Title
Embedding Intelligence into Robotic Systems - Programming, Learning, and Planning

Author
Lin, Hsien-Chung

Publication Date
2018

Peer reviewed|Thesis/dissertation

Embedding Intelligence into Robotic Systems – Programming, Learning, and Planning

by

Hsien-Chung Lin

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Professor Kameshwar Poolla
Professor Ruzena Bajcsy

Summer 2018

Embedding Intelligence into Robotic Systems – Programming, Learning, and Planning

Abstract

Embedding Intelligence into Robotic Systems − Programming, Learning, and Planning

by

Hsien-Chung Lin

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Although robots play increasingly important roles in automated production due to their high efficiency, high accuracy, and high repeatability, new challenges for robots arise from different aspects as market demands shift and technology improves. The increasing product complexity and shorter product life cycle bring more difficulties into factories. Hence, higher intelligence for robots is necessary to perform various complicated tasks and to safely assist human workers.

In order to enhance robot intelligence, one could consider referencing the pattern of human development. When dealing with a completely new task, humans would learn or ask for assistance from experienced people or experts. Once learned, humans would apply such skill to various similar tasks. Furthermore, rather than merely completing the task, humans would make plans to accomplish the mission with better quality and efficiency. The following three phases could be referred to if we apply the same pattern to robot intelligence: 1) Programming, 2) Learning, and 3) Planning. Programming is to retrieve the information/knowledge from human. Learning is to generalize the learned skill to similar tasks. Planning is to plan an optimal policy to achieve the goal given constraints.

Following the aforementioned phases, this dissertation is divided into three parts to study the three phases. The programming part investigates several alternative programming approaches and introduces an online collision avoidance algorithm for human guidance programming. Following the framework of learning from demonstration, the learning part proposes remote lead through teaching (RLTT) for assembling and grinding skill learning and applies the non-rigid registration algorithm - coherent point drift (CPD) to transfer the learned grasp examples to similar objects. The planning part presents a fast robot motion planner by using the convex feasible set (CFS) to solve the nonconvex optimization problem in collision avoidance path planning and operational time reduction.

To My Family and My Friends

Contents

List of Figures

List of Tables

Acknowledgments

In the past five years, I grew and matured from an inexperienced young man to a Ph.D. scientist, and perhaps more importantly to become a husband and a father. The life at Berkeley was an amazing journey. I would like to express my deepest appreciation to Professor Masayoshi Tomizuka, who brought me to UC Berkeley and unconditionally supported me in research and life. He gave me the freedom to explore my own research interest, but steered me in the right direction whenever he thought I needed it. Without his guidance, this dissertation would not have been possible.

I would like to acknowledge Professor Ruzena Bajcsy and Kameshwar Poolla for serving my dissertation committee members and qualifying exam committee. Their encouragement and valuable feedback facilitate the completion of this dissertation. In addition, a sincere thank you to Professor Oliver O'Reilly, whose enthusiasm in research and teaching had lasting effect. I am gratefully indebted to the valuable lesson in my qualifying exam.

A very sincere gratitude goes out to all down at Berkeley Fellowship and FANUC Corporation for sponsoring my Ph.D. study. The close collaboration with FANUC enrich the work in this dissertation. Especially, I am very thankful to Dr. Wenjie Chen, Mr. Tetsuaki Kato, and Mr. Kaimeng Wang for the tremendous support on my robotics research.

Furthermore, a special thank goes to Hamburg University of Technology Foundation Scholarship for providing the precious research experience in Europe. In particular, I would like to thank to Dr. Eugen Solowjow for the warm hospitality and the career advice. I also appreciate his help in polishing my writing in part of this dissertation.

I would also like to thank many interesting engineers I worked with at Autodesk Inc.: David Thomasson, Heather Kerrick, Jason Cuenco, Fereshteh Shahmiri, Nicholas Cote, and Hui Li. It was fantastic to have the internship opportunity to work on part of my research in Autodesk Pier 9 Workshop.

I am glad to be a member of Mechanical Systems Control (MSC) laboratory. In the past few years, I received a lot of help from the former members: Professor Changliu Liu, Dr. Chung-Yen Lin, and Professor Cong Wang, who unselfishly shared their knowledge and experience with me. I am also very fortunate to work with my best robotics research partners: Te Tang, Yu Zhao, and Yongxiang Fan. The inspiring discussion and the numerous support from them helped me overcome various challenges. Also, I wish to give my thanks to all the past and current MSC lab members: Chi-Shen Tsai, Xu Chen, Wenlong Zhang, Yizhou Wang, Raechel Tan, Chen-Yu Chan, Junkai Lu, Minghui Cheng, Liting Sun, Xiaowen Yu, Kevin Haniger, Shiying Zhou, Dennis Wai, Shuyang Li, Cheng Peng, Wei Zhan, Daisuke Kaneishi, Zining Wang, Chianyu Chen, Kiwoo Shin, Yu-Chu Huang, Yujiao Cheng, Cheng Tang, Jiachen Li, Yeping Hu, Zhuo Xu, Shiyu Jin, Jessica Leu, and Hengbo Ma.

Nobody has been more important to me in the pursuit of this doctoral degree than the members of my family. I would like to express my very profound gratitude to my parents, whose love and guidance are with me in whatever I pursue. Most importantly, I must give my wholehearted appreciation to my wife, Wei-Ming Chan, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. I could not achieve those accomplishments without her. Thank you.

Chapter 1

Introduction

1.1 Motivation and Contribution

Since the industrial robot started to operate on the factory floor in 1960s, the application of robots has continuously grown and evolved, which made significant impacts to manufacturing processes and production lines. Today, robots play an increasingly important role in automated production due to their high efficiency, high accuracy, and high repeatability.

New challenges for robots arise as market demands shift and technology improves. The increasing product complexity and shorter product life cycles introduce more difficulties to the shopfloor. The current shifting from mass production to mass customization [26] requires manufacturers to combine human dexterity and robot productivity in flexible production lines [55]. Hence, robots should be made more intelligent perform various complicated tasks and to safely assist human workers. In addition, the rapid growth of the Internet of Thing (IoT) and the e-commerce business revolutionize customer behavior and expand the need of robots from factories to warehouse. For instance, online shopping replaces traditional in-store shopping and further induces the demands of automatic packaging and logistic. Unlike traditional "pick and place" tasks in conventional production lines, versatile grasping skills are required for robots to adapt the variation among different products. Nevertheless, robots nowadays are not cognitive enough to satisfy the aforementioned needs. Even for a simple task, it still requires tremendous programming efforts. Moreover, robots are not able generalize the programmed trajectories to similar tasks, not to mention to taking initiative to assist human by themselves.

In order to better innovate robot intelligence, one could consider referencing the pattern of human development. Human intelligence consists of the abilities to learn from experience, adapt to new situations, understand and handle abstract concepts, and use knowledge to manipulate one's environment [97]. When dealing with a complete new task, humans would learn or ask for assistance from experienced people or experts. Once learned, humans would apply such skill to various similar tasks. Furthermore, rather than merely completing the task, humans would make plans to accomplish the mission with better quality and efficiency. The following three phases could be referred to if we apply the same pattern to robot intelligence: 1) Programming, 2) Learning, and

3) Planning. Programming is to retrieve the information/knowledge from humans. Learning is to generalize the learned skill to similar tasks. Planning is to plan an optimal set of control actions to achieve the goal given constraints.

Following the aforementioned pattern of intelligence development, this dissertation is divided into three parts to study the "programming, learning and planning" phases. The first part discusses alternative robot teaching methodologies. The second part follows the framework of learning from demonstration and studies the skill learning of motion-force tasks and versatile grasping. The third part introduces an optimization-based motion planning algorithm to generate safe and efficient motion to perform tasks. This content can be further broken down into the following topics: 1) robot programming interface for demonstration, 2) human guidance programming with collision avoidance, 3) learning from demonstration with remote lead through teaching, 4) robot grasp transferring by non-rigid transformation, and 5) fast robot motion planning by convex feasible set. Each of these topics is introduced below, and its details will be given in the respective chapter of this dissertation.

Robot Programming Interface for Demonstration

Programming by Demonstration (PbD) allows to teach the skill to robots by demonstrating how to achieve a task through examples instead of explicitly programming every detail. Chapter 3 focuses on the programming interface, which is one important keystone to gather and transmit the information to robots. The robot programming interface for demonstration can be categorized into three major trends: robot imitation from human motion, kinesthetic teaching, and immersive teleoperation. Each programming interface has its own strength and drawback and may be appropriate for various tasks and scenarios. Three case studies for each programming interface are provided with a discussion on their features and implementations.

Human Guidance Programming with Collision Avoidance

In the application of physical human-robot interaction (pHRI), the collaboration between human and robot can significantly improve the production efficiency through combination of the human's flexible intelligence and the robot's consistent performance. In this application, however, it is an important concern to ensure the safety of the human and the robot. In the human guidance programming scenario, the operator plans a collision-free path for the robot end-effector, but the robot body might collide with an obstacle while being guided by the operator. Chapter 4 focuses on developing a novel on-line velocity based collision avoidance algorithm to solve the problem in this particular scenario. The proposed algorithm gives an explicit solution to deal with both collision avoidance and human guidance command at the same time, which provides the operator a better and safer lead through programming experience. Real-time experiment is performed in three different obstacle scenarios. Part of this work was published in [62, 63].

Learning from Demonstration with Remote Lead Through Teaching

Many production applications require both position and force control; however, tuning the position-force controller is nontrivial. To simplify this process, the learning from demonstration (LfD) is proposed to transfer the human skills directly into robot applications. However, the current teaching approaches are not perfectly fit in incorporating both the force and motion information for robot skill learning. Chapter 5 proposes a framework of Remote Lead Through Teaching (RLTT) for robots to learn control policies from human knowledge. To learn the human skill model, the demonstration data is firstly synchronized by dynamic time warping (DTW), then decomposed into several actions by a support vector machine (SVM) based classifier. Lastly, the learning controller is trained by a Gaussian mixture regression (GMR). The experimental validation is performed on a H7/h7 peg-hole insertion task and a surface grinding task. Part of this work was published in [64, 65].

Robot Grasp Transfer through Non-Rigid Transformation

Grasp planning is essential for robots to execute dexterous tasks. Solving the grasping problem on-line, however, is challenging due to the heavy computation load during exhaustive sampling, and the difficulties to consider task requirements. Chapter 6 focuses on combining analytic approach with learning for efficient grasp generation. The example grasps are taught by human demonstration and mapped to similar objects by a non-rigid transformation. The mapped grasps are evaluated analytically and refined by an orientation search to improve the grasp robustness and robot reachability. The proposed approach is able to plan high-quality grasps, avoid collision, satisfy task requirements, and achieve efficient on-line planning. The effectiveness of the proposed method is verified by a series of experiments. Part of this work was published in [61].

Fast Robot Motion Planning with Convex Feasible Set Algorithm

Considering the growing demand of quick motion planning in robot applications, Chapter 7 focuses on developing a fast robot motion planner to plan a collision-free and time-optimal trajectory. The convex feasible set algorithm (CFS) is applied to solve both, the trajectory planning problem and the temporal optimization problem. The performance of CFS in trajectory planning is compared with sequential quadratic programming (SQP) in simulation. CFS shows a significant decrease in iteration numbers and computation time to converge to a solution. The effectiveness of temporal optimization is shown through the cycle time reduction in the experiment.

1.2 Dissertation Outline

The structure of this dissertation is depicted in Fig. 1.1. More specifically, the remainder of this dissertation is organized as follows. Chapter 2 describes robot modeling and the experimental system setup. Part I studies the robot programming methodologies and is comprised of Chapters 3 and 4. Thereby, Chapter 3 investigates several alternative programming approaches, and Chapter 4 introduces the safe motion in kinesthetic teaching. Part II discusses the robot skill learning with its applications. Chapter 5 introduces how to teach robots assembly and grinding skill by demonstration. Chapter 6 discusses the transfer of robot grasping among various objects through non-rigid registration. Part III focuses on the topic on robot motion planning, where Chapter 7 presents an optimization-based motion planning algorithm to deal with collision avoidance and operational time reduction. Lastly, Chapter 8 summarizes the main results and contribution of this dissertation and discuss the possible extension work.

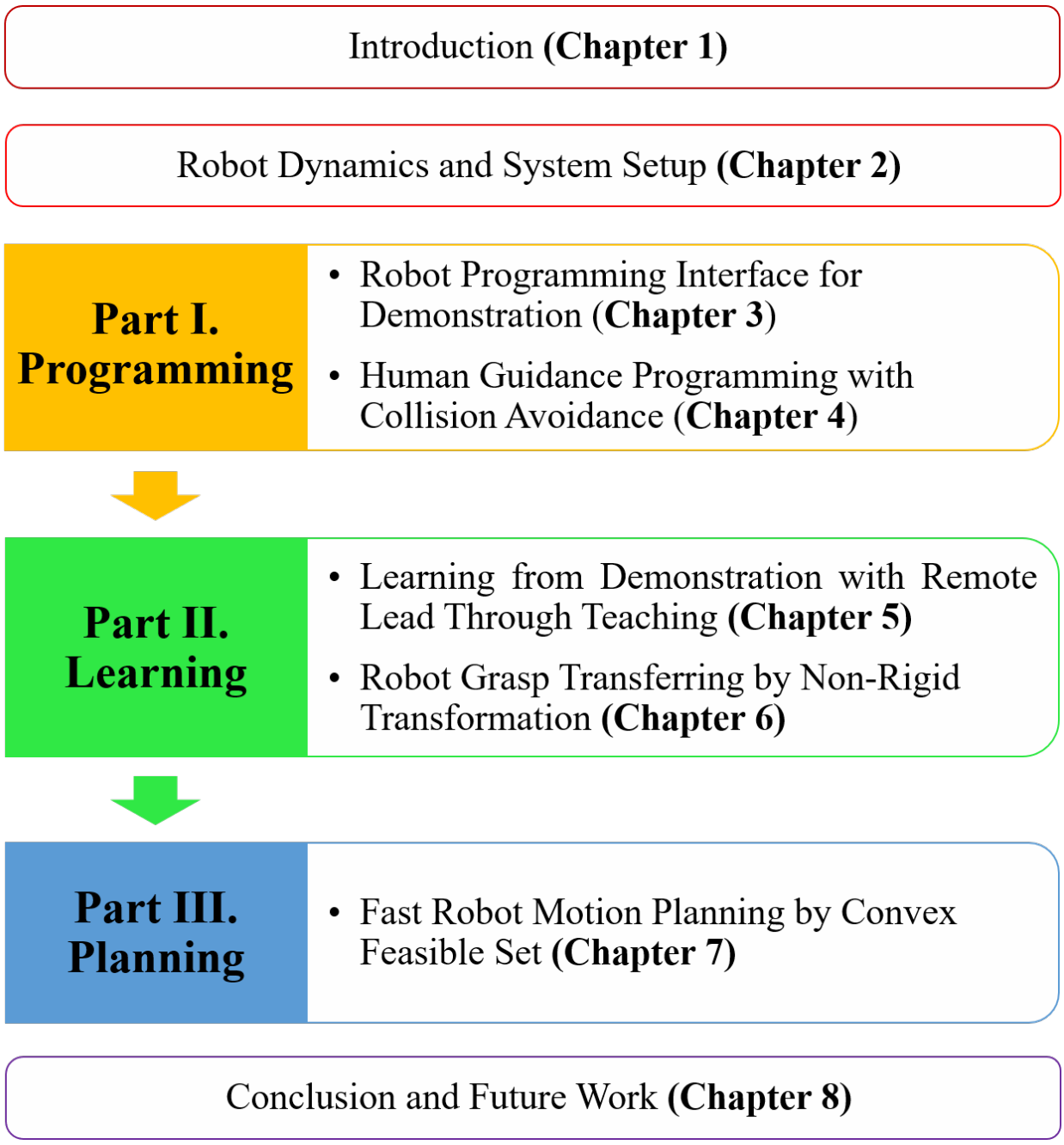


Figure 1.1: The structure of the dissertation

Chapter 2

Robot Dynamics and System Setup

2.1 Multi-Joint Indirect Drive Robot Model

The multi-joint indirect drive trains considered in this dissertation are assumed to be connected in serial. This is an appropriate assumption since the serial industrial robots with rotatory joints are examples of this kind of system. The dynamics of an n-joint robot with the gear compliance can be expressed as [102, 103]

$$M_\ell(q_\ell)\ddot{q}_\ell + C(q_\ell, \dot{q}_\ell)\dot{q}_\ell + G(q_\ell) + D_\ell\dot{q}_\ell + F_{\ell c}\text{sgn}(\dot{q}_\ell) + J(q_\ell)^{\mathrm{T}} f_{ext} \\ = K_J\left(N^{-1}q_m - q_\ell\right) + D_J\left(N^{-1}\dot{q}_m - \dot{q}_\ell\right) \tag{2.1}$$

$$M_m\ddot{q}_m + D_m\dot{q}_m + F_{mc}\text{sgn}(\dot{q}_m) \\ = \tau_m - N^{-1}\left[K_J\left(N^{-1}q_m - q_\ell\right) + D_J\left(N^{-1}\dot{q}_m - \dot{q}_\ell\right)\right] \tag{2.2}$$

where $q_\ell = \begin{bmatrix} q_{\ell 1} & q_{\ell 2} & \cdots & q_{\ell n}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^n$ and $q_m = \begin{bmatrix} q_{m1} & q_{m2} & \cdots & q_{mn}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^n$ are the load side and the motor side position vectors, respectively. $\tau_m = \begin{bmatrix} \tau_{m1} & \tau_{m2} & \cdots & \tau_{mn}\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^n$ is the motor torque vector. $M_\ell(q_\ell) \in \mathbb{R}^{n\times n}$ is the load side inertia matrix, $C(q_\ell,\, \dot{q}_\ell) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix, and $G(q_\ell) \in \mathbb{R}^n$ is the gravity vector. M_m, K_J, D_J, D_ℓ, D_m, $F_{\ell c}$, F_{mc}, and $N \in \mathbb{R}^{n\times n}$ are all diagonal matrices. The (i,i)-th elements of these matrices, J_{mi}, K_{Ji}, D_{Ji}, $D_{\ell i}$, D_{mi}, $F_{\ell ci}$, F_{mci}, and N_i, represent the motor side inertia, joint stiffness, joint damping, load side damping, motor side damping, load side Coulomb friction, motor side Coulomb friction, and gear ratio of the i-th joint, respectively. $f_{ext} \in \mathbb{R}^6$ denotes the external force acting on the robot due to the contact with the environment. The matrix $J(q_\ell) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix of the end effector. Notice that (2.1) and (2.2), is a simplified model, since the dynamics of each motor does not include the angular motion of the previous carrying links, and thus the coupling inertia and forces between every actuator and every link due to their relative positions are not considered [102, 103, 78, 2]. For robots with large gear ratios, this simplification is reasonable, since the angular velocities and accelerations of the links are much smaller than those of the motors.

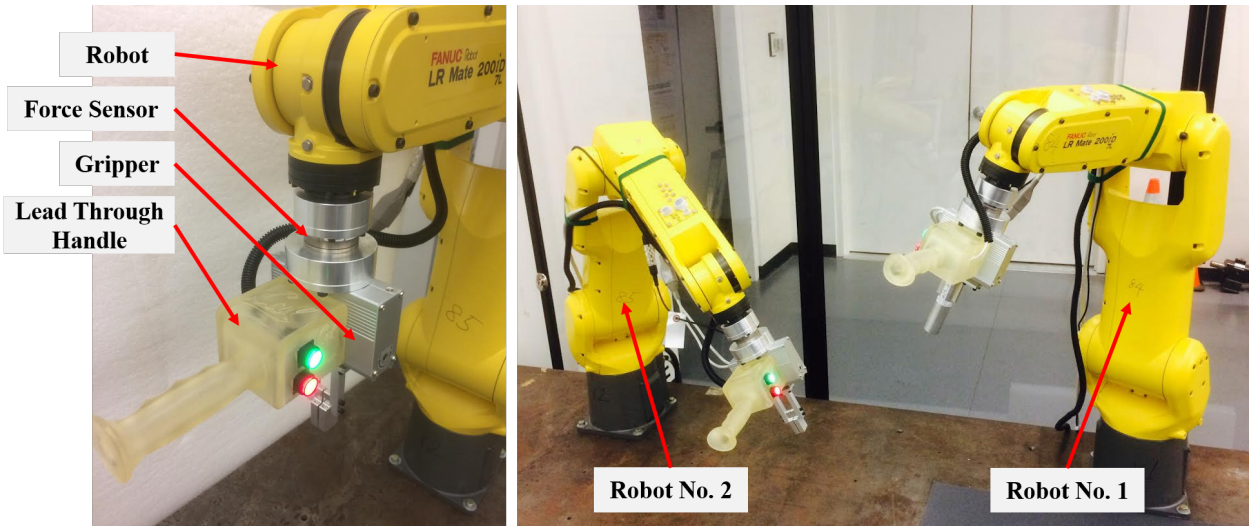


Figure 2.1: Dual LR Mate 200*i*D/7L Robot system

The overall robot dynamics combined with load side and motor side can be written as

$$\tau_m = M_m\ddot{q}_m + D_m\dot{q}_m + F_{mc}\text{sgn}(\dot{q}_m) + \tag{2.3}$$

$$N^{-1}\left[M_\ell(q_\ell)\ddot{q}_\ell + C(q_\ell,\dot{q}_\ell)\dot{q}_\ell + G(q_\ell) + D_\ell\dot{q}_\ell + F_{\ell c}\text{sgn}(\dot{q}_\ell) + J(q_\ell)^{\mathrm{T}} f_{ext}\right] \tag{2.4}$$

If there is no contact with the environment, $f_{ext} = \mathbf{0}$. Furthermore, if the joints are rigid joints instead of flexible joints, i.e., K_J and D_J become infinity, and $q_m = Nq_\ell$, (2.1) and (2.2) is further simplified as follows [74, 94]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + D\dot{q} + F_c\text{sgn}(\dot{q}) = \tau \tag{2.5}$$

where $q = q_\ell$ and $\tau = N\tau_m$ are the vectors of load side joint angles and applied load side joint torques respectively. $M(q) = M_\ell(q_\ell) + N^2M_m$ is the positive-definite inertia matrix. $C(q,\dot{q}) = C(q_\ell,\dot{q}_\ell)$ accounts for Coriolis and centrifugal effects. $F_c = F_{\ell c} + NF_{mc}$ represents Coulomb friction effects, $D = D_\ell + N^2D_m$ represents the viscous damping effects, and $G(q) = G(q_\ell)$ represents the torques due to gravity.

2.2 Dual Robot Experimental Setup

The dual robot experimental setup used at the University of California, Berkeley, courtesy of FANUC, is two LR Mate 200*i*D/7L industrial robots as shown in Fig. 2.1. LR Mate 200*i*D/7L is a six-axis, small size robot. It is designed to carry objects up to 7 kg. Its main applications are picking, handling and packaging. The specification of this robot can be found in [38].

The hardware scheme of the experimental setup is shown in Fig. 2.2. Each FANUC LR Mate 200*i*D/7L robot has built-in motor encoders. Additionally, each robot is equipped with a

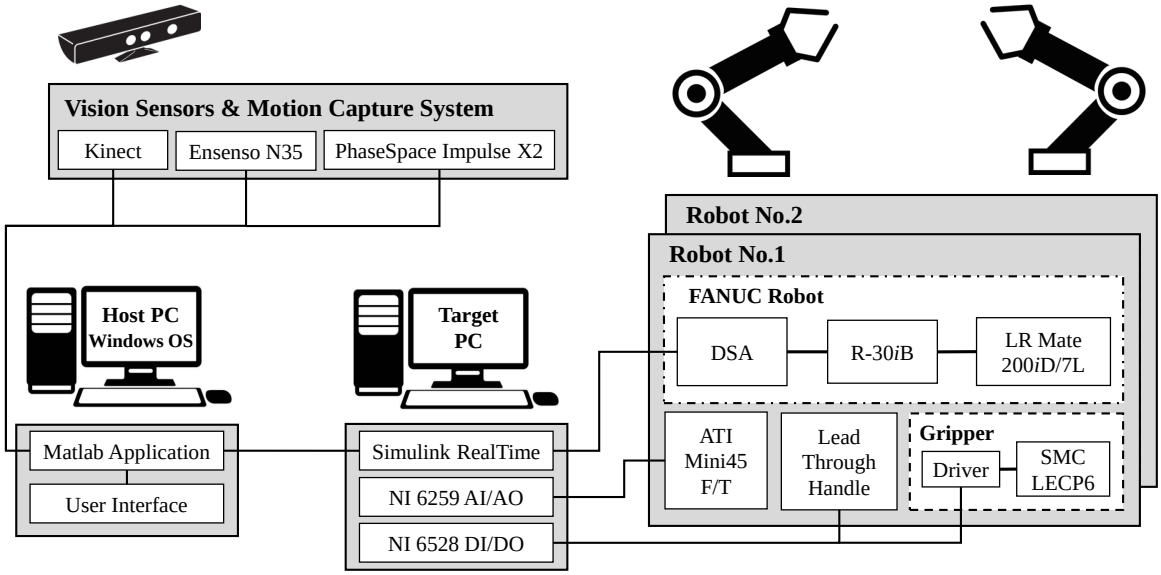


Figure 2.2: Dual LR Mate 200*i*D/7L Robot system setup scheme

force/torque transducer, a parallel gripper, and a lead through handle bar on its end-effector. The force/torque transducer (ATI mini 45[5]) is to measure the end-effector three-dimensional force and three-dimensional torque. The parallel gripper is SMC LEHF20K2 [95], which has 11 to 28 N grasping force and up to 48 mm grasping width. The gripper has a servo control unit, SMC LECP6 [95], where it can be programmed up to 64 grasping mode.The 3D-printed lead through handle bar is used to trigger the lead through teaching mode.

The basic controller used for this multi-joint robot is a decentralized PID feedback controller with pre-calculated feedforward torques (Fig. 2.3). In this controller, only motor side encoder signals are used for real-time feedback. The feedforward torques are calculated on-line based on the the multi-joint robot model (2.4) with the feedback motion signals. This controller can be customized and added more high level controller to accomplish more sophisticated tasks. Fig. 2.2 shows the hardware connection diagram. The Simulink RealTime developed by MathWorks is used to implement the proposed control algorithms. All real-time controls are deployed on the target PC, which uses FANUC PCI interface to communicate with a FANUC digital servo adapter (DSA) through EtherCAT. The force sensors, parallel grippers, and lead through bars are communicated with the target PC through National Instrument data acquisition (DAQ) boards, NI 6528 and NI 6259. The host PC provides a user interface to operate some basic functions, such as brake on/off, joint jogging, and run programmed trajectories, etc. The sampling rate for control and sensing is 1kHz. The vision sensors and motion capture system are communicated with the host PC through the local network.

In this dissertation, vision sensors and motion capture systems are used in different applications. Kinect [72] is a motion sensing input device that was introduced by Microsoft. It consists an

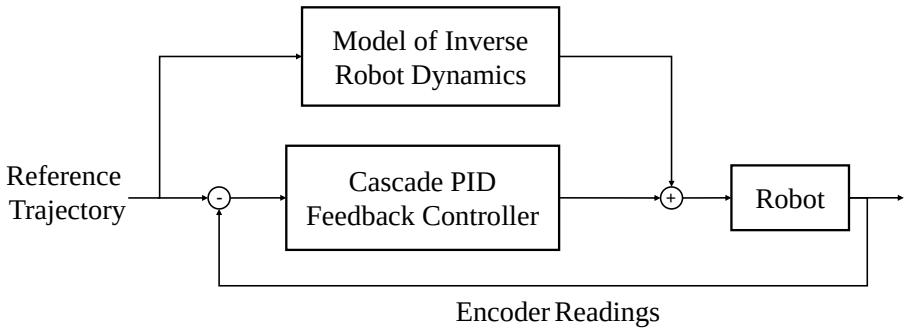


Figure 2.3: Basic controller structure

RGB camera and a depth sensor. The RGB camera outputs data in three channels with a 1280×960 pixel resolution. The depth sensor has two parts, IR emitter and IR depth sensor. The IR emitter emits infrared light beams, and the IR depth sensor receives the beams reflected by objects. The distance from the sensor to the object is calculated from the traveling time of infrared light beams. Combining the RGB camera and the depth sensor, the 3D point cloud in the workspace can be reproduced for robot perception. The Kinect sample rate is up to 30 Hz, and can track human motion and dynamic objects. The Ensenso N35 [46] operates using stereo vision, which acquires images from the same scene from two different positions. Although the cameras see the same scene content, there are different object positions according to the cameras projection rays. Special matching algorithms compare the two images, search for corresponding points and visualize all point displacements in a Disparity Map. Knowing intrinsic parameters of the camera, the Ensenso software calculates 3D coordinates of each image pixel, which results to a 3D point cloud. Although Ensenso only has 10 fps frame rate, it has much accurate perception and higher resolution, which is suitable for applications that require high precision. The motion capture system, PhaseSpace Impulse X2 [82], is to measure the three-dimensional position of LED markers in the work space. Its sampling rate is up to 960 Hz, and each camera achieves 3600×3600 optical resolution by using two linear detectors with 16-bit dynamic range.

Part I

Programming

Chapter 3

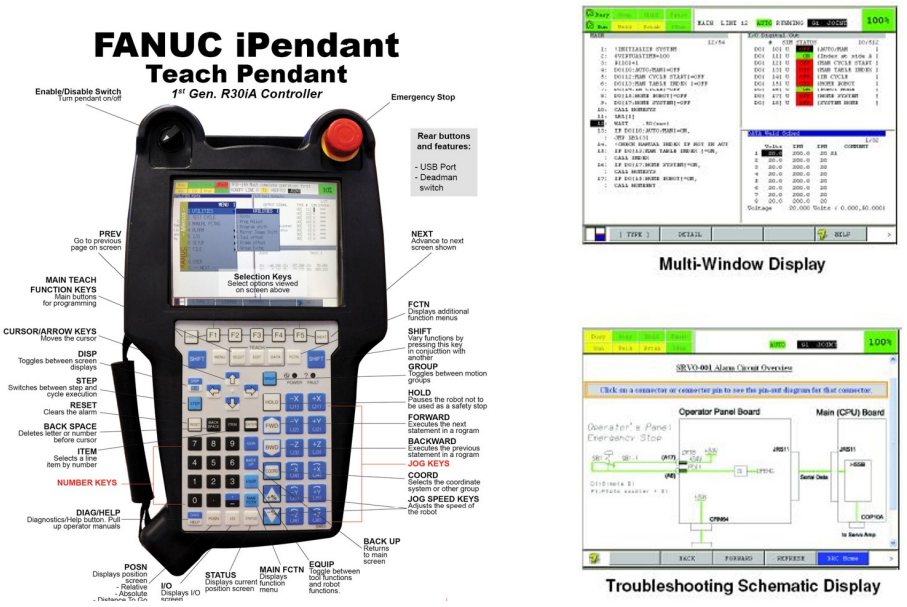
Robot Programming Interface for Demonstration

3.1 Introduction

Robot programming is the essential way to "teach" the robot the desired motion/trajectory to complete assigned tasks. The most common robot programming method is the teach pedant programming. To program a robot, the operator moves it passing through all the via points using the buttons on the panel and save each pose individually. Once the whole trajectory has been programmed, the robot can replay the same motion at full speed. This method allows precise positioning since the robot can be programmed using numerical coordinates which can be either the world coordinate, the robot base coordinate, or any other defined coordinate systems. However, this method is not intuitive to people without the related knowledge. Specifically speaking, the teach pendant has many buttons on the penal as shown in Fig. 3.1a, which makes the programming complicated. Even for a well-trained robotics technician, it still takes time to program a delicate trajectory. Moreover, the system information is presented by various code numbers on the display; however, those codes are not straightforward to understand and operate the robot system (see Fig. 3.1b).

In computer science field, Programming by Demonstration (PbD) is an end-user development technique that teaches a computer or a machine new behavior by demonstrating the task to transfer directly instead of programming it through lines of codes [29]. This idea is very attractive to robotics applications due to the aforementioned problems. To be more specific, the operator often has implicit knowledge on the tasks but does not have the programming skill to deploy a robot. PbD allows to teach the skill to robots by demonstrating how to achieve the task through examples instead of explicitly programming every detail.

In [11], Billard and Grollman further extend this idea to "Learning from Demonstration" (LfD), which is not just a *record and play* technique but a methodology to generalize a task skill to various scenarios from observing several demonstrations. The robotics research on LfD has been conducted and steadily grown recent years with the interest intersection of engineering and machine learning. Several research surveys can be found in [4, 12, 90]. In this dissertation, the related



(a) (b)

Figure 3.1: The teach pendant of FANUC robot system. (a) The FANUC iPendant (b) The display of teach pendant.

contribution in LfD will be further discussed in Part II. This chapter will focus on the programming interface, which is one important keystone to gather and transmit the information to robots.

In [11], the robot programming interface for demonstration is categorized into three major trends: 1) Robot imitation from human motion, 2) Kinesthetic teaching, and 3) Immersive teleoperation. Robot imitation from human motion is directly recording the kinematic of human motion by motion tracking systems that based on vision, exoskeleton, or other wearable motion sensors. These tracking systems return the measurement of the human joints. Some works directly teach the humanoid robot walking or arm swing by exploiting the full body motion of the human demonstration [99, 57]. Kinesthetic teaching, also called lead through teaching, is to let robot be physically guided through the task by the humans. By using this teaching method, there is no explicit physical correspondence needed since the user demonstrates the skill with the robot's own body. It also provides a natural teaching interface to correct a skill reproduced by the robot. This technique becomes one of common programming methods in the current generation of robot products [100, 86, 37]. Immersive teleoperation can be achieved by using the remote control devices such as joysticks or haptic devices. One advantage of teleoperation is that the teacher/demonstrator no longer needs to stay in the same workspace with the robot. Ocean One [52] is one example that illustrates the strength of teleoperation in ocean discovery by underwater robots. da Vinci Surgical System [47] shows a success application of teleoperation in microsurgery.

Each programming interface has its own strength and drawback and may be appropriate for various tasks and scenarios. The chapter provides the case studies for each programming inter-

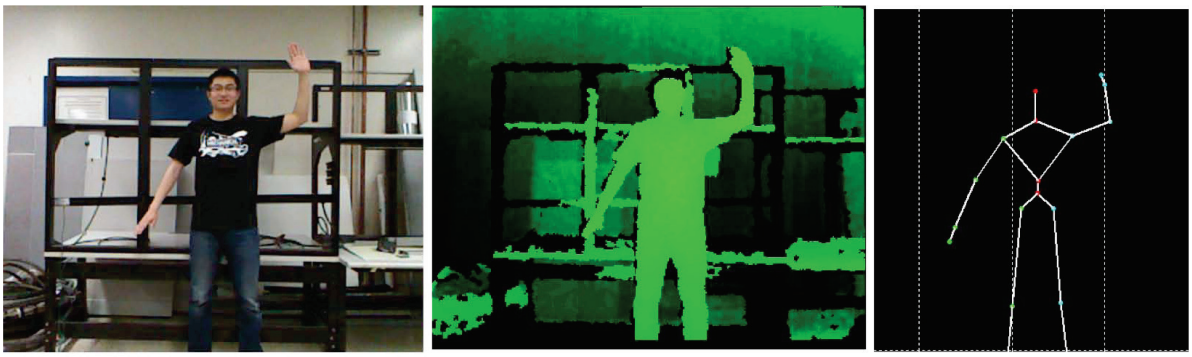


Figure 3.2: Three image formates provided by Kinect

face. In each case study, an example of the programming method is discussed, which includes its framework and implementation. The remainder of this chapter is organized as follow. Section 3.2 discusses the Robot imitation from human motion by a vision sensor. Section 3.3 talks about the implementation of kinesthetic teaching by a force damping control. Section 3.4 presents a immersive teleoperation platform in virtual reality, which is also a collaborative project with Autodesk Inc. [6]. Finally, Section 3.5 concludes the case studies of robot programming interfaces.

3.2 Robot Imitation from Human Motion

Robot imitation enables the robot to imitate human motion from demonstration. Human can observe and correct the robot trajectory whenever the robot is moving in an undesired manner. To achieve robot motion imitation, the vision sensor is used to capture human motion for determining the desired trajectory. In this case study, we use the Kinect as the vision sensor to track the human motion and further implement the robot imitation.

Figure 3.2 shows the three types of data stream provided by Kinect, which are the RGB image, depth image, and skeleton data, respectively. The first two are obtained from the RGB camera and the depth sensor. The skeleton data is calculated from the estimates of the sensor. Since this case study focus on the implementation for robot imitation, we directly exploit the skeleton data as the human motion tracking result. For other human pose tracking research, the related works can be found in [58, 3].

Kinematic Control Scheme of Robot Imitation

The basic framework for kinematic control of robot imitation on industrial robot M-16*i*B is shown in Fig. 3.3. The motion of human arm is tracked by Kinect while demonstrating a specific trajectory. Considering the sensor noise, a low-pass filter is embedded to smooth out the jittery signal.

