Clemson University
TigerPrints

All Dissertations | Dissertations

May 2021

Human-Robot Collaboration in Automotive Assembly

Yi Chen
Clemson University, yc4@clemson.edu

Follow this and additional works at: https://tigerprints.clemson.edu/all_dissertations

Recommended Citation

Chen, Yi, "Human-Robot Collaboration in Automotive Assembly" (2021). *All Dissertations*. 2813.
https://tigerprints.clemson.edu/all_dissertations/2813

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

HUMAN-ROBOT COLLABORATION IN AUTOMOTIVE ASSEMBLY

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Automotive Engineering

---

by
Yi Chen
May 2021

Accepted by:
Dr. Yunyi Jia, Committee Chair
Dr. Venkat N. Krovi, Committee Co-Chair
Dr. Bing Li
Dr. Matthias Schmid

ABSTRACT

In the past decades, automation in the automobile production line has significantly increased the efficiency and quality of automotive manufacturing. However, in the automotive assembly stage, most tasks are still accomplished manually by human workers because of the complexity and flexibility of the tasks and the high dynamic unconstructed workspace. This dissertation is proposed to improve the level of automation in automotive assembly by human-robot collaboration (HRC).

The challenges that eluded the automation in automotive assembly including lack of suitable collaborative robotic systems for the HRC, especially the compact-size high-payload mobile manipulators; teaching and learning frameworks to enable robots to learn the assembly tasks, and how to assist humans to accomplish assembly tasks from human demonstration; task-driving high-level robot motion planning framework to make the trained robot intelligently and adaptively assist human in automotive assembly tasks.

The technical research toward this goal has resulted in several peer-reviewed publications. Achievements include: 1) A novel collaborative lift-assist robot for automotive assembly; 2) Approaches of vision-based robot learning of placing tasks from human demonstrations in assembly; 3) Robot learning of assembly tasks and assistance from human demonstrations using Convolutional Neural Network (CNN); 4) Robot learning of assembly tasks and assistance from human demonstrations using Task Constraint-Guided Inverse Reinforcement Learning (TC-IRL); 5) Robot learning of assembly tasks from non-expert demonstrations via Functional Objective-Oriented

Network (FOON); 6) Multi-model sampling-based motion planning for trajectory optimization with execution consistency in manufacturing contexts.

The research demonstrates the feasibility of a parallel mobile manipulator, which introduces novel conceptions to industrial mobile manipulators for smart manufacturing. By exploring the Robot Learning from Demonstration (RLfD) with both AI-based and model-based approaches, the research also improves robots' learning capabilities on collaborative assembly tasks for both expert and non-expert users. The research on robot motion planning and control in the dissertation facilitates the safety and human trust in industrial robots in HRC.

DEDICATION

Dedicated to My Mom and Dad

ACKNOWLEDGMENTS

In this long journey to my doctoral degree, I am grateful to have found so many mentors, colleagues, friends, and families who have supported me to keep going forward without hesitation. A page is too less to name all the people who have encouraged me and influenced me in some way during this period. Nevertheless, the dissertation would not be completed without mentioning some people from the following groups.

Academic advisors: My sincerest gratitude goes to my advisor, Dr. Yunyi Jia, for being patient and supportive to let me explore different areas of robotics and his wise, insightful suggestions on my research. My appreciation to my co-advisor, Dr. Venkat N. Krovi, for his friendly and patient high-level guidance in both my research and paper writing.

Colleagues: Many thanks to my colleagues and friends, Dr. Weitian Wang and Dr. Rui Li, for the wise suggestions and wonderful cooperation on all the aspects of my research work. I am also grateful to my colleagues and friends, Dr. Zhujun Zhang, Zebin Wang, and Yi Sun, who greatly helped in the experiments to complete this dissertation.

Friends and family: I would like to say thank you to Kurt Repsher, Bo Winter, and Angela Winter, who have introduced me in wonderful track days and made me keep energetic and optimistic in this period.

Last but not the least, my deepest gratitude goes toward my parents, Jianmin Chen and Hong Guan, who have loved me and supported me unconditionally in my life.

