics* 11.03 (2014), p. 1430001.

[5] N. Hogan. “Impedance control: An approach to manipulation: Part I—Theory”. In: (1985).

[6] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger. “Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms”. In: *2003 IEEE International conference on robotics and automation (Cat. No. 03CH37422)*. Vol. 3. IEEE. 2003, pp. 3704–3709.

[7] D. Heck, D. Kostić, A. Denasi, and H. Nijmeijer. “Internal and external force-based impedance control for cooperative manipulation”. In: *2013 European Control Conference (ECC)*. IEEE. 2013, pp. 2299–2304.

[8] L. Le-Tien and A. Albu-Schäffer. “Robust adaptive tracking control based on state feedback controller with integrator terms for elastic joint robots with uncertain parameters”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2017), pp. 2259–2267.

[9] C. Ott, A. Dietrich, and A. Albu-Schäffer. “Prioritized multi-task compliance control of redundant manipulators”. In: *Automatica* 53 (2015), pp. 416–423.

[10] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. “Task-priority based redundancy control of robot manipulators”. In: *The International Journal of Robotics Research* 6.2 (1987), pp. 3–15.

[11] A. Del Prete, F. Nori, G. Metta, and L. Natale. “Prioritized motion–force control of constrained fully-actuated robots:“Task Space Inverse Dynamics””. In: *Robotics and Autonomous Systems* 63 (2015), pp. 150–157.

[12] M. Liu, Y. Tan, and V. Padois. “Generalized hierarchical control”. In: *Autonomous Robots* 40.1 (2016), pp. 17–31.

[13] C.-M. Chen, Y.-p. Chen, and Q. Zhang. “Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization”. In: *2009 IEEE Congress on Evolutionary Computation*. IEEE. 2009, pp. 209–216.

[14] K. Zhang, G. G. Yen, and Z. He. “Evolutionary algorithm for knee-based multiple criteria decision making”. In: *IEEE transactions on cybernetics* (2019).

[15] J. H. Palep. “Robotic assisted minimally invasive surgery”. In: *Journal of minimal access surgery* 5.1 (2009), p. 1.

[16] M. Zhou, Q. Yu, K. Huang, S. Mahov, A. Eslami, M. Maier, C. P. Lohmann, N. Navab, D. Zapp, A. Knoll, et al. “Towards robotic-assisted subretinal injection: A hybrid parallel–serial robot system design and preliminary evaluation”. In: *IEEE Transactions on Industrial Electronics* 67.8 (2019), pp. 6617–6628.

[17] J. Sandoval, G. Poisson, and P. Vieyres. “A new kinematic formulation of the RCM constraint for redundant torque-controlled robots”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4576–4581.

[18] H. Su, J. Sandoval, M. Makhdoomi, G. Ferrigno, and E. De Momi. “Safety-enhanced human-robot interaction control of redundant robot for teleoperated minimally invasive surgery”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6611–6616.

[19] M. Minelli and C. Secchi. “Dynamic-based RCM Torque Controller for Robotic-Assisted Minimally Invasive Surgery”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 733–740.

[20] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. “Statistical dynamical systems for skills acquisition in humanoids”. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE. 2012, pp. 323–329.

[21] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel. “Dynamic Movement Primitives in Robotics: A Tutorial Survey”. In: *arXiv preprint arXiv:2102.03861* (2021).

[22] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. “Control, planning, learning, and imitation with dynamic movement primitives”. In: *Workshop on Bilateral Paradigms on Humans and Humanoids: IEEE International Conference on Intelligent Robots and Systems (IROS 2003)*. 2003, pp. 1–21.

[23] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. “Dynamical movement primitives: learning attractor models for motor behaviors”. In: *Neural computation* 25.2 (2013), pp. 328–373.

[24] A. Paraschos, C. Daniel, J. Peters, G. Neumann, et al. “Probabilistic movement primitives”. In: *Advances in neural information processing systems* (2013).

[25] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters. “Adaptation and robust learning of probabilistic movement primitives”. In: *IEEE Transactions on Robotics* 36.2 (2020), pp. 366–379.

[26] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell. “Kernelized movement primitives”. In: *The International Journal of Robotics Research* 38.7 (2019), pp. 833–852.

[27] E. Pignat and S. Calinon. “Bayesian Gaussian mixture model for robotic policy imitation”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4452–4458.

[28] N. Jaquier, D. Ginsbourger, and S. Calinon. “Learning from demonstration with model-based Gaussian process”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 247–257.

[29] Y. Xia and J. Wang. “Recurrent neural networks for optimization: the state of the art”. In: *RECURRENT NEURAL NETWORKS* (2001).

[30] Y. Zhang, X. Lv, Z. Li, Z. Yang, and K. Chen. “Repetitive motion planning of PA10 robot arm subject to joint physical limits and using LVI-based primal–dual neural network”. In: *Mechatronics* 18.9 (2008), pp. 475–485.