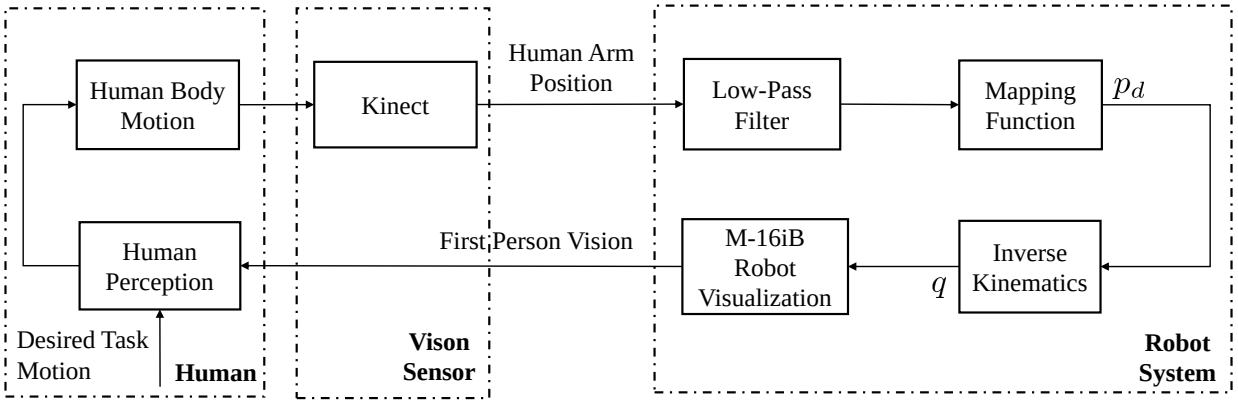


Figure 3.3: Scheme of robot imitation kinematic control on M-16*i*B

(a) (b)

Figure 3.4: Simulation result of robot imitation kinematic control on M-16*i*B. (a) Human motion capture from Kinect. (b) FANUC M-16*i*B Visualization in the simulator

The end-effector position is converted from human arm reference by a mapping function. In this work, the operator's left hand and hip center are mapping to the robot end-effector and its base, respectively. The desired tool center point (TCP), p_d, in Cartesian space is defined as,

$$p_d = p_h - p_c, \tag{3.1}$$

where p_h and p_c are the operator's left hand and hip center measured by Kinect, respectively. Since the orientation of the hand joint has not been used in this work, the orientation of the robot end-effector is assumed to be fixed, and the half-solution of inverse kinematics is used to solve for

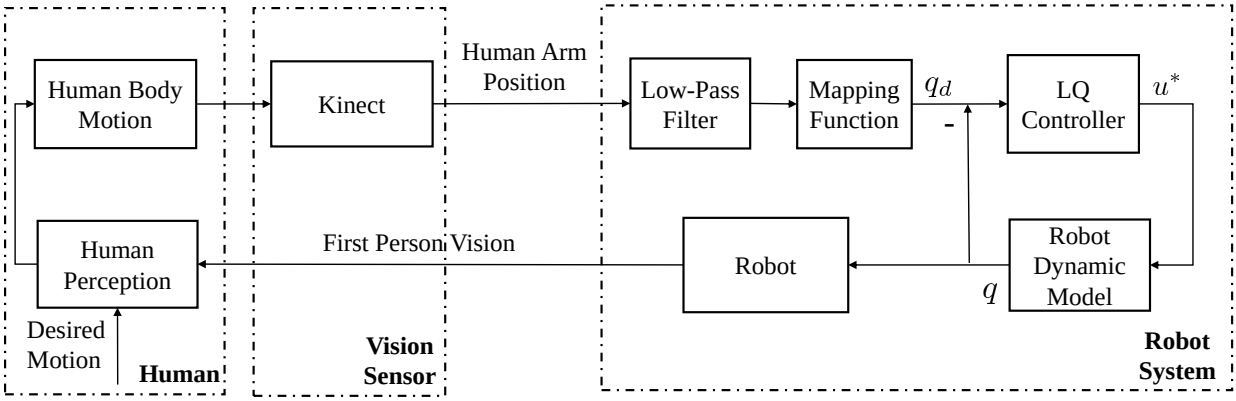


Figure 3.5: Scheme of robot imitation dynamic control on SCARA robot

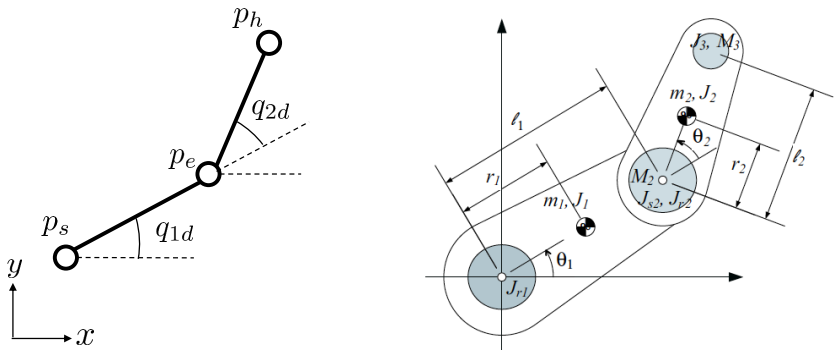


Figure 3.6: Mapping from human arm to SCARA robot

the desired joint position for the first three joints of the robot. Figure 3.4 shows the simulation interface. Fig. 3.4a is the RGB image from Kinect that shows human demonstration, and Fig 3.4b is the 3D visualization of M-16*i*B that illustrating the simulation of robot imitation, which helps the operator correct the robot motion.

Dynamic Control Scheme of Robot Imitation

The dynamic control scheme of robot imitation is similar to the kinematic one that introduced previously. The main difference is that the dynamic model is taken into consideration. In this work, the dynamic controller is applied to a two-link direct-drive SCARA robot. Figure 3.5 shows the scheme of dynamic control on SCARA robot. The additional modules are the LQ controller and a computed torque module to generate the desired control input to the robot dynamic system.

Since SCARA robot works in a two-dimensional plane, the motion of the human arm is projected to $x - y$ plane so that the calculation can be performed in the same dimension. Figure 3.6 shows the mapping from human arm to SCARA robot, where $p_s = (x_s, y_s)$, $p_e = (x_e, y_e)$, and $p_h = (x_h, y_h)$ are the joint positions of the operator's shoulder, elbow, and hand respectively. Hence, the desired joints q_{1d} and q_{2d} are obtained from the human arm geometric relation is formulated as

$$q_{1d} = \tan^{-1}\left(\frac{y_e - y_s}{x_e - x_s}\right), \tag{3.2}$$

$$q_{2d} = \tan^{-1}\left(\frac{y_h - y_e}{x_h - x_e}\right) - q_{1d}. \tag{3.3}$$

In order to smoothly track the human motion, an LQ tracking controller is introduced. Since the human arm is mapped to the robot configuration, we directly exploit the kinematic model in the joint space, where the discrete-time system is given by

$$x(k+1) = Ax(k) + Bv(k) \tag{3.4}$$

$$y(k) = Cx(k) \tag{3.5}$$

where

$$\begin{aligned}
q(k) &= [q_1(k)\ \dot{q_1}(k)\ q_2(k)\ \dot{q_2}(k)]^{\mathrm{T}}, \\
u(k) &= [\ddot{q}_1(k)\ \ddot{q}_2(k)]^{\mathrm{T}}, \\
A &= \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Ts \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
B &= \begin{bmatrix} \frac{1}{2}T_s^2 & T_s & 0 & 0 \\ 0 & 0 & \frac{1}{2}T_s^2 & T_s \end{bmatrix}^{\mathrm{T}}, \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},
\end{aligned}$$

with T_s denoting the sampling time. Note that the linearized time invariant system is controllable and observable, which guarantees closed-loop asymptotic stability. Therefore, we design the controller via solving the infinite horizontal LQ problem [17]. Our goal is to track human motion with reasonably sized control input; thus, the performance index is formulated as

$$J = \frac{1}{2}\sum_{i=0}^{\infty} \left(q(i) - q_d(k)\right)^T Q \left(q(i) - q_d(k)\right) + u(i)^T R u(i), \tag{3.6}$$

where $q_d(k) = [q_{1d}(k)\ 0\ q_{2d}(k)\ 0]^T$ is the desired joint position at time k. The indices i and k represent for LQ horizon and time, respectively. In each time step k, the equilibrium point is

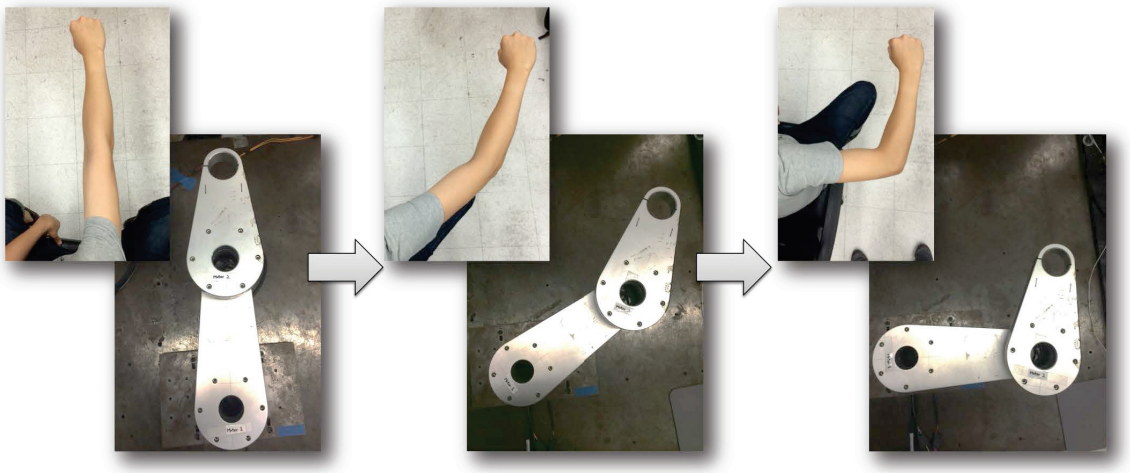


Figure 3.7: Human demonstration and the SCARA robot imitation

assigned at $q_d(k)$, and the system state $q(i)$ asymptotically approaches $q_d(k)$. $Q \succeq 0$ and $R \succ 0$ are the weighting matrices of system outputs and control inputs, respectively. The optimal control input is obtained by solving the algebraic Riccati equation.

$$P_s = C^T Q C + A^T P_s A - A^T P_s B((R + B^T P_s B))^{-1} B P_s A, \tag{3.7}$$

where P_s is the stationary solution of Riccati equation. The optimal control law is given by

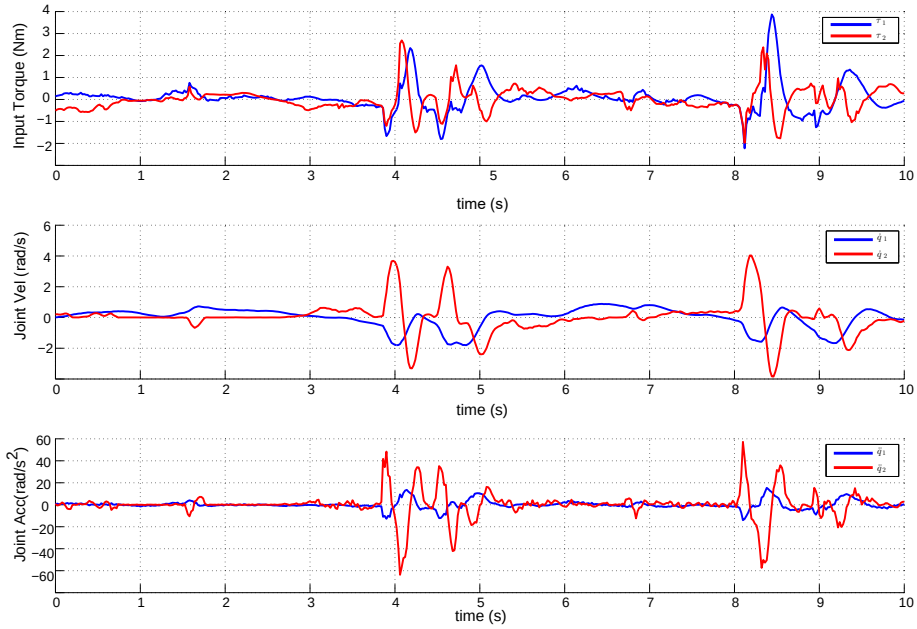
$$u^*(k) = -(R + B^T P_s B)^{-1} B P_s A e(k), \tag{3.8}$$

where $e(k) = q(k) - q_d(k)$ is the tracking error.

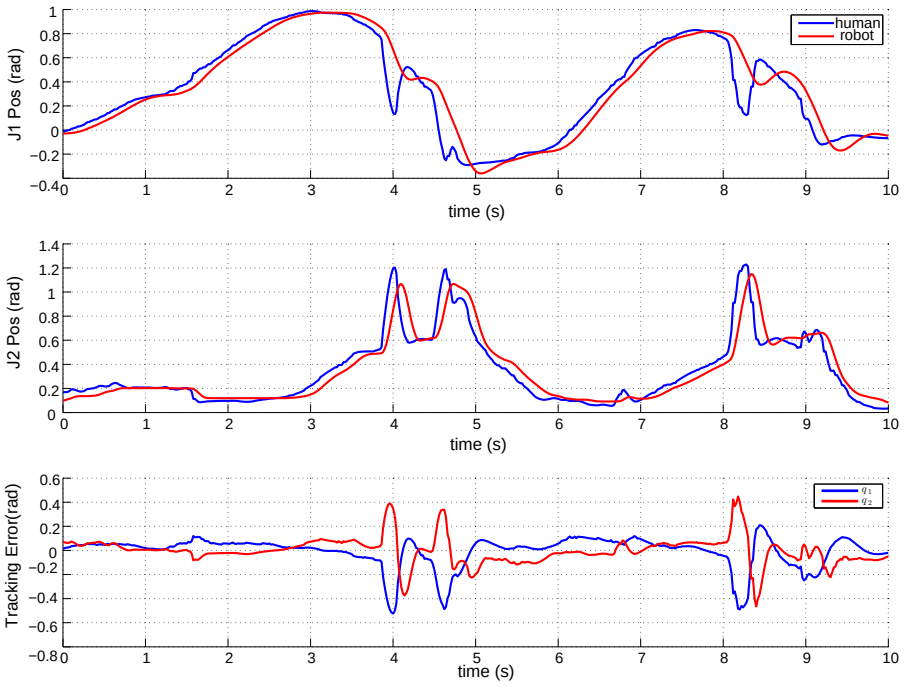
Lastly, the optimal tracking control law is directly plugged into the robot dynamics equation to obtain the desired control torque input.

$$\tau = M(q)u^* + C(q, \dot{q})\dot{q} + G(q). \tag{3.9}$$

To validate the dynamic control scheme of robot imitation, an experiment was performed on a NSK SCARA robot. In the experiment, the motion of the operator's arm was tracked by the Kinect sensor, and the robot tended to follow the corresponding motion as shown in Fig. 3.7. The further details are provided in Fig. 3.8a and Fig. 3.8b. Figure 3.8a shows the input torque, velocity, and acceleration of each robot joint, and Fig. 3.8b shows the joint position of human and robot and the corresponding tracking error. The root mean squares (RMS) tracking errors of joints are 0.1531 rad and 0.1846 rad respectively, which is mainly led from signal latency mentioned previously. If the delay is compensated in 0.1 second, the RMS tracking errors reduce to 0.0103 rad and 0.0305 rad. Although 0.1 second delay is large for the control system, the operator barely detects the latency in experiments.



(a)



(b)

Figure 3.8: The experimental result of SCARA robot imitation motion. (a) The joint position and tracking error between human and robot. (b) The robot joint torque, velocity, and acceleration.

In Fig. 3.8b, there exists a notch in human joint position at $t = 4s$. The reason is that the skeleton estimate algorithm in Kinect is not robust enough, which leads to inaccurate estimates. The problem becomes obvious when part of the operator's arm is occluded by the human body or obstacles. This problem can be solved by using more accurate sensors or optimizing the filter.

3.3 Kinesthetic Teaching

The kinesthetic teaching methodology provides a more intuitive robot programming approach, which becomes more and more popular nowadays. With the kinesthetic teaching, the programmer can directly grasp the robot end-effector or the lead through handle that usually mounted on the robot tool and manually guide the robot through a desired path or successive points to define the path and/or points used for the task.

There are several approaches to implement the kinesthetic teaching. The key idea is to transfer the operator's intention to the robot motion. One approach is to mount a 6D mouse or joystick on the robot end-effector so that the operator can directly guide the robot by maneuvering the device. Another approach is to estimate the force that the operator applied on the robot end-effector and convert to the desired robot motion. For the direct-drive robot or light-weight robot such as UR series [100], the motor current can also be exploited as a signal to estimate the operator's intention. Because it becomes more common that robot end-effector equipped a force sensor, the force-sensor based kinesthetic teaching method will be discussed in this case study.

Compliance Control

In the scheme of the kinesthetic teaching, the operator addressed his/her desired pose by pushing or dragging with the robot end-effector, and the interaction force between robot and human can be regarded as a motion reference signal. Therefore, it falls into the category of compliance control, which achieves force control via motion control without explicit closure of a force feedback loop.

In [89], a compliance control scheme is proposed to design the robot end-effector stiffness along degrees of freedom in Cartesian space by modifying its position gain. Considering a general spring with six degrees of freedom, its action could be described by

$$F_E = K_p \delta x, \tag{3.10}$$

where $F_E = [f_e^T,\ \tau_e^T] \in \mathbb{R}^6$ is a wrench vector that includes the external force and moment at the robot end-effector, $\delta x = x_d - x_e$ is the displacement of the robot end-effector from the desired position, and $K_p \in \mathbb{R}^{6\times 6}$ is a diagonal matrix with three linear stiffness followed by three torsional stiffness on the diagonal entries. Recalling the definition of robot Jacobian is $\delta x = J(q)\delta q$, the external force is thus given by

$$F_E = K_p J(q) \delta q. \tag{3.11}$$

For a robot with six degrees of freedom, K_p describes the desired stiffness of the robot end-effector in Cartesian space. Through use of the Jacobian, the Cartesian stiffness is transformed to a joint-space stiffness.

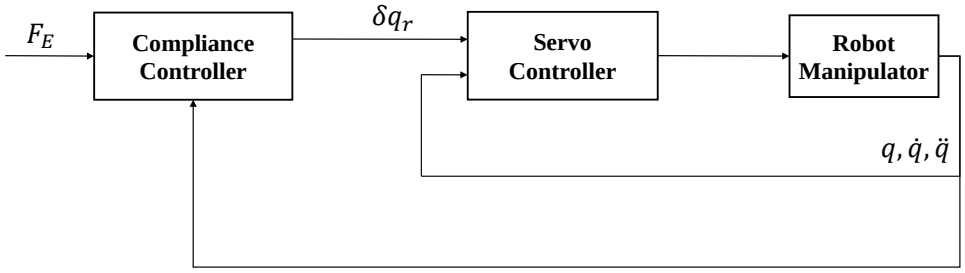


Figure 3.9: The Scheme of the robot compliance control

By taking the inverse of K_p in (3.10), the desired end-effector displacement in Cartesian space is given by

$$\delta x = CF_E, \tag{3.12}$$

where $C = K_p^{-1}$ is the compliance matrix that describes the desired behavior of the robot end-effector by giving an external force.

In kinesthetic teaching, the operator continuously applies a force/moment to the robot end-effector, and the corresponding displacement is considered as a reference signal for the robot servo system, which is given by

$$\delta q_r = J(q)^{\dagger} C F_E, \tag{3.13}$$

where $J(q)^{\dagger} = J^T(JJ^T)^{-1}$ is the pseudo inverse of the robot Jacobian. Figure 3.9 summarizes the scheme of robot compliance control, where the compliance controller is given by (3.13) and the servo controller is similar to the control structure introduced in Chapter 2.

External Force Sensor Identification

Although most force sensors have been calibrated by their manufacturers, there are several factors that affect the force measurement in the actual robot operations. For example, the robot end-effector induces a force offset by its own weight, and the displacement from the center of mass of the payload to the sensor frame generates a torsional offset. Besides the effect from the payload, mounting the sensor on the robot would also introduce a sensor offset to the force measurement. Since the compliance control relies on the force measurement, those external offsets can not be neglected. Hence, an external force sensor identification is essential to assure the correct measurement.

Note that the force measurement by a six-axis transducer can be written as a wrench vector, i.e. $F = [f^T, \tau^T]^T \in \mathbb{R}^6$. Considering the aforementioned effects and the measurement noise, the force measurement is given by

$$F = F_E + F_P + F_B + F_N, \tag{3.14}$$

where F_P is the payload wrench vector, F_B is the sensor bias generated by the assembly force, and the F_N represents the measurement noise, where $F_\bullet$ is a wrench vector that includes the corresponding force and the torque denoted as $f_\bullet$ and $\tau_\bullet$ respectively.

When there is no external force applied on the robot end-effector, the force measurement in (3.14) only consists the latter three terms. Since the measurement noise results from high frequency signal, a law-pass filter can be designed to filter out the noise effect. The remainder includes the effect from the payload weight and the sensor bias due to the assembly. The identification process includes two parts: the force identification and the torque identification.

Force Identification

For noise-free and no external force case, the translational force measurement in the sensor frame is given by

$$f = R_{ee}^T f_P + f_B, \tag{3.15}$$

where the R_{ee} is the orientation of the end-effector. For the case that the sensor frame and the robot end-effector share the same orientation, $R_{ee}^T = R_{ee}^{-1}$ can project f_P from the robot world frame to the sensor frame. Rewriting (3.15), we can get

$$\underbrace{\begin{bmatrix} R_{ee}^T & I_3 \end{bmatrix}}_{A_f} \begin{bmatrix} f_P \\ f_B \end{bmatrix} = \underbrace{\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}}_{b_f}, \tag{3.16}$$

where I_3 is a 3×3 identity matrix. Then the estimate of the payload weight and the sensor bias can be solved by least square, i.e.

$$\begin{bmatrix} \hat{f}_P \\ \hat{f}_B \end{bmatrix} = A_f^\dagger b_f, \tag{3.17}$$

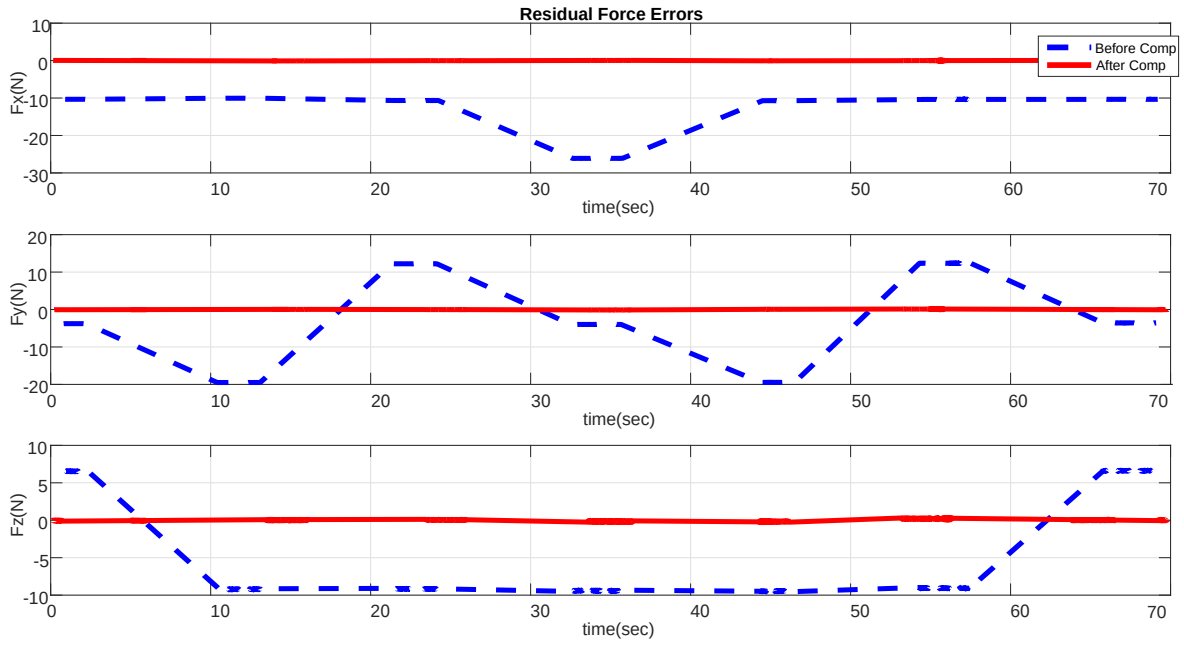
where A_f is the pseudo-inverse of A_f.

Torque Identification

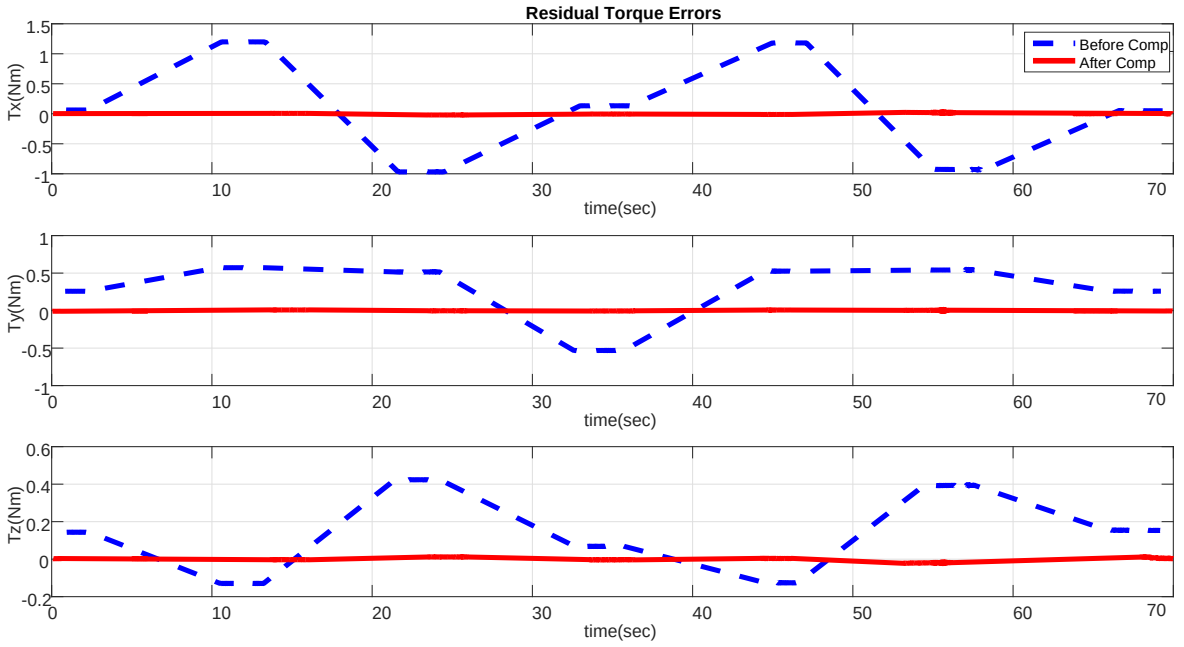
For noise-free and no external force case, the torsional moment measurement in the sensor frame is given by

$$\tau = r_P \times f_P + \tau_B, \tag{3.18}$$

where $r_P = [r_{P,x}, r_{P,y}, r_{P,z}]^T$ is the center of mass position in the sensor frame. Substituting the payload weight by the estimate $\hat{f}_P = [\hat{f}_{P,x}, \hat{f}_{P,y}, \hat{f}_{P,z}]^T$ retrieved from the force identification,



(a)



(b)

Figure 3.10: The force/torque identification result. (a) The residual force error before/after identification. (b) The residual torque error before/after identification.

(3.18) can be rewritten as

$$\tau = \underbrace{\begin{bmatrix} 0 & \hat{f}_{P,z} & -\hat{f}_{P,y} \\ -\hat{f}_{P,z} & 0 & \hat{f}_{P,x} \\ \hat{f}_{P,y} & -\hat{f}_{P,x} & 0 \end{bmatrix}}_{[\hat{f}_P\times]} \begin{bmatrix} r_{P,x} \\ r_{P,y} \\ r_{P,z} \end{bmatrix} + \tau_B, \tag{3.19}$$

Similar to the aforementioned process, the center of mass location of the payload and the torque bias can be estimated by the least square solution of

$$\underbrace{\begin{bmatrix} [\hat{f}_P\times] & I_3 \end{bmatrix}}_{A_\tau} \begin{bmatrix} r_P \\ \tau_B \end{bmatrix} = \underbrace{\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}}_{b_\tau}. \tag{3.20}$$

The force/torque identification is performed by collecting the force sensor measurement in various robot configurations. The result of identification is as shown in Fig. 3.10, where the red solid line and the blue dash line are the residual error before and after the offset compensation. Since there is no other external force acting on end-effector, the ideal case of the force sensor reading should be zero and invariant to the pose of the end-effector. However, due to the aforementioned effect, the sensor reading (the blue dash line in Fig. 3.10) changes with respect to the end-effector pose. After the identification process, both the payload weight and the sensor bias effect are compensated by the estimated parameters, and thus the residual error of sensor reading on the same trajectory is close to zero.

With the external force sensor identification, the force sensor can measure the external force more accurately by eliminating other force effects. Hence, while the human operator using the compliance control to perform the kinesthetic teaching, the remainder of the force sensor reading can be used to exploiting the operator's intention.

Kinesthetic Teaching Path Reproduction

Kinesthetic teaching is intuitive for human operators to design a path to target pose; however, it is not data-efficient to record/reproduce the whole guided path. Moreover, the operator might induce some redundant motion while guiding the robot. One alternative approach is to record the key points for the task and then interpolate the trajectory between these points.

As shown in Fig. 3.11, the blue line is the recorded path guided by a human operator and the orange line is the path that generated by a Cartesian space trajectory generator. The Cartesian space trajectory generator produces line segments between key points and interpolates a smooth trajectory by cubic or higher order polynomials, where the joint space velocity reference is provided in Fig. 3.12.

In this case study, a basic kinesthetic teaching is implemented through compliance control. Chapter 4 will further discuss the kinesthetic teaching with real-time collision avoidance as well as its experimental validations.

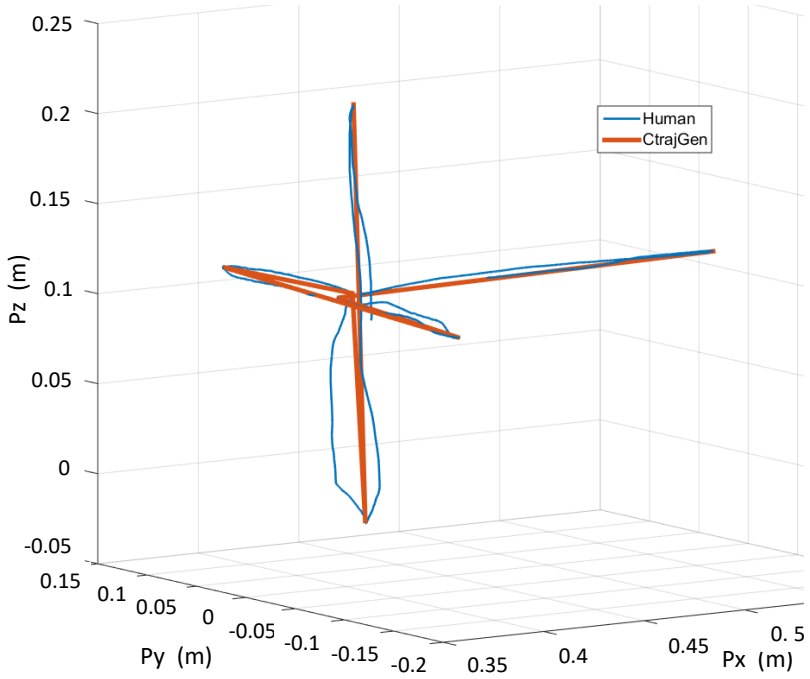


Figure 3.11: The kinesthetic teaching path and the robot reproduced path.

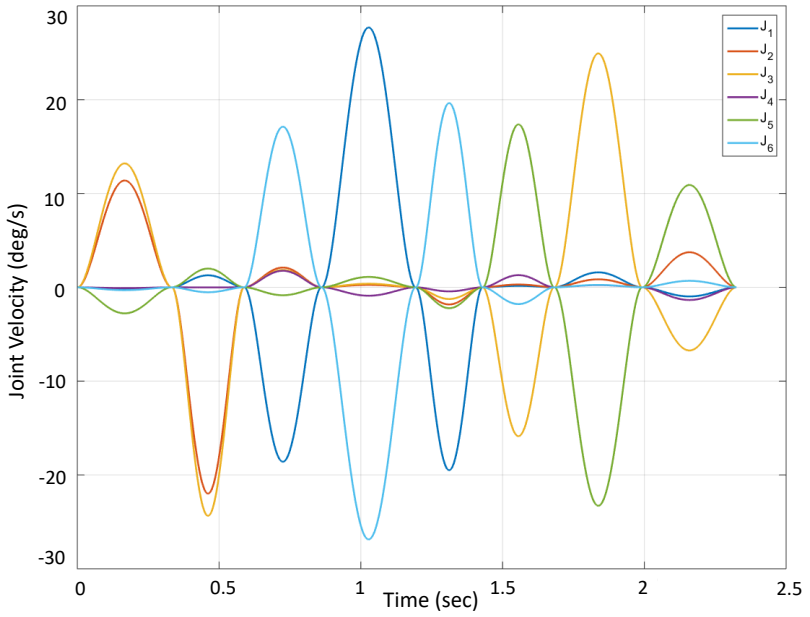


Figure 3.12: The joint velocity reference generated by trajectory generator.

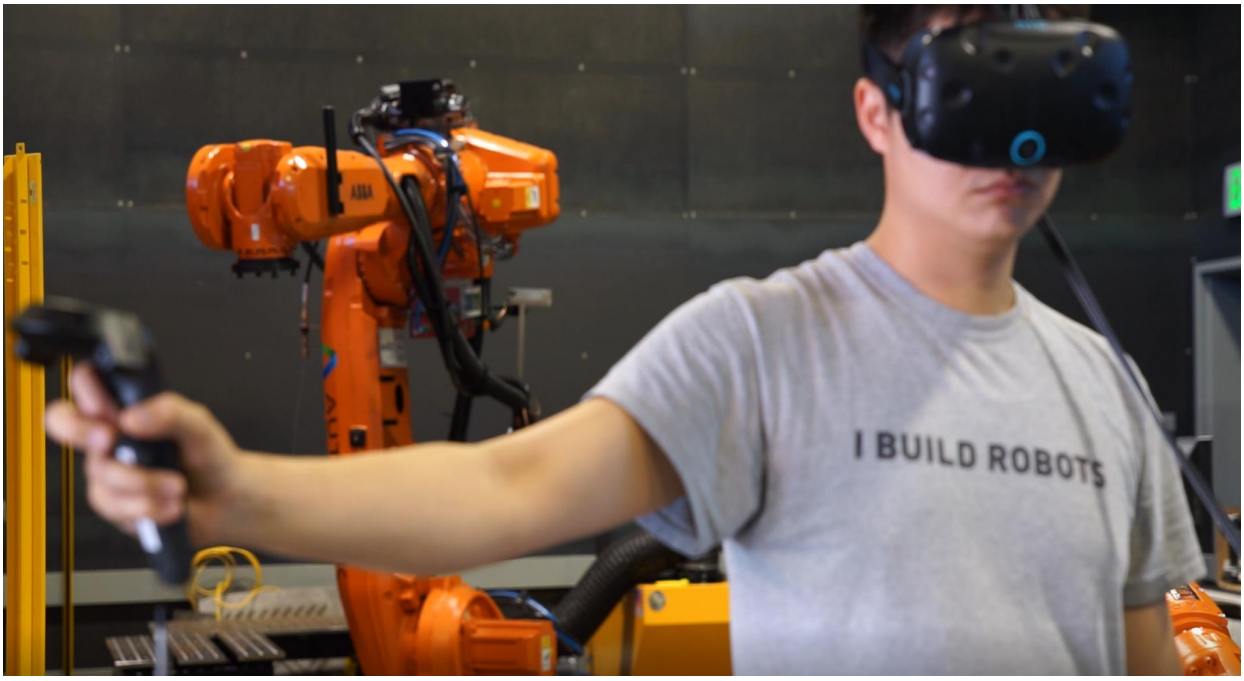


Figure 3.13: The operator teleoperate the robot through virtual reality.

3.4 Immersive Teleoperation

In Immersive teleoperation scenarios, the human operator is performs the task through using the robot's own sensors and actuators. Although the flexibility is limited, it resolves the problem of the discrepancies in how robots and humans are embodied, i.e. the perceptual and physical difference between human and robot. The teleoperation can be achieved by using joysticks or other remote control devices such as haptic devices. The latter devices provide the force feedback to the operator that allows operators to teach tasks that require force information, while the joysticks can only provide kinematic information such as position and velocity.

Over the past few years, the technology of virtual reality (VR) has grown rapidly due to the significant progress in computer graphics. VR is a computer-generated scenario that simulates experience through senses and perception. The immersive environment could be either a simulation of the real world or the imaginary world. Hence, it has been applied to various applications such as entertainment, medical, and aerospace. Using VR as the platform to interact with robots is another novel application that provides a safe interaction experience at a low cost.

This case study investigates the implementation of immersive teleoperation through VR, which is mainly based on the result of the Autodesk collaborative project - Human Interaction with Robot through Virtual Reality (HIROVR) [6].