TABLE OF CONTENTS

LIST OF TABLES

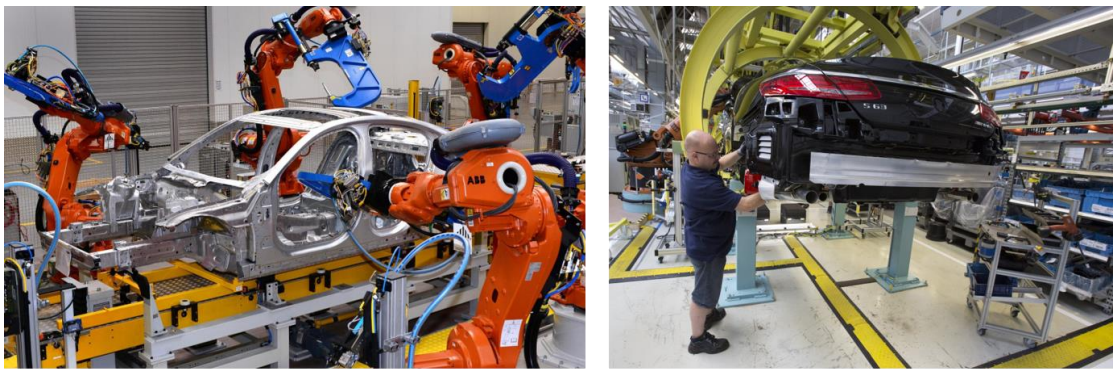
LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In this chapter, the motivation of this dissertation is presented at first. Then the challenges are summarized. Afterward, the conducted research and the corresponding contributions and impacts are discussed to clarify the research scope of this dissertation. Last but not the least, the organization of the dissertation is listed.

1.1 Motivation

Prototypes of industrial robots have been implemented in automotive manufacturing since the 1960s, which were started with performing spot-welding tasks. By the 1980s, billions of dollars were spent by companies worldwide to automate basic tasks in their assembly lines to improve efficiency, productivity, and competitiveness. Nowadays, industrial robots have been widely deployed in many aspects of automotive manufacturing, such as welding, gluing, material handling, and material transport. Although human involvement in manufacturing for the automotive industry has decreased dramatically in recent years, over 60% of automotive assembly tasks are still accomplished manually by human workers [1].



(a) (b)

Figure 1.1. The environment of the automotive assembly line.
Figure (a) is the manufacturing of the white body, which is fully automated by fenced robotic arms and well-constructed environment. Figure (b) is the final assembly process, which is usually a hybrid process involving both robots and humans in a unconstructed environment.

Many early stages of automotive manufacturing, such as white body manufacturing, have achieved full automation with conventional industrial robots in a well-constructed environment, where the robots are locked away in a fence that prohibiting access of humans as shown in Figure 1.1 (a). However, the automotive final assembly presents numerous challenges, such as significant variability of tools and parts, flexible tasks, and unstructured and dynamic environments shown in Figure 1.1 (b), that preclude direct automation via traditional fenced robotic work-cells. For example, different humans may have different preferences to complete the same work, which may require robots to adjust their speed or motions accordingly to adapt to humans' preferences. Humans may be distracted and do not pay attention to the moving robots in the shared workspace, which may demand robots to stay alert and adjust the speed or even stop to ensure safety. The dissertation is motivated by addressing some of these challenges and facilitating the level of automation in automotive assembly by HRC.

1.2 Challenges

The challenges in our research would include the following: First, the lack of suitable robotic systems remains a challenge, especially a compact-size collaborative mobile manipulator that can handle the heavy payload. Secondly, online teaching and learning through human demonstration is not easy to achieve. Because of the flexible tasks and the uncertainty of human motions in a specific demonstration, the robot is required to abstract the common conceptions of the tasks based on only a few inconsistent demonstrations. Thirdly, it is not easy to make robots adaptively assist human workers in the shared workspace. Human workers can get tired, can make mistakes, and can have personal preferences, which means robots will need to correctly react to some untrained situations only base on environment sensing and intelligently decision-making strategy. Finally, all these components in this complicated system need to be carefully designed with system thinking and make sure they are closely connected to and work properly with each other.

1.3 Contributions

This dissertation is proposed to improve the level of automation in automotive assembly by HRC. Facets of this problem are explored including the novel collaborative robotic systems for HRC in automotive assembly contexts, robot learning capabilities that enable robots to learn the assembly tasks and how to assist humans to accomplish assembly tasks from human demonstration, and task-driven high-level robot motion planning framework to make the trained robot assist human intelligently and adaptively in the collaborative process of automotive assembly applications.