[31] Z. Li and S. Li. “Kinematic Control of Manipulator with Remote Center of Motion Constraints Synthesised by a Simplified Recurrent Neural Network”. In: *Neural Processing Letters* (2021), pp. 1–20.

[32] H. Su, J. Sandoval, P. Vieyres, G. Poisson, G. Ferrigno, and E. De Momi. “Safety-enhanced collaborative framework for tele-operated minimally invasive surgery using a 7-DoF torque-controlled robot”. In: *International Journal of Control, Automation and Systems* 16.6 (2018), pp. 2915–2923.

[33] Y. Hu, H. Su, L. Zhang, S. Miao, G. Chen, and A. Knoll. “Nonlinear Model Predictive Control for Mobile Robot Using Varying-Parameter Convergent Differential Neural Network”. In: *Robotics* 8.3 (2019), p. 64.

[34] Y. Hu, J. Li, Y. Chen, Q. Wang, C. Chi, H. Zhang, Q. Gao, Y. Lan, Z. Li, Z. Mu, et al. “Design and Control of a Highly Redundant Rigid-Flexible Coupling Robot to Assist the COVID-19 Oropharyngeal-Swab Sampling”. In: *IEEE Robotics and Automation Letters* (2021).

[35] X. Wang, L. Tan, X. Wang, W. Liu, Y. Lu, L. Cheng, and Z. Sun. “Comparison of nasopharyngeal and oropharyngeal swabs for SARS-CoV-2 detection in 353 patients received tests with both specimens simultaneously”. In: *International Journal of Infectious Diseases* (2020).

[36] B. Xu, M. U. Kraemer, B. Gutierrez, S. Mekaru, K. Sewalk, A. Loskill, L. Wang, E. Cohn, S. Hill, A. Zarebski, et al. “Open access epidemiological data from the COVID-19 outbreak”. In: *The Lancet Infectious Diseases* 20.5 (2020), p. 534.

[37] J. Hindson. “COVID-19: faecal–oral transmission?” In: *Nature Reviews Gastroenterology & Hepatology* 17.5 (2020), pp. 259–259.

[38] G.-Z. Yang, B. J. Nelson, R. R. Murphy, H. Choset, H. Christensen, S. H. Collins, P. Dario, K. Goldberg, K. Ikuta, N. Jacobstein, et al. *Combating COVID-19—The role of robotics in managing public health and infectious diseases*. 2020.

[39] S.-Q. Li, W.-L. Guo, H. Liu, T. Wang, Y.-Y. Zhou, T. Yu, C.-Y. Wang, Y.-M. Yang, N.-S. Zhong, N.-F. Zhang, et al. “Clinical Application of Intelligent Oropharyngeal-swab Robot: Implication for COVID-19 Pandemic”. In: *European Respiratory Journal* (2020).

[40] H. Liu. *Application of artificial intelligence robots in respiratory diseases*. 2000.

[41] S. Wang, K. Wang, R. Tang, J. Qiao, H. Liu, and Z.-G. Hou. “Design of a low-cost miniature robot to assist the COVID-19 nasopharyngeal swab sampling”. In: *IEEE Transactions on Medical Robotics and Bionics* 3.1 (2020), pp. 289–293.

[42] *Danish startup develops throat swabbing robot for COVID-19 testing*. https://www.therobotreport.com/danish-startup-develops-throat-swabbing-robot-for-covid-19-testing/.

[43] Z. Li, J. Feiling, H. Ren, and H. Yu. “A novel tele-operated flexible robot targeted for minimally invasive robotic surgery”. In: *Engineering* 1.1 (2015), pp. 073–078.

[44] H. Su, Y. Hu, H. R. Karimi, A. Knoll, G. Ferrigno, and E. De Momi. “Improved recurrent neural network-based manipulator control with remote center of motion constraints: Experimental results”. In: *Neural Networks* 131 (2020), pp. 291–299.

[45] H. Su, C. Yang, G. Ferrigno, and E. De Momi. “Improved human–robot collaborative control of redundant robot for teleoperated minimally invasive surgery”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1447–1453.

[46] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard. “A unified framework for coordinated multi-arm motion planning”. In: *The International Journal of Robotics Research* 37.10 (2018), pp. 1205–1232.

[47] Z. Wang, Y. Torigoe, and S. Hirai. “A prestressed soft gripper: design, modeling, fabrication, and tests for food handling”. In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 1909–1916.

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

[49] Z. Li, B. Huang, A. Ajoudani, C. Yang, C.-Y. Su, and A. Bicchi. “Asymmetric bimanual control of dual-arm exoskeletons for human-cooperative manipulations”. In: *IEEE Transactions on Robotics* 34.1 (2017), pp. 264–271.

[50] J. Fei and C. Lu. “Adaptive sliding mode control of dynamic systems using double loop recurrent neural network structure”. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.4 (2017), pp. 1275–1286.