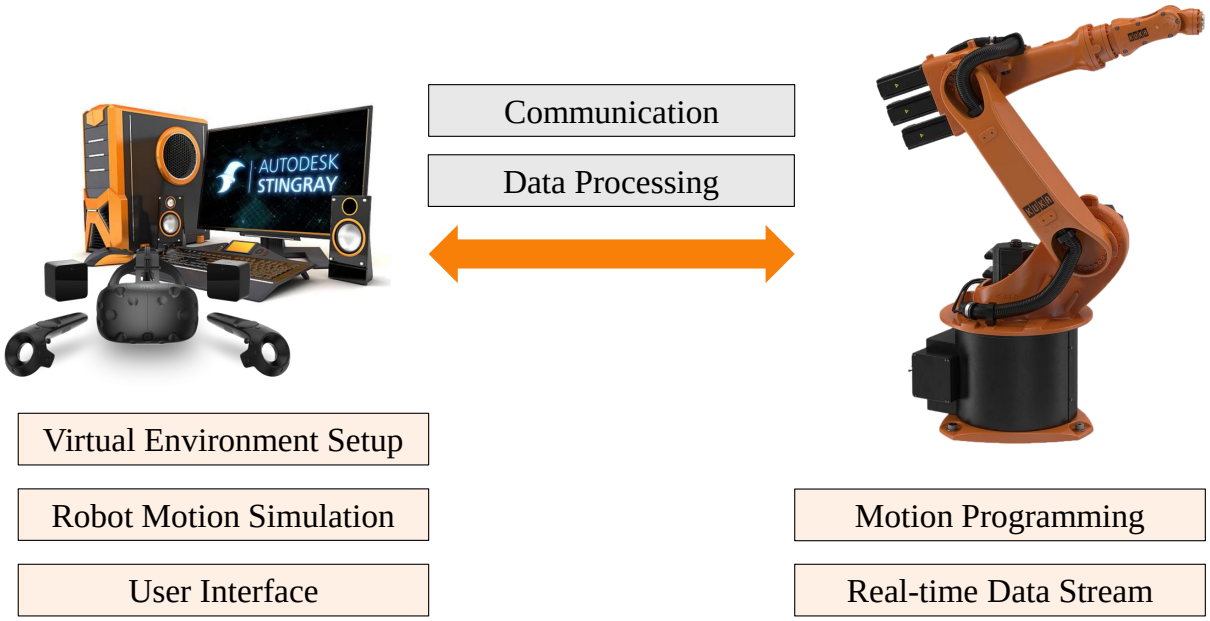


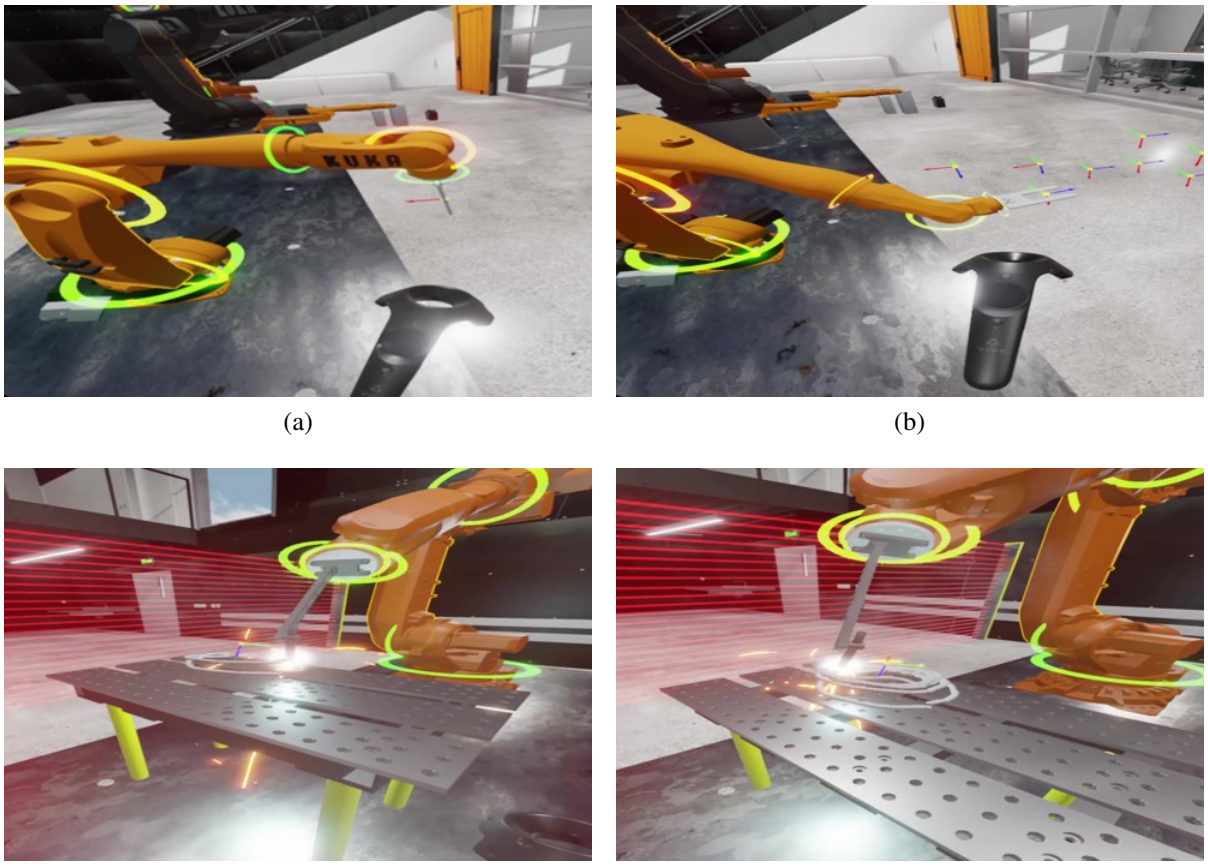
Figure 3.14: The system framework of robot teleoperation through virtual reality.

Scheme of Robot Teleoperation through Virtual Reality

Figure. 3.13 shows how the human operator teleoperates a robot through VR. The human operator wears a VR headset to immerse himself in a digital factory, which provides an identical environment to the actual workspace. By manipulating the VR controller, the human operator directly controls the virtual robot as well as the actual robot to follow his motion.

As shown in Fig. 3.14, the system framework of robot teleoperation through VR includes three parts: VR system, communication module, and robot controller. In this work, HTC Vive VR system [45] operates on Stingray, a game engine produced by Autodesk [7]. The VR program constructs the virtual environment and simulates the robot motion, including forward kinematics and inverse kinematics. In addition, the user interface is built to allow the VR operator to interact with the virtual environment. The robot controller is connected to the VR system through EtherCAT. In order to transfer the correct data protocol for the robot system, a communication socket is embedded to transmit the desired joint positions or the desired end-effector pose in Cartesian space to the robot. The robot controller generates the real-time motion command through the online trajectory generator. In this work, a small size industrial robot, KUKA KR 5 Arc, and a large size industrial robot, ABB IRB 6700, serve as the teleoperated robot subjects.

The work flow of robot teleoperation through VR is as follows. The human operator and the robot share the same coordinate reference in the virtual environment. The pose of the operator's hand that tracked by the VR controller is regarded as the target pose of the robot end-effector. The VR program calculates the corresponding joint position by an analytic inverse kinematic module



(a) (b)

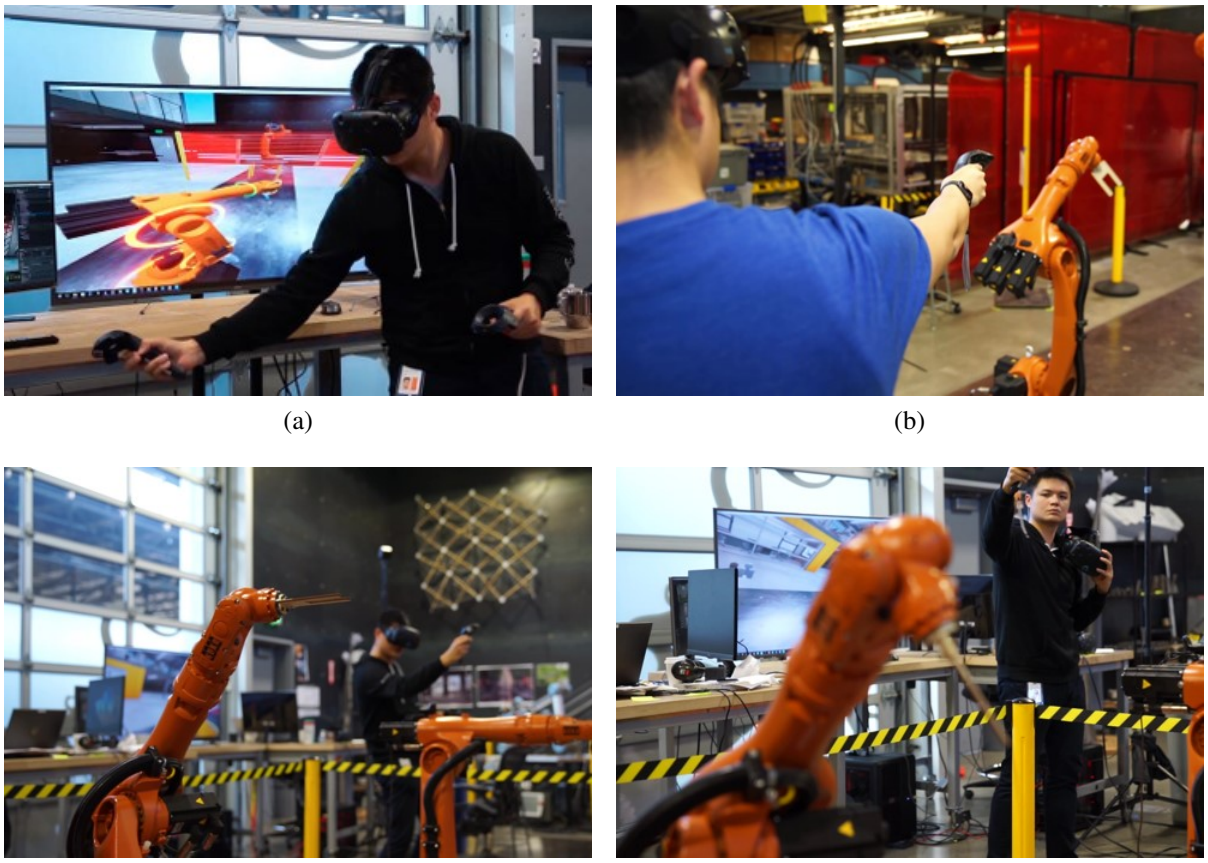
(c) (d)

Figure 3.15: The virtual reality scene that the human operator sees through the VR headset.

and visualizes the robot configuration in real-time. In the same time, the robot controller receives the motion data streaming from the VR side and generates the motion accordingly.

Implementation of Robot Teleoperation through Virtual Reality

Figure 3.15 reveals several scenes that the human operator sees through the VR headset. In Fig. 3.15a and Fig. 3.15b, the operator performs a path programming through VR. The operator guides the virtual robot to several waypoints and presses the trigger of the VR controller to record these waypoints, which are visualized as the coordinates in Fig. 3.15b. The robot would repeat exactly the same path if the operator pressed the replay button on the VR controller. In Fig. 3.15c and Fig. 3.15d, the operator remotely controls a robot to conduct a welding task. With the help of *telepresence*, the operator could keep a safe distance away from the dangerous welding site but closely observe the detail of welding process through VR screen, e.g., the feed rate of solder with respect to the motion of robot end-effector.



(a) (b)

(c) (d)

Figure 3.16: The experiment footages of the robot teleoperation through virtual reality.

Figure 3.16 shows the footages taken in the experiment of robot teleoperation through VR. In Fig. 3.16a, the human operator using VR system, while the television monitor (behind the human operator) displays the current scene in the VR headset. It allows other people to observe the interaction occurred in the virtual environment. The remaining three subfigures show the robot motion synchronized with the motion of operator's hand. The operator can even take off the VR headset to see the real robot behave the same movement as his arm as shown in Fig. 3.16d.

The pipeline of this VR teleoperation is very natural, because humans can simply move their hand and the target pose for the robot end-effector to follow in the same way, making it intuitive for even first-time users to accomplish the robot programming. Moreover, the kinematic configuration difference is reduced to minimal in this setting.

3.5 Chapter Summary

This chapter discussed the programming interface for demonstration. Unlike conventional programming methods that required tremendous programming efforts to implement a task, programming by demonstration (PbD) allows human operator teaching robots by more intuitive ways. This chapter investigated three main trends of robot programming interfaces: robot imitation from human motion, kinesthetic teaching, and immersive teleopeation.

In the case study of robot imitation from human motion, the human gesture was captured and mapped to the robot configuration by a Kinect sensor. A simulation and an experimental validation were performed on a FANUC M-16*i*B simulator and a NSK SCARA robot system, respectively. In the case study of kinesthetic teaching, the compliance control was introduced and implemented on FANUC LR Mate 200*i*D robots, where the guided path could be recorded and reproduced. In the case study of immersive teleoperation, the framework of robot teleoperation through virtual reality (VR) was presented. Combining the VR system and the robot controller with a real-time data streaming module, the robot motion could be synchronized with the motion of the operator's hand. The experiment was performed on both KUKA KR 5 Arc (a small size industrial robot) and ABB IRB 6700 (a large size industrial robot).

Chapter 4

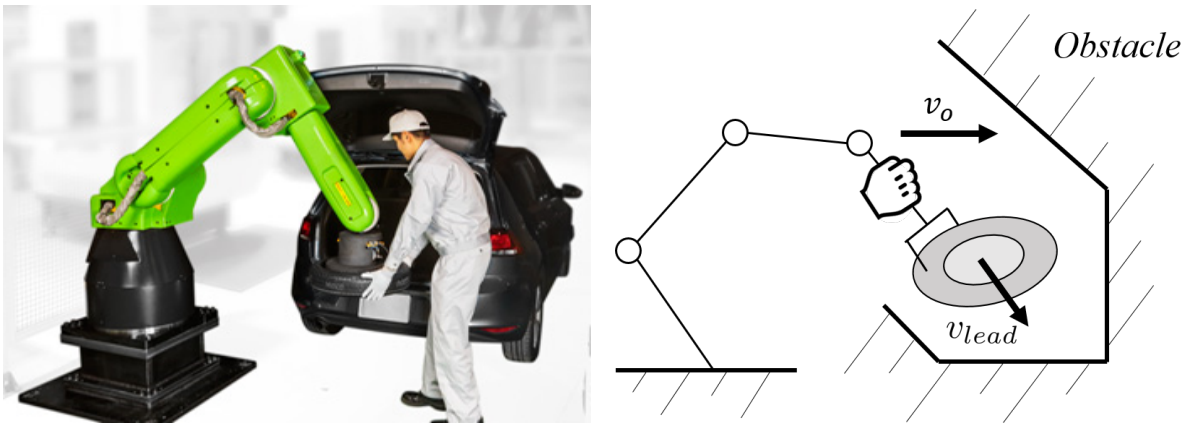
Human Guidance Programming with Collision Avoidance

4.1 Introduction

Human-robot collaboration is one of the most important application in physical human-robot interaction (pHRI). It has great potential to improve production efficiency. Human laborers can often do flexible and intelligent tasks, whereas robots excel in repetitive and assistive tasks. However, the risk increases when humans and robots work together. Not only it is potentially hazardous that the robot can harm a human, but it is also that the robot can damage its environment through human mistake.

A human-robot collaboration scenario is given in Fig. 4.1a to highlight the safety issue. The robot is handling a heavy tire, while the operator is guiding the end-effector to put the tire into the car body. The operator might focus only on the task of placing the object at the right location without being aware of the entire robot configuration. Therefore, the robot arm might collide with the rear door. Fig. 4.1b is a simplified drawing to show how this type of collision might happen. v_{lead} is the lead through command from the operator's guiding force, which leads the end-effector towards the bottom of the trunk. Meanwhile, there is a point on the robot body, which is approaching the obstacle with a velocity v_o. A collision would occur if the operator kept on guiding the robot at the current direction.

With regards to collision avoidance, several methods have been proposed. Brock and Khatib [18] presents the framework of robot motion modification in the dynamical environment. Khatib [51] introduces the potential field method to prevent collision in manipulators and mobile robots. De Santis [31] and Täubig et. al. [98, 34] prevent a dual-arm robot from self-collision by adding a reactive force at the minimum distance pair. Flacco et al. [40] develops an algorithm to estimate the obstacle distance and its velocity by the depth sensor. Although these works provide very good results in the real-time applications, they do not deal with the end-effector task performance. However, the behavior of the robot end-effector should be cautiously considered in pHRI applications.



(a) (b)

Figure 4.1: The scenario of human-robot collaboration. (a) The human laborer is guiding the robot to place a tire. [37] (b) The collision might happen if the labor does not notice the distance of the robot body and the obstacle.

Maciejewski and Klein [69] illustrate the benefit of redundancy. The redundant degrees of freedom give the robot the ability to track the desired trajectory and avoid obstacles at the same time. Petrič and Lajpah [81] formulate the redundant robot command for multiple tasks and analyze its stability. However, these redundant degree of freedom approaches are not applicable in non-redundant robots such as CR 35-*i*A in Fig. 4.1a. Although traditional collision avoidance methods will move the robot in a direction away from the obstacle, these methods often fail to take advantage of a human's intuition in collision-avoidance path planning.

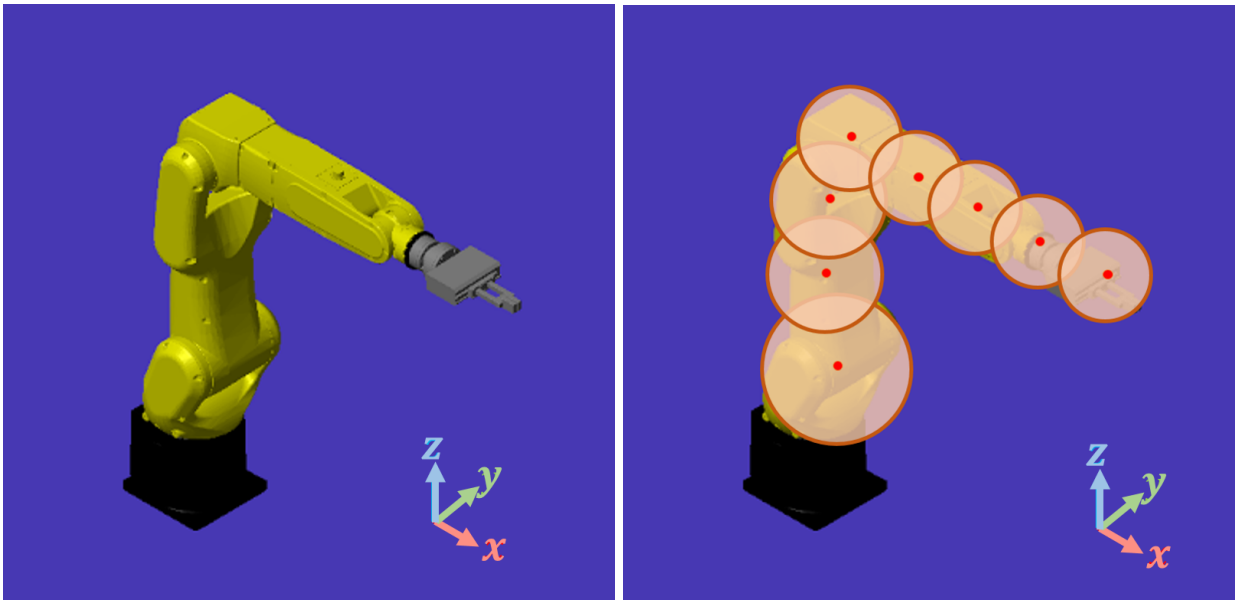
To deal with the problem mentioned above, this chapter introduces how to implement both robot collision avoidance and human lead through following on a 6-DoF robot simultaneously. Then, a novel collision constraint relaxation is proposed to give the robot more flexibility to deal with obstacles, which not only preserves the robot safety but also improves user's flexibility.

The remainder of this chapter is organized as follows. Section 4.2 briefly introduces the relevant works such as collision avoidance by a repulsive action and a null space projection for simultaneous collision avoidance and end-effector positioning. Section 4.3 presents the collision avoidance algorithm for a 6-DoF robot. The experimental verification is presented in Section 4.4. Lastly, the conclusion is given in Section 4.5.

4.2 Related Works

Repulsive Action

For the collision avoidance, it is necessary to calculate the distance between the robot and the obstacle. In order to accelerate the distance calculation, it is necessary to simplify the robot links into a simple geometric model such as ellipsoids [88] and cylinders [98]. In [9], the manipulator



(a) (b)

Figure 4.2: The robot geometry approximation. (a) The original robot CAD model. (b) The robot model is approximated by several spheres.

is modeled by several spheres. The advantages of the spherical approximation are the geometrical simplicity and the computational efficiency.

The sphere-based geometry approximation is illustrated in Fig. 4.2. The red points are the control points of the robot body. The orange balls are the spheres that centered at the control points. The distance between the robot and the obstacle is reformulated as the distance between the ball to the obstacle surface. Then, the repulsive action is applied to the control point that has the shortest distance to the obstacle.

The repulsive action in traditional collision avoidance methods is adding a repulsive vector v_{rep} at the control point. v_{rep} is defined as an artificial force assigned to push the robot away from the obstacle. In [51], this artificial force is obtained by the derivative of a designed potential function. [40] uses a sigmoid function to present a smooth repulsive vector. These repulsive actions have the same property: the closer the distance between robot and obstacle, the larger the magnitude of v_{rep}.

Null Space Projection

Although the repulsive action prevents collisions, it may also affect the robot end-effector position. [69] shows the possibility of both preventing of collision and desired positioning of the end-effector on a redundant robot. To be more specific, the motion of a redundant robot can be decomposed into a least square solution of Cartesian space motion and a homogeneous solution created by the

null space projector,

$$\dot{q} = J^{\dagger}\dot{x} + Nz \tag{4.1}$$

where $\dot{q}$ and $\dot{x}$ are the joint and Cartesian velocity, respectively; $J^{\dagger}$ is the pseudo inverse of the robot Jacobian, J, i.e. $J^{\dagger} = J^T(JJ^T)^{-1}$, $N = I - J^{\dagger}J$ is the null space projector, and z is an arbitrary vector. [81] gives a more general form for a redundant robot to satisfy multiple tasks requirement,

$$\dot{q} = J_1^{\dagger}\dot{x}_1 + \sum_{i=2}^{K} N_{i-1}J_i^{\dagger}\dot{x}_i \tag{4.2}$$

where $N_i = I - J_i^{\dagger}J_i$, and $i = 1, \cdots K$ is the task priority order. In this formulation, the lower index number has higher task priority. The lower priority task is satisfied by adding robot velocity in the robot Jacobian's null space. In other words, redundant robots can leverage the fact that there are extra degrees of freedom to achieve the lower priority tasks while ensuring higher priority tasks are still satisfied. Hence, the robot velocity command for a collision avoidance and an end-effector tracking can be formulated as

$$\dot{q} = J_o^{\dagger}\dot{x}_o + N_o J_e^{\dagger}\dot{x}_e \tag{4.3}$$

where J_o is the Jacobian at the control point, $N_o = I - J_o^{\dagger}J_o$ is the null space of J_o, J_e is the Jacobian at the end-effector, $\dot{x}_o$ is the desired velocity for collision avoidance at control point, and $\dot{x}_e$ is the desired end-effector velocity.

4.3 Algorithm

Jacobian Decomposition

Because of redundancy, the null space projector is applicable to a redundant robot and can perform the desired task as well as collision avoidance at the same time. However, the Jacobian of a 6-DoF robot is full rank in nonsingular configuration. There is no additional degree of freedom to directly apply the original null space projection method.

Note that if a robot is to avoid the obstacle at the control point, the repulsive motion, $\dot{x}_{\text{rep}}$, is given by

$$\dot{x}_{\text{rep}} = J_o(q)\dot{q}_{\text{rep}} \tag{4.4}$$

where $\dot{q}_{\text{rep}} \in \mathbb{R}^n$ and $J_o \in \mathbb{R}^{6\times n}$ are the joint velocity and Jacobian at the control point respectively, and n is the degrees of freedom at the robot control point. Notice that the repulsive vector that is introduced previously only deals with translation, not rotation. i.e. $v_{\text{rep}} \in \mathbb{R}^3$. Thus, (4.4) can be written as

$$\begin{bmatrix} v_{\text{rep}} \\ 0 \end{bmatrix} = \begin{bmatrix} J_{o,1:3} \\ J_{o,4:6} \end{bmatrix} \dot{q}_{\text{rep}} \tag{4.5}$$

Because only the first three rows of Jacobian projects the repulsive vector to joint space, a decomposed Jacobian is defined as

$$J_{\text{rep}} := J_{o,1:3} \in \mathbb{R}^{3\times n} \tag{4.6}$$

It gives $v_{\text{rep}} = J_{\text{rep}}\dot{q}_{\text{rep}}$. The physical meaning of J_{rep} is the mapping of the joint velocity at the control point to the repulsive vector in Cartesian space. For $n > 3$, J_{rep} has a null space of dimension $n-3$. For example, if $n = 4$, the rank of J_{rep} is 3, then the null space has one dimension and the null space projection method can be applied. Because there exists at least one redundant degree of freedom, the robot arm can achieve multiple tasks as described in (4.3). i.e.

$$\dot{q}_{\text{rep}} = J^{\dagger}_{\text{rep}} v_{\text{rep}} + N_{\text{rep}} z \tag{4.7}$$

where $N_{\text{rep}} = I - J^{\dagger}_{\text{rep}} J_{\text{rep}}$ is the null space projector of J_{rep}, $z \in \mathbb{R}^n$ is the weighted coefficient of the null space vectors. Since $J_{\text{rep}} N_{\text{rep}} = 0$, the joint velocity at the control point is deterministic. Even if $n = 3$ and N_{rep} is an empty set, (4.7) still gives a proper repulsive joint velocity by

$$\dot{q}_{\text{rep}} = J^{\dagger}_{\text{rep}} v_{\text{rep}} \tag{4.8}$$

The case with $n < 3$ is not discussed here. In such cases, since the robot degree of freedom is less than the dimension of the repulsive vector, there exists at least one direction that the robot would not be able to avoid the collision.

Collision Avoidance Optimization

To incorporate both the human guidance command and the collision avoidance, the problem can be formulated as an optimization problem,

$$\min_{\dot{q}_{\text{cmd}}} \quad \|\dot{x}_{\text{lead}} - J(q)\dot{q}_{\text{cmd}}\|_2^2 \tag{4.9a}$$

$$s.t. \quad J_{\text{rep}}\dot{q}_{\text{cmd,n}} = v_{\text{rep}} \tag{4.9b}$$

where $\dot{x}_{\text{lead}} \in \mathbb{R}^6$ is the human lead through motion command, $\dot{q}_{\text{cmd}} \in \mathbb{R}^6$ is the 6-DoF robot joint velocity command, and $\dot{q}_{\text{cmd,n}}$ is the first n-th joint velocity command. The cost function is the velocity difference between human's desired motion and the robot actual motion command. The constraint indicates that the robot command should give an equivalent velocity to the repulsive vector at the control point so as to avoid collision.

Let $\dot{q}_{\text{cmd,n}} = \dot{q}_{\text{rep}}$ and plug (4.7) into (4.9b) The original problem can be reformulated as a standard least square problem, where its optimizer is given by

$$\begin{bmatrix} z^* \\ \dot{q}_c^* \end{bmatrix} = -\begin{bmatrix} J_1 N_{\text{rep}} & J_2 \end{bmatrix}^{\dagger} \left(\dot{x}_{\text{lead}} - J_1 J^{\dagger}_{\text{rep}} v_{\text{rep}} \right) \tag{4.10}$$

where $z^* \in \mathbb{R}^n$ is the optimal weighting coefficient of the null space vectors, $\dot{q}_c^* \in \mathbb{R}^{6-n}$ is the optimal complementary joint velocity command, J_1 is the first n columns of J, and J_2 is the rest of J. This analytical form solution makes it feasible for real-time applications.

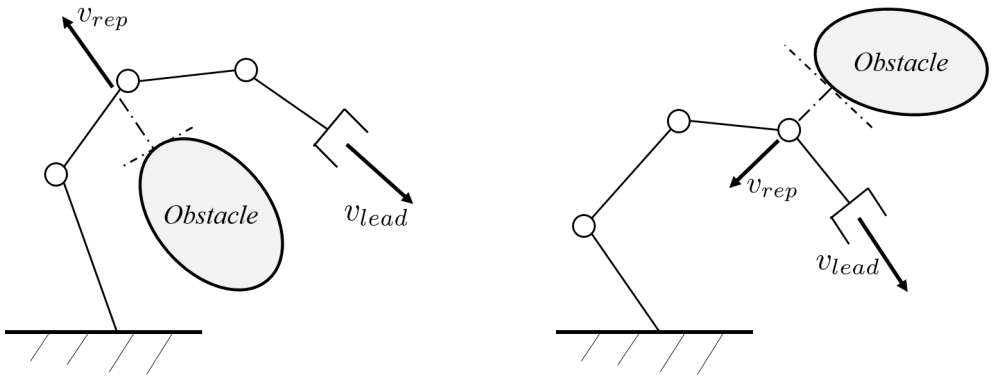


Figure 4.3: The collision avoidance scenarios. (a) In order to prevent the collision avoidance, the repulsive vector v_{rep} is necessary. (b) The repulsive vector v_{rep} is not necessary in this case, since v_{lead} can guide the robot to escape the obstacle.

Note that (4.9b) is a strong restriction that fixes the robot velocity. Thus, the solution (4.10) makes the robot move in the direction of the repulsive vector at the control point, which is always orthogonal to the obstacle surface. However, we may ask whether this is necessary in all cases when the human is guiding the robot.

Fig. 4.3 shows two different scenarios of robot collision avoidance while the human is guiding the robot. In Fig. 4.3a, the repulsive vector is critical in preventing the robot from colliding into the obstacle. Because the lead through command leads the end-effector going down, the robot body might approach the obstacle underneath.

In contrast, the repulsive vector in Fig. 4.3b is not necessary, because the human lead through command provides enough downward velocity component to move the robot away from the obstacle above the robot body. Moreover, the velocity component from the repulsive vector would counteract the forward velocity component of the lead through command, which in turn would make the operator feel resistance in guiding the robot. This example reflects the limitation of traditional repulsive action methods. Although the repulsive vector is the fastest way to prevent obstacle collision, it does not consider user's comfort and is inflexible in the context of human-robot collaboration.

Constraint Relaxation

A novel approach to formulate the collision avoidance is illustrated in Fig. 4.4, where $u_{\text{rep}} = \frac{v_{\text{rep}}}{\|v_{\text{rep}}\|}$ is the unit repulsive vector. The shaded area is a half ball with its center located at the control point, P. It is also a safety set, where any velocity within this region would not move the robot arm towards the obstacle.

v_1 and v_2 are two different example velocities at the same control point P. The inner product of v_1 and u_{rep} is positive. Thus, v_1 does not hit the boundary of the half ball. On the other hand, the

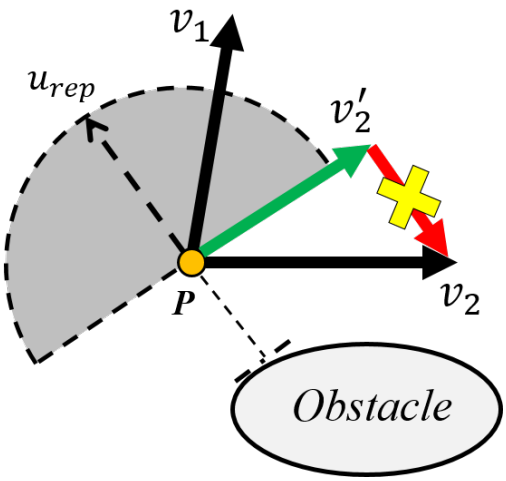


Figure 4.4: The velocity constraint at the control point, P, is a half unit ball to bound any velocity that approaches to the obstacle.

inner product of v_2 and u_{rep} is negative. It means v_2 that has a velocity component that counteracts u_{rep}. A feasible velocity is v_2', which stays within the safety set and is tangent to the obstacle surface.

Hence, the original equality constraint can be replaced by an inequality constraint, and the original problem can be reformulated as,

$$\min_{\dot{q}_{\text{cmd}}} \quad \|\dot{x}_{\text{lead}} - J(q)\dot{q}_{\text{cmd}}\|_2^2 \tag{4.11a}$$

$$s.t \quad u_{\text{rep}}^T J_{\text{rep}} \dot{q}_{\text{cmd,n}} \geq 0 \tag{4.11b}$$

where (4.11b) implies that the robot could not have any velocity component that opposes the repulsive vector at the control point.

Since this is a standard quadratic programming problem, it has a global optimal solution. Moreover, the explicit solution can be found because the dimension of the constraint is small. In fact, the dimension of the constraint is equal to the number of obstacles in the environment. For a single obstacle case, the problem can be simplified by checking the active set of collision constraints.

Velocity Composition

Although (4.11) prescribes a more flexible velocity command to the robot compared to traditional collision avoidance algorithms, the constraint would still limit the robot mobility if the obstacle is far away. Thus, it is necessary to design a switch mechanism to turn on the collision avoidance optimization when the obstacle is within a certain distance.

Note that the discontinuous velocity command would cause an uncomfortable user experience. Instead of using a distance threshold as a trigger condition, the velocity command here is generated

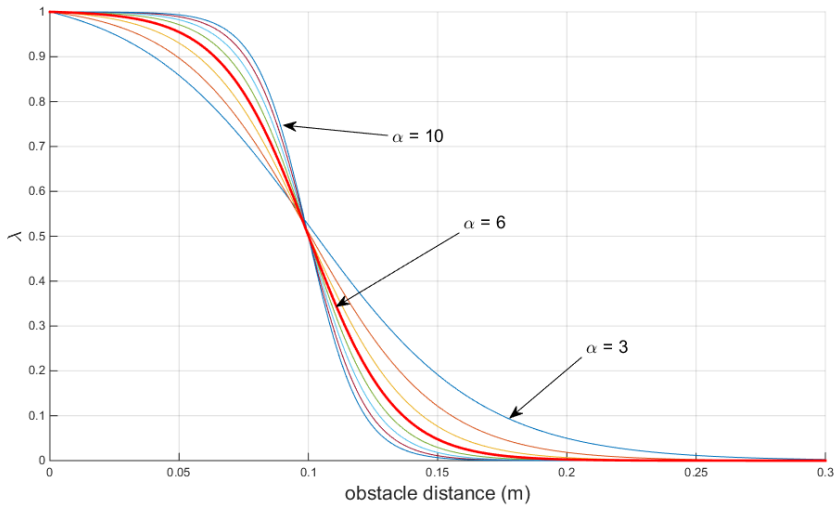


Figure 4.5: The weighted coefficient λ with $d_0 = 0.15$ m and various shaping factor α.

by a composition of the lead through command and the collision avoidance command.

$$\dot{q}'_{\text{cmd}} = \lambda(d)\dot{q}^*_{\text{cmd}} + (1 - \lambda(d))\dot{q}_{\text{lead}} \tag{4.12}$$

where $\dot{q}^*_{\text{cmd}}$ is the optimal solution of (4.11), $\dot{q}_{\text{lead}}$ is the lead through command given by the operator, and $\lambda(d) \in [0, 1]$ is the weighted coefficient for the velocity combination. Here λ is given by a sigmoid function,

$$\lambda = \frac{1 + \exp(-\alpha)}{1 + \exp(\alpha\|d\|(2/d_0) - 1)} \tag{4.13}$$

where d is the distance between the control point and the obstacle, d_0 is the safety margin, and α is a positive shaping factor. Fig. 4.5 shows the relationship between λ and the obstacle distance with various α.

When the robot stays in a safe distance (i.e far away from the obstacle), λ is close to zero, and the operator can freely guide the robot. Once the control point of the robot reaches the safe distance margin, λ will increase gradually. The weight of the collision avoidance becomes larger when λ increases. When $\lambda = 1$, the collision avoidance command fully controls the robot to prevent collision. Thus, the velocity composition not only preserves the switch feature but also smoothen the transition between two different velocity commands.

Table 4.1: The cylinder obstacle position in the three scenarios

Obstacle Scenario	Obstacle Cylinder Parameters, Units: mm			
	Diameter	Center x	Center y	Center z
I. Lower front	70	480	± 500	13
II. Upper front		710	± 500	546
III. Beside		230	430	-330, 670

4.4 Experiment

Experimental Setup

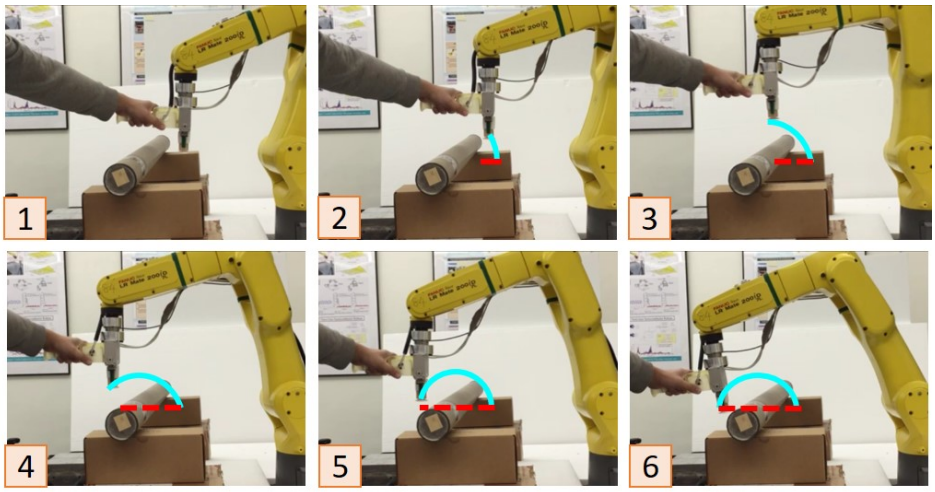
The experimental verification is implemented on the robot setup as introduced in Chapter 2. The operator steer the robot by applying a guidance force on the lead through handle bar. Note that the experiment does not use a sensor or a vision camera to locate obstacle, but the distance to obstacle is calculated based on the geometry parameters predefined in the program. Experiments are performed for three scenarios in Table 4.1, where it lists the cylinder diameter and the two circular center positions in the robot base frame. The parameters of the velocity composition weighting factor λ are assigned as $\alpha = 6$ and $d_0 = 0.15$ m. In future experiments, we will introduce a sensing means to detect and find the distance to obstacles in real time. The video of experiments can be found in [60].