[51] Y. Li, S. Li, and B. Hannaford. “A novel recurrent neural network for improving redundant manipulator motion planning completeness”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2956–2961.

[52] F. Facchinei, C. Kanzow, and S. Sagratella. “Solving quasi-variational inequalities via their KKT conditions”. In: *Mathematical Programming* 144.1-2 (2014), pp. 369–412.

[53] Y. Chen, Q. Wang, C. Chi, C. Wang, Q. Gao, H. Zhang, Z. Li, Z. Mu, R. Xu, Z. Sun, et al. “A collaborative robot for COVID-19 oropharyngeal swabbing”. In: *Robotics and Autonomous Systems* (2021), p. 103917.

[54] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[55] A. Radonić, S. Thulke, I. M. Mackay, O. Landt, W. Siegert, and A. Nitsche. “Guideline to reference gene selection for quantitative real-time PCR”. In: *Biochemical and biophysical research communications* 313.4 (2004), pp. 856–862.

[56] Y. Hu, H. Su, G. Chen, G. Ferrigno, E. De Momi, and A. Knoll. “Hierarchical optimization Control of Redundant Manipulator for Robot-assisted Minimally Invasive Surgery”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2929–2934.

[57] H. Su, Y. Hu, J. Li, J. Guo, Y. Liu, M. Li, A. Knoll, G. Ferrigno, and E. De Momi. “Improving Motion Planning for Surgical Robot with Active Constraints”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3151–3156.

[58] A. M. Okamura. “Haptic feedback in robot-assisted minimally invasive surgery”. In: *Current opinion in urology* 19.1 (2009), p. 102.

[59] A. R. Lanfranco, A. E. Castellanos, J. P. Desai, and W. C. Meyers. “Robotic surgery: a current perspective”. In: *Annals of surgery* 239.1 (2004), p. 14.

[60] K. A. Nichols and A. M. Okamura. “A framework for multilateral manipulation in surgical tasks”. In: *IEEE Transactions on Automation Science and Engineering* 13.1 (2015), pp. 68–77.

[61] N. Enayati, A. M. Okamura, A. Mariani, E. Pellegrini, M. M. Coad, G. Ferrigno, and E. De Momi. “Robotic assistance-as-needed for enhanced visuomotor learning in surgical robotics training: An experimental study”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6631–6636.

[62] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. “Manipulability analysis.” In: *Humanoids*. 2012, pp. 568–573.

[63] B. Siciliano. “Kinematic control of redundant robot manipulators: A tutorial”. In: *Journal of intelligent and robotic systems* 3.3 (1990), pp. 201–212.

[64] H. Su, J. Sandoval, M. Makhdoomi, G. Ferrigno, and E. De Momi. “Safety-enhanced Human-Robot Interaction Control of Redundant Robot for Teleoperated Minimally Invasive Surgery”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 6611–6616.

[65] N. Aghakhani, M. Geravand, N. Shahriari, M. Vendittelli, and G. Oriolo. “Task control with remote center of motion constraint for minimally invasive robotic surgery”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 5807–5812.

[66] T. M. Sobh and D. Y. Toundykov. “Optimizing the tasks at hand [robotic manipulators]”. In: *IEEE robotics & automation magazine* 11.2 (2004), pp. 78–85.

[67] T. Ortmaier and G. Hirzinger. “Cartesian control issues for minimally invasive robot surgery”. In: *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*. Vol. 1. IEEE. 2000, pp. 565–571.

[68] A. Ebrahimi, N. Patel, C. He, P. Gehlbach, M. Kobilarov, and I. Iordachita. “Adaptive control of sclera force and insertion depth for safe robot-assisted retinal surgery”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9073–9079.

[69] L. Jin, S. Li, H. M. La, and X. Luo. “Manipulability optimization of redundant manipulators using dynamic neural networks”. In: *IEEE Transactions on Industrial Electronics* 64.6 (2017), pp. 4710–4720.

[70] H. Su, S. Li, J. Manivannan, L. Bascetta, G. Ferrigno, and E. De Momi. “Manipulability optimization control of a serial redundant robot for robot-assisted minimally invasive surgery”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 1323–1328.

[71] Z. Li, T. Zhao, F. Chen, Y. Hu, C.-Y. Su, and T. Fukuda. “Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoid-like mobile manipulator”. In: *IEEE/ASME Transactions on Mechatronics* 23.1 (2017), pp. 121–131.

[72] A. Dietrich, C. Ott, and J. Park. “The Hierarchical Operational Space Formulation: Stability Analysis for the Regulation Case”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 1120–1127.

[73] J. A. Petersen and M. Bodson. “Constrained quadratic programming techniques for control allocation”. In: *IEEE Transactions on Control Systems Technology* 14.1 (2005), pp. 91–98.

[74] H. Xiao, Z. Li, C. Yang, L. Zhang, P. Yuan, L. Ding, and T. Wang. “Robust stabilization of a wheeled mobile robot using model predictive control based on neurodynamics optimization”. In: *IEEE Transactions on Industrial Electronics* 64.1 (2016), pp. 505–516.