Results

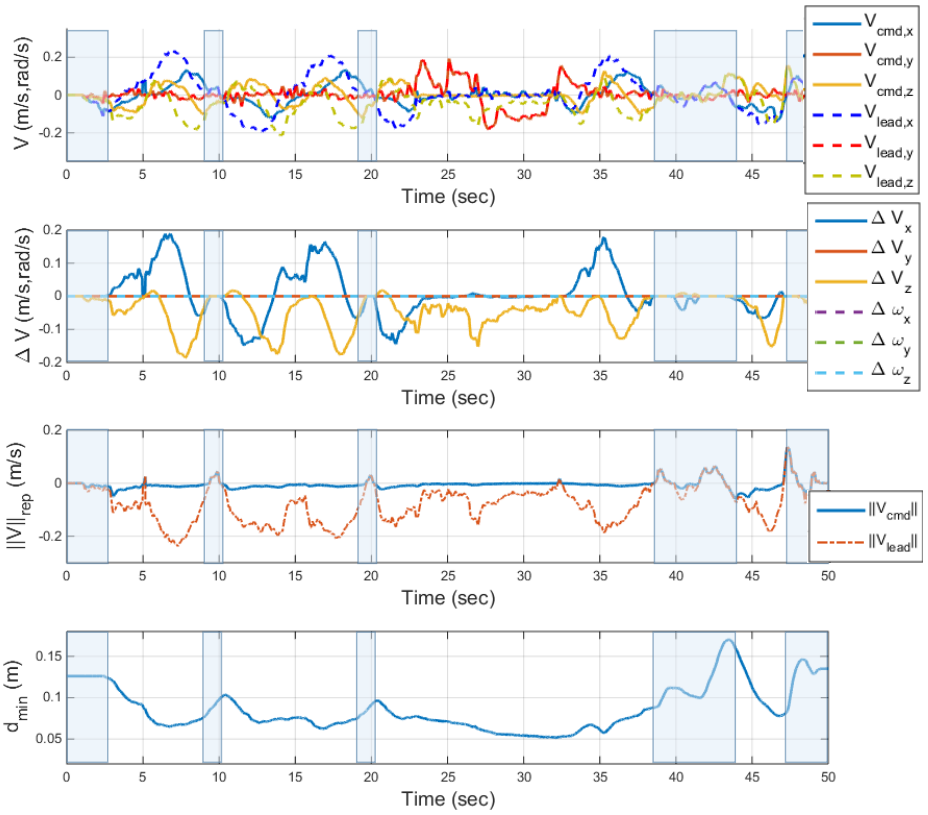
Fig. 4.6a, Fig. 4.7a, and Fig. 4.8a show part of the operator lead through sequence motions in the three experiment scenarios, where the cyan solid line and the red dash line represent the robot actual path and the operator's lead through intention, respectively. The experimental measurements are plot in Fig. 4.6b, Fig. 4.7b, and Fig. 4.8b. Each experiment has four plot. The first plot shows both the robot translational velocity commands, V_{cmd}, and the operator's lead through commands V_{lead} at the tool center point. V_{lead} is determined by the operator's lead through force, F_{lead}, i.e. $V_{\text{lead}} = K_d F_{\text{lead}}$, where K_d is the damping gain. The damping gain is 0.07 (m/Ns) for translational velocity, and 0.06 (rad/Ns) for angular velocity. The difference between these two velocity commands, $\Delta V = V_{\text{cmd}} - V_{\text{lead}}$, is shown in the second plot. The third plot shows the projection of the velocity command at u_{rep} direction. i.e. $\|V\|_{\text{rep}} = V^T u_{\text{rep}}$. The last plot shows the minimum obstacle distance to the robot body.

In the first experiment (See Fig. 4.6a), the obstacle is placed at the lower front position, where the end-effector might collide the obstacle if the operator guided the robot to move forward directly. The collision avoidance algorithm modifies the velocity command so that the robot moves slightly upward to cross the obstacle.

The detail of the robot motion can be found in Fig. 4.6b. The shaded areas in the plots are the time segments that the collision algorithm is inactive. Thus, there is no difference between V_{lead} and V_{cmd}, and the operator can freely guide the robot. In the unshaded area, the collision algorithm

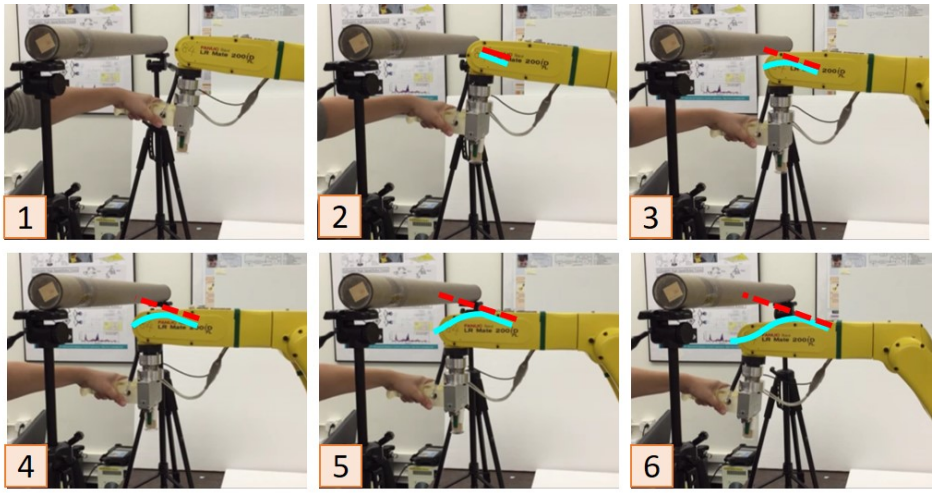


(a)

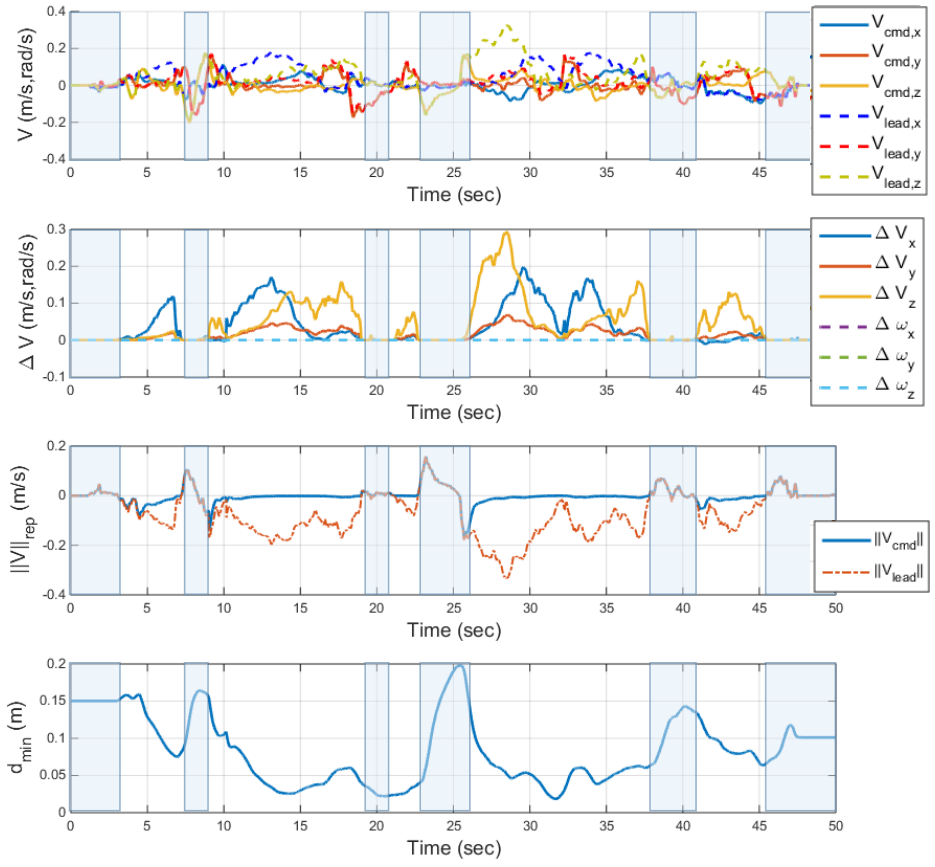


(b)

Figure 4.6: Lead through teaching with collision avoidance in the obstacle scenario I. (a) The sequence of figures that human operator is guiding the robot (b) The experiment plot.

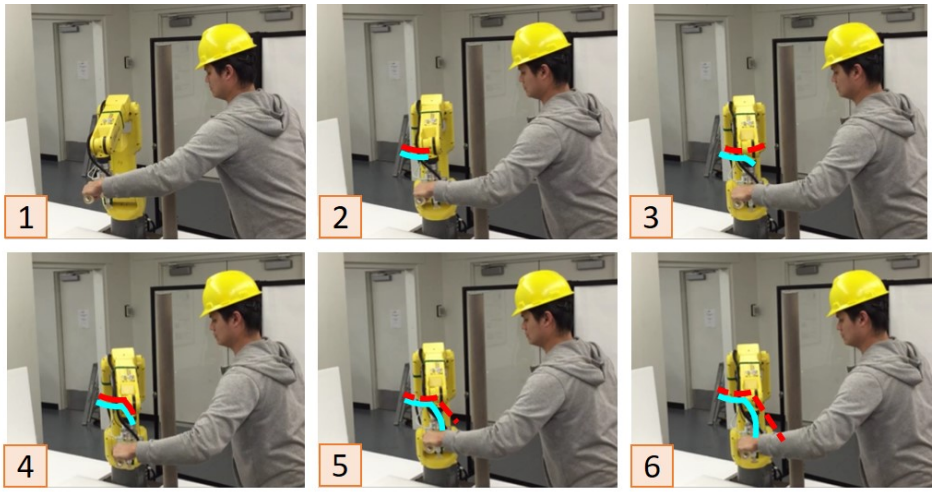


(a)

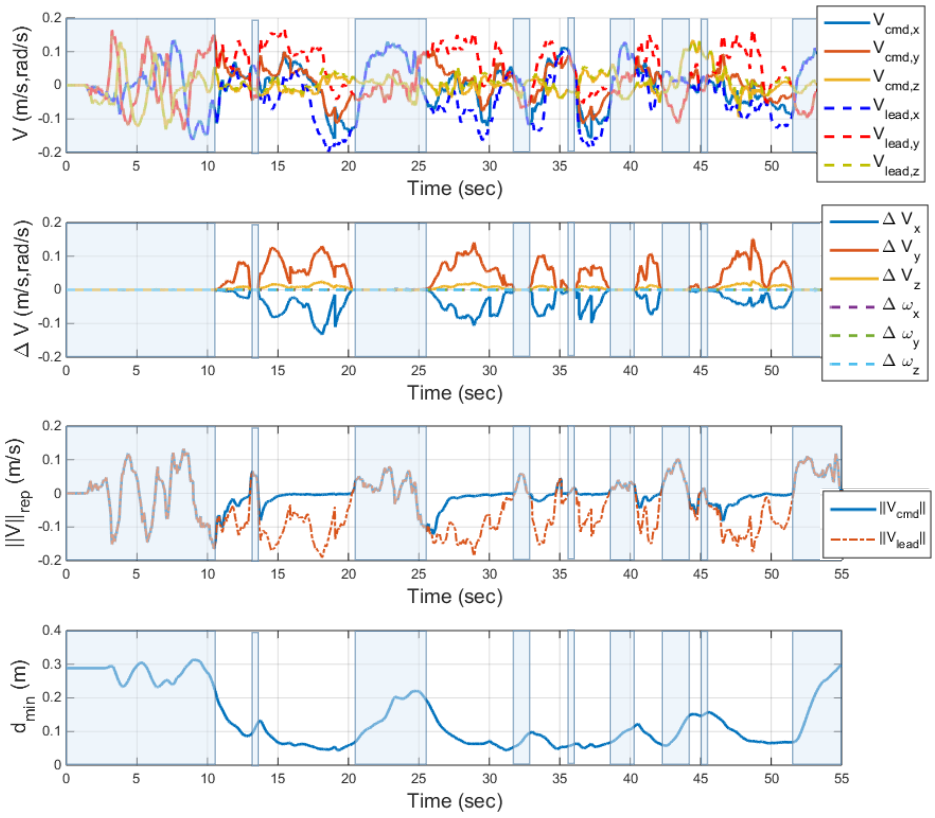


(b)

Figure 4.7: Lead through teaching with collision avoidance in the obstacle scenario II. (a) The sequence of figures that human operator is guiding the robot (b) The experiment plot.



(a)



(b)

Figure 4.8: Lead through teaching with collision avoidance in the obstacle scenario III. (a) The sequence of figures that human operator is guiding the robot (b) The experiment plot.

protects the robot by contributing a nonnegative velocity command in the direction of the repulsive vector. Hence, the blue solid line in the third plot of Fig. 4.6b is regularized at zero in the unshaded area, i.e. $u_{\text{rep}} \perp V_{\text{cmd}}$. This prevents the robot from moving closer to the obstacle. If the robot is operated by the lead through command only, the result would be more like the orange dash line, which would collide into the obstacle. Moreover, when the algorithm is activated, it tends to generate a tangential velocity along the obstacle consequently, and the modified path looks like a half circle in this experiment.

In the second experiment (see Fig. 4.7a), the obstacle is placed on the upper front position. Similar to the first experiment, a collision might happen if the operator does not pay attention to the obstacle during the lead through demonstration. However, the collision avoidance algorithm lowers the robot body to sidestep through the obstacle. Even if the operator intends to steer the robot arm to collide with the obstacle, the algorithm is still able to protect the robot. As shown in the third plot of Fig. 4.7b, there is a large velocity difference in z direction from $t = 25$ sec to $t = 30$ sec, which indicates that the operator intends to move the robot upward. However, the proposed method prevents this behavior. The operator would feel a heavy load when he tends to move the robot upward.

Unlike the first two experiment, the third experiment has a cylinder obstacle standing beside the robot (see Fig. 4.8a). This simulates the case where there is a pillar or a standing object beside the robot. In this experiment, the operator tests how fast the algorithm could switch between the free lead through mode and collision avoidance mode. As shown in Fig 4.8b, the operator moves around the boundary of the avoidance criteria from $t = 30$ sec to $t = 45$ sec. The third plot illustrates the blue solid line converges to 0 at a very short time. Hence, this algorithm can instantly protect the robot from colliding into the obstacle.

To summarize the experimental results, two trigger conditions have to be satisfied simultaneously to activate the collision avoidance algorithm. The first condition is that the obstacle distance is below the safety margin. The second one is that the projection between the lead through velocity command with the repulsive direction is non-negative. Thus, it is possible that the robot is close to an obstacle, but the collision avoidance algorithm is not activated because the operator's lead through velocity would not collide with the obstacle. This is a good property in the human guidance programming because the velocity difference is minimized during the operation. Moreover, because the repulsive vector affects the translational velocity only, the angular velocity difference remains at zero during this experiment. Lastly, the collision avoidance algorithm converges to zero in a short time even if the mode is frequently switched. This shows that this method can provide instant protection for the robot.

4.5 Chapter Summary

This chapter proposed a novel collision avoidance algorithm for physical human-robot interaction (pHRI). The Jacobian decomposition modified the original null space projection method and led it applicable to a non-redundant robot. The human guidance command and collision avoidance were formulated into an optimization problem, where the closed form solution could be found.

Unlike other conventional collision avoidance methods that only considers the obstacle distance, the proposed collision algorithm took the human intention into consideration as well. When the operator's lead through command was in a safe direction, the collision avoidance would not be activated. This not only preserved the safety from the traditional method, but also improved the flexibility and comfort in the human guidance programming. Lastly, the proposed algorithm was verified on a 6 DoF industrial manipulator in three different obstacle scenarios. The video for the experiment is in [60].

Part II

Learning

Chapter 5

Learning from Demonstration with Remote Lead Through Teaching

5.1 Introduction

Many industrial tasks performed by robots require position-force control. For example, the assembly and the surface polishing are realized by controlling not only the position of the end-effector but also the applied force to the workpiece. The position-force control can be categorized into two methods [83], the impedance control and the hybrid position/force control. The impedance control first proposed by Hogan [43] establishes a relationship between the velocity/position of the robot and the interaction force with the environment. The hybrid position/force control proposed by Raibert and Craig [84] separates the position and force control into two independent channels. Hence, the desired position or force for each can be individually specified. These controllers can achieve a good performance when the desired task and the environment are well-defined. However, tuning a set of control parameters is nontrivial. Also, even when the task changes slightly, tuning has to be repeated.

Compared to the robot, human can learn from and adapt to various tasks with shorter time and fewer trials. Based on this observation, many researchers have investigated how to transfer the human knowledge and skill to the robot. To simplify the robot programming, the idea of Learning from Demonstration (LfD) is introduced by Schaal [92]. The main principle of robot LfD is that the users can teach the robots through demonstration instead of programming. Then, a question comes up: what is the interface for demonstration? The first intuitive answer is direct demonstration by human. Namely, operators use their own bodies to demonstrate the task, and the motion capture suits or markers record the demonstration. Calinon et al. [20] and Schaal [91] illustrate robot imitation learning from demonstration. This approach may be well suited to the humanoid or anthropomorphic robots. However, the different configurations between human and industrial robot make the mapping difficult. Furthermore, the wearable sensors usually only record the human motions, but many industrial applications require the force information as well.

The lead through teaching [42, 25] is another common technique in LfD. The operator directly

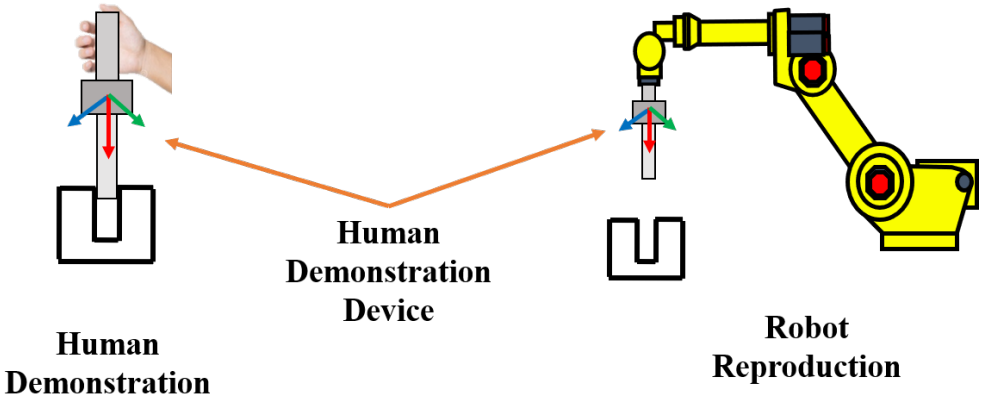


Figure 5.1: The common tool frame of the human demonstration device is defined for both the human and robot workspace.

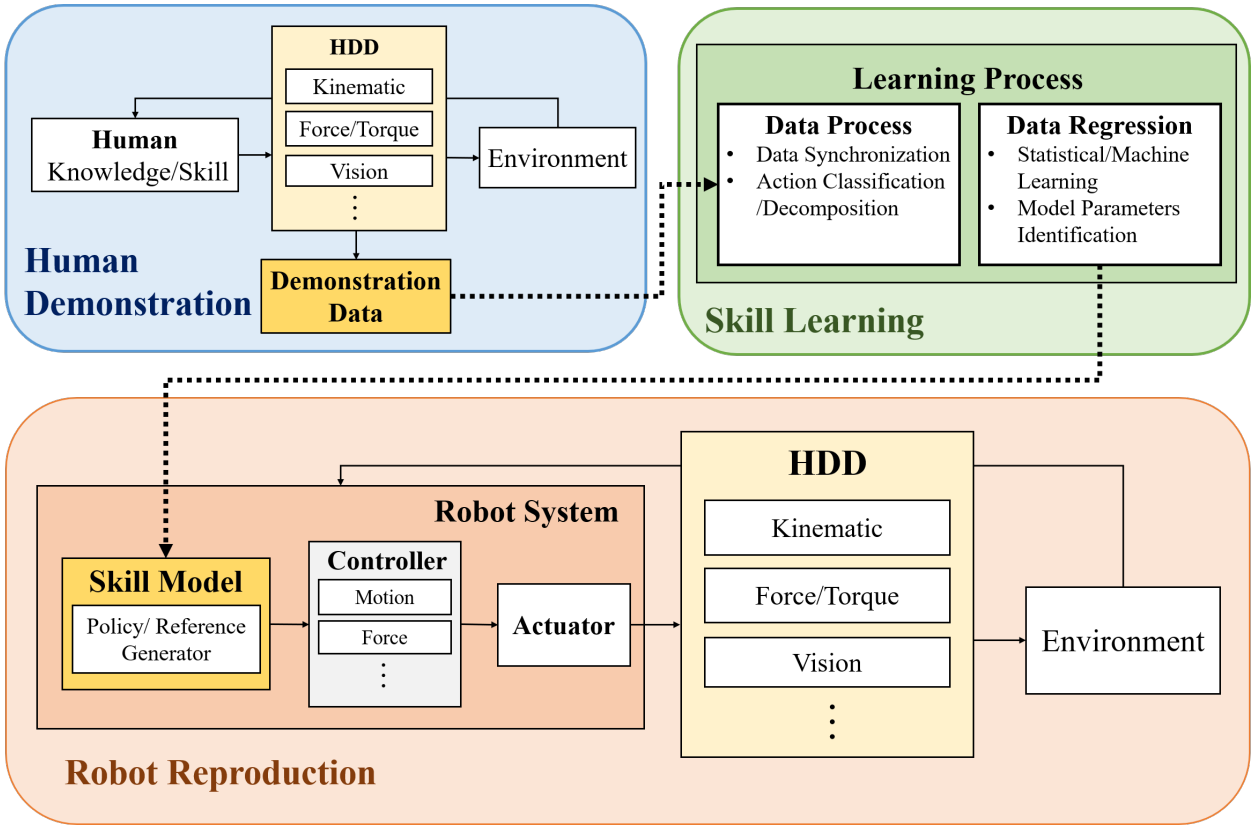


Figure 5.2: The framework of remote lead through teaching, where it is decomposed into two phase, human demonstration and robot reproduction. In order to transfer human knowledge/skill to the robot, a skill learning process is established between the two phases.

grasps the link or the handle mounted on the robot. The robot force controller allows the operator manually move the robot arm to pass though a desired path or a sequence of successive points so as to define the task. The lead through teaching provides a convenient and intuitive path planning approach, but it requires physical contact between the operator and the robot, which poses a potential danger to the operator. Also, the force measurement is the external force applied by human, the lead through teaching can not be used to teach the interactive force between the robot and the environment.

Teleoperation [23, 21] or virtual robot teaching [50] separates the workspace of human and robot. The operator manipulates the robot in a virtual reality environment by maneuvering a haptic interface. A sequence of robot commands are generated by recording the human motion/force on the haptic device. The remote operation ensures the operator's safety, but insufficient tactile feedback limits the applications. For instance, the teleoperation method may not be applicable to the complicated industrial tasks such as surface polishing.

In this chapter, a novel approach called remote lead through teaching (RLTT) is proposed to simplify the robot programming process. Under the framework of RLTT, human and robot share the common reference. Hence, the human demonstration data can be directly utilized by the robot. RLTT preserves the properties of the lead through teaching method. It also provides to the users a natural and safe demonstration approach.

The remainder of this chapter is organized as follows. The basic concepts and the framework of RLTT are outlined in Section 5.2. The demonstration data processing and learning are introduced in Section 5.3. Section 5.4 presents two applications of RLTT teaching method. Finally, the conclusion is given in Section 5.5.

5.2 Remote Lead Through Teaching

The basic idea of RLTT is illustrated in Fig. 5.1. A human demonstration device (HDD) is designed as a common tool for both human and robot. The tool frame in human demonstration phase is aligned to that in robot reproduction phase. HDD is a senor fusion system to record all the task required information during human demonstration. While the human is performing the task, the HDD records the demonstrator's motion and force. When the robot is assigned to reproduce the task, the HDD is mounted on the robot end-effector, and sends the measurement as feedback signal to the robot controller.

The framework of RLTT is shown in Fig. 5.2. As previously discussed, RLTT is decomposed into two phase, human demonstration and robot reproduction. In order to transfer human knowledge/skill to the robot, a skill learning process is established between the two phases.

Human Demonstration Phase

In human demonstration phase, the demonstrator uses HDD to naturally perform the task. At the same time, HDD records the information in the tool frame. As mentioned previously, HDD is a

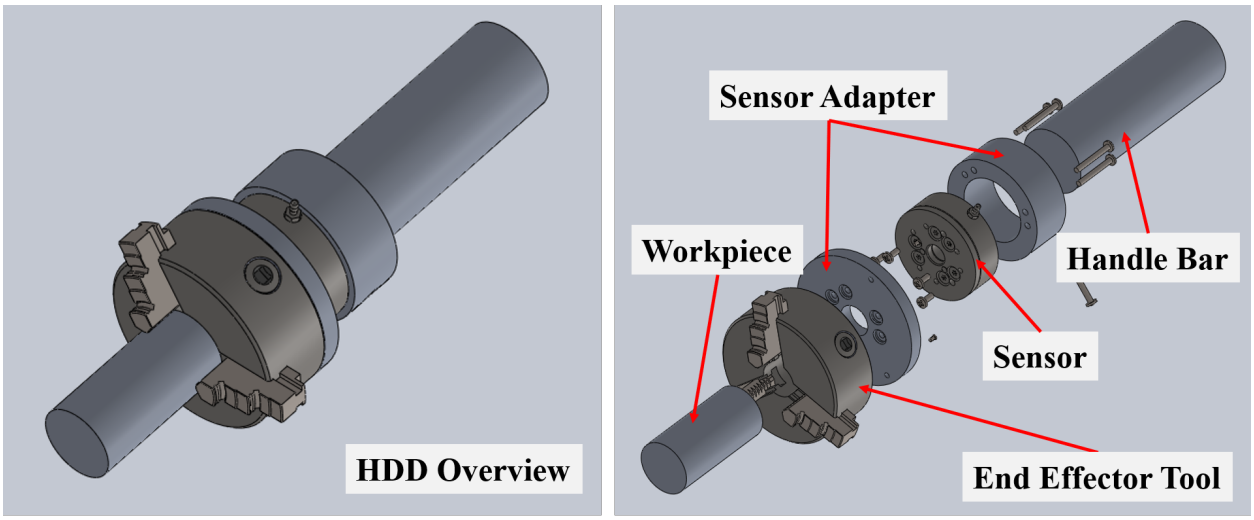


Figure 5.3: The design of the human demonstration device

combination of different sensor devices, which may include the measurement of position, velocity, and force, etc.

The structure of the HDD is briefly illustrated in Fig. 5.3, which consists of the handle bar, the sensor, the end-effector tool, and the adapters. The handle bar provides the place held by the human operator and mounted on the robot end-effector. A rapid release mechanism is placed on the robot end-effetor so that handle bar can be easily installed/uninstalled on the robot. As previously introduced, the HDD requires both force and motion sensing abilities. In this prototype design, a six-DoF force/torque transducer is fixed in the center of the HDD. The motion capture markers are placed on different surfaces of the HDD so that motion capture camera would not lose the tracking signal. The end-effector tool can be changed with various tools corresponding to the task requirement. For example, a gripper is placed for the peg-hole insertion, while a grinder is used for the surface polishing. The adapters are designed to protect the sensor and to link the handle bar with end-effector tool. Figure 5.4 shows the completed prototype of the HDD. The force sensor model is ATI mini 45 F/T transducer[5]. The PhaseSpace Impulse X2 LED markers[82] are placed on the top and body of the HDD.

There are several benefits of HDD in the human demonstration phase. Firstly, the human and robot workspaces are separated. Hence, the user's safety is guaranteed. Secondly, the HDD can be regarded as an add-on of the tool. It does not make significant changes of the user operating the task. Thus, the natural demonstration behavior can be preserved by RLTT.

Skill Learning Process

The purpose of skill learning process is to build the skill model from the demonstration data. The skill model is a policy or a reference generator for the robot system. When the robot is given a

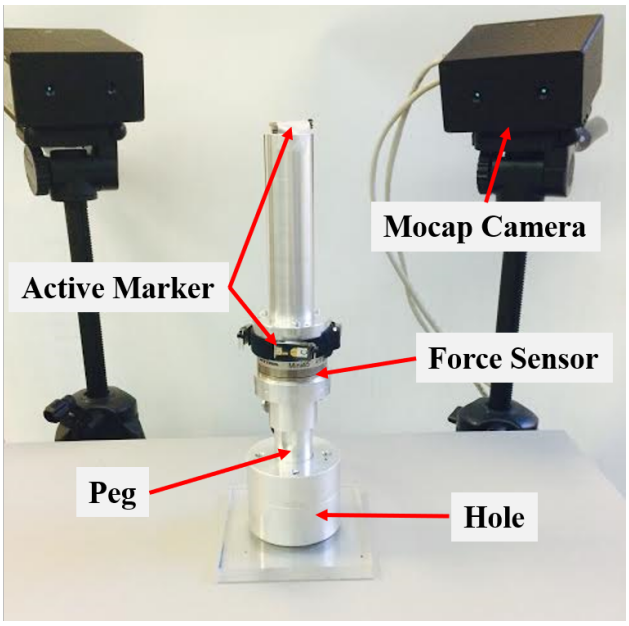


Figure 5.4: The prototype of the human demonstration device

feedback signal from the HDD, the skill model generates a corresponding command to the robot control loops, where the policy is learned from the human demonstration data.

Before training the model from the demonstration data, some processes are required for improving the learning quality. For instance, the demonstrator has different motion speed in each demonstration. Although the demonstration behaviors are similar, the trajectories might look very different due to the mismatched timing. Besides, robot and human have their own expertise. It is not necessary to learn every single action from human demonstration. For example, the robot moves more precisely and faster than human, while human is more intelligent in assembly. Then, the robot does not need to imitate how human approaches the workpiece, but to learn how human assembles the parts together. Hence, the data processing involves two steps. The demonstration data is firstly synchronized, then decomposed into several action segments. The data in the target segments are further utilized for skill model learning.

Since it is difficult to directly derive the human skill model, the data must be first analyzed to describe the human behavior. The statistical learning and machine learning are the methods to identify the model by the training data. If the structure of model is known, the model parameter identification technique is applicable to estimate the model parameters. The detail of the skill learning process is discussed in Section 5.3.

Robot Reproduction Phase

In the robot reproduction phase, the HDD is mounted on the robot end-effector. In addition, the skill model established from the previous process is embedded into the robot control system. Because the HDD frame in the robot reproduction phase is aligned to that in the human demonstration phase, the sensory information in these two phases are shared in the same reference. Hence, the mechanism of skill model finds the closest human demonstrated policy to the current HDD measurements.

5.3 Skill Learning From Demonstration

This section introduces how to process and learn from the demonstration data, so as to build the human skill model. The three steps are (1) Data Synchronization, (2) Data Decomposition, and (3) Data Regression.

Data Synchronization

Berndt at el [10] proposed the dynamic time warping (DTW) method to deal with the speech recognition problem. The technique of DTW uses a dynamic programming approach to align the time series and a specific pattern so that the distance is minimized. Because each human demonstration is a time series with a specific pattern, DTW is applicable to synchronize multiple demonstrations.

Suppose there are three demonstration trajectories with different speed (see Fig. 5.5a). T_r is the trajectory with reference speed, while T_f and T_s are the fast and slow trajectories, respectively. In practice, the reference trajectory is determine by the user.

As shown in Fig. 5.5a, the node i is the value of the trajectory at time step i. $w_k(i,j)$ is the warping distance between two trajectories. The goal of DTW algorithm is to find the optimal sequence of $w_k(i,j)$ such that the total warping distance is minimized.

$$\min_{i_k,j_k} \quad \sum_{k=1}^{K} \|w_k(i_k,j_k)\| \tag{5.1}$$

$$s.t \quad i_1 = 1, \quad j_1 = 1, \tag{5.2}$$

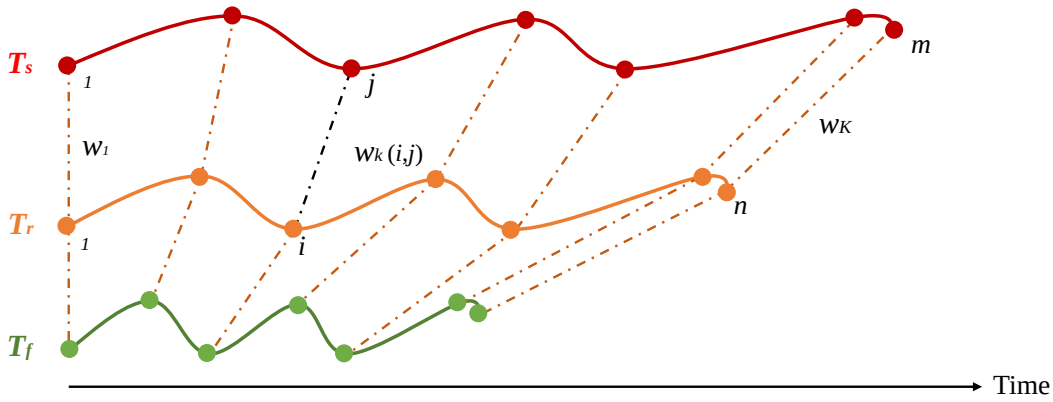
$$i_K = m, \quad j_K = n, \tag{5.3}$$

$$i_{k-1} \leq i_k, j_{k-1} \leq j_k \tag{5.4}$$

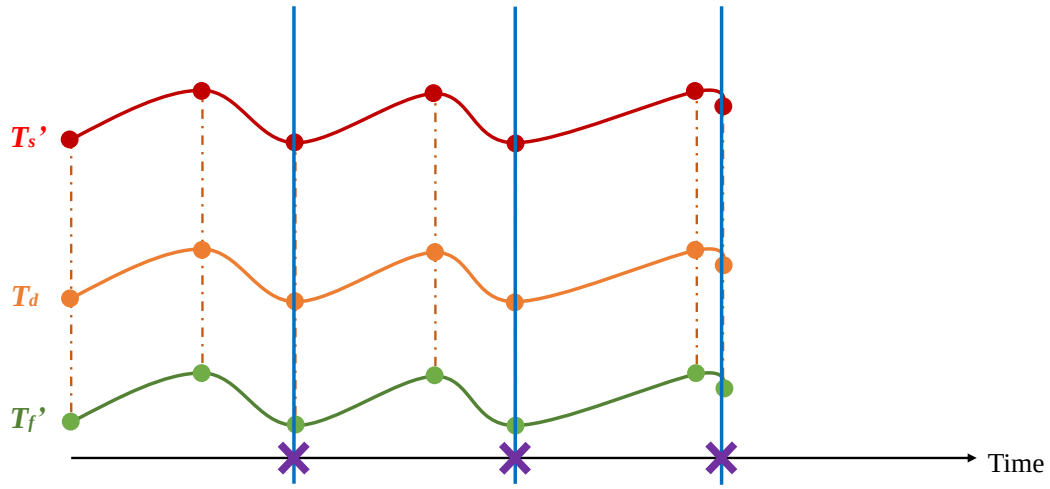
$$i_k - i_{k-1} \leq 1, j_k - j_{k-1} \leq 1 \tag{5.5}$$

$$|i_k - j_k| \leq r \tag{5.6}$$

The constraints are designed to reduce the space of possible warping paths and to make the warping time index more reasonable. (5.2) and (5.3) are the initial and final condition of the time series, respectively. (5.4) implies that all the grid points are monotonically ordered with respect to time.



(a)



(b)

Figure 5.5: The illustration of data synchronization and data decomposition (a)Different trajectories synchronized by dynamic time warping (b) The synchronized data decomposed by the split points

The continuity of the time series is restricted by (5.5). The last constraint is to define the size of the warping window, $r \in \mathbb{Z}^+$, which makes indices searching more efficient.

As shown in Fig. 5.5b, T'_f and T'_s are the trajectories aligned with the reference trajectory. With the synchronized trajectories, the pattern of the demonstration is more distinct to users. The data decomposition is thus applicable to the multiple demonstrations.

Data Decomposition

The data decomposition is to find the time steps that the demonstrator changes the action behavior. e.g. the purple points in Fig. 5.5b. Since each action has its own pattern in motion/force trajectory, searching the action sequence can be formulated as an action classification problem. The support vector machine is a mature algorithm in classification problems [16, 27]. We introduce a SVM-based action classifier to decompose the demonstration data. A demonstration trajectory is given by

$$T = \begin{bmatrix} p_1 & \cdots & p_t & \cdots & p_n \\ f_1 & \cdots & f_t & \cdots & f_n \end{bmatrix} \in \mathbb{R}^{d \times n} \tag{5.7}$$

where p_t, and f_t are the position and the force measurement at t, d is the total dimension of the measurements, and n is the total time steps of the demonstration. The feature vector is written as

$$\Phi = \mathbf{vec}(T) \tag{5.8}$$

$$= \begin{bmatrix} p_1^T & f_1^T & \cdots & p_t^T & f_t^T & \cdots & p_n^T & f_n^T \end{bmatrix}^T \in \mathbb{R}^{nd} \tag{5.9}$$

Since the actions are the segments of the demonstration, each action can be represented similarly in the vector form.

To illustrate the action classification by SVM, a simple example is given. Suppose there are two sets of actions. The mechanism of SVM is to construct an optimal hyperplane in the middle of the two classes, so that the margin to the nearest positive or negative example is maximized. The decision function of the action classification is

$$f(\Phi) = \theta^\top \Phi + b \tag{5.10}$$

where $\theta \in \mathbb{R}^{nd}$ is the weighting parameter vector, and $b \in \mathbb{R}$ is the bias or offset scalar.

Although the actions are assumed to have different patterns, there are some cases that the different actions have a similar pattern, which makes it difficult to determine the decision boundary. Regarding the robustness issue, the soft margin SVM algorithm [104] is proposed by adding the slack variables $\xi_i \geq 0$. Hence, the training example Φ_i can satisfy the constraint even if it is on the wrong side of the decision boundary. The soft-margin SVM is given by

$$\min_{\theta,b,\xi} \quad \|\theta\| + C \sum_{i=1}^{N} \xi_i \tag{5.11}$$

$$s.t. \quad y_i(\theta^\top \Phi_i + b) \geq 1 - \xi_i, \qquad \forall i = 1, \cdots, N \tag{5.12}$$

$$\xi_i \geq 0 \tag{5.13}$$

Algorithm 1 Demonstration Decomposition

Input the demonstration trajectory
Initialize: $P_s = 0$, Iter = 1
if Iter $< k$ **then**
  Crop the segment by the sliding window
  Align the segment dimension by **DTW**
  Classify by **SVM**
  if classified label = sequence(Iter) **then**
    Map Score = 1
  else Map Score = 0
  end if
  Calculate the center of the local score map for the separate point P_s
  Update: Iter $\leftarrow$ Iter +1, $P_s \leftarrow P_s$
else **return** the decomposition with P_s
end if

where $C \geq 0$ is a penalized constant, and $y \in \{-1, 1\}$ is defined as the class label. The SVM classifier can be extended to multi-classes classification without losing the generality.

Hence, a SVM-based classifier can decompose the demonstration into several actions by classifying the motion/force trajectory based on trained class sets.

The demonstration is assumed to be a specific sequence, and the total number of actions is known. e.g. in a peg-hole insertion task, the demonstrator first approaches the hole, then rotates the peg to align with the hole, lastly inserts the peg into the hole. The total number of actions in the insertion task is three. The basic idea of the data decomposition is to find the split points, P_s, in the demonstration, where the split points are the timings that the demonstrator changes his/her action or move on to the next step in the task. To search the split points along the trajectory, the sliding window method is realized, which is a common technique in computer vision for finding the target objects in a picture[105].

The whole algorithm is presented in Algorithm 1. First, the $P_s \in \mathbb{R}^k$ is initialized as zeros, where k is the total number of actions in the task. In each iteration, a segment is firstly cropped by the sliding window, where the width of the window is designed by users. The time steps of cropped segments are aligned with the trained actions by DTW. Then, a classified label is assigned by SVM. The matched segments are labeled with a map score. By calculating the score map, the locations of split points are determined. Keep the iteration until the whole split points are found. With the set of split points, the demonstration can be decomposed into several actions, $\mathcal{A}(p, f)$.

Data Regression

To mathematically quantify the human skill is not a trivial work. There is no convincing deterministic model to describe the human decision during a single task [44]. Thus, the statistical learning

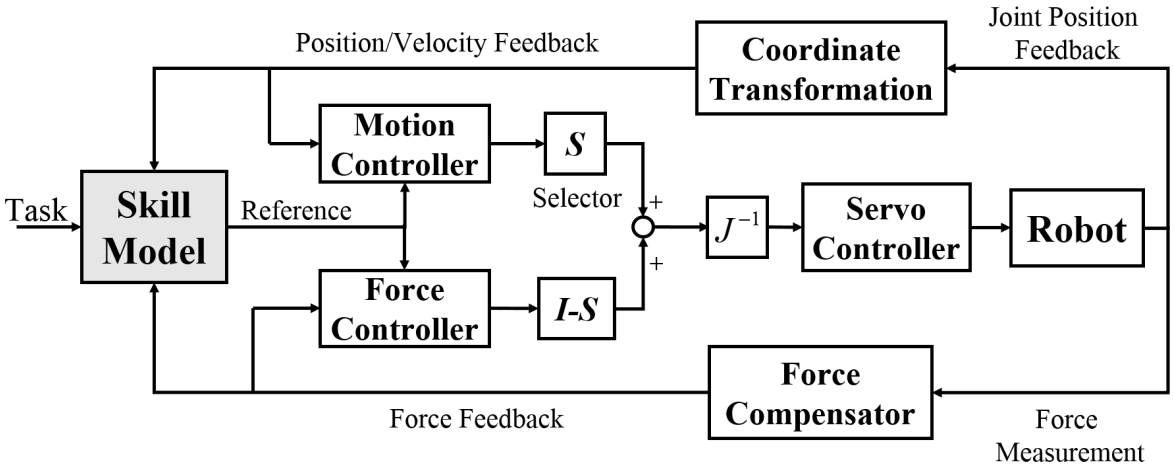


Figure 5.6: Skill model in the position-motion hybrid control scheme

model is utilized to represent the human skill. Suppose the human skill model is a black box, then the human perception Ψ and policy Π are the model input and output, respectively.

Although the relationship between Ψ and Π are ambiguous, the pair of (Ψ, Π) of the human skill is obtained by observing the target action $\mathcal{A}(p, f)$. For instance, when demonstrator performs the insertion task, he/she usually senses the contact force first, then adjusts the motion. In surface grinding case, the demonstrator would think a desired shape first, then apply force on the workpiece.

The skill model is estimated by a mixture of Gaussian models [71]. The joint probability of a human perception/policy is estimated by N Gaussian components

$$\Pr(\Psi, \Pi) = \sum_{i=1}^{N} \alpha^i \mathcal{N}(\mu^i, \Sigma^i) \tag{5.14}$$

where μ^i and Σ^i are the mean and covariance matrix, and α^i is the weighting factor of the i-th Gaussian component.

The purpose of skill model is to generate a policy/reference when given a perception. The conditional probability is given by

$$\Pr(\Pi|\Psi) = \mathcal{N}(\mu_{\Pi|\Psi}, \Sigma_{\Pi|\Psi}) \tag{5.15}$$

and is also a Gaussian due to the properties of Gaussian [71]. The GMR algorithm [14] determines the optimal policy Π^* by maximizing the likelihood of $\Pr(\Pi|\Psi)$

$$\Pi^* = \arg\max_{\Psi} \Pr(\Pi|\Psi) \tag{5.16}$$

$$= \sum_{i=1}^{N} \frac{\alpha^i \mathcal{N}(\mu^i_{\Pi|\Psi}, \Sigma^i_{\Pi|\Psi})}{\sum_{j=1}^{N} \alpha^j \mathcal{N}(\mu^j_{\Pi|\Psi}, \Sigma^j_{\Pi|\Psi})} \mu^i_{\Pi|\Psi} \tag{5.17}$$

where α^i is the weighting factor of the i-th Gaussian component. To implement the GMR, the Gaussian parameters $(\mu^i_{\Pi|\Psi}, \Sigma^i_{\Pi|\Psi}, \alpha^i)$ are estimated by EM algorithm from the demonstration data [13]. Because the EM algorithm is usually sensitive to initialization, K-means clustering [8] is applied to the dataset for a good initial condition.

The skill model trained by (5.17) is embedded into the position-force hybrid control scheme, which is shown as a shaded block in Fig. 5.6. The skill model uses the current measurements to represent Ψ, and then generates a policy Π as a robot reference. In this way, the human skill model could be transferred for robot applications. Note that the choice of perception will vary by tasks. For example, in the peg-hole insertion application, the force measurement is used as a feedback to generate the corrective velocity for insertion. In the surface grinding application, the current pose is used to generate the desired force.

5.4 Experiment

In order to validate the proposed RLTT framework, it is applied on two classical position-force industrial tasks. The first application scenario is the assembly task, which is represented by peg-hole insertion. The second one is the grinding scenario, which is represented by a simplified testbed. In this section, the experimental setup is firstly introduced, followed by the illustration of data acquisition, synchronization and decomposition in peg-hole insertion demonstration. Lastly, the skill learning by GMR and the robot execution in the two scenarios are provided.

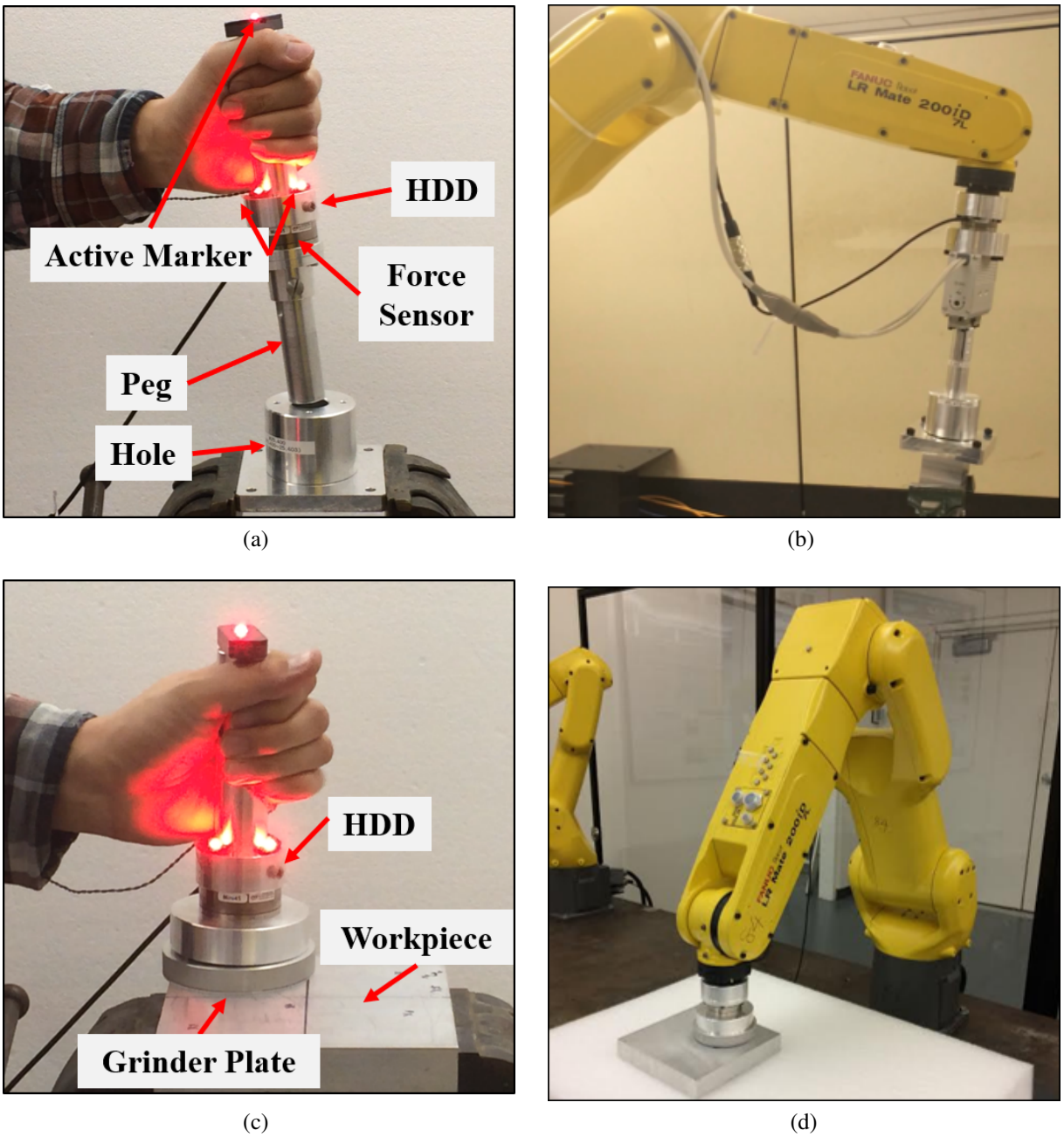
Experiment Setup

Fig. 5.7 shows the experimental setup for the remote lead through teaching. Fig. 5.7a and 5.7b are the photos of the peg-hole insertion in human demonstration and robot reproduction phases, respectively. Fig. 5.7c and 5.7d show both phases in the surface grinding. In the human demonstration phase, the demonstrator's motion is captured by tracker cameras. The robot in Fig. 5.7b and 5.7d is FANUC LR Mate 200*i*D/7L as introduced in Chapter 2, where the force sensor is installed on the end-effector. However, markers are not necessary to place on the robot body because the motion of the end-effector can be obtained by calculating the forward kinematics of the current robot joint position.

Data Acquisition

The demonstration trajectory and force during the peg-hole insertion task were shown in Fig. 5.8. The first plot illustrated the motion of the marker on the top of the HDD. The second plot shows the orientation of the HDD. The direction angles are the HDD relative to the basis axes. The third plot illustrated the force measured by the HDD.

The peg-hole insertion is decomposed to four phases: approaching, rotation, insertion, and completion, where each phase was labeled under the bottom of the Fig. 5.8. The approaching phase is from 0s to 2s, where the operator holds the tilted HDD to approach the hole. The position



(a) (b)

(c) (d)

Figure 5.7: Remote lead through teaching in two industrial applications. (a) The human demonstrator performs the peg-hole insertion task. (b) The robot reproduces the peg-hole insertion experiment. (c) The human demonstrator performs the grinding task. (d) The robot reproduces the grinding experiment.

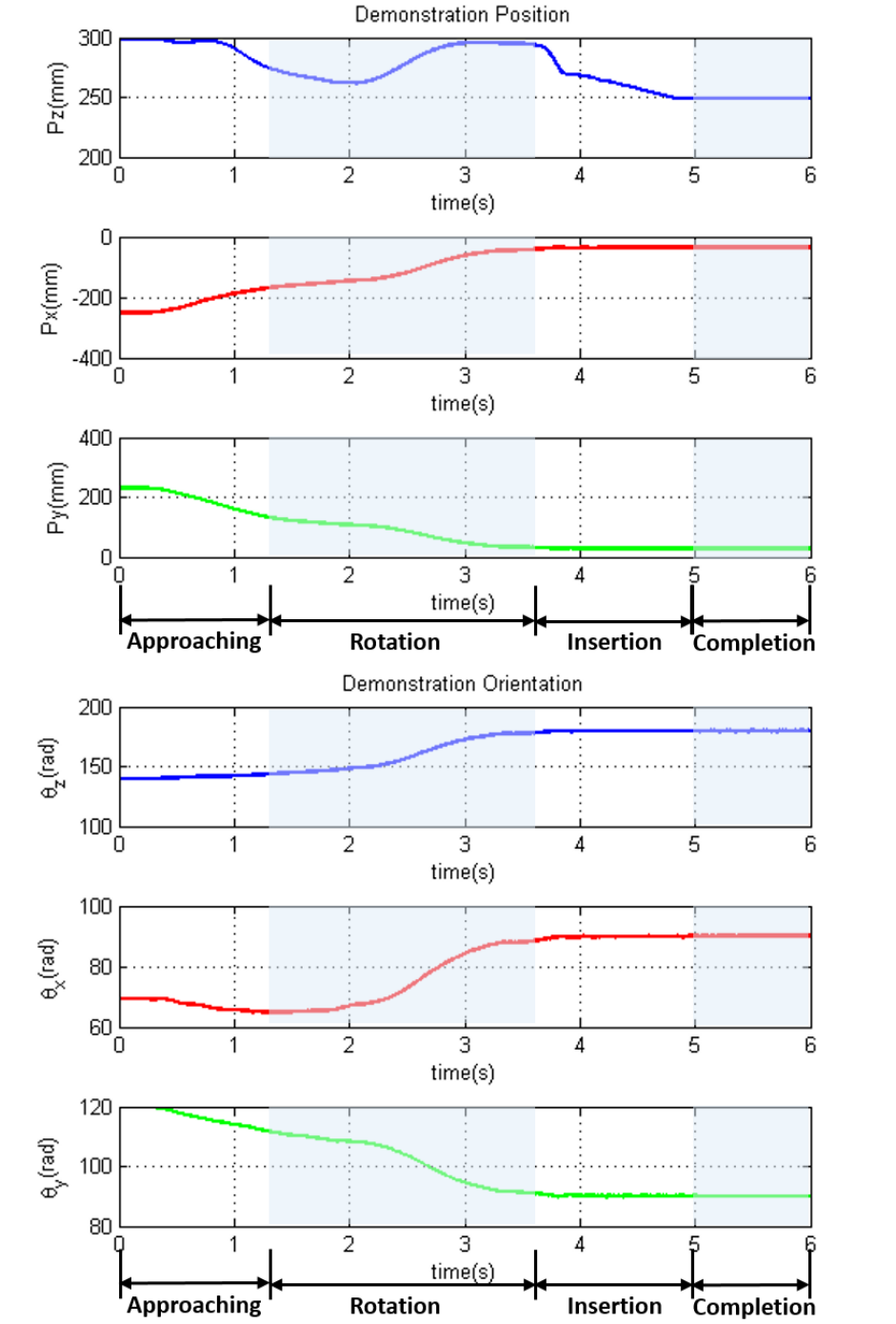


Figure 5.8: The demonstration motion and force

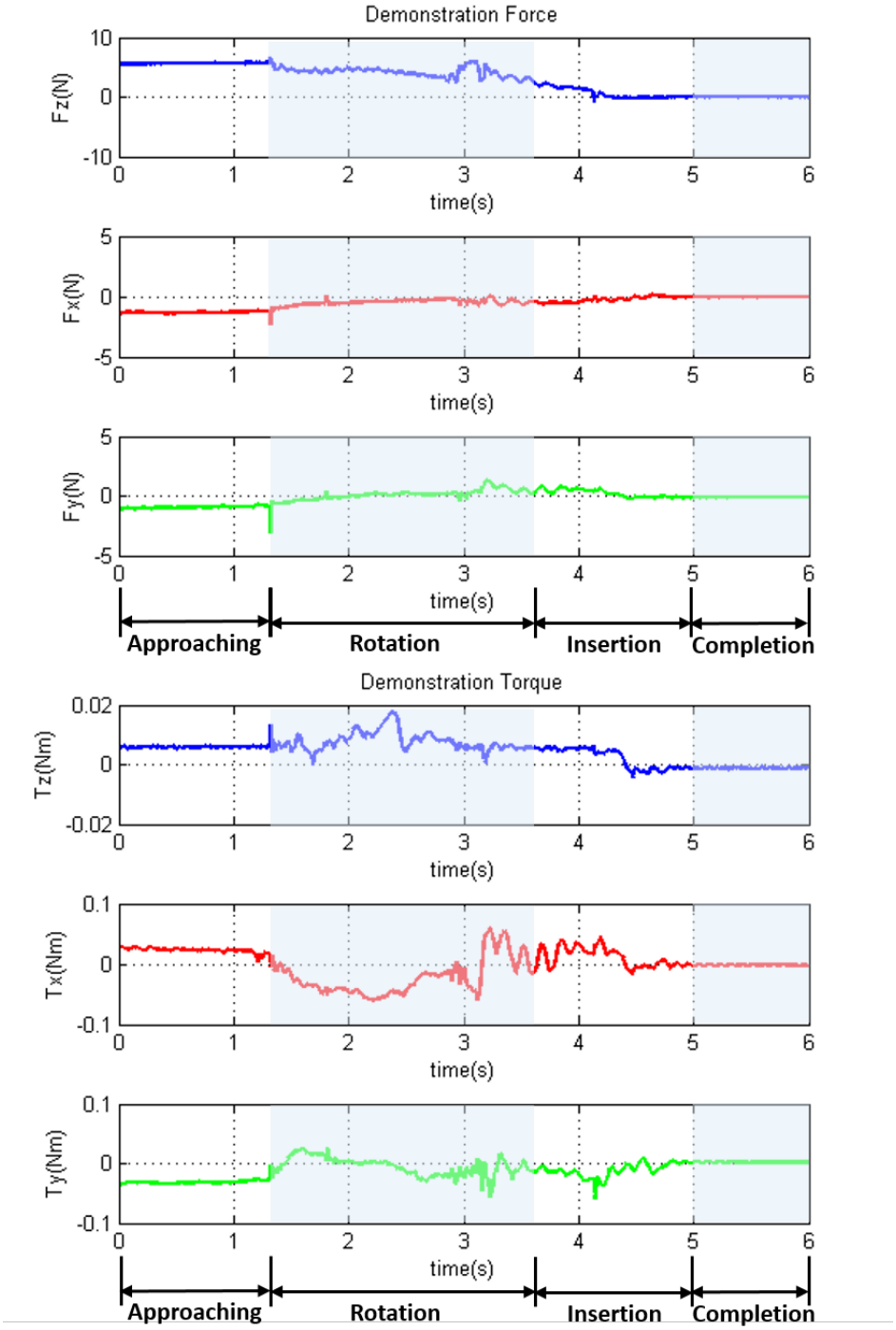
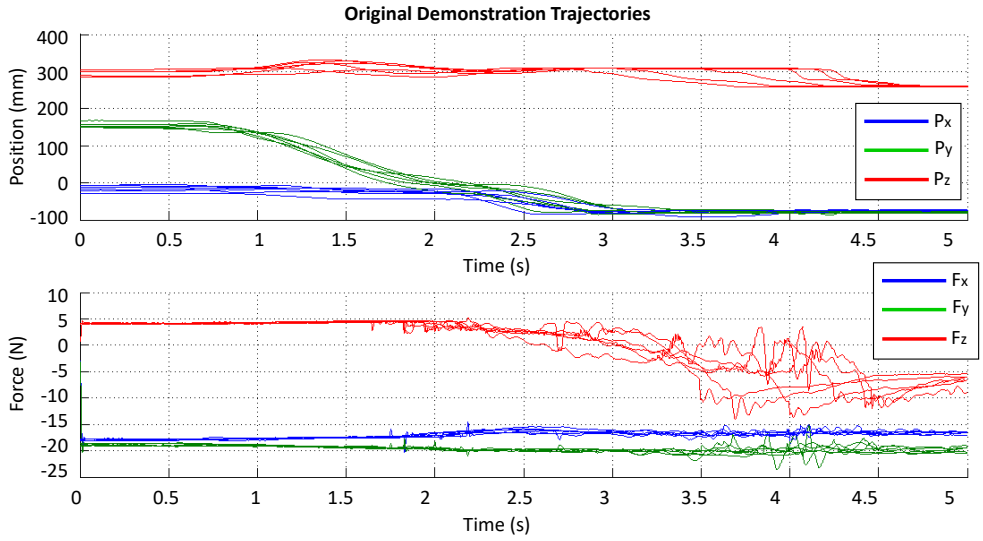
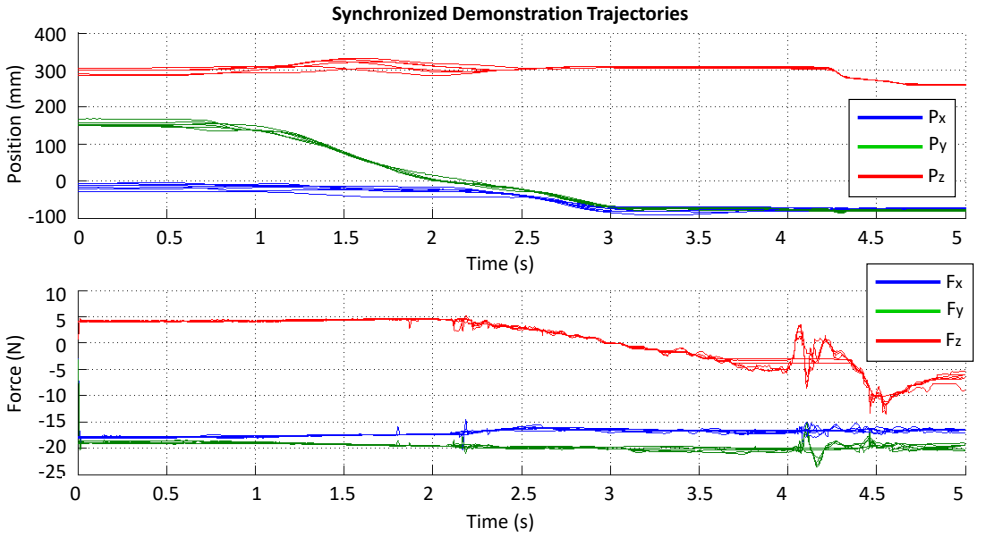


Figure 5.9: The demonstration motion and force



(a)



(b)

Figure 5.10: The demonstration data processing. (a) The original demonstration data. (b) The synchronized demonstration data.

of the HDD decreases to a certain height and approached the entrance of the hole. The force is caused by the gravity and inertia of the HDD. Notice that when the HDD contacts the hole in this phase, there is a support force compensating the gravity effect. The rotation phase is the shaded area from 2s to 3.5s. The orientation of the HDD is adjusted to the insertion direction by the operator in this interval. Due to the orientation adjustment, the height of the marker increases until the HDD was perpendicular to the table. The insertion phase is from 3.5s to 5.2s. The HDD moves

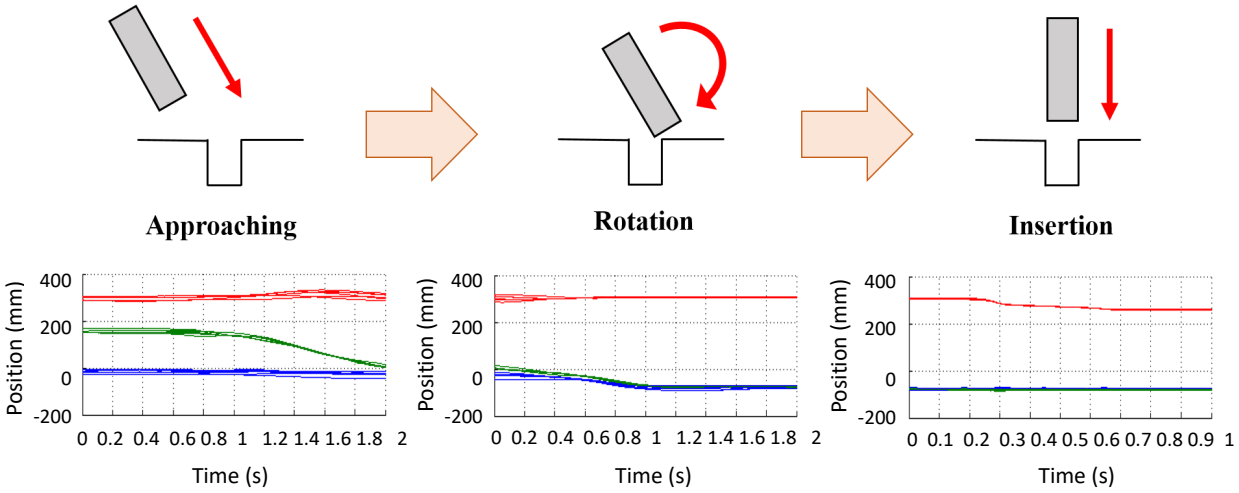


Figure 5.11: The demonstration decomposed into three actions: approaching, rotation, insertion.

downward to the bottom of the hole, and the orientation remains the same. The force sensor reflects the fluctuation caused by the friction during the insertion. In the completion step, the HDD was in the static status, and the measurements remained the same. The experimental results clearly show the motion/force, which satisfied the expectation. Also, the results provide users with a insightful information of the demonstration.

Data Processing

To illustrate the data processing, sixty demonstrations of peg-hole insertion are recorded. The original and synchronized demonstration motion/force trajectories of peg-hole insertion are shown in Fig. 5.10a and 5.10b, respectively. In order to have a better illustration, there are only five demonstrations plotted in the figures. The original insertion timings are inconsistent due to the different demonstrator's motion speed. The data decomposition is realized by the SVM-based action classifier with the SVM toolbox package [24]. The training data is acquired by manually decomposing and labeling in the 20 demonstrations, and the rest of demonstrations are automatically decomposed by the SVM-based action classifier. The result is shown in Fig. 5.11, where the three actions decomposed from the peg-hole insertion demonstration are approaching, rotation, and insertion.

Assembly Scenario

The assembly scenario is designed as an industrial standard H7/h7 peg-hole insertion task. The material and the dimension of the workpieces are listed in Table 5.1, where the tolerance is below 0.030 mm. The action segments of the insertion from the data processing are used to train the skill model. The sensed wrench is regarded as the input of the model, and the corrective velocity is regarded as the output of the model. Since the dimensions of wrench is six, it is difficult to

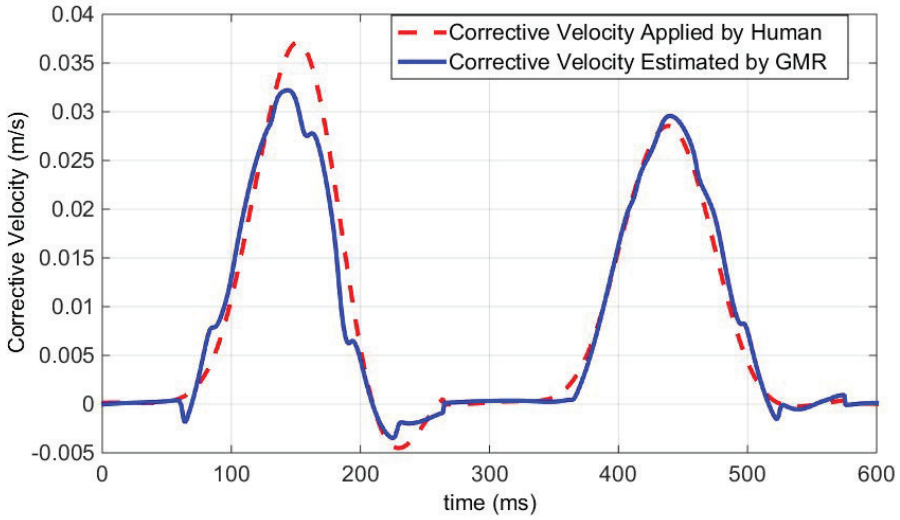


Figure 5.12: The GMR model for the assembly scenario

Table 5.1: The specification of peg-hole insertion testbed

	Material	mm (SI)	inch (UI)	Tolerance
Hole	T6101	$\phi 25.400^{+0.003}_{-0.000}$	$\phi 1.000^{+0.0001}_{-0.0000}$	H7
Peg	T6101	$\phi 25.400^{+0.000}_{-0.030}$	$\phi 1.000^{+0.0012}_{-0.0000}$	h7

Table 5.2: The robot execution result in peg-hole insertion

Total trials	Success	Fail	Success Rate(%)
50	48	2	96.00%

visualize the model. An insertion action segment is used as a query data, and the comparison of the skill model output and the original demonstration is shown in Fig. 5.12. The red-dash line is the corrective velocity from human demonstration, while the blue solid line is the velocity reference generated by the skill model. The robot has completed 50 trials in the experiment. The statistical result for the robot reproduction is shown in Table 5.2. The robot achieves the 96% success rate in the H7/h7 peg-hole testbed. The two failure cases is caused by the fact that the peg is tilted before insertion.

Grinding Scenario

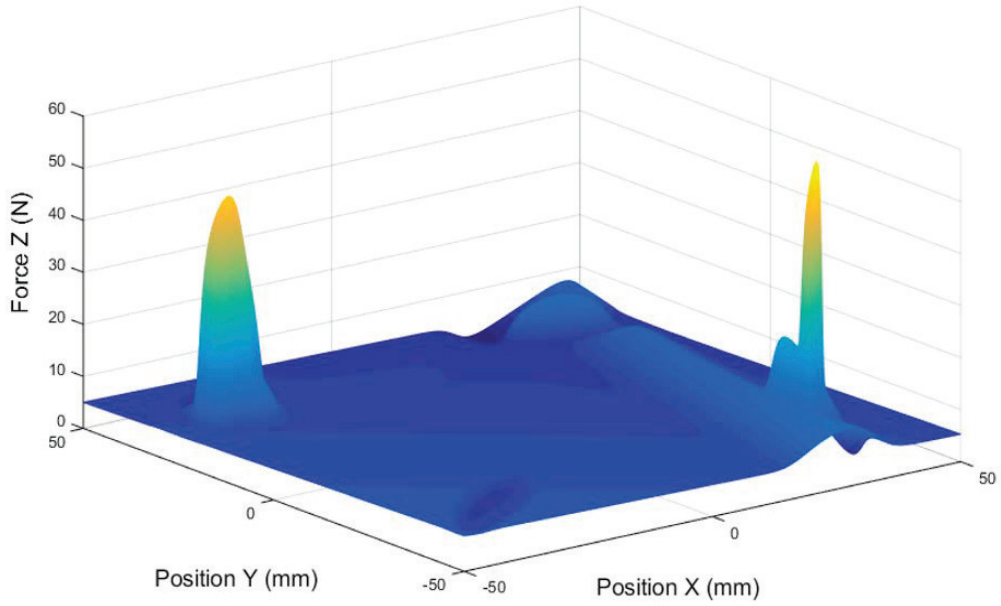
In the grinding scenario, the demonstrator performs the grinding task as shown in Fig 5.7c. For the safety consideration and the process simplification, an aluminum plate represents a grinder in this experiment. Also, the demonstrator only presses large forces at two specific regions of the workpiece, and moves on the surface with small force. The skill model is trained by the demonstration, where the demonstrator's pose is the input and the pressing force is the output of the model. The skill model of the demonstrator's force distribution on the workpiece is shown in Fig. 5.13a. The orange dash line in Fig. 5.13b is the force reference that sliced in the diagonal direction from Fig. 5.13a, and the blue solid line is the force that the robot applies on the workpiece along the path. The result indicates that the robot learns the human's intension in applying the different forces along the trajectory.

5.5 Chapter Summary

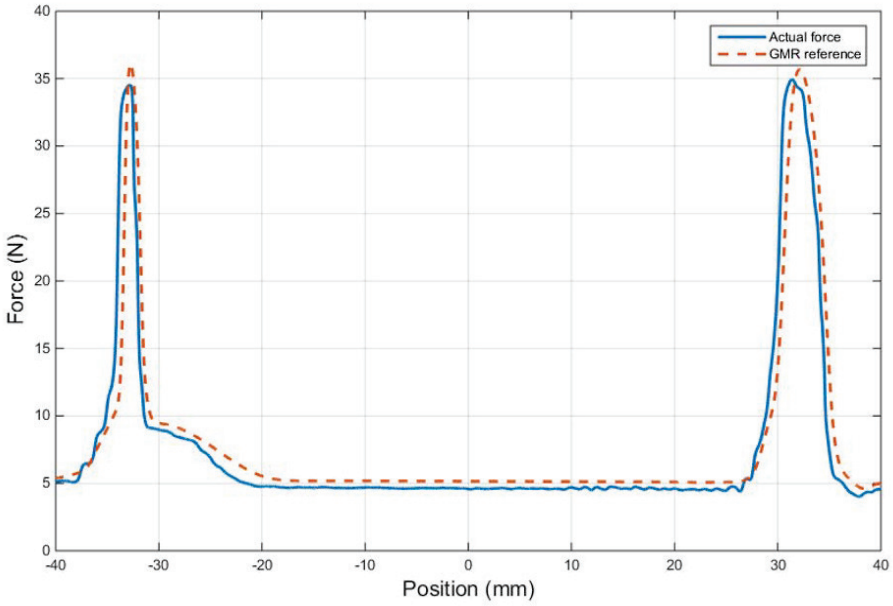
In this chapter, a novel framework of the remote lead through teaching (RLTT) was introduced to simplify the robot programming problem. The idea of RLTT is to design a common reference between human demonstration and robot reproduction. Hence, the demonstration data from human can be directly utilized in the robot reproduction.

This chapter also introduced the skill learning process by three steps. First, the dynamic time warping (DTW) synchronizes the demonstration data in the same time horizon. Second, a support vector machine (SVM) based action classifier is designed to decompose the demonstration data into several action segments. Lastly, the Gaussian mixture regression (GMR) is used to train the human skill model.

Two experimental verifications in the classical industrial application scenarios were given. In the assembly scenario, the robot has achieved 96% success rate in H7/h7 peg-hole insertion, where the tolerance was below 0.030 mm. In the grinding scenario, the robot successfully imitated the human behavior to press large forces in the desired spots.



(a)



(b)

Figure 5.13: The skill model for the grinding scenario. (a) The grinding force reference distribution over the workpiece trained by GMR. (b) The force reference of the skill model and the actual robot force along the desired path.

Chapter 6

Robot Grasp Transferring by Non-Rigid Transformation

6.1 Introduction

Grasping is an essential capability for robots to accomplish complex manipulation tasks. In traditional grasping scenarios, such as pick-and-place in assembly lines, object-specific grippers might be designed for the robust grasping of a single type of objects. In recent years, however, more and more applications require a versatile ability for robots to grasp various objects with general purpose grippers. For example, human robot interaction requires collaboration and assistance between robots and humans, during which robots may pass different tools to humans or help holding various workpieces for assembly tasks. The increasing demand for massive customization and warehouse automation also promotes the development of dexterous grasping.

However, the grasp planning for various objects with general purpose grippers is challenging to solve due to heavy computational loads, large task variance and imperfect perceptions. First of all, many analytic planners such as Ferrari-Canny metric [39] and grasp isotropy [53] require considerable time for exhaustive searching and complex computation for evaluation. Secondly, these planners generally assume point contact, and calculate the quality based on the local features such as contact position and contact normal, while the global task requirements such as robot reachability and collision avoidance are not under consideration. Moreover, analytic planners are usually sensitive to the noises and distortion of point clouds caused by hardware limitations and calibration errors. Therefore, the grasp quality evaluated by analytic planners is usually inconsistent with the empirical success rate and cannot resemble reality effectively [15].