[75] J. Sandoval, G. Poisson, and P. Vieyres. “Improved dynamic formulation for decoupled cartesian admittance control and RCM constraint”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1124–1129.

[76] R. C. Locke and R. V. Patel. “Optimal remote center-of-motion location for robotics-assisted minimally-invasive surgery”. In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1900–1905.

[77] Y. Hu, G. Chen, X. Ning, J. Dong, S. Liu, and A. Knoll. “Mobile Robot Learning from Human Demonstrations with Nonlinear Model Predictive Control”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 5057–5062.

[78] J. Sandoval, H. Su, P. Vieyres, G. Poisson, G. Ferrigno, and E. De Momi. “Collaborative framework for robot-assisted minimally invasive surgery using a 7-DoF anthropomorphic robot”. In: *Robotics and Autonomous Systems* 106 (2018), pp. 95–106.

[79] B. Lacevic, P. Rocco, and A. M. Zanchettin. “Safety assessment and control of robotic manipulators using danger field”. In: *IEEE Transactions on Robotics* 29.5 (2013), pp. 1257–1270.

[80] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias. “Safety in human-robot collaborative manufacturing environments: Metrics and control”. In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2015), pp. 882–893.

[81] J. Woolfrey, W. Lu, and D. Liu. “A control method for joint torque minimization of redundant manipulators handling large external forces”. In: *Journal of Intelligent & Robotic Systems* 96.1 (2019), pp. 3–16.

[82] K. Glass, R. Colbaugh, D. Lim, and H. Seraji. “Real-time collision avoidance for redundant manipulators”. In: *IEEE transactions on robotics and automation* 11.3 (1995), pp. 448–457.

[83] M. Galicki. “Path-constrained control of a redundant manipulator in a task space”. In: *Robotics and Autonomous Systems* 54.3 (2006), pp. 234–243.

[84] F. Chen, M. Selvaggio, and D. G. Caldwell. “Dexterous grasping by manipulability selection for mobile manipulator with visual guidance”. In: *IEEE Transactions on Industrial Informatics* 15.2 (2018), pp. 1202–1210.

[85] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

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

[87] S. Karaman and E. Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The international journal of robotics research* 30.7 (2011), pp. 846–894.

[88] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. “Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 2997–3004.

[89] F. Burget, M. Bennewitz, and W. Burgard. “BI 2 RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3714–3721.

[90] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. “Chomp: Covariant hamiltonian optimization for motion planning”. In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1164–1193.

[91] Y. Zhang and Z. Zhang. *Repetitive motion planning and control of redundant robot manipulators*. Springer Science & Business Media, 2014.

[92] N. Kumar, J.-H. Borm, V. Panwar, and J. Chai. “Tracking control of redundant robot manipulators using RBF neural network and an adaptive bound on disturbances”. In: *International Journal of precision engineering and manufacturing* 13.8 (2012), pp. 1377–1386.

[93] J. He, M. Luo, Q. Zhang, J. Zhao, and L. Xu. “Adaptive fuzzy sliding mode controller with nonlinear observer for redundant manipulators handling varying external force”. In: *Journal of Bionic Engineering* 13.4 (2016), pp. 600–611.

[94] Y. Jiang, C. Yang, Y. Wang, Z. Ju, Y. Li, and C.-Y. Su. “Multi-hierarchy interaction control of a redundant robot using impedance learning”. In: *Mechatronics* 67 (2020), p. 102348.

[95] A. A. Hassan, M. El-Habrouk, and S. Deghedie. “Inverse kinematics of redundant manipulators formulated as quadratic programming optimization problem solved using recurrent neural networks: A review”. In: *Robotica* 38.8 (2020), pp. 1495–1512.

[96] A. Escande, N. Mansard, and P.-B. Wieber. “Hierarchical quadratic programming: Fast online humanoid-robot motion generation”. In: *The International Journal of Robotics Research* 33.7 (2014), pp. 1006–1028.

[97] Y. Hu, Z. Li, G. Li, P. Yuan, C. Yang, and R. Song. “Development of sensory-motor fusion-based manipulation and grasping control for a robotic hand-eye system”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47.7 (2016), pp. 1169–1180.

[98] X. Le and J. Wang. “A two-time-scale neurodynamic approach to constrained minimax optimization”. In: *IEEE Transactions on Neural Networks and Learning Systems* 28.3 (2016), pp. 620–629.

[99] L. Xiao, Y. Zhang, Z. Hu, and J. Dai. “Performance benefits of robust nonlinear zeroing neural network for finding accurate solution of Lyapunov equation in presence of various noises”. In: *IEEE Transactions on Industrial Informatics* 15.9 (2019), pp. 5161–5171.

[100] C. A. Klein and C.-H. Huang. “Review of pseudoinverse control for use with kinematically redundant manipulators”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 2 (1983), pp. 245–250.