Another common approach for grasp planning is to learn optimal grasps from previous grasp examples. For example, the Dex-Net [70] trains a neural network from a database which is built by analytic planners. The network is able to estimate optimal grasps for unseen objects after training. In [101], the grasp is calculated from heatmaps that generated by deep learning. However, these methods usually require considerable data for the training process and the optimal grasp is planned without considering the task constraints.

Despite the variance of object shapes, we notice that objects to grasp can be classified into several categories. For example, in the tool picking scenario, objects can often be specified into categories such as wrenches, pliers, and screwdrivers. Objects in each category share similar topological structures but may have difference in shapes and sizes.

Some research has been conducted based on this observation. In [19], the perceived cloud of the object is fitted to different objects templates in the database, and the grasp is estimated by superimposing all representations considering their confidence levels. A semantic grasping is proposed in [30] to consider task requirements. The task constraints are implicitly represented by a grasp example in each object category and the desired grasp on the novel object is retrieved by mapping the grasp example and refined by eigen-grasp planner. A dictionary of object parts is learned in [33] to generate grasps across partially similar objects. The dictionary assumes that the segments that shared by objects are rigid and have similar sizes. However, this assumption cannot hold in many scenarios.

In this work, we propose a novel framework for efficient and effective grasp generation from previous grasp examples[1]. Firstly, a 'learning from human demonstration' approach is introduced to teach robots candidate grasping poses by human experts. In the test stage, the category of the target object will be classified by its similarities towards the taught objects. Then a grasping pose transferring is performed between similar objects based on the concept of coherent point drift method [75, 76]. Moreover, the transformed poses will be rated by analyzing the grasp isotropy metric [53]. An orientation search method will also be introduced to improve the robot reachability and avoid collisions.

The remainder of this chapter is organized as follows: Section II introduces the normal formulation of grasping problems and the benefits of involving human demonstration. The background of coherent point drift, together with its application on grasping pose transferring is introduced in Section III. Section IV presents the dissimilarity measure between objects and the refinement of poses after transferring. A series of experiments on grasping multiple categories of objects are shown in Section V. Experimental videos can be found in [36]. Section VI concludes the chapter.

6.2 Grasp Planning with Human Demonstration

A basic grasp planning for parallel grippers can be formulated as

$$\max_{\boldsymbol{c},\boldsymbol{n_c}} Q(\boldsymbol{c},\boldsymbol{n_c}) \tag{6.1a}$$

$$s.t. \quad c_i \in \partial O \quad i=1,2 \tag{6.1b}$$

$$\|c_1 - c_2\| \leq w_{\max}, \tag{6.1c}$$

where Q denotes the grasp quality to be maximized, $\boldsymbol{c} = \{c_1, c_2\}$ denotes the contact pair with $c_i \in \mathbb{R}^D$, and $\boldsymbol{n_c} = \{n_{c,1}, n_{c,2}\}$ denotes the normals of the contact pair with $n_{c,i} \in \mathbb{S}^{D-1}$. Con-

[1]This work published in [61] was co-authored with Dr. Te Tang. The similar content may be appeared in his dissertation.

straint (6.1b) shows the contacts should lie on the surface of object ∂O, and (6.1c) shows that the distance of the contact pair should be less than the width of the gripper $w_{\max}$.

Equation (6.1) is challenging to solve gradient based methods because of the high complexity of surface modeling, the discrete representation of surface points, and the discontinuity of surface normal. Compared with gradient based searching, the sampling based method is able to adapt to discrete object representation and escape from local optimum. However, it requires considerable computation for sampling and quality evaluation to find a reasonable grasp due to the complicated structure of the object and the feasibility constraints such as gripper width, task requirements and collisions, thus the direct sampling method is generally not affordable for real-time implementation.

We assume that the objects to grasp can be clustered into various categories. The objects in the same category share similar topological structures but can have different shapes, sizes and configurations. The objective of this work is to provide an efficient framework to grasp objects in the same category without overwhelmed training, modeling and computation. To achieve this, we introduce human demonstration to accelerate grasp searching by providing heuristics to guide sampling. Instead of directly using human demonstration as the sampling pool for the target object to grasp, we use a mapping function to transfer the example grasps based on the topological similarity between the source object and the target object. Therefore, (6.1) becomes:

$$\max_{\boldsymbol{c},\boldsymbol{n_c}} Q(\boldsymbol{c},\boldsymbol{n_c}) \tag{6.2a}$$

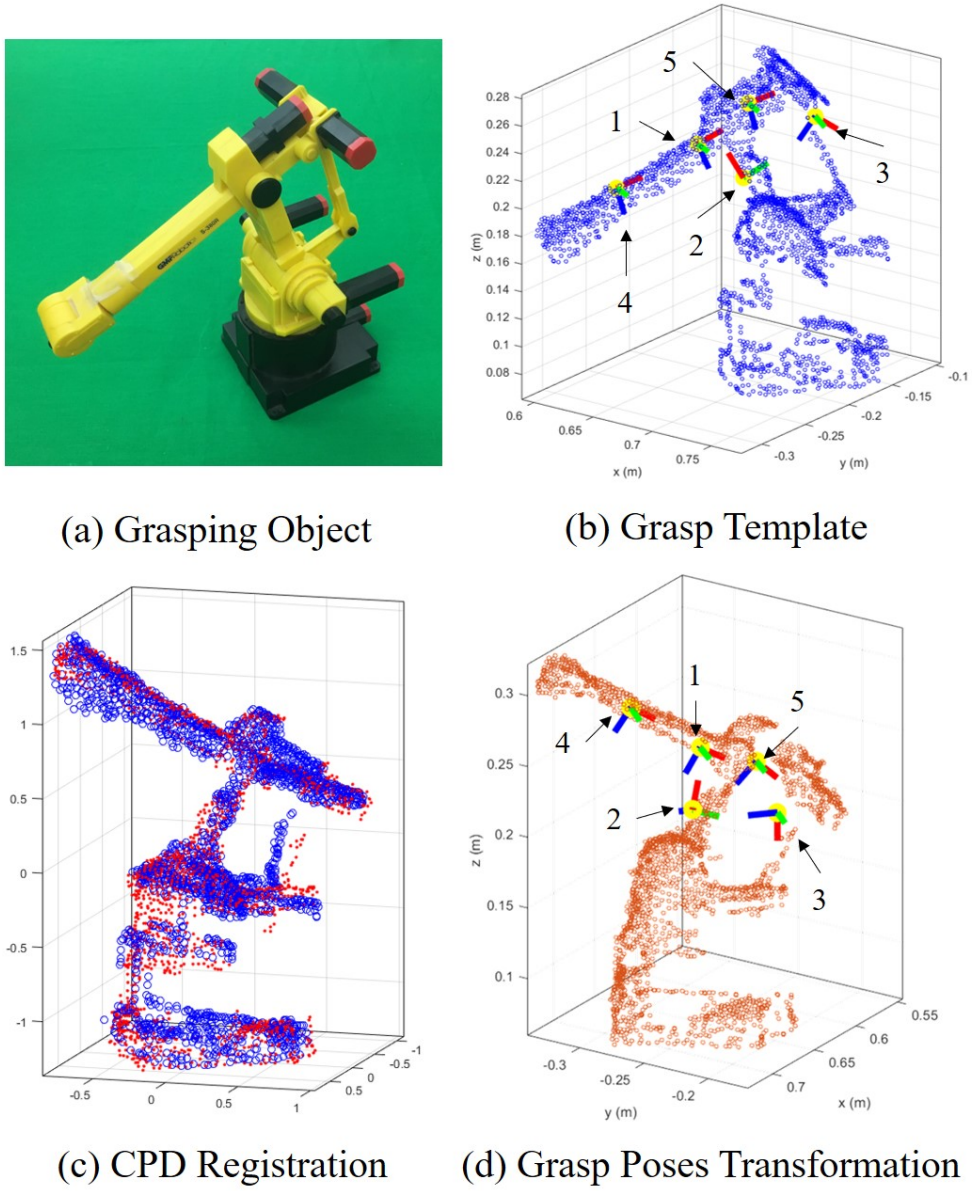
$$s.t. \quad \{\boldsymbol{c},\boldsymbol{n_c}\} \in map(\mathcal{H}) \tag{6.2b}$$

$$\|c_1 - c_2\| \leq w_{\max}, \tag{6.2c}$$

where $\mathcal{H}$ denotes a human demonstration database containing example grasps on the source object, and the function $map(\cdot)$ represents a grasp transferring. Compared with (6.1), the introduction of human demonstration in (6.2) has the following advantages. First, incorporating human intelligence into the framework will improve the empirical success rate, since the human demo usually considers a variety of factors such as the local structure of the object and the global geometry for collision avoidance. Second, some tasks have special requirements. For example, some workpieces have fragile parts or polished surfaces which are not suitable for grasping. Some workpieces have some preferred grasping poses for the ease of following assembly procedures. Explicitly imposing such constraints to traditional approaches is nontrivial, while these requirements can be easily encoded by human demonstration. Moreover, by mapping the grasp examples to novel objects, the proposed method exploits much fewer but reasonable grasp samples compared to traditional exhaustive search methods. Therefore, the searching time is greatly reduced.

6.3 Grasping Pose Transferring by Point Registration

Assume a grasp template consists of a source object (Fig. 6.1a) and multiple demonstrated grasping poses (Fig. 6.1b), where the blue dots are the point clouds of the source object and each coordinate labeled with a number represents a demonstrated grasping pose. The source object is represented



(a) Grasping Object (b) Grasp Template

(c) CPD Registration (d) Grasp Poses Transformation

Figure 6.1: The Grasping Pose Transferring: (a) A toy manipulator model as a grasping object. (b) The grasp example that contains a source object and several grasping poses. (c) The non-rigid point registration by Coherent Point Drift. (d) The target object with the warped grasping poses. The grasping poses are labeled with number.

by a point set $\mathbf{X} = (x_1, \cdots, x_N) \in \mathbb{R}^{N \times D}$, where $x_n \in \mathbb{R}^D$ is the n-th point in the point set. The grasping poses are denoted as $\boldsymbol{g}_i = (\boldsymbol{t}_i, \boldsymbol{R}_i) \in \mathbb{R}^D \otimes \mathbf{SO}(D)$, $i = 1, 2, \cdots, I$, where $\boldsymbol{t}_i \in \mathbb{R}^D$ is the center of the grasping point, $\boldsymbol{R}_i \in \boldsymbol{SO}(D)$ represents the grasping orientation, and i is the index among the total I grasping poses. The target object is represented by another point set $\mathbf{Y} = (y_1, \cdots, y_M) \in \mathbb{R}^{M \times D}$, where $y_m \in \mathbb{R}^D$ is the m-th point in the target point set. Our objective is to find a smooth transformation $\mathcal{T} : \mathbb{R}^D \to \mathbb{R}^D$ that maps the source object to the target object as well as transferring the grasp examples to new grasping poses $\boldsymbol{g}'_i = (\boldsymbol{t}'_i, \boldsymbol{R}'_i)$ on the target object (Fig. 6.1d).

The transformation can be found by aligning the source object to the target object. Then the task can be formulated as a point set registration problem as shown in Fig. 6.1c. Considering variation and deformation between the source object and the target object, the mapping should have more flexibility than rigid transformation. In the meantime, the topological structure of point sets must be preserved during the alignment process so that the the grasping pose can be transferred to a reasonable location. In this work, we use the coherent point drift (CPD) algorithm [75] to perform a smooth non-rigid registration.

Coherent Point Drift

In order to align the source object toward the target object, CPD considers source points in $\mathbf{X}$ as the centroids of Gaussian mixtures, and transforms them to fit the target points in $\mathbf{Y}$ coherently. The source points are assumed to deform toward the target points according to a continuous displacement field $v(\cdot)$, and the transformed source point is written as

$$\mathcal{T}(x_n) = x_n + v(x_n). \tag{6.3}$$

The goal of CPD is to retrieve the displacement field v that maximizes the likelihood of $\mathbf{Y}$ sampled from $\mathbf{X}$.

With the Gaussian mixture model, the probability distribution of y_m can be described as

$$\begin{aligned} p(y_m) &= \sum_{n=1}^{N} p(n) p(y_m|n) \\ &= \sum_{n=1}^{N} \frac{1}{N} \mathcal{N}(y_m; \mathcal{T}(x_n), \sigma^2) \\ &= \sum_{n=1}^{N} \frac{1}{N} \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left(-\frac{\|y_m - x_n - v(x_n)\|^2}{2\sigma^2}\right), \end{aligned} \tag{6.4}$$

where it is assumed that each Gaussian shares the same isotropic covariance σ^2 and has equal membership probability $p(n) = 1/N$.

Since there might be some noise and outliers from the measurement, which may deteriorate the result of registration, an additional uniform distribution is added to the mixture model to take

account of these effects. Thus, the complete mixture model is reformulated as

$$\begin{aligned} p(y_m) &= \sum_{n=1}^{N+1} p(n)p(y_m|n) \\ &= (1-\mu)\sum_{n=1}^{N}\frac{1}{N}\mathcal{N}(y_m;\mathcal{T}(x_n),\sigma^2)+\frac{\mu}{M}, \end{aligned} \tag{6.5}$$

where $\mu \in [0,1]$ denotes the weight of the uniform distribution. The log-likelihood function of $\mathbf{Y}$ is given by

$$\begin{aligned} l(v,\sigma^2) &= \log\prod_{m=1}^{M} p(y_m) \\ &= \sum_{m=1}^{M}\log\sum_{n=1}^{N+1} p(n)p(y_m|n). \end{aligned} \tag{6.6}$$

The parameter (v,σ^2) can be estimated by maximizing (6.6); however, it is nontrivial to directly optimize over the log-likelihood function, since the summation inside the $\log(\cdot)$ leads to a non-convex formulation. An alternative log-likelihood function L can be constructed as

$$L(v,\sigma^2) = \sum_{m=1}^{M}\sum_{n=1}^{N+1} p(n|y_m)\log\left(p(n)p(y_m|n)\right). \tag{6.7}$$

It can be proven by Jensen's inequality [56] that L is the lower bound of l. Hence, increasing the value of L will always 'push' the value of l increased until it reaches the local optimum. Compared with the structure of l, the inside summation of L is moved to the front of the $log(\cdot)$ function, which provides much convenience to maximize the log-likelihood by the EM algorithm [32].

The EM algorithm runs the expectation step (E-step) and maximization step (M-step) iteratively to estimate the parameters by maximizing L.

E-step: The expectation step computes the posterior probability distribution of $p(n|y_m)$ with the previous estimated parameters from the last M-step,

$$p(n|y_m) = \frac{\exp\left(-\frac{\|y_m-x_n-v(x_n)\|^2}{2\sigma^2}\right)}{\sum_{n=1}^{N}\exp\left(-\frac{\|y_m-x_n-v(x_n)\|^2}{2\sigma^2}\right)+c}, \tag{6.8}$$

where $c = (2\pi\sigma^2)^{D/2}\frac{\mu}{(1-\mu)}\frac{N}{M}$.

M-step: Ignoring the terms that are independent of v and σ^2, the log-likelihood function can be written as

$$\begin{aligned} L(v,\sigma^2) = &-\frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{m=1}^{M} p(n|y_m)\|y_m-x_n-v(x_n)\|^2 \\ &-\frac{D}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} p(n|y_m)\log\sigma^2. \end{aligned} \tag{6.9}$$

The maximization step is to substitute (6.8) into (6.9) and take partial derivative with respect to v and σ^2 to find its maximum.

Although alternating between E-step and M-step will converge to a local optimal, it can not guarantee that the topological structure of the source object is preserved after the transformation. That is because there is no topological constraints to restrict the locations of these Gaussian centroids. Therefore, a regularization term is added to the log-likelihood function to regularize the smoothness of the deformation function, and the modified likelihood function is given by

$$\tilde{L}(v, \sigma^2) = L(v, \sigma^2) - \frac{\lambda}{2}\|v\|_{\mathcal{F}}^2, \tag{6.10}$$

where $\|v\|_{\mathcal{F}}^2 = \int_{\mathbb{R}^D} \frac{|V(s)|^2}{G(s)} ds$ is a norm to quantitatively measure the function smoothness [41]. $V(s)$ is a Fourier transform of v and $G(s)$ presents a symmetric filter that approaches to zero as $s \to \infty$. The overall Fourier domain norm here basically captures the energy of high frequency components of $V(s)$. Intuitively, the larger the norm $||v||_{\mathcal{F}}$, the more 'oscillating' v will be, i.e., less smoothness. $\lambda \in \mathbb{R}^+$ is a weighting coefficient that represents the trade off between the fitting of the point sets and the smoothness constraints on the transformation.

It can be proved by variational calculus that the maximizer of (6.10) has the form of the radial basis function [75],

$$v(z) = \sum_{n=1}^{N} w_n g(z - x_n), \tag{6.11}$$

where $g(\cdot)$ is a kernel function retrieved from the inverse Fourier transform of $G(s)$, and w_n is the unknown kernel weights. In general, $g(\cdot)$ can be any formulation with positive definiteness, and $G(s)$ behaves like a low-pass filter. For simplicity, a Gaussian kernel is chosen so that $g(z - x_n) = \exp(-\frac{1}{2\beta^2}||z - x_n||^2)$, where $\beta \in \mathbb{R}^+$ is a parameter that defines the width of smoothing Gaussian filter. Larger β corresponds to more rigid transformation, whereas smaller β produces more local deformation.

Substituting (6.7) and (6.11) to (6.10), we get

$$\begin{aligned}\tilde{L} = &\frac{-1}{2\sigma^2} \sum_{n=1}^{N} \sum_{m=1}^{M} p(n|y_m) \|y_m - x_n - \sum_{k=1}^{N} w_k g(x_n - x_k)\|^2 \\ &- \frac{D}{2} \sum_{n=1}^{N} \sum_{m=1}^{M} p(n|y_m) \log \sigma^2 - \frac{\lambda}{2} tr(\mathbf{W}^T \mathbf{G} \mathbf{W}),\end{aligned} \tag{6.12}$$

where $\mathbf{G} \in \mathbb{R}^{N \times N}$ is a Gramian matrix with element $\mathbf{G}_{ij} = g(x_i - x_j)$ and $\mathbf{W} = [w_1, \cdots, w_n]^T \in \mathbb{R}^{N \times D}$ is the vectorization of kernel weights in (6.11).

From (6.12), the regularized log-likelihood function is now parameterized by $(\mathbf{W}, \sigma^2)$. Similar to (6.7), the EM algorithm can be performed to estimate the parameters iteratively. In E-step, the posterior $p(n|y_m)$ is calculated by using the previous estimated parameters. In M-step, take

$\partial\tilde{L}/\partial\mathbf{W} = 0$ and $\partial\tilde{L}/\partial\sigma^2 = 0$ to obtain a new estimate of $(\mathbf{W}, \sigma^2)$. The closed-form solution for M-step requires further mathematical derivation, more details can be found in [75, 76].

After $\tilde{L}$ is converged, the point set of the source object $\mathbf{X}$ can be aligned toward the target object by

$$\mathcal{T}(\mathbf{X}) = \mathbf{X} + \mathbf{GW}. \tag{6.13}$$

Grasp Pose Transferring

As shown in Fig. 6.1d, after finding the mapping from source object $\mathbf{X}$ to target object $\mathbf{Y}$, the demonstrated grasping poses on $\mathbf{X}$ will also be transferred to achieve new grasping poses that are suitable for object $\mathbf{Y}$. The grasping poses can be decomposed to two parts: the position and the orientation of the robot end-effector. With regard to the position, the non-rigid transformation $\mathcal{T} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ can directly map the center of grasp from grasp example to the target object by

$$\boldsymbol{t}_i' \leftarrow \mathcal{T}(\boldsymbol{t}_i), \quad i = 1, 2, \cdots, I. \tag{6.14}$$

As for the orientation, it can be considered as transferring x, y, and z axes of the original grasp orientation to the new object space. One natural way to transform a vector $\mathbf{v}$ at a point $\boldsymbol{t}$ through a function is to multiply the vector with the gradient of $\mathcal{T}(\boldsymbol{t})$ [1], i.e. $\nabla\mathcal{T}(\boldsymbol{t})\boldsymbol{v}$. Considering the properties of the special orthogonal group, the singular value decomposition (SVD) of the matrix is performed to construct the new orientation of the grasp, which is

$$\boldsymbol{R}_i' \leftarrow \mathbf{U}_i\mathbf{V}_i^T, \quad i = 1, 2, \cdots, I. \tag{6.15}$$

where $\mathbf{U}_i\Sigma\mathbf{V}_i^T = svd(\nabla\mathcal{T}(\boldsymbol{t}_i)\boldsymbol{R}_i)$, $\mathbf{U}_i, \mathbf{V}_i$ are the orthonormal basis of the matrix, and Σ is a diagonal matrix that consists of the singular values of the matrix $\nabla\mathcal{T}(\boldsymbol{t}_i)\boldsymbol{R}_i$.

Hence, the new grasping poses on the target can be transferred by

$$\boldsymbol{g}_i' = (\boldsymbol{t}_i', \boldsymbol{R}_i') \leftarrow (\mathcal{T}(\boldsymbol{t}_i), \mathbf{U}\mathbf{V}_i^T), \quad i = 1, 2, \cdots, I. \tag{6.16}$$

6.4 Grasping Poses for Various Objects

Dissimilarity Measure

During the training stage, multiple grasping poses for different categories of objects are demonstrated by human experts. Given a new object at test, it is necessary to first classify which category the object belongs to, then use the Section III method to transfer the corresponding grasping poses from the correct category to get a new feasible grasp. Therefore, an object classifier is essential for pose transferring.

There are some researches that apply surface matching technique to rigidly fit the object template to the measured point clouds and calculate the dissimilarity [79, 54]. The source objects from different categories are exploited as the templates to match the target object. By measuring

the dissimilarity between each of the source objects $\mathbf{X}$ and the target object $\mathbf{Y}$, the most similar pair will be selected to determine the category of the target object. In our work, since CPD can be applied to warp the template $\mathbf{X}$ to $\mathcal{T}(\mathbf{X})$ which is aligned with $\mathbf{Y}$, the residual dissimilarity between $\mathcal{T}(\mathbf{X})$ and $\mathbf{Y}$ instead of the dissimilarity between $\mathbf{X}$ and $\mathbf{Y}$ will be checked to provide a more robust category classification.

The average minimum distance between the two point sets can be designed as:

$$d(\mathcal{T}(\mathbf{X}), \mathbf{Y}) = \frac{1}{N} \sum_{n=1}^{N} \min_{m \in [1,M]} ||\mathcal{T}(x_n) - y_m||, \tag{6.17}$$

where $||\mathcal{T}(x_n) - y_m||$ is the Euclidean distance between point $\mathcal{T}(x_n)$ and y_m. Equation (6.17) is an error function that is commonly used for point cloud alignment. However, (6.17) is asymmetric. The dissimilarity between a source object and a target object can be formulated as

$$D(\mathbf{X}', \mathbf{Y}) = d(\mathbf{X}', \mathbf{Y}) + d(\mathbf{Y}, \mathbf{X}'), \tag{6.18}$$

where $\mathbf{X}' = \mathcal{T}(\mathbf{X})$ is the source points warped toward $\mathbf{Y}$ by CPD. The function $D(\cdot, \cdot)$ sums the two asymmetric dissimilarity measurements together so that D is symmetric to its input arguments, i.e. $D(\mathbf{X}', \mathbf{Y}) = D(\mathbf{Y}, \mathbf{X}')$.

Suppose there are K object categories, the most possible category that the target object belongs to can be estimated by

$$\mathbf{k}^* = \arg \min_{k \in [1,K]} D(\mathbf{X}'_k, \mathbf{Y}). \tag{6.19}$$

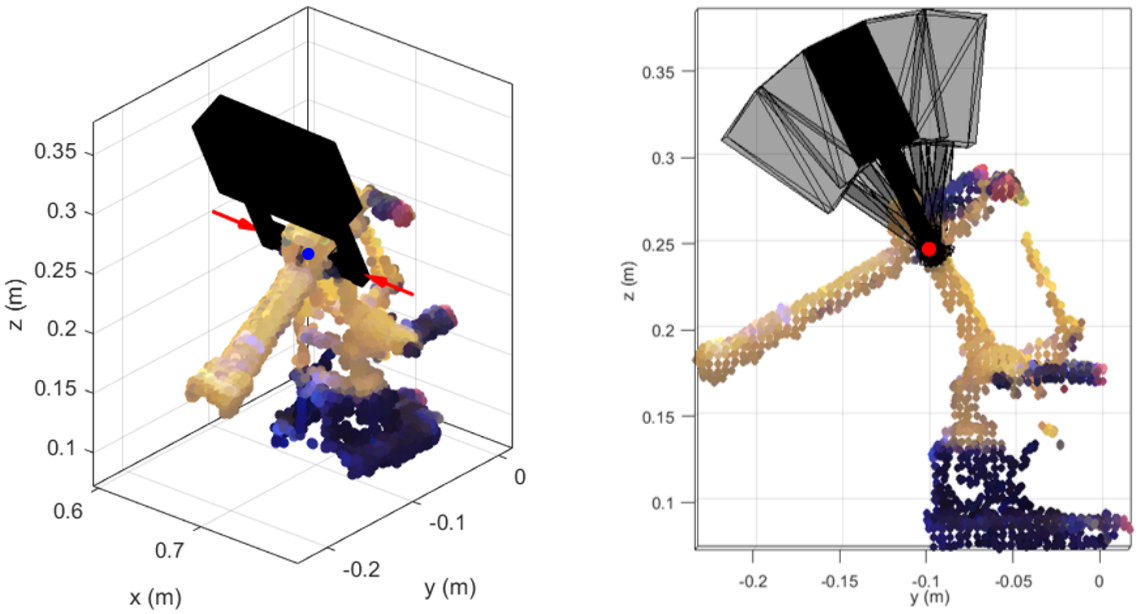
Grasping Pose Optimization

Once the object category is determined, we can map the example poses from the corresponding category to the target object. The grasp quality of the mapped poses will then be evaluated by analytic methods using the grasp isotropy index [53]. The grasp isotropy index measures the uniformness of different contact forces to the total wrench. More concretely, it can be written as

$$Q_i = \frac{\sigma_{\text{min}} \mathcal{G}(\boldsymbol{g}'_i, \boldsymbol{g}_o)}{\sigma_{\text{max}} \mathcal{G}(\boldsymbol{g}'_i, \boldsymbol{g}_o)}, \tag{6.20}$$

where $\boldsymbol{g}_o$ denotes the pose of the object, $\mathcal{G}(\boldsymbol{g}'_i, \boldsymbol{g}_o)$ represents the grasp map determined by the contacts and the object [73], and σ_{min} and σ_{max} respectively denote the minimum and maximum singular values of the grasp map. The contacts are inferred by the line search along the grasp axis. The line search tries to locate the nearest neighbor of the grasp center on the object's point cloud. The contacts are represented by the nearest neighbors search in the positive and negative directions of the grasp axis respectively. The transferred grasp would be treated as a bad pose if the contacts deviate from grasp axis too much, in which case a negative quality will be allocated.

Apart from the grasp quality, we have to consider the feasibility constraints such as the reachability and the gripper-object collision. The feasibility constraints are guaranteed by an orientation search introduced below.



(a) (b)

Figure 6.2: (a) a grasp example, where the red arrows indicate the direction of gripper closing (which is also the grasp axis). (b) The side view of the grasp example, and the transparent grippers are shared the same grasp center and grasp axis but different orientations.

Because the robot grasping pose with a parallel-jaw gripper is composed of a center of grasping and a grasp axis, it does not necessarily restrict all the rotation axis. i.e. the grasping quality is not affected by rotating along the grasp axis. A parallel-jaw gripper grasp example is shown in Fig. 6.2a, where the center of grasping point is the blue dot and the red arrows represent the operational direction of the jaw which parallels to the grasp axis. By rotating along the grasp axis, the grasping pose can be modified as the translucent grippers as shown in Fig. 6.2b, where the modified poses are also valid for grasping. If the initial grasping pose is not feasible, then the modification can be made by searching the various orientations around the initial one. Suppose the initial orientation is denoted as $\boldsymbol{R}_0$, the sampled orientation is denoted as $\boldsymbol{R}_i$, and $\mathcal{R}$ is the set of all the sampled orientations. The orientation search can be formulated as

$$\min_{\boldsymbol{R}_i \in \mathcal{R}} \ \Delta\xi(\boldsymbol{R}_0, \boldsymbol{R}_i) + C\left[f_{IK}(\boldsymbol{t}, \boldsymbol{R}_i) + f_{col}(\boldsymbol{t}, \boldsymbol{R}_i, \mathbf{Y})\right], \tag{6.21}$$

where $\Delta\xi(\boldsymbol{R}_0, \boldsymbol{R}_i) = 1 - \xi(\boldsymbol{R}_0)^T \xi(\boldsymbol{R}_i) \in [0, 1]$ is the rotation deviation in quaternion between $\boldsymbol{R}_0$ and $\boldsymbol{R}_i$, $\xi(\cdot)$ converts a rotation matrix to a quaternion. We use quaternion rather than Euler angles to represent rotation difference to avoid singular representation in rotations. $f_{IK}(\boldsymbol{t}, \boldsymbol{R})$ is a boolean function that returns 1 when the inverse kinematics of $(\boldsymbol{t}, \boldsymbol{R})$ is invalid and returns 0

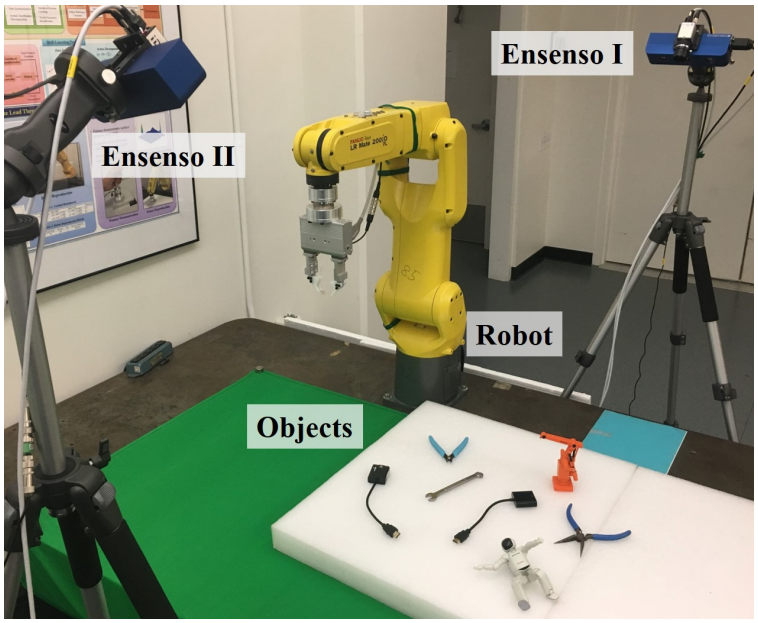


Figure 6.3: The experimental setup of grasp transferring: a FANUC LR Mate 200*i*D/7L and dual Ensenso stereo cameras

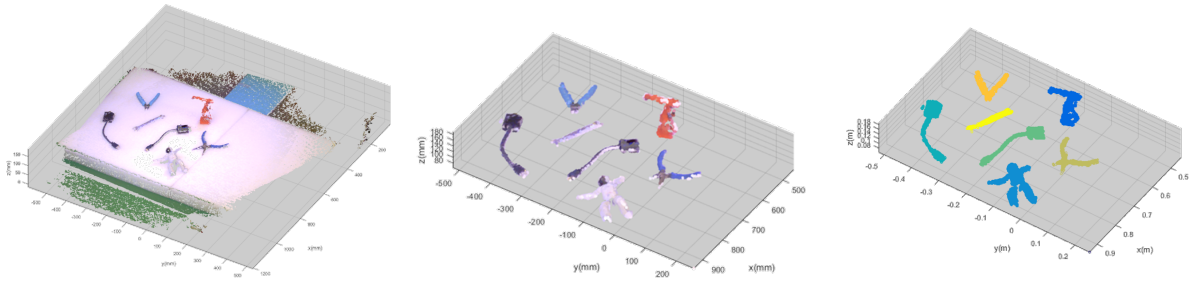
otherwise. $f_{col}(\boldsymbol{t}, \boldsymbol{R}, \mathbf{Y})$ is another boolean function that return 1 when the gripper with the pose $(\boldsymbol{t}, \boldsymbol{R}_i)$ is collided with $\mathbf{Y}$ and returns 0 otherwise. C is a large constant number to penalize the condition of both the infeasible inverse kinematics and the gripper-object collision. If all the sampled orientations are invalid, the value of (6.21) will be greater than or equal to C. Then the orientation search is applied to exploit the other candidates until it finds a feasible grasping pose to perform the task.

6.5 Experimental Results

In order to verify the proposed grasping approach, a series of experiments were conducted to grasp various objects on the robot system that introduced in Chapter 2, where the grasping task is as shown in Fig. 6.3.

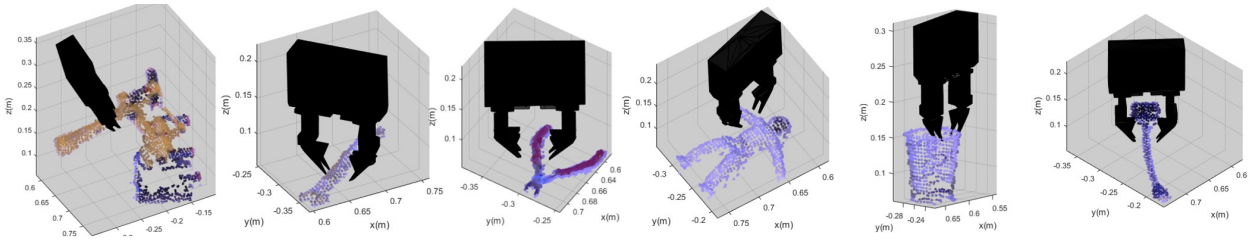
The point clouds retrieved from the dual Ensenso stereo cameras were shown in Fig. 6.4a. By applying the snapshot of the empty workspace as a filter mask, the point clouds of objects were extracted from the background as shown in Fig. 6.4b. Then running the density-based spatial clustering application with noise (DBSCAN) algorithm [35], the point clouds can be separated to several clusters to represent different objects (Fig. 6.4c). A voxel grid filter with step size 5mm was implemented to downsample the point clouds uniformly.

Six categories of objects, including cups, pliers, wrenches, cable adapters, toy manipulator models and toy humanoid models, were tested in the experiment (Fig. 6.5). Note that neither CAD models nor mesh files were used in this work. For each category, a specific source object was



(a) (b) (c)

Figure 6.4: The point cloud process, (a) The raw data was captured by the dual Ensenso stereo camera. (b) The objects were extracted from the background by predefined region. (c) The point cloud was clustered by DBSCAN.



(a) Manipulator (b) Wrench (c) Plier (d) Humanoid (e) Cup (f) Adapter

Figure 6.5: Grasp examples: the first row shows one of the grasp pose on each source object, and the second row provides the snapshots of the actual demonstrated grasping poses.

selected, and the human operator taught multiple preferred grasp poses on it through kinesthetic teaching. The point cloud of the object and the demonstrated grasp poses were recorded as training database.

At the test stage, objects with different sizes and configurations across all the categories were randomly placed in the workspace. For example, multiple types of cups and wrenches were tested for grasping; the pliers were either open or closed; the cable adapter were twisted to various shapes; the joints of the two toy manipulator models and the toy humanoid model were rotated to random angles. All the target objects were shown in Fig. 6.6.

Before running the grasping experiment, an object classification test was performed by mea-

Figure 6.6: Target objects, which are similar to the source objects but different in size, shape, and configuration.

Manipulator – 1	20	0	0	0	0	0
Wrench – 2	0	19	0	0	0	1
Plier – 3	0	2	18	0	0	0
Humanoid – 4	2	0	0	18	0	0
Cup – 5	0	0	0	0	20	0
Adapter – 6	0	2	0	0	0	20
	1	2	3	4	5	6

Figure 6.7: The confusion matrix of object classification. Each column represents a predicted class, and each row represents a actual class.

suring the dissimilarity between the target object and all the source objects (see Section IV.A). The target objects in Fig. 6.6 were randomly placed, with each category of objects collecting 20 different configurations. The parameters of CPD were set as $\beta = 2$ and $\lambda = 50$.

As shown in Fig. 6.7, the performance of object classification was presented by a confusion matrix, where each column represented the predicted class and each row represented the actual class. The diagonal entries of the confusion matrix indicated the correct classification, whereas the off-diagonal entries were misclassification. The overall classification accuracy was 94.17% (113/120).

Table 6.1: Grasping Quality Evaluation

Grasping Pose No.	1	2	3	4	5
Isotropy Index	0.0098	-1.000	0.0001	0.0089	-1.000

Table 6.2: Grasping Results

class	success/trials	avg. CPD time (ms)	avg. numbers of points
manipulator	19/20	1276.4	1563.7
wrench	20/20	111.3	316.7
plier	18/20	706.1	1419.0
humanoid	17/20	369.5	773.3
cup	20/20	350.5	609.3
adapter	19/20	480.2	917.0
average	18.8/20	549.0	933.2

Each category of objects was tested 20 times for grasping with different orientations, shapes, sizes, and configurations. The parameters of CPD were the same as the ones in object classification.