[101] A. S. Deo and I. D. Walker. “Minimum effort inverse kinematics for redundant manipulators”. In: *IEEE transactions on robotics and automation* 13.5 (1997), pp. 767–775.

[102] Y. Zhang, J. Wang, and Y. Xu. “A dual neural network for bi-criteria kinematic control of redundant manipulators”. In: *IEEE Transactions on Robotics and Automation* 18.6 (2002), pp. 923–931.

[103] S. Kim, K. Jang, S. Park, Y. Lee, S. Y. Lee, and J. Park. “Whole-body control of non-holonomic mobile manipulator based on hierarchical quadratic programming and continuous task transition”. In: *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. 2019, pp. 414–419.

[104] E. Falotico, L. Vannucci, A. Ambrosano, U. Albanese, S. Ulbrich, J. C. Vasquez Tieck, G. Hinkel, J. Kaiser, I. Peric, O. Denninger, et al. “Connecting artificial brains to robots in a comprehensive simulation framework: the neurorobotics platform”. In: *Frontiers in neurorobotics* 11 (2017), p. 2.

[105] L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2001.

[106] S. Li, L. Jin, and M. A. Mirza. *Kinematic control of redundant robot arms using neural networks*. John Wiley & Sons, 2019.

[107] S. Liu and J. Wang. “A simplified dual neural network for quadratic programming with its KWTA application”. In: *IEEE Transactions on Neural Networks* 17.6 (2006), pp. 1500–1510.

[108] N. Koenig and A. Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. IEEE. 2004, pp. 2149–2154.

[109] H. Su, Y. Hu, Z. Li, A. Knoll, G. Ferrigno, and E. De Momi. “Reinforcement learning based manipulation skill transferring for robot-assisted minimally invasive surgery”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2203–2208.

[110] A. M. Okamura, C. Simone, and M. D. O’leary. “Force modeling for needle insertion into soft tissue”. In: *IEEE transactions on biomedical engineering* 51.10 (2004), pp. 1707–1716.

[111] B. Eldridge, K. Gruben, D. LaRose, J. Funda, S. Gomory, J. Karidis, G. McVicker, R. Taylor, and J. Anderson. “A remote center of motion robotic arm for computer assisted surgery”. In: *Robotica* 14.1 (1996), pp. 103–109.

[112] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario. “Medical robotics and computer-integrated surgery”. In: *Springer handbook of robotics*. Springer, 2016, pp. 1657–1684.

[113] G. A. Fontanelli, M. Selvaggio, L. R. Buonocore, F. Ficuciello, L. Villani, and B. Siciliano. “A new laparoscopic tool with in-hand rolling capabilities for needle reorientation”. In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2354–2361.

[114] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis. “Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature”. In: *The international journal of medical robotics and computer assisted surgery* 7.4 (2011), pp. 375–392.

[115] D. Hu, Y. Gong, B. Hannaford, and E. J. Seibel. “Semi-autonomous simulated brain tumor ablation with ravenii surgical robot using behavior tree”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3868–3875.

[116] M. Capurso, M. M. G. Ardakani, R. Johansson, A. Robertsson, and P. Rocco. “Sensorless kinesthetic teaching of robotic manipulators assisted by observer-based force control”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 945–950.

[117] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell. “Non-parametric imitation learning of robot motor skills”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5266–5272.

[118] S. Calinon, D. Bruno, M. S. Malekzadeh, T. Nanayakkara, and D. G. Caldwell. "Human–robot skills transfer interfaces for a flexible surgical robot". In: *Computer methods and programs in biomedicine* 116.2 (2014), pp. 81–96.

[119] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell. "Learning task priorities from demonstrations". In: *IEEE Transactions on Robotics* 35.1 (2018), pp. 78–94.

[120] F. Meier, E. Theodorou, F. Stulp, and S. Schaal. "Movement segmentation using a primitive library". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3407–3412.

[121] P. Kormushev, S. Calinon, and D. G. Caldwell. "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input". In: *Advanced Robotics* 25.5 (2011), pp. 581–603.

[122] F. Stulp, E. A. Theodorou, and S. Schaal. "Reinforcement learning with sequences of motion primitives for robust manipulation". In: *IEEE Transactions on robotics* 28.6 (2012), pp. 1360–1370.

[123] O. Koç, G. Maeda, and J. Peters. "Optimizing the execution of dynamic robot movements with learning control". In: *IEEE Transactions on Robotics* 35.4 (2019), pp. 909–924.

[124] P. Englert and M. Toussaint. "Combined Optimization and Reinforcement Learning for Manipulation Skills." In: *Robotics: Science and systems*. Vol. 2016. 2016.

[125] S. Parisi, S. Ramstedt, and J. Peters. "Goal-driven dimensionality reduction for reinforcement learning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4634–4639.

[126] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. "Reinforcement learning to adjust parametrized motor primitives to new situations". In: *Autonomous Robots* 33.4 (2012), pp. 361–379.

[127] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. "Learning physical collaborative robot behaviors from human demonstrations". In: *IEEE Transactions on Robotics* 32.3 (2016), pp. 513–527.

[128] Y. Hu, G. Chen, L. Zhang, H. Su, M. Li, Y. Schmirander, H. Cao, and A. Knoll. "Fuzzy Adaptive Control-based Real-time Obstacle Avoidance under Uncertain Perturbations". In: *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. 2020, pp. 50–55.

[129] Z. S. Abo-Hammour, O. M. Alsmadi, S. I. Bataineh, M. A. Al-Omari, and N. Affach. "Continuous genetic algorithms for collision-free cartesian path planning of robot manipulators". In: *International Journal of Advanced Robotic Systems* 8.6 (2011), p. 74.

[130] M. Duguleana, F. G. Barbuceanu, A. Teirelbar, and G. Mogan. "Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning". In: *Robotics and Computer-Integrated Manufacturing* 28.2 (2012), pp. 132–146.

[131] J.-J. Kim and J.-J. Lee. "Trajectory optimization with particle swarm optimization for manipulator motion planning". In: *IEEE Transactions on Industrial Informatics* 11.3 (2015), pp. 620–631.

[132] Y. Hu, X. Wu, P. Geng, and Z. Li. "Evolution Strategies Learning With Variable Impedance Control for Grasping Under Uncertainty". In: *IEEE Transactions on Industrial Electronics* 66.10 (2018), pp. 7788–7799.

[133] R. Lioutikov, O. Kroemer, G. Maeda, and J. Peters. “Learning manipulation by sequencing motor primitives with a two-armed robot”. In: *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 1601–1611.

[134] C. Yang, C. Chen, W. He, R. Cui, and Z. Li. “Robot learning system based on adaptive neural control and dynamic movement primitives”. In: *IEEE transactions on neural networks and learning systems* 30.3 (2018), pp. 777–787.

[135] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. “Learning movement primitives”. In: *Robotics research. the eleventh international symposium*. Springer. 2005, pp. 561–572.

[136] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal. “Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields”. In: *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. 2008, pp. 91–98.

[137] Y. Su, P. C. Muller, and C. Zheng. “Global asymptotic saturated PID control for robot manipulators”. In: *IEEE Transactions on Control systems technology* 18.6 (2009), pp. 1280–1288.

[138] Z. Li, J. Li, S. Zhao, Y. Yuan, Y. Kang, and C. P. Chen. “Adaptive neural control of a kinematically redundant exoskeleton robot using brain–machine interfaces”. In: *IEEE transactions on neural networks and learning systems* 30.12 (2018), pp. 3558–3571.

[139] T.-F. Ding, M.-F. Ge, Z.-W. Liu, Y.-W. Wang, and H. R. Karimi. “Discrete-communication-based bipartite tracking of networked robotic systems via hierarchical hybrid control”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.4 (2020), pp. 1402–1412.

[140] Z. Li, C.-Y. Su, G. Li, and H. Su. “Fuzzy approximation-based adaptive backstepping control of an exoskeleton for human upper limbs”. In: *IEEE Transactions on Fuzzy Systems* 23.3 (2014), pp. 555–566.

[141] Y. Wang, H. R. Karimi, H.-K. Lam, and H. Yan. “Fuzzy output tracking control and filtering for nonlinear discrete-time descriptor systems under unreliable communication links”. In: *IEEE Transactions on Cybernetics* 50.6 (2019), pp. 2369–2379.

[142] I. Petersén and O. Eeg-Olofsson. “The development of the electroencephalogram in normal children from the age of 1 through 15 years–non-paroxysmal activity”. In: *Neuropädiatrie* 2.03 (1971), pp. 247–304.

[143] M. Deng, Z. Li, Y. Kang, C. P. Chen, and X. Chu. “A learning-based hierarchical control scheme for an exoskeleton robot in human–robot cooperative manipulation”. In: *IEEE transactions on cybernetics* 50.1 (2018), pp. 112–125.

[144] O. Kroemer, S. Niekum, and G. Konidaris. “A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms.” In: *J. Mach. Learn. Res.* 22 (2021), pp. 30–1.

[145] S. M. Khansari-Zadeh and A. Billard. “Learning stable nonlinear dynamical systems with gaussian mixture models”. In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[146] Khansari-Zadeh, S Mohammad and Billard, Aude. “Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions”. In: *Robotics and Autonomous Systems* 62.6 (2014), pp. 752–765.

[147] J. A. Primbs, V. Nevistić, and J. C. Doyle. “Nonlinear optimal control: A control Lyapunov function and receding horizon perspective”. In: *Asian Journal of Control* 1.1 (1999), pp. 14–24.

[148] J. Duan, Y. Ou, J. Hu, Z. Wang, S. Jin, and C. Xu. “Fast and stable learning of dynamical systems based on extreme learning machine”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.6 (2017), pp. 1175–1185.