Take one target object (Fig. 6.1d) as an example. The grasping qualities of the transferred grasps are provided in Table 6.1. Note that the qualities of the second and fifth transferred poses are marked as negative based on the isotropy index analysis, since the second pose was mapped to a region with sparse points, and the contacts for the the fifth grasp was wider than the width of the gripper. The remaining pose with the highest grasping quality, i.e., the first pose, was selected. The selected pose was then refined by the orientation search to improve the reachability and avoid collision. The final grasp performed in the experiment is shown in Fig. 6.8b. The grasp was regarded as success when the object could be robustly lifted up at least 10 cm without slipping. The success rate, average computation time and average point numbers for each category of objects are provided in Table 6.2. The experimental video can be found at [36]. The snapshots of grasping experiments are shown in Fig. 6.8.

Although the shapes and configurations of target objects were different to the ones of the source object, they shared the similar structures. Therefore, the grasping poses on the source object could be transferred to reasonable locations on the target objects. For instance, the grasping poses on the various toy manipulator models were invariant in terms of topological structures (see the first row of Fig. 6.8). The grasping poses taught by kinesthetic teaching had the intuition from human such as the task specific consideration and fairly good grasping quality, and CPD transferred the insight to the target objects. Therefore, the test can be successful in most of the cases.

The failure case happened when there was a very large distortion to transform the source object to the target, which degraded the accuracy of the transformation estimated by CPD. As a result, the grasping pose was not accurately transformed, which caused the grasp failed. Although CPD did

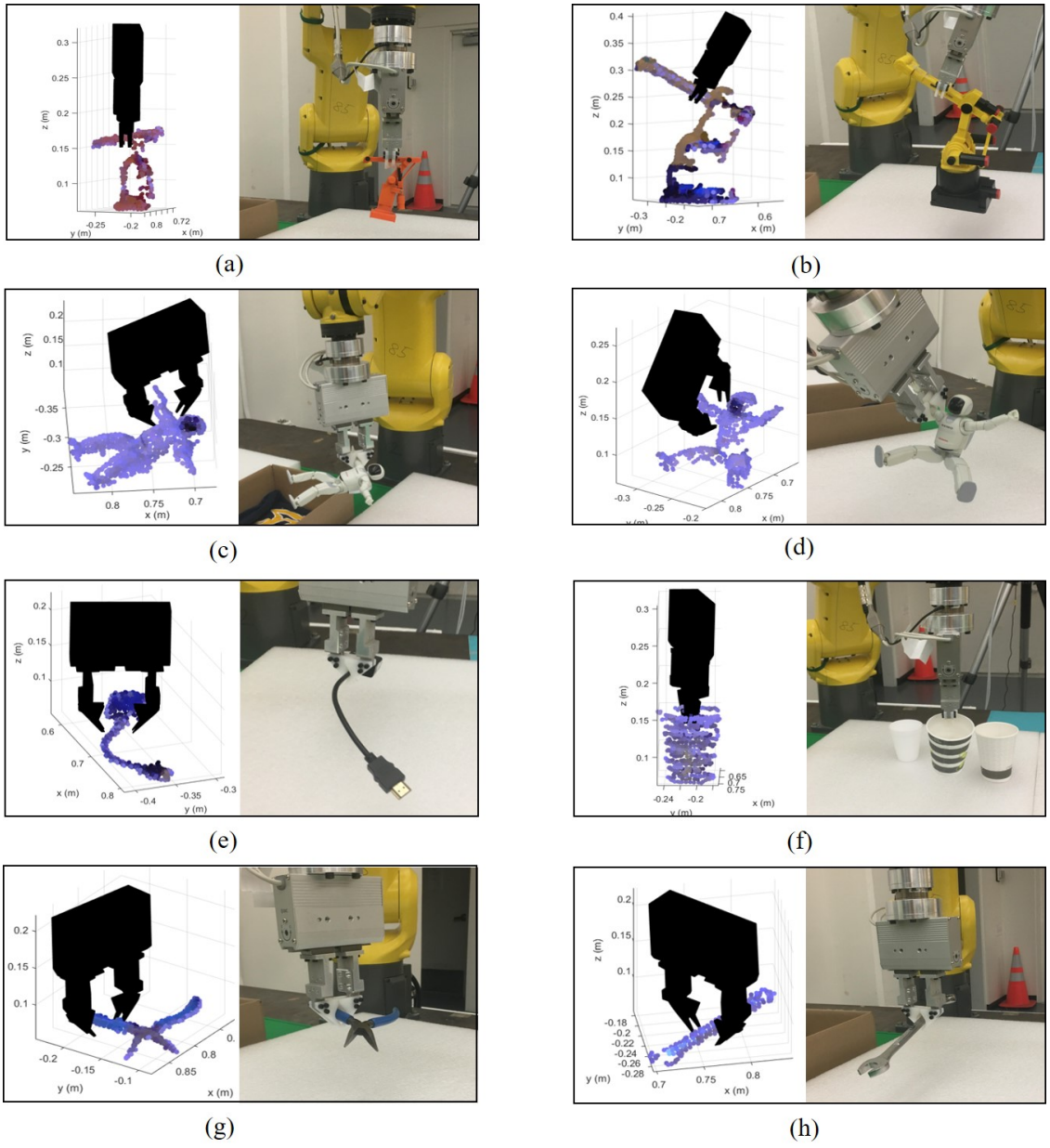


Figure 6.8: The planned grasping poses and the corresponding snapshots of the grasping results.

not transfer grasping poses with high accuracy in this situation, it provided a relatively close one. In the future, we may include an adaptation on the warped grasping pose to avoid this failure.

6.6 Chapter Summary

This Chapter proposed a framework for efficient grasp generation by combining analytic approach with learning from demonstration. A database containing multiple categories of source objects with demonstrated grasping poses were constructed by human experts. During the test scenario, a novel object was firstly classified into one of the example categories by measuring its dissimilarity to each source object. Then the grasping poses on the most similar source object were transferred to the novel object by the coherent point drift (CPD) method. All the transferred grasping poses were evaluated and sorted by the grasp isotropy metric. The selected pose was further refined by an orientation search mechanism, which improves the robot reachability and avoids collision. A series of experiments were performed to grasp six categories of objects with various shapes, sizes and configurations. The average success rate was 18.8 out of 20 grasp trials. The experimental video is available at [36].

Part III

Planning

Chapter 7

Fast Robot Motion Planning by Convex Feasible Set

7.1 Introduction

Robot motion planning has been a popular topic for several decades. Most researches address the two key factors: safety and efficiency. Safety indicates the protection for the robot system from any risk of collision. Several sampling-based methods [59, 49] and optimization-based approaches [85, 80] have been developed to plan collision-free trajectories for robots. Efficiency refers the minimization of the cost such as control input and operation time. Some algorithms[28, 87] are proposed to find the time-optimal trajectory on a specified path.

However, this field is still open for research especially from the viewpoint of computationally efficient motion planning because of the increasing demand of massive customization and the growing application of human-robot interaction (HRI). Massive customization is unlike conventional massive production, where every product looks very similar but slightly different. However, the whole trajectory is required to be reprogrammed due to these subtle changes, which is nontrivial and time-consuming. On the other hand, HRI requires the robot to frequently re-plan its motion so that it can safely interact with human in a dynamic environment. Both applications show the need of a fast robot motion planning, which helps the robot to efficiently generate new motion trajectories to adapt to varying environment.

In this chapter, the framework of fast robot motion planner (FRMP) is proposed as shown in Fig.7.1. It has two layers, the trajectory planning layer and the temporal optimization layer, where each layer deals with safety and efficiency respectively. Trajectory planning is to plan a collision-free trajectory from a given reference trajectory or several way points, whereas the temporal optimization is to minimize the cycle time over the planned trajectory. This chapter utilizes optimization-based algorithms to deal with the problems in both layers in FRMP. The optimization-based motion planning methods, such as model-predictive control (MPC)[22], often need to face highly nonlinear dynamics constraints and highly non-convex constraints for obstacle avoidance, which make it difficult to solve the problem efficiently.

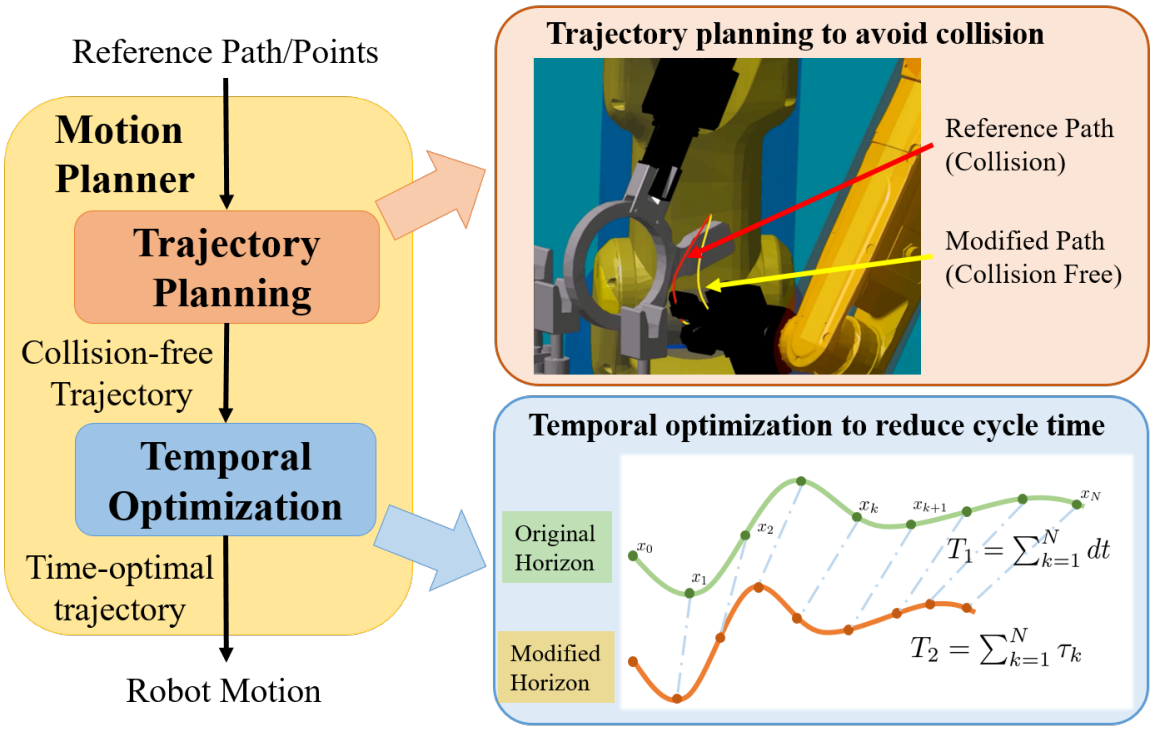
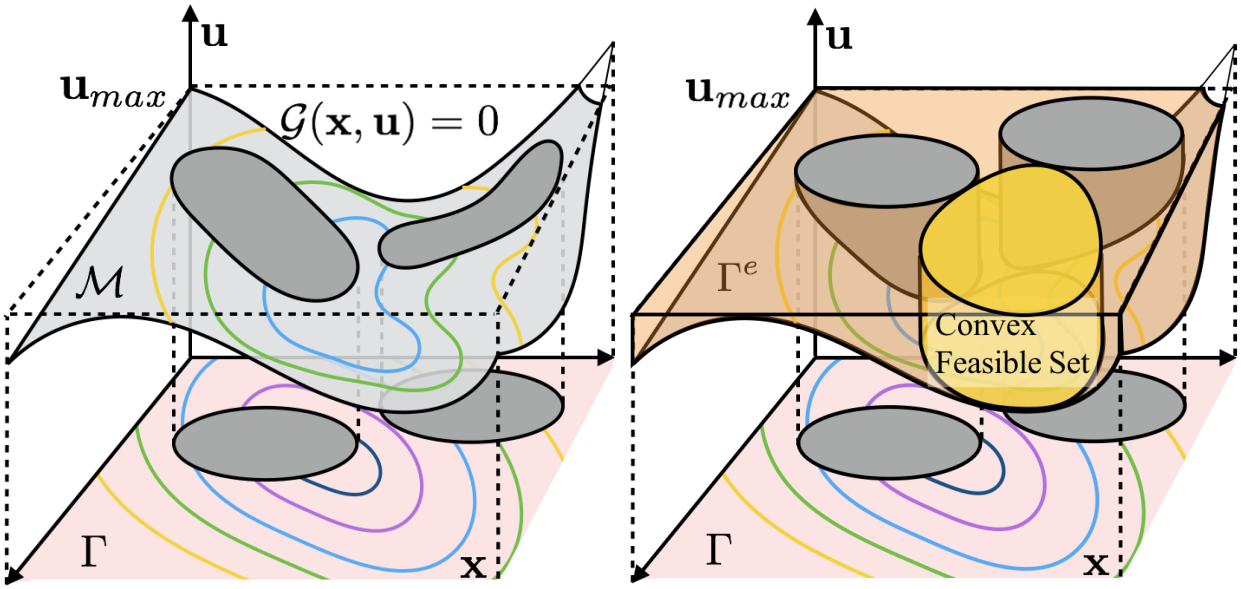


Figure 7.1: The framework of fast robot motion planner (FRMP)

Convexification is a common technique to transform a non-convex problem into a convex one. One of the most popular convexification method is the sequential quadratic programming (SQP)[96, 77], which iteratively solves a quadratic subproblem obtained by quadratic approximation of the Lagrangian function and linearizion of all constraints. The method has been successfully applied to robot motion planning as discussed in [48] and [93]. However, SQP is designed for general purposes, and it often takes multiple iterations to find the solution.

Considering the specific geometric structure in motion planning problems, the convex feasible set algorithm (CFS)[66] and the slack convex feasible set algorithm (SCFS)[67] are proposed for the real time motion planning, a successful application to path planning for autonomous vehicles is given in [68]. FRMP follows the similar algorithmic structure to plan a collision-free trajectory for robot manipulators. Moreover, in order to achieve the time optimality in a short computational time, temporal optimization is formulated into another CFS problem and solved in a fast manner.

The rest of this chapter is organized as follow. Section 7.2 reviews the convex feasible set algorithm for trajectory planning. Section 7.3 firstly formulates the temporal optimization problem, then introduces how to apply CFS to this problem. Section 7.4 compares the performance between CFS and SQP in trajectory planning and provides the experimental results of FRMP conducted on FANUC LR Mate 200*i*D/7L. Section 7.5 concludes the chapter.



(a) (b)

Figure 7.2: Illustration of (a) The geometric structure of the trajectory planning problem (b) The convex feasible set in the trajectory space

7.2 Trajectory Planning

Problem Formulation

Denote the state of the robot as $x \in \mathcal{X} \subset \mathbb{R}^n$, where $\mathcal{X}$ is the feasible configuration state space of the robot and n is its dimension; $u \in \mathcal{U} \subset \mathbb{R}^m$ is the the control input of the robot, where $\mathcal{U}$ is the constraints of u and m is its dimension. In general robot trajectory planning, the input constraint is often formulated as a box constraint, e.g. $-u_{max} \leq u \leq u_{max}$, where u_{max} is the maximum magnitude of the control input.

Suppose the robot needs to move from the initial position to the goal position, its discrete-time trajectory is denoted as $\mathbf{x} = \left[x_0^T, x_1^T, \cdots, x_N^T\right]^T \in \mathcal{X}^{N+1}$, where $x_t \in \mathcal{X}$ is the robot state at time step t and N is the horizon. The sampling time is defined as dt. The reference trajectory and the reference state at time step t are denoted as $\mathbf{x}^r \in \mathcal{X}^{N+1}$ and $x_t^r \in \mathcal{X}$, respectively. Similarly, $\mathbf{u} = \left[u_0^T, u_1^T, \cdots, u_{N-1}^T\right]^T \in \mathcal{U}^N = \Omega$ is the control input for the whole trajectory.

The Cartesian space occupied by the robot body at the state x_t is denoted as $C(x_t) \in \mathbb{R}^3$, whereas the area occupied by the obstacles in the environment at time t is denoted as $\mathcal{O}_t \in \mathbb{R}^3$. Suppose the Euclidean distance between p_A and p_B in the Cartesian space is denoted as $d_E(p_A, p_B) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$, then the minimum distance between the robot and the obstacles is given by $d(x_t, \mathcal{O}_t) := \min_{p_R \in C(x_t), p_O \in \mathcal{O}_t} d_E(p_R, p_O)$.

With the notation above, the robot trajectory planning problem can be formulated as a general

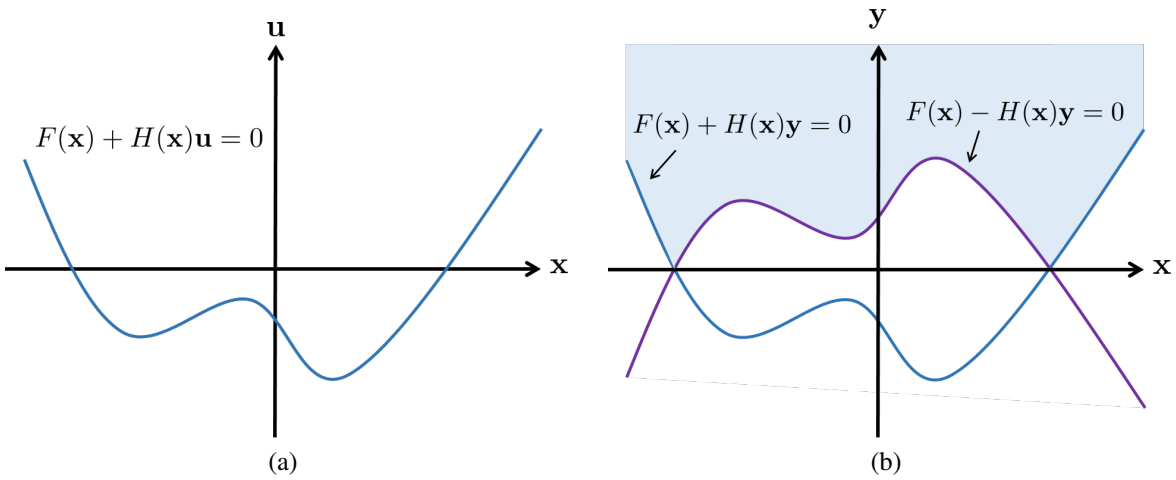


Figure 7.3: Illustration of (a) the nonlinear equality constraint and (b) the nonlinear inequality constraints with slack variables

non-convex optimization problem,

$$\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x},\mathbf{u}) = w_1\|\mathbf{x}-\mathbf{x}^r\|_Q^2 + w_2\|\mathbf{u}\|_R^2 \tag{7.1a}$$

$$s.t. \quad x_t \in \mathcal{X}, u_t \in \mathcal{U}, \tag{7.1b}$$

$$x_t = f(x_{t-1}, u_{t-1}), \tag{7.1c}$$

$$d(x_t, \mathcal{O}_t) \geq d_{min}, \quad \forall t = 1, \cdots, N \tag{7.1d}$$

Equation (7.1a) is designed to be a quadratic cost function for the task performance, where $w_1, w_2 \in \mathbb{R}_+$, $\|\mathbf{x}-\mathbf{x}^r\|_Q^2 := (\mathbf{x}-\mathbf{x}^r)^T Q(\mathbf{x}-\mathbf{x}^r)$ penalizes the deviation of the planned trajectory from the reference trajectory, and $\|\mathbf{u}\|_R^2 = \mathbf{u}^T R\mathbf{u}$ penalizes the control effort over the trajectory. Note that the matrices Q, R are designed to be positive definite. Equation (7.1b) are the feasible constraints of the state and the input, e.g. joint limits, singularity points, and saturation of the control input. Equation (7.1c) is the dynamics constraints. Equation (7.1d) describes the collision avoidance constraints, where $d_{min} \in \mathbb{R}_+$ is the safety distance margin.

The Geometric Structure of Trajectory Planning

Combining the feasible state constraint in 7.1b and the safety constraint 7.1d together, the feasible trajectory constraint in the trajectory space is given by $\mathbf{x} \in \Gamma = \mathcal{X}^{N+1} \cap \mathcal{D}$, where $\mathcal{D}$ describes the safety set in the trajectory space. Moreover, the dynamics constraint in (7.1c) can be rewritten as

$\mathcal{G}(\mathbf{x}, \mathbf{u})$ in the trajectory space. Hence, the robot trajectory planning problem can be rewritten as

$$\min_{\mathbf{x},\mathbf{u}} \quad J(\mathbf{x}, \mathbf{u}) \tag{7.2a}$$

$$s.t. \quad \mathbf{x} \in \Gamma,\ \mathbf{u} \in \Omega, \tag{7.2b}$$

$$\mathcal{G}(\mathbf{x}, \mathbf{u}) = 0 \tag{7.2c}$$

The geometry of this problem in the trajectory space is illustrated in Fig. 7.2a, where $\mathcal{M}$ is the manifold of the robot dynamics, and the gray patch on the manifold is the infeasible region. Notice that there are two important features in this problem.

Feature 1 (*Symmetry*): The cost function given in (7.1a) is designed to be $J(\mathbf{x}, \mathbf{u}) = J_1(\mathbf{x}) + J_2(\mathbf{u})$, where the minimum of $J_2(\boldsymbol{u})$ is achieved at $\mathbf{u} = 0$. Moreover, the box constraint of Ω is also symmetric to $\mathbf{u} = 0$.

Feature 2 (*Affine Dynamics*): Considering the robot dynamics equation is written as $M(x)\ddot{x} + N(x, \dot{x}) = u$, it can be regarded as an affine dynamics equation, i.e. $\mathcal{G}(\mathbf{x}, \mathbf{u}) = F(\mathbf{x}) + H(\mathbf{x})u = 0$.

Due to these two features, the problem can be relaxed by introducing the slack variable $\mathbf{y}$,

$$\min_{\mathbf{x},\mathbf{y}} \quad J(\mathbf{x}, \mathbf{y}) \tag{7.3a}$$

$$s.t. \quad \mathbf{x} \in \Gamma,\ \mathbf{y} \leq \mathbf{u}_{max}, \tag{7.3b}$$

$$F(\mathbf{x}) + H(\mathbf{x})\mathbf{y} \geq 0,\ F(\mathbf{x}) - H(\mathbf{x})\mathbf{y} \geq 0 \tag{7.3c}$$

It is proven in [67] that the optimizer of this relaxed problem is equivalent to that of the original problem. The intuition behind the relaxation is that: by introducing the slackness, the feasible region is augmented from the original nonlinear manifold $\mathcal{M}$ in Fig. 7.2a to the volume Γ^e with linear structure in Fig. 7.2b. Due to Feature 1, as J_2 achieves the minimum at $\mathbf{u} = 0$, the algorithm will pull the optimal solution down to $\mathcal{M}$, which is on the "bottom" of Γ^e, so that we may still get the same optimal solution as in the original problem.

The difference between these two problems is illustrated in Fig. 7.3, where the curve in Fig. 7.3a represents the nonlinear equality constraint (7.2c) and the shaded area in Fig. 7.3b represents the nonlinear inequality constraints (7.3c).By introducing the slack variable $\mathbf{y}$, the nonlinear equality constraint is successfully removed. Then, let $\mathbf{z} = [\mathbf{x}^T\ \mathbf{y}^T]^T$, and the problem becomes $\min_{\mathbf{z}\in\Gamma^e} J(\mathbf{z})$, which will be solved by the CFS algorithm.

The Convex Feasible Set Algorithm

The idea of the CFS algorithm proposed in [66] is to transform the original non-convex problem into a sequence of convex subproblems by obtaining convex feasible sets within the non-convex inequality constraints and linear equality constraints, then iteratively solve the quadratic programming (QP) subproblems until convergence. Note that the CFS algorithm is applied to the problem under the following assumption: 1) The cost function J is assumed to be smooth, strictly convex. 2) The constraint Γ^e is assumed as the intersection of N supersets Γ_i which can be represented by continuous, semi-convex and piecewise smooth functions ϕ_i, e.g. $\Gamma^e = \cap_i\{\mathbf{z} : \phi_i(\mathbf{z}) \geq 0\}$. The

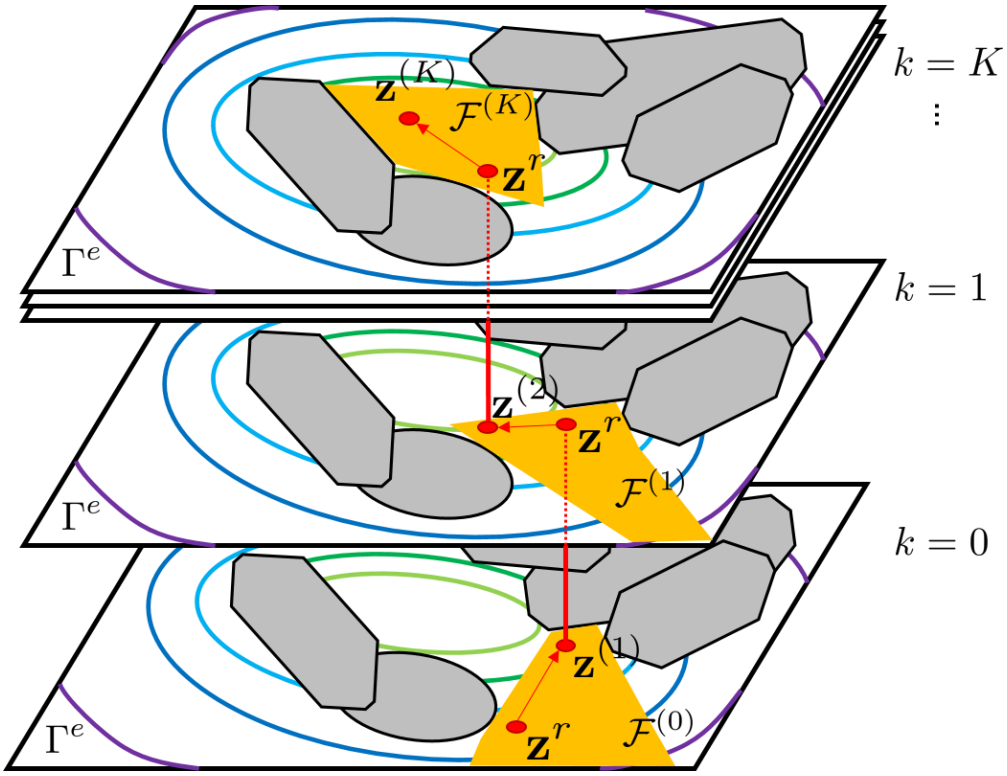


Figure 7.4: The illustration of the convex feasible set algorithm, where the yellow polygons are the convex feasible set obtained at each iteration and the gray areas are the infeasible set in the space Γ^e

Algorithm 2 The Convex Feasible Set Algorithm

$\mathbf{z}^{(0)} = \mathbf{z}^r$
while $\|\mathbf{z}^{(k+1)} - \mathbf{z}^k\| > \epsilon$ **do**
  Find the convex feasible set: $\mathcal{F}(\mathbf{z}^{(k)}) \subset \Gamma^e$
  Solve QP: $\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z} \in \mathcal{F}^{(k)}} J(\mathbf{z})$
  $\mathbf{z}^r \leftarrow \mathbf{z}^{(k+1)}$, $k = k + 1$
end while

semi-convexity of ϕ_i implies that the hessian of ϕ_i is lower-bounded. i.e. there exists a positive semi-definite matrix H_i such that for any $\mathbf{z}$ and v, $\phi_i(\mathbf{z} + v) - 2\phi_i(\mathbf{z}) + \phi_i(\mathbf{z} - v) \geq -v^T H_i v$.

Fig. 7.4 shows how CFS computes the solution iteratively, where the space Γ^e is filled contour plots of $J(\boldsymbol{z})$, and the infeasible sets are the gray polygons on the plot. At iteration k, given a reference point $\mathbf{z}^{(k)}$, a convex feasible set $F^{(k)} := F(\mathbf{z}^{(k)}) \subset \Gamma^e$ is computed around $\mathbf{z}^{(k)}$, which are the yellow polygons in Fig. 7.4. Then a new reference point $\mathbf{z}^{(k+1)}$ will be obtained by solving

the following QP problem

$$\mathbf{z}^{(k+1)} = \arg\min_{\mathbf{z}\in\mathcal{F}^{(k)}} J(\mathbf{z}) \tag{7.4}$$

The iteration will be terminated until the solution converges, i.e. $\|\mathbf{z}^{(k+1)} - \mathbf{z}^k\| \leq \epsilon$.

Given a reference point $\mathbf{z}^r$, the desired convex feasible set $\mathcal{F}(\mathbf{z}^r)$ is obtained by $\mathcal{F}(\mathbf{z}^r) := \cap_i \mathcal{F}_i(\mathbf{z})$, where $\mathcal{F}_i(\mathbf{z}^r) \subset \Gamma_i$. The different cases of $\mathcal{F}_i(\mathbf{z}^r)$ are illustrated in Fig. 7.5, where the gray shaded areas represent the infeasible set. The mathematical definition of $\mathcal{F}_i(\mathbf{z}^r)$ is stated below,

Case 1: Γ_i is convex.
Define $\mathcal{F}_i = \Gamma_i$.

Case 2: The complementary of Γ_i is convex.
In this case, ϕ_i can be designed to be convex, then $\phi_i(\mathbf{z}) \geq \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z} - \mathbf{z}^r)$. At the point where ϕ_i is not differentiable, $\nabla\phi_i$ is a sub-gradient which should be chosen as such that the steepest descent of J in the set of Γ^e is always included in the convex set $\mathcal{F}$. the convex feasible set $\mathcal{F}_i$ with respect to a reference point $\mathbf{z}^r$ is defined as

$$\mathcal{F}_i(\mathbf{z}^r) := \{\mathbf{z} : \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z} - \mathbf{z}^r) \geq 0\} \tag{7.5}$$

Case 3: neither Γ_i nor its complementary is convex.
In this case, ϕ_i is neither convex nor concave, but we can define a new convex function as $\tilde{\phi}_i(\mathbf{z}) := \phi_i(\mathbf{z}) + \frac{1}{2}(\mathbf{z} - \mathbf{z}^r)^T H_i(\mathbf{z} - \mathbf{z}^r)$. Then $\tilde{\phi}_i(\mathbf{z}) \geq \tilde{\phi}_i(\mathbf{z}^r) + \nabla\tilde{\phi}_i(\mathbf{z}^r)(\mathbf{z} - \mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z} - \mathbf{z}^r)$, where $\nabla\phi_i$ is identified with the sub-gradient of $\tilde{\phi}_i$ at points that are not differentiable. The convex feasible set with respect to $\mathbf{z}^r$ is then defined as

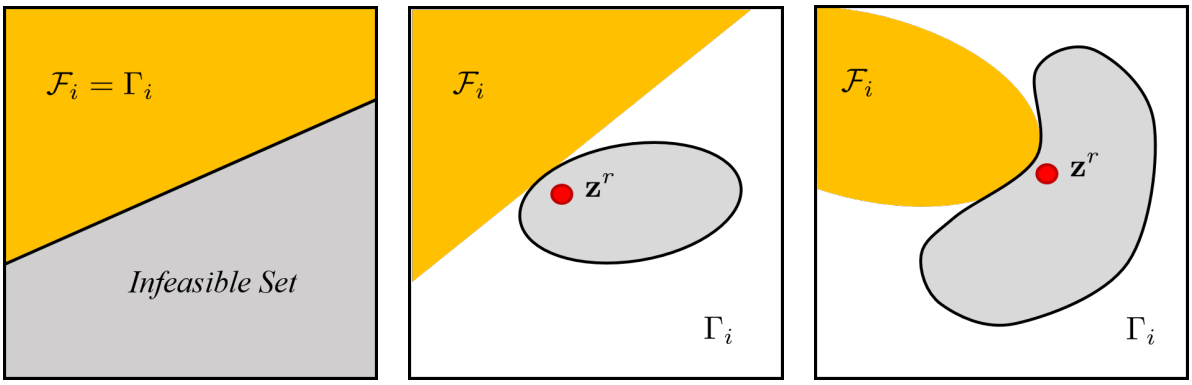
$$\begin{aligned}\mathcal{F}_i(\mathbf{z}^r) := \{\mathbf{z} :& \phi_i(\mathbf{z}^r) + \nabla\phi_i(\mathbf{z}^r)(\mathbf{z} - \mathbf{z}^r) \\ &\geq \frac{1}{2}(\mathbf{z} - \mathbf{z}^r)^T H_i(\mathbf{z} - \mathbf{z}^r)\}\end{aligned} \tag{7.6}$$

The CFS algorithm is summarized in Algorithm 2, and the detail of its convergence and feasibility are proven in [66].

7.3 The Temporal Optimization

Concept

In the previous section, a collision-free trajectory with fixed time step is optimized in the feasible set. However, the operational time of this trajectory has not been optimized yet. In order to obtain the time optimality, the time step should be considered as a variable as well. For example, suppose there is a N-step trajectory planning problem as the green line shown in Fig. 7.6. Since the sampling time dt is fixed, the operational time is given by $\sum_{k=1}^{N} dt = N \cdot dt$. On the other hand, the same trajectory planning problem with variable time step is shown as the orange line in Fig. 7.6. Its operational time is determined by the summation of time steps, i.e. $\sum_{k=1}^{N} \tau_k$. In short, the idea of the temporal optimization is to penalize the time variables over the horizon of the defined path.



(a) (b) (c)

Figure 7.5: The geometry illustration of the feasible set $\mathcal{F}$ at the reference point $\mathbf{z}^r$. (a) Case 1:Γ_i is convex. (b) Case 2: The infeasible set is convex. (c) Case 3: Neither Γ_i nor the infeasible set is convex.

Problem Formulation

Denote that the time variable at time step t as $\tau_k \in \mathbb{R}_+$, and $\boldsymbol{\tau} = [\tau_1, \tau_2, \cdots, \tau_N]^T \in \mathbb{R}_+^N$ is denoted as the time profiles over the horizon. Considering the smoothness of the trajectory in the defined path, the acceleration should be limited, i.e. $-a_{max} \leq a_t \leq a_{max}$, where $a_t,\ a_{max} \in \mathcal{A} \subset \mathbb{R}^m$ are denoted as the acceleration and the acceleration bound. Suppose the initial velocity and the final velocity are given as $v_0, v_N \in \mathbb{R}^m$ respectively, then acceleration are computed by

$$a_t = \begin{cases} \frac{1}{\tau_1}\left(\frac{x_1 - x_0}{\tau_1} - v_0\right) & t = 1 \\ \frac{1}{\tau_t}\left(\frac{x_{t+1} - x_t}{\tau_t} - \frac{x_t - x_{t-1}}{\tau_{t-1}}\right) & t = 2, \cdots, N-1 \\ \frac{1}{\tau_N}\left(v_N - \frac{x_N - x_{N-1}}{\tau_N}\right) & t = N \end{cases} \tag{7.7}$$

Denote the acceleration profile in the horizon by $\boldsymbol{a} = \left[a_1^T, a_2^T, \cdots, a_N^T\right]^T \in \mathcal{A}^N = \mathbf{A}$. $\boldsymbol{a}$ and $\boldsymbol{\tau}$ define a nonlinear dynamics, i.e. $\mathcal{G}_T(\boldsymbol{\tau}, \boldsymbol{a}) = 0$. Then, the temporal optimization problem can be formulated as

$$\min_{\boldsymbol{\tau}, \boldsymbol{a}} \quad J_T(\boldsymbol{\tau}, \mathbf{a}) = w_3 \|\boldsymbol{\tau}\|_1 + w_4 \|\boldsymbol{a}\|_R^2 \tag{7.8a}$$

$$s.t. \quad \boldsymbol{\tau} > 0,\ \boldsymbol{a} \in \mathbf{A} \tag{7.8b}$$

$$\mathcal{G}_T(\boldsymbol{\tau}, \boldsymbol{a}) = 0 \tag{7.8c}$$

where $w_3, w_4 \in \mathbb{R}_+$, J_T is designed as the convex and smooth cost function of the temporal optimization, where $\|\boldsymbol{\tau}\|_1 = \sum_{t=1}^N \tau_t$ penalizes the operational time, and $\|\boldsymbol{a}\|_R^2 = \boldsymbol{a}^T R \boldsymbol{a}$ penalizes the acceleration with a positive definite matrix R. Note that the cost function can be decoupled as $J_T(\boldsymbol{\tau}, \boldsymbol{a}) = J_{T1}(\boldsymbol{\tau}) + J_{T2}(\boldsymbol{a})$, and the minimum of the second term is achieved at $\boldsymbol{a} = 0$.