[149] S. Jin, Z. Wang, Y. Ou, and W. Feng. “Learning Accurate and Stable Dynamical System Under Manifold Immersion and Submersion”. In: *IEEE transactions on neural networks and learning systems* 30.12 (2019), pp. 3598–3610.

[150] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. “Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications”. In: *IEEE transactions on cybernetics* 50.9 (2020), pp. 3826–3839.

[151] A. Duan, R. Camoriano, D. Ferigo, Y. Huang, D. Calandriello, L. Rosasco, and D. Pucci. “Learning to Sequence Multiple Tasks with Competing Constraints”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 2672–2678.

[152] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal. “Learning variable impedance control”. In: *The International Journal of Robotics Research* 30.7 (2011), pp. 820–833.

[153] F. J. Abu-Dakka and M. Saveriano. “Variable Impedance Control and Learning—A Review”. In: *Frontiers in Robotics and AI* 7 (2020).

[154] J. Rey, K. Kronander, F. Farshidian, J. Buchli, and A. Billard. “Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies”. In: *Autonomous Robots* 42.1 (2018), pp. 45–64.

[155] M. Bogdanovic, M. Khadiv, and L. Righetti. “Learning variable impedance control for contact sensitive tasks”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6129–6136.

[156] F. Stulp and O. Sigaud. “Robot skill learning: From reinforcement learning to evolution strategies”. In: *Paladyn, Journal of Behavioral Robotics* 4.1 (2013), pp. 49–61.

[157] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi. “Bidirectional relation between CMA evolution strategies and natural evolution strategies”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2010, pp. 154–163.

[158] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. “Natural evolution strategies”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 949–980.

[159] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell. “Force-based variable impedance learning for robotic manipulation”. In: *Robotics and Autonomous Systems* 109 (2018), pp. 156–167.

[160] A. K. Tanwani and S. Calinon. “Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model”. In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 235–242.

[161] S. Yi, D. Wierstra, T. Schaul, and J. Schmidhuber. “Stochastic search using the natural gradient”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009, pp. 1161–1168.

[162] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber. “Exponential natural evolution strategies”. In: *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 2010, pp. 393–400.

[163] M. Y. Ata. “A convergence criterion for the Monte Carlo estimates”. In: *Simulation Modelling Practice and Theory* 15.3 (2007), pp. 237–246.

[164] J. R. Hershey and P. A. Olsen. “Approximating the Kullback Leibler divergence between Gaussian mixture models”. In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP’07*. Vol. 4. IEEE. 2007, pp. IV–317.

[165] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni. “Sofa-an open source framework for medical simulation”. In: *MMVR 15-Medicine Meets Virtual Reality*. Vol. 125. IOP Press. 2007, pp. 13–18.

List of Figures

1.1 Thesis structure. 4
1.2 The main research framework of this thesis. 5

2.1 9-degree-of-freedom (DOF) redundant rigid–flexible coupling robot. 8
2.2 Control framework of COVID-19 OP-swab sampling with a redundant rigid–flexible coupling robot. 8
2.3 (a) Cross section of the micro-pneumatic actuator(MPA). (b) bending process of MPA. and (c) 2D modeling of MPA. 10
2.4 Experimental results of MPA. (a) Deformation of MPA with a 50-shore hardness under different air pressures. (b) Relationship between bending angle γ and air pressure. 11
2.5 Relationship between F and ΔU. 12
2.6 Coordinate system of the 9-DOF redundant RFC manipulator. 13
2.7 Constraint with the oral cavity center. 14
2.8 Structure of VP-ZNN. 16
2.9 Demonstration with an oral cavity phantom. In our experiments, the 1:1 human oral cavity is tested. 17
2.10 The visual system for oral detection, segmentation and localization using Mask R-CNN [53]. 18
2.11 Sampling process from vision detection to sampling tasks. In our experiments, many volunteers were tested, and the recognition rate, control precision, sampling time, and sampling contact force were recorded. 18
2.12 Tracking results: the manipulator track of the desired trajectories (sampling tasks) in the Cartesian space. (a) The 3D trajectories of the manipulator; (b) the joint angle trajectories; (c) the joint velocity trajectories (located in bounds); and (d) comparison experiments with [44]. 19
2.13 PCR-test. 21
2.14 RT-PCR test results. 21
2.15 Multiple operational tasks in the Robot-assisted Minimally Invasive Surgery. . 24
2.16 VPRNN based Hierarchical Control Framework, where $N = m + 2n^2 + 2n$. The details are shown in Sec. 2.2.3. 28
2.17 Demonstration with Kuka manipulator. 30
2.18 Experimental setup scene: 1) hands-on control to move the robot manipulator to pass through the RCM constraint (small incision on the patient's body); 2) autonomous tracking is activated to run the application. 31
2.19 Sine curve: tracking results of end-effector's trajectory, trajectory errors, RCM errors and manipulability. 32
2.20 Joint angles solution using VPRNN of sine curve. 33
2.21 Angle curve: tracking results of end-effector's trajectory, trajectory errors, RCM errors and manipulability. 34