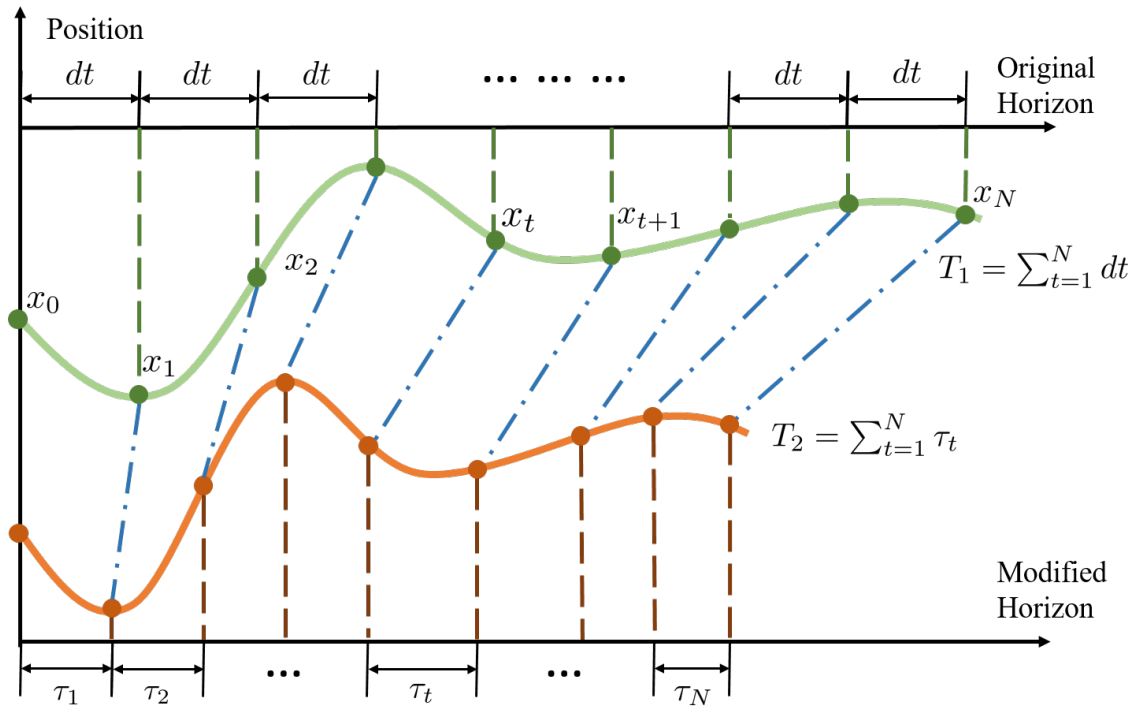


Figure 7.6: Illustration of the temporal optimization, where the green line is represents original trajectory, whereas the orange line represents the modified trajectory.

Furthermore, $\mathcal{G}_T$ is affine with respect to $\boldsymbol{a}$. For example, for $t = 2, \cdots, N-1$, (7.7) can be reformulated as,

$$\underbrace{\tau_t \left(x_t - x_{t-1}\right) - \tau_{t-1}\left(x_{t+1} - x_t\right)}_{F_T^t(\tau_t)} + \underbrace{\tau_t^2 \tau_{t-1}}_{H_T^t(\tau_t)} a_t = 0 \tag{7.9}$$

where $F_T^t(\tau) \in \mathbb{R} \to \mathbb{R}^m$ and $H_T^t(\tau_t) \in \mathbb{R} \to \mathbb{R}^{m \times m}$. Therefore, this problem has the same geometric features as (7.2). Hence, (7.8) can be solved by CFS.

In the fast robot motion planning framework, both the trajectory planning and the temporal optimization can be translated to CFS-solvable problems, which are formulated as several QP subproblems and solved iteratively. This results in a significant reduction of the computation time, comparing to the conventional motion planning methods.

7.4 Experiment

In order to verify the proposed algorithm, a series of experimental validations were perform on the robot system introduced in Chapter 2. The robot reference path was shown in Fig. 7.8a. The red points were the way points that the robot needed to pass and stop by. The cyan lines represented

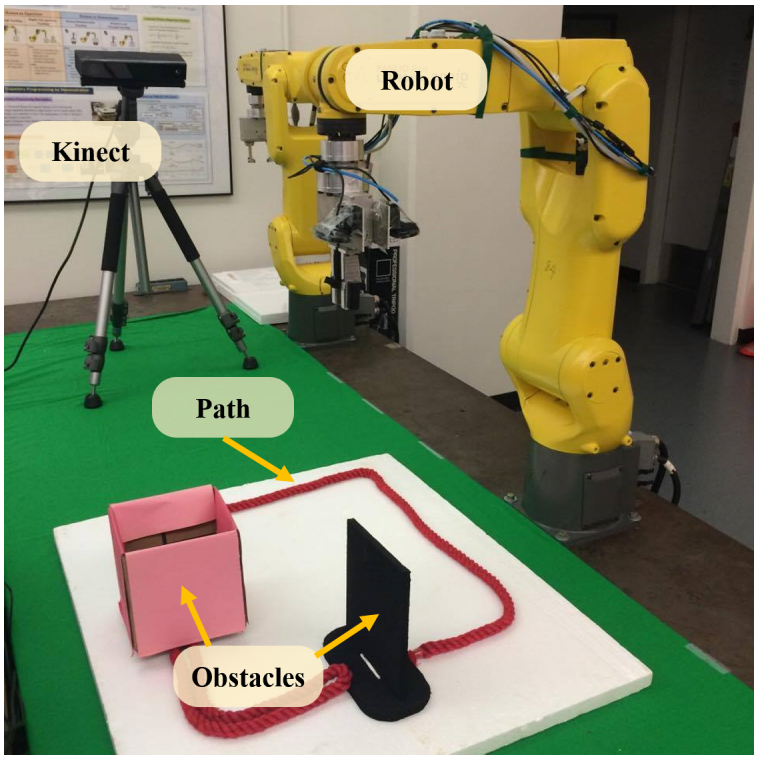


Figure 7.7: The experimental setup, where the robot is a FANUC LR Mate 200iD/7L, the Microsoft Kinect is used to detect the pink and black obstacles, and the red line represents the reference path.

the desired path, and path segments were sequentially numbered. Table 7.1 showed the performance of both algorithms on the planning of a trajectory with 95 steps. Paths 1,2, and 5 were collision-free, while Paths 3 and 4 were occupied by obstacles. On the collision-free paths, there were not significant difference between two algorithms. On the blocked paths, the CFS algorithm exhibited much less computation time and iterations than SQP to converge to local optima. This was because SQP was developed for general purposes, where it was more conservative the step size selection. On the other hand, CFS considered the specific geometric structure of the trajectory planning problem, and the computational efficiency was significantly improved. The results of the simulation was shown in Fig. 7.8b, where the red line and the yellow line represented the SQP and CFS trajectory respectively. Although these algorithms converged to different solution, both of them achieved the collision avoidance motion.

We used the same path to evaluate the performance of FRMP, which was the CFS trajectory planning with the temporal optimization. The computation time of FRMP was shown in Table 7.2, where the average computation per path of the temporal optimization was 31 ms. Hence, the temporal optimization would not become a burden in FRMP; moreover, it could significantly reduce the operational time, where the improvement by FRMP could be found in Table 7.3 and Fig. 7.9.

Table 7.1: The comparison of SQP and CFS

	SQP		CFS	
	Total Horizon: 95			
unit: sec	Computation	Iterations	Computation	Iterations
Path 1	0.0043	0	0.0033	0
Path 2	0.0031	0	0.0033	0
Path 3	46.8098	94	0.8938	2
Path 4	48.8767	96	0.8397	5
Path 5	0.0032	0	0.0031	0
Total	95.725	190	1.7434	7

Table 7.2: The computation time of FRMP

	Fast Robot Motion Planner (FRMP)	
unit: sec	Path Planning	Temporal Optimization
Path 1	0.0033	0.0279
Path 2	0.0033	0.0272
Path 3	0.8938	0.0272
Path 4	0.8397	0.0485
Path 5	0.0031	0.0252
Total	1.7434	0.1561

Table 7.3: The comparision of CFS and FRMP

unit: sec	CFS	FRMP
Operational Time	15.00	6.63
Computation Time	1.74	1.90

7.5 Chapter Summary

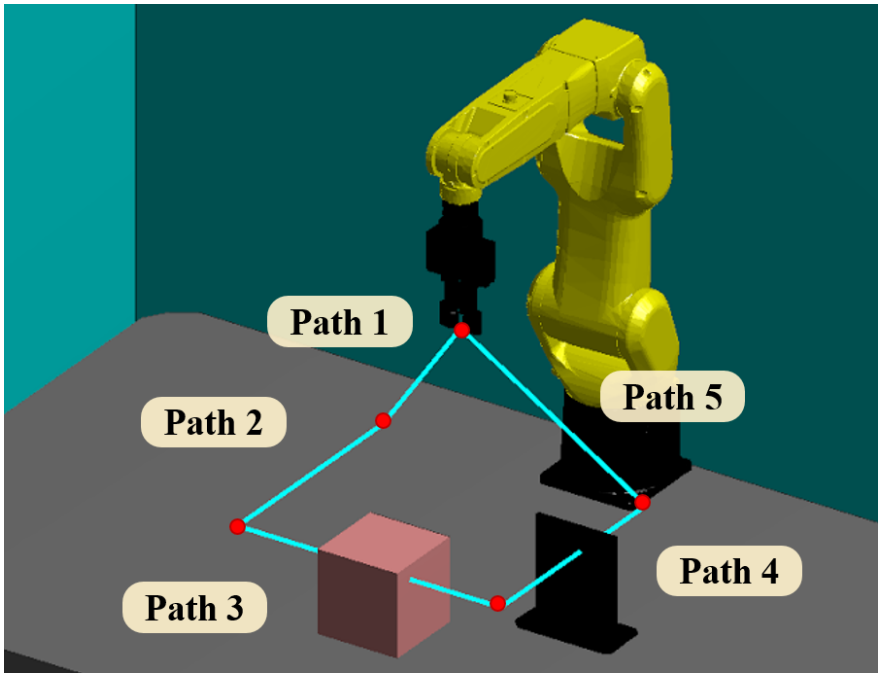
This chapter proposed the fast robot motion planner (FRMP) by formulating trajectory planning and temporal optimization as two optimization problems and solving them by the convex feasible set (CFS) algorithm.

The CFS algorithm is developed to deal with a problem with specific geometric structure, and it outperforms the benchmark optimization algorithm, sequential quadratic programming (SQP) in the trajectory planning problem, where both the computation time and iteration numbers are significantly reduced.

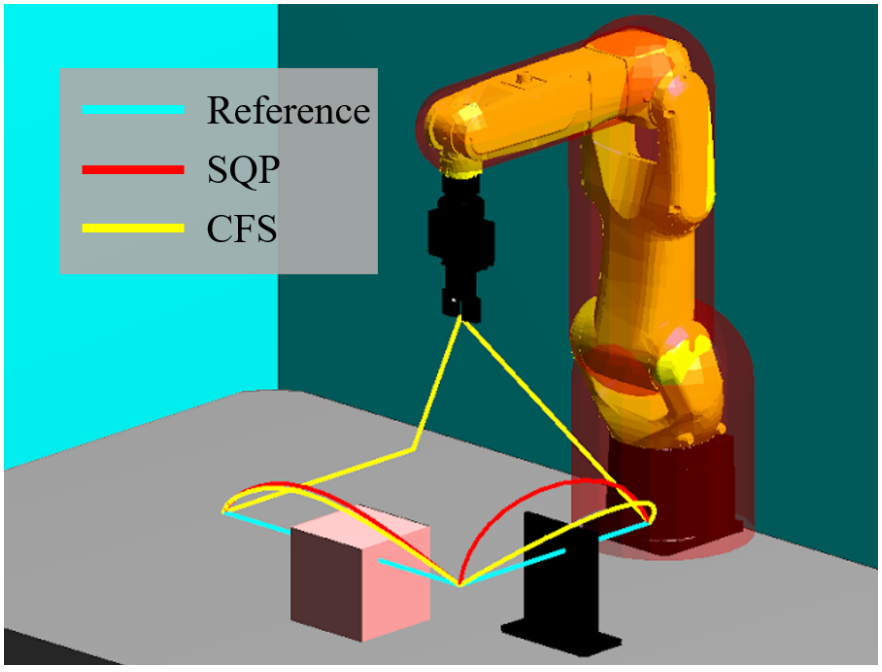
In order to achieve the time optimality on a defined path, the time variables are introduced to the

temporal optimization problem, which can be solved by CFS. It was shown by the experimental that the average computation time of the temporal optimization only takes 31 ms, and the operational time is reduced from 15 second to 6.6 second.

The experiment demonstrated that FRMP can plan a time-optimal trajectory on a 95th-step horizon within 2 second. In future, a more complicated and practical scenario will be designed to evaluate the performance of FRMP.



(a)



(b)

Figure 7.8: The geometry illustration. (a) The geometric structure of the path planning problem (b) The convex feasible set in the trajectory space

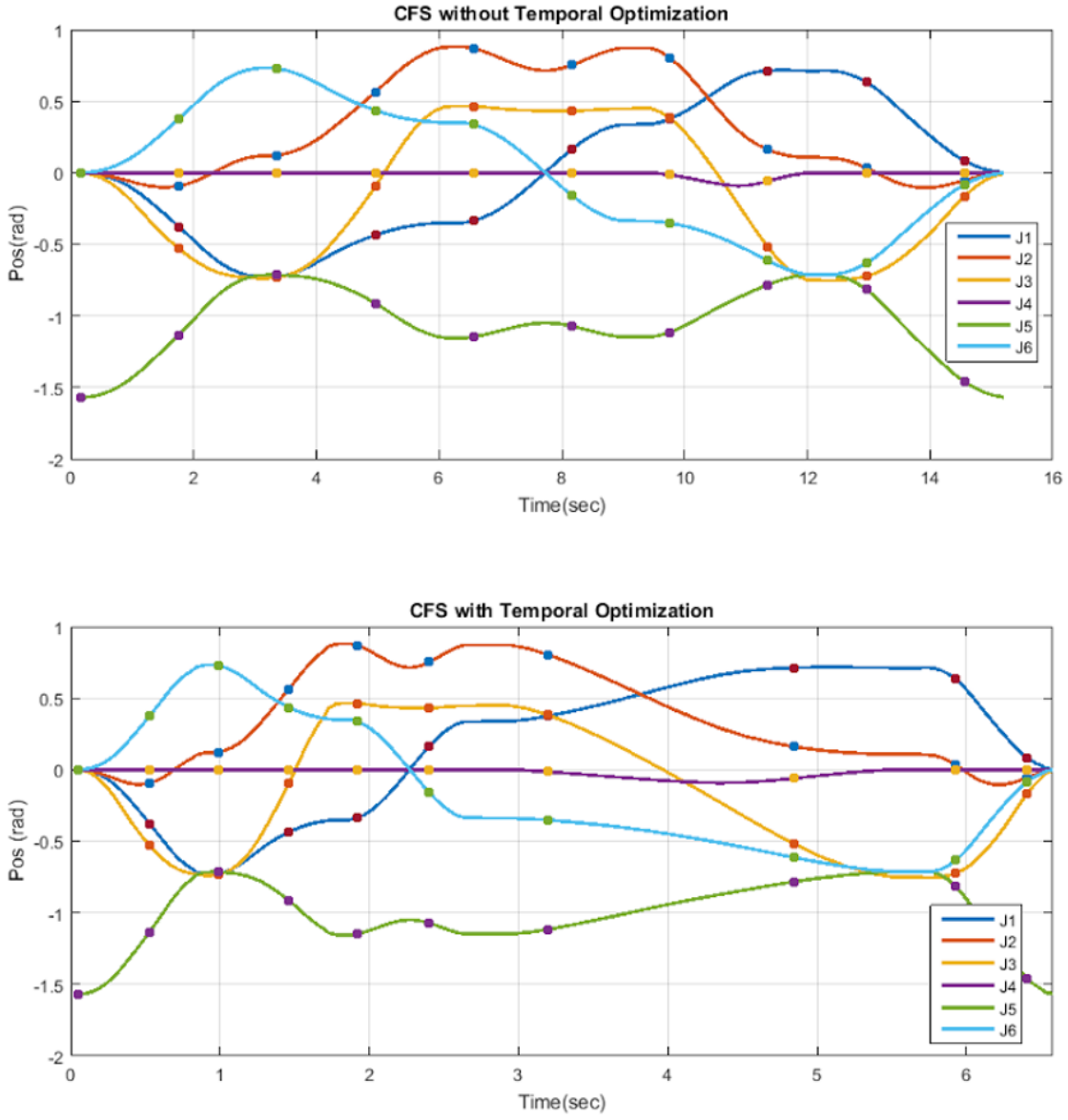
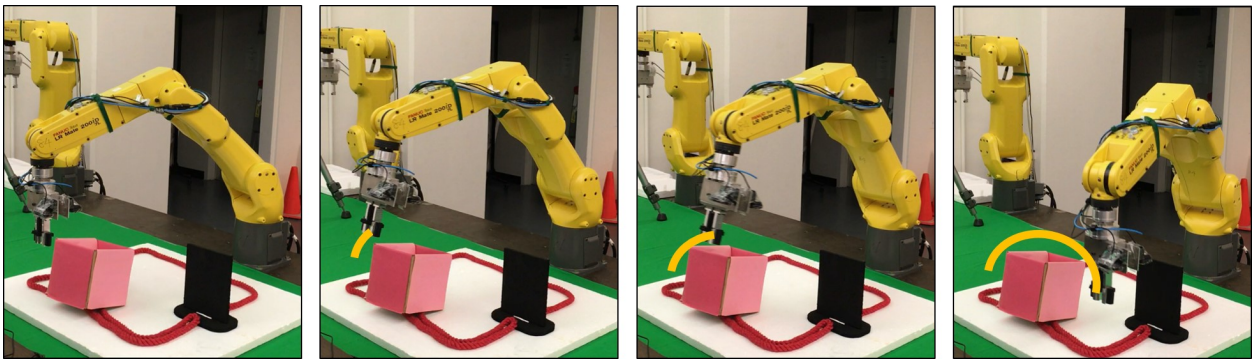
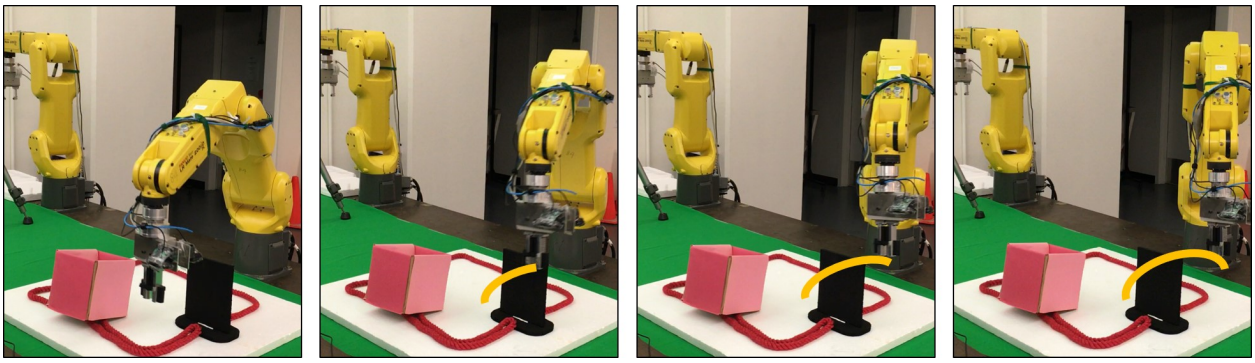


Figure 7.9: The trajectory reference of the CFS and FRMP



(a)



(b)

Figure 7.10: The sequential of the figures shows the robot motion in the experiment, where the executive motion of the robot planned by FRMP is shown as the orange line (a) The robot avoids the pink obstacle. (b) The robot avoids the black obstacle.

Chapter 8

Conclusions

8.1 Summary

Inspired by the pattern of human intelligence development, this dissertation studied to improve robot intelligence from three parts: Programing, Learning, and Planning. The programming part investigated various interfaces that retrieve the information/knowledge from humans. The learning part studied the methodologies to generalize the learned skills to similar tasks. The planning part discussed the algorithm to plan an optimal set of control actions to achieve the goal given constraints. The contributions of each chapter are as follows.

Chapter 3 discussed the alternative programming methodology - Programming by Demonstration (PbD). Three case studies are given to investigate three major trends of robot programming interfaces: robot imitation from human motion, kinesthetic teaching, and immersive teleopeation. Each case study introduced the framework and the implementation of each programming mechanisms.

Chapter 4 proposed a novel collision avoidance algorithm for kinesthetic teaching. The human guidance command and collision avoidance were formulated into an constrained least square problem. The proposed collision algorithm not only preserved the safety from the conventional collision avoidance methods, but also improved the flexibility and comfort in the kinesthetic teaching. Chapter 5 introduced a novel framework of the remote lead through teaching (RLTT) to simplify the robot programming problem in the tasks that required both motion and force. The human demonstration device was developed to align both human demonstration and robot reproduction phase in the same tool frame. In addition, this chapter further introduced the skill learning process from synchronizing demonstration data to constructing the skill model by Gaussian mixture regression. The experimental verifications were performed in two classical industrial applications. The robot achieved 96% success rate in the H7/h7 insertion task and successfully imitate human behavior in the grinding task.

Chapter 6 proposed a framework for efficient grasp generation by combining analytic approach with learning from demonstration. A database containing multiple categories of source objects with demonstrated grasping poses were constructed by human experts. The grasping poses to the

new target object were transferred from the most similar grasp example by the coherent point drift (CPD) method. All the transferred grasping poses were evaluated and sorted by the grasp isotropy metric. A series of experiments were performed to grasp objects with various shapes, sizes and configurations. The average success rate was 18.8 out of 20 grasp trials.

Chapter 7 proposed the fast robot motion planner (FRMP) by formulating trajectory planning and temporal optimization as two optimization problems and solving them by the convex feasible set (CFS) algorithm. CFS outperformed the benchmark nonconvex optimization algorithm, sequential quadratic programming (SQP) in the trajectory planning problem. The experiment demonstrated that FRMP can plan a time-optimal trajectory on a 95th-step horizon within 2 second.

8.2 Future Topics

Several open issues were raised during this dissertation study. This section discusses several directions for future research topics.

Programming

Chapter 3 introduced three different kinds of programming interface for demonstration with their frameworks and preliminary implementations. The performance of those programming interfaces can be further improved from the perception part. For instance, the noisy signal retrieved from the Kinect sensor might deteriorate the estimation of human motion. Hence, the computer vision research in human pose recovery would be very useful in the articulated body motion tracking.

The online collision avoidance algorithm proposed in Chapter 4 was applied to the scenarios with predefined environments. One possible direction to improve the ability of the proposed method would be integrating distance sensors such as infrared sensors or acoustic sensors to help robot deal with an unknown or dynamic environment.

Learning

In Chapter 5, robots had learned the peg-hole insertion and grinding task from human demonstration and successfully performed these learned tasks. However, a question then is raised: “Could robots further improve and outperform than the human teacher?” One possible answer may be found in reinforcement learning. By formulating the task as a reward function, the robot could explore and exploit the state space to further improve its control policy.

Chapter 6 applied the coherent point drift (CPD) algorithm to transfer grasping pose from grasp example to similar objects. There is one strong assumption that the point clouds of different objects could be perfectly separated. In practice, however, this assumption could not hold in bin-picking tasks or cluttered scenarios.Thus, one improvement direction would be pushing the registration algorithm from the object level to the feature level. Instead of mapping the grasping pose from similar objects, transferring the grasping pose based on the similar features would be more robust.

Planning

Chapter 7 discussed the planning problem in motion level, where it focused on the path collision avoidance and the operational time reduction. The intelligence of robots may be further enhanced to perform higher level planning. For instance, a hierarchical task planning would enable robots to manage complex tasks. Moreover, environment cognition would help robots understand the current status and needs and further improve the ability of robot planning.

Bibliography

[1] Ralph Abraham, Jerrold E Marsden, and Jerrold E Marsden. *Foundations of mechanics*. Vol. 36. Benjamin/Cummings Publishing Company Reading, Massachusetts, 1978.

[2] J. Adongo Ochier, C. D. Mladenova, and P. C. Müller. "An approach to automatic generation of dynamic equations of elastic joint manipulators in symbolic language". In: *Journal of Intelligent & Robotic Systems* 14.2 (Oct. 1995), pp. 199–218. ISSN: 0921-0296. DOI: 10.1007/BF01559612. URL: http://www.springerlink.com/index/10.1007/BF01559612.

[3] Ankur Agarwal and Bill Triggs. "Recovering 3D human pose from monocular images". In: *IEEE transactions on pattern analysis and machine intelligence* 28.1 (2006), pp. 44–58.

[4] Brenna D Argall et al. "A survey of robot learning from demonstration". In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[5] ATI Industrial Automation. URL: http://www.ati-ia.com.

[6] Autodesk Inc. *Human Interaction with Robot through Virtual Reality*. URL: https://goo.gl/yZZYBx.

[7] Autodesk Inc. *Stingray*. URL: https://www.autodesk.com/products/stingray/overview.

[8] F Badano, A Jutard, and M Betemps. "Chamferless robotic assembly using random search". In: *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*. IEEE. 1991, pp. 1598–1601.

[9] Lucian Balan and Gary M Bone. "Real-time 3D collision avoidance method for safe human and robot coexistence". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference On*. IEEE. 2006, pp. 276–282.

[10] Donald J Berndt and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series." In: *KDD workshop*. Vol. 10. 16. Seattle, WA. 1994, pp. 359–370.

[11] A. Billard and D. Grollman. "Robot learning by demonstration". In: *Scholarpedia* 8.12 (2013). revision #138061, p. 3824. DOI: 10.4249/scholarpedia.3824.

[12] Aude G Billard, Sylvain Calinon, and Rüdiger Dillmann. "Learning from humans". In: *Springer handbook of robotics*. Springer, 2016, pp. 1995–2014.

[13] Jeff A Bilmes et al. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models". In: *International Computer Science Institute* 4.510 (1998), p. 126.

[14] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[15] Jeannette Bohg et al. "Data-driven grasp synthesis—a survey". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 289–309.

[16] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152. ISBN: 0-89791-497-X. DOI: 10.1145/130385.130401. URL: http://doi.acm.org/10.1145/130385.130401.

[17] Stephen P Boyd et al. *Linear matrix inequalities in system and control theory*. Vol. 15. SIAM, 1994.

[18] Oliver Brock and Oussama Khatib. "Elastic strips: A framework for motion generation in human environments". In: *The International Journal of Robotics Research* 21.12 (2002), pp. 1031–1052.

[19] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. "Collaborative grasp planning with multiple object representations". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2851–2858.

[20] S. Calinon and A. Billard. "Learning of Gestures by Imitation in a Humanoid Robot". In: *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*. K. Dautenhahn and C.L. Nehaniv (Eds). Cambridge University Press, 2007, pp. 153–177.

[21] Sylvain Calinon et al. "Learning collaborative manipulation tasks by demonstration using a haptic interface". In: *Advanced Robotics, 2009. ICAR 2009. International Conference on*. IEEE. 2009, pp. 1–6.

[22] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[23] Francisco Andrés Candelas Herías, Carlos Alberto Jara Bravo, and Fernando Torres Medina. "Flexible virtual and remote laboratory for teaching Robotics". In: (2006).

[24] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[25] Sang Choi et al. "Lead-through robot teaching". In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 1–4.

[26] Mica Comstock, Kerstin Johansen, and Mats Winroth. "From mass production to mass customization: enabling perspectives from the Swedish mobile telephone industry". In: *Production Planning & Control* 15.4 (2004), pp. 362–372.

[27] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[28] D Costantinescu and EA Croft. "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths". In: *Journal of robotic systems* 17.5 (2000), pp. 233–249.

[29] Allen Cypher and Daniel Conrad Halbert. *Watch what I do: programming by demonstration*. MIT press, 1993.

[30] Hao Dang and Peter K Allen. "Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 1311–1317.

[31] Agostino De Santis et al. "The skeleton algorithm for self-collision avoidance of a humanoid manipulator". In: *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*. IEEE. 2007, pp. 1–6.

[32] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[33] Renaud Detry et al. "Generalizing grasps across partly similar objects". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3791–3797.

[34] Alexander Dietrich et al. "Extensions to reactive self-collision avoidance for torque and position controlled humanoids". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 3455–3462.

[35] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[36] Experimental Videos for Grasping Objects. URL: {http://me.berkeley.edu/\%7Ehclin/IROS2018/GraspByCPD.html}.

[37] FANUC Corporation. *CR 35-iA*. URL: http://robot.fanucamerica.com/products/robots/collaborative-robot-fanuc-cr-35ia.aspx.

[38] FANUC Corporation. *LR Mate 200iD series*. URL: https://www.fanucamerica.com/products/robots/productsbyseries/?seriesId=3&robotseries=LR%20Mate%20Series.

[39] Carlo Ferrari and John Canny. "Planning optimal grasps". In: *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE. 1992, pp. 2290–2295.

[40] F. Flacco et al. "A depth space approach to human-robot collision avoidance". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012, pp. 338–345. DOI: 10.1109/ICRA.2012.6225245.

[41] Federico Girosi, Michael Jones, and Tomaso Poggio. "Regularization theory and neural networks architectures". In: *Neural computation* 7.2 (1995), pp. 219–269.

[42] Timothy L Graf. *Lead-through robot programming system*. US Patent 5,880,956. 1999.

[43] Neville Hogan. "Impedance control: An approach to manipulation: Part II—Implementation". In: *Journal of dynamic systems, measurement, and control* 107.1 (1985), pp. 8–16.

[44] Geir E Hovland, Pavan Sikka, and Brenan J McCarragher. "Skill acquisition from human demonstration using a hidden markov model". In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 3. IEEE. 1996, pp. 2706–2711.

[45] HTC Corporation. *HTC VIVE VR System*. URL: https://www.vive.com/us/.

[46] IDS Imaging Development Systems GmbH. URL: https://www.ensenso.com/portfolio-item/n3x/.

[47] Intuitive Surgical. *da Vinci Surgical System*. URL: https://www.intuitivesurgical.com/.

[48] Tor Arne Johansen, Thor I Fossen, and Stig P Berge. "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming". In: *IEEE Transactions on Control Systems Technology* 12.1 (2004), pp. 211–216.

[49] Lydia E Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4 (1996), pp. 566–580.

[50] H. Kawasaki et al. "Virtual robot teaching for humanoid hand robot using muti-fingered haptic interface". In: *Communications, Computing and Control Applications (CCCA), 2011 International Conference on*. 2011, pp. 1–6. DOI: 10.1109/CCCA.2011.6031431.

[51] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *The international journal of robotics research* 5.1 (1986), pp. 90–98.

[52] Oussama Khatib et al. "Ocean one: A robotic avatar for oceanic discovery". In: *IEEE Robotics & Automation Magazine* 23.4 (2016), pp. 20–29.

[53] Byoung-Ho Kim et al. "Optimal grasping based on non-dimensionalized performance indices". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Vol. 2. IEEE. 2001, pp. 949–956.

[54] Ulrich Klank et al. "Real-time cad model matching for mobile manipulation and grasping". In: *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*. IEEE. 2009, pp. 290–296.

[55] Ralf Koeppe et al. "Robot-robot and human-robot cooperation in commercial robotics applications". In: *Robotics research. the eleventh international symposium*. Springer. 2005, pp. 202–216.

[56] Marek Kuczma. *An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality*. Springer Science & Business Media, 2009.

[57] Dana Kulić et al. "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 330–345.

[58] Xiangyang Lan and Daniel P Huttenlocher. "Beyond trees: Common-factor models for 2d human pose recovery". In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 1. IEEE. 2005, pp. 470–477.

[59] Steven M LaValle. "Rapidly-exploring random trees: A new tool for path planning". In: (1998).

[60] Hsien-Chung Lin. *Human Guidance Programming with Collision Avoidance*. URL: https://youtu.be/lqkp6g-RmxE.

[61] Hsien-Chung Lin et al. "A Framework for Robot Grasping Transferring with Non-rigid Transformation". In: *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE. 2018, in review.

[62] Hsien-Chung Lin et al. "Human guidance programming on a 6-DoF robot with collision avoidance". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 2676–2681.

[63] Hsien-Chung Lin et al. "Real-time collision avoidance algorithm on industrial manipulators". In: *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE. 2017, pp. 1294–1299.

[64] Hsien-Chung Lin et al. "Remote lead through teaching by human demonstration device". In: *ASME 2015 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers. 2015, V002T30A003–V002T30A003.

[65] Hsien-Chung Lin et al. "Robot learning from human demonstration with remote lead hrough teaching". In: *Control Conference (ECC), 2016 European*. IEEE. 2016, pp. 388–394.

[66] Changliu Liu, Chung-Yen Lin, and Masayoshi Tomizuka. "The convex feasible set algorithm for real time optimization in motion planning". In: *arXiv preprint arXiv:1709.00627* (2017).

[67] Changliu Liu and Masayoshi Tomizuka. "Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification". In: *Systems & Control Letters* 108 (2017), pp. 56–63.

[68] Changliu Liu et al. "Convex feasible set algorithm for constrained trajectory smoothing". In: *American Control Conference (ACC), 2017*. IEEE. 2017, pp. 4177–4182.

[69] Anthony A Maciejewski and Charles A Klein. "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments". In: *The international journal of robotics research* 4.3 (1985), pp. 109–117.

[70] Jeffrey Mahler et al. "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1957–1964.

[71] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[72] Microsoft. URL: https://developer.microsoft.com/en-us/windows/kinect/.

[73] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[74] RM Murray, Z Li, and S. Sastry. *A mathematical introduction to robotic manipulation*. 1st. CRC Press, 1994.

[75] Andriy Myronenko and Xubo Song. "Point set registration: Coherent point drift". In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.

[76] Andriy Myronenko, Xubo Song, and Miguel A Carreira-Perpinán. "Non-rigid point set registration: Coherent point drift". In: *Advances in Neural Information Processing Systems*. 2007, pp. 1009–1016.

[77] Jorge Nocedal and Stephen J Wright. *Sequential quadratic programming*. Springer, 2006.

[78] J. Adongo Ochier and P. C. Muller. "Control of multi-axis robots with elastic joints via cascade compensation using the method of exact linearization". In: *Journal of Intelligent & Robotic Systems* 17.4 (Dec. 1996), pp. 351–370. ISSN: 0921-0296. DOI: 10.1007/BF00571698.

[79] Chavdar Papazov et al. "Rigid 3D geometry matching for grasping of known objects in cluttered scenes". In: *The International Journal of Robotics Research* 31.4 (2012), pp. 538–553.

[80] Chonhyon Park, Jia Pan, and Dinesh Manocha. "ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments." In: *ICAPS*. 2012.

[81] Tadej Petrič and Leon Lajpah. "Smooth Continuous Transition Between Tasks on a Kinematic Control Level: Obstacle Avoidance As a Control Problem". In: *Robot. Auton. Syst.* 61.9 (Sept. 2013), pp. 948–959. ISSN: 0921-8890. DOI: 10.1016/j.robot.2013.04.019. URL: http://dx.doi.org/10.1016/j.robot.2013.04.019.

[82] PhaseSpace. URL: http://http://www.phasespace.com/.

[83] Javier Pliego-Jiménez and Marco A Arteaga-Pérez. "Adaptive position/force control for robot manipulators in contact with a rigid surface with uncertain parameters". In: *European Journal of Control* 22 (2015), pp. 1–12.

[84] Marc H Raibert and John J Craig. "Hybrid position/force control of manipulators". In: *Journal of Dynamic Systems, Measurement, and Control* 103.2 (1981), pp. 126–133.

[85] Nathan Ratliff et al. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 489–494.

[86] Rethink Robotics. *http://www.rethinkrobotics.com/*.

[87] P Reynoso-Mora, W Chen, and M Tomizuka. "A convex relaxation for the time-optimal trajectory planning of robotic manipulators along predetermined geometric paths". In: *Optimal Control Applications and Methods* (2016).

[88] Elon Rimon and Stephen P Boyd. "Obstacle collision detection using best ellipsoid fit". In: *Journal of Intelligent and Robotic Systems* 18.2 (1997), pp. 105–126.

[89] J Kenneth Salisbury. "Active stiffness control of a manipulator in cartesian coordinates". In: *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*. Vol. 19. IEEE. 1980, pp. 95–100.

[90] Stefan Schaal, Auke Ijspeert, and Aude Billard. "Computational approaches to motor learning by imitation". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 358.1431 (2003), pp. 537–547.

[91] Stefan Schaal, Auke Ijspeert, and Aude Billard. "Computational approaches to motor learning by imitation". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 358.1431 (2003), pp. 537–547.

[92] Stefan Schaal et al. "Learning from demonstration". In: *Advances in neural information processing systems* (1997), pp. 1040–1046.

[93] John Schulman et al. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." In: *Robotics: science and systems*. Vol. 9. 1. Citeseer. 2013, pp. 1–10.

[94] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. 1st. Advanced Textbooks in Control and Signal Processing. Springer-Verlag, 2009.

[95] SMC Corporation. URL: `http://www.smcpneumatics.com/`.

[96] Peter Spellucci. "A new technique for inconsistent QP problems in the SQP method". In: *Mathematical Methods of Operations Research* 47.3 (1998), pp. 355–400.

[97] Robert J. Sternberg. *Human intelligence*. 2017. URL: `https://www.britannica.com/science/human-intelligence-psychology/Introduction`.

[98] Holger Täubig, Berthold Bäuml, and Udo Frese. "Real-time swept volume and distance computation for self collision detection". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 1585–1592.

[99] Aleš Ude, Christopher G Atkeson, and Marcia Riley. "Programming full-body movements for humanoid robots by observation". In: *Robotics and autonomous systems* 47.2-3 (2004), pp. 93–108.

[100] Universal Robotics. *https://www.universal-robots.com/*.

[101] Jacob Varley et al. "Generating multi-fingered robotic grasps via deep learning". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE. 2015, pp. 4415–4420.

[102] Chun-Chih Wang. "Motion Control of Indirect-drive Robots: Model Based Controller Design and Performance Enhancement Based on Load-side Sensors". PhD thesis. University of California at Berkeley, 2008.

[103] Carlos Canudas de Wit, Georges Bastin, and Bruno Siciliano. *Theory of Robot Control*. 1st. Secaucus, NJ: Springer-Verlag New York, Inc., Jan. 1996. ISBN: 3540760547.

[104] Qiang Wu and Ding-Xuan Zhou. "SVM soft margin classifiers: linear programming versus quadratic programming". In: *Neural computation* 17.5 (2005), pp. 1160–1187.

[105] Shin-Shan Zhuang and Shang-Hong Lai. "Face detection directly from h. 264 compressed video with convolutional neural network". In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE. 2009, pp. 2485–2488.