2.22 Joint angles solution using VPRNN of angle curve. 35
2.23 Top view of 9DOF mobile-based manipulator. 42
2.24 Coordinate frames of 9DOF mobile based manipulator. 44
2.25 Tracking a circular predefined trajectory in simulation environment. 50
2.26 Circular-shape tracking results including the joint position and state variable. 50
2.27 Circular-shape tracking results including the trajectory and trajectory error. . . 51
2.28 Circular-shape comparison for different metrics: the 2.28b is about the comparison results between TNN with TNN slack variable, and 2.28a is the comparison results between TNN with TNN infinity-norm. 51
2.29 Tracking an infinite shape predefined trajectory in simulation environment. . . 52
2.30 '8'-shape tracking results including the joint position and state variable. 52
2.31 '8'-shape tracking results including the trajectory and trajectory error. 53
2.32 '8'-shape comparison for different metrics: the 2.32b is about the comparison results between TNN with TNN slack variable, and 2.32a is the comparison results between TNN with TNN infinity-norm. 53
2.33 Tracking the trajectory calculated by path-planning in Gazebo simulation. . . . 53
2.34 Obstacle avoidance task tracking results including the joint position and state variable. 55
2.35 Obstacle avoidance task tracking results including the trajectory and trajectory error. 55

3.1 Path planning for puncture in robot-assisted surgery operation. The planned path should always pass trough the small incision on the abdonimal wall (RCM constriant). 59
3.2 Experimental procedures. 60
3.3 Demonstration of puncture procedure. The numbers (1-6) indicate the puncture procedure by hands-on demonstration. The 1st picture shows the start point of the tracking tasks, and the 6th picture represents the corresponding end point. The "surgeon" use hands to hold on end-effector and insert the tool tip into the abdominal cavity. 64
3.4 Reproduction from random initial position via RCM tasks by multi-demonstrations using GMM-GMR. The black curves denote the multi-demonstrations. 65
3.5 Learning with GMM. 66
3.6 Reinforcement learning process for via-point task. 66
3.7 Learning results with Reinforcement learning. 67
3.8 Demonstration of autonomous puncture using reinforcement learning. The numbers (1-6) indicate the puncture procedure. 67
3.9 Demonstrated trajectory curve. 68
3.10 The original DMPs vs the enhanced DMPs for changing targets. 72
3.11 Imitation learning in 2D space for obstacle avoidance. 73
3.12 Imitation learning in 3D space for obstacle avoidance. 73
3.13 Imitation learning in 2D space for obstacle avoidance in complex scenario. . . 74
3.14 7-DOFs kuka robot's model. 75
3.15 Tracking results in Cartesian space using fuzzy adaptive control. 78

4.1 The control flow of the learning-control framework using the proposed methods to derive a stable and robust control policy for robot system. Θ denotes the policy parameter, K_p is the stiffness of the impedance controller, $\dot{q}, q$ are the joint angle and velocity, and $\dot{x}, x$ are the state variable dynamical system. . . 84
4.2 The learning process of the dynamical system and stiffness using exponential NES. 94

4.3 Physical experiments are conducted using Franka Emika robot, where ① is the expert data from human demonstrations; ② ③ ④ are the different running phases at the final policy. 96
4.4 The robot is controlled to repeat the motion 6 times from human demonstration. The initial position of the reproduction is selected randomly and learned by policy improvement. 96
4.5 The learning process of the dynamical system for obstacle avoidance. The first roll-outs collide with the bucket, the 300 roll-outs reach into the bucket but still collide with bucket, and the 2000 roll-outs reach the goal without collision and obtain the shortest possible path under the condition of (4.43). 96
4.6 Autonomous motion planning from any initial position to goal position. 97
4.7 Autonomous motion planning from $goal_1$ position to to $goal_2$ position. 97
4.8 Autonomous motion planning from $goal_1$ position to to $goal_3$ position. 98
4.9 Cost value along the update numbers in the learning process. 99
4.10 Policy learning for obstacle avoidance case using exponential NES. 99
4.11 The learning process of the dynamical system for stiffness learning. The first roll-outs deviate from the original trajectories with the disturbance, while the 200 roll-outs gradually converge to the original trajectories along with the iterations but still with a high error rate, and 1000 roll-outs coincide with the original trajectories and adaptation towards via -points under the condition of (4.43). 100
4.12 The learning process of the dynamical system and stiffness using exponential NES. 101
4.13 Policy learning for disturbance case using exponential NES. 101
4.14 Learning results at final policy using exponential NES. 102

List of Tables

2.1 Average sampling parameters. 22
2.2 Ct distribution. 22
2.3 Experimental controller parameters 30
2.4 Explanation of the used notations shown in Fig.2.23 41