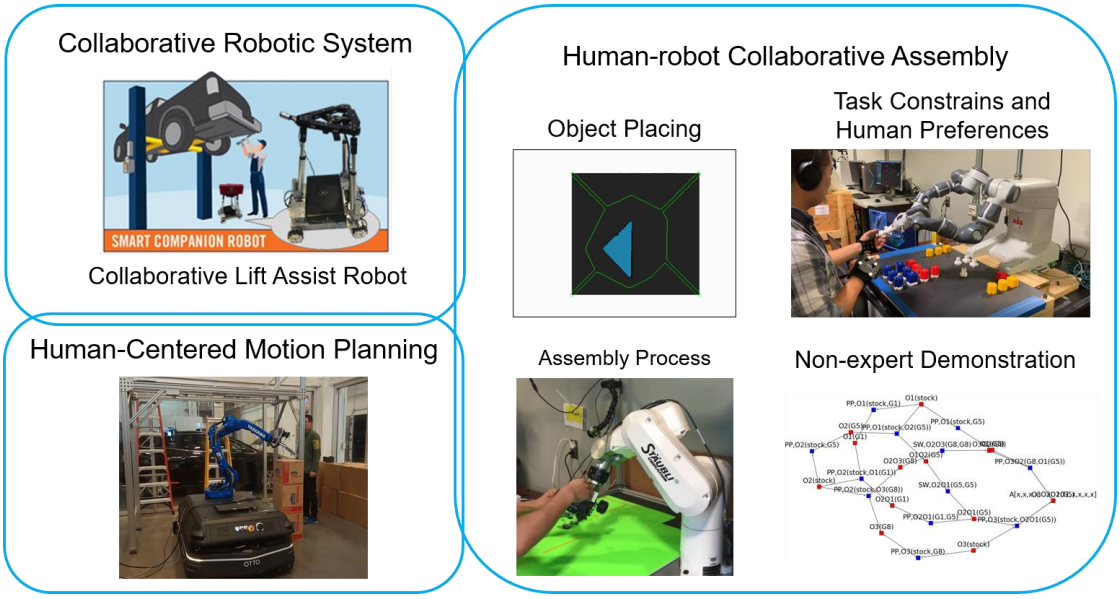


Figure 1.2. Contributions of the dissertation.

In Chapter 3, the design of a collaborative lift assist robot for automotive assembly is proposed. Comparing to the conventional lift assist mechanism and the commercial-off-the-shelf mobile manipulators in the market, the proposed design of the parallel mobile manipulator has a compact size, unlimited workspace, high task flexibility, intuitive human-robot interaction, and can handle the heavy payload. The research demonstrates the feasibility of a parallel mobile manipulator, which introduces novel conceptions to industrial mobile manipulators for smart manufacturing.

By exploring the Robot Learning from Demonstration (RLfD) with both AI-based and model-based approaches, the research also improves robots' learning capabilities on collaborative assembly tasks for both expert and non-expert users. In Chapter 4, a graph-based approach is proposed to model the object placing tasks in assembly. The proposed approach eliminates the pre-define parameters for tasks and makes the robot can learn object placing tasks from human demonstrations. To further enable robots to assist humans

in more complicated assembly tasks, in Chapter 5, An AI-based approach is proposed to enable robots to learn the assembly process and assist humans in assembly tasks using convolutional neural networks (CNN). The proposed approach makes robots assist humans actively in low-precision and high-strength jobs with easy-to-use human demonstrations. Also, it eliminates the complexity of the assembly task modeling and system setup. In Chapter 6, a model-based approach is proposed to enable robots to learn assembly task constraints and human preferences and then assist humans in the collaboration. The robot can not only repeat the demonstrated tasks but also actively assist humans according to their preferences in larger geometry scale tasks from a few rounds of small-scale human demonstrations. The proposed approach improves the task scalability and reduces the requirements of training data and computational efforts comparing to the conventional inverse reinforcement learning (IRL) approaches. In Chapter 7, a graph-based approach is proposed to enable robots to learn assembly tasks from non-expert demonstrations. The proposed approach enables robots to learn assembly tasks from imperfect demonstrations and then find optimal solutions for the demonstrated tasks.

In Chapter 8, a multi-model sampling-based motion planning framework is proposed to generate predictable, efficient, and consistent robot motions via the popular sampling-based motion planning algorithms. A cost-function-based trajectory optimization algorithm is proposed in the framework, which considers the predictability, efficiency, manipulability, and safety in trajectory optimization. A constraint-guided and similarity-based motion planning algorithm is proposed to improve the consistency of the robot

motions. The research on robot motion planning and control in the dissertation facilitates the safety and human trust in industrial robots in HRC.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter 2 discusses the related work and summaries the research gaps. Chapter 3 presents the research work on collaborative robotic systems, which provides a design and evaluation of a novel parallel mobile manipulator for heavy-payload lift assistance in automotive assembly. Chapter 4 to Chapter 7 introduce the research on robot learning from demonstration for automotive assembly, which includes the learning of object placing tasks in assembly, the learning and assistance generation for assembly tasks via both model-based and AI-based approaches, and the learning of assembly tasks from non-expert human demonstration. Chapter 8 provides the research work on human-centered robot motion planning in automotive assembly. Finally, the conclusions and future work are discussed in Chapter 9.

CHAPTER 2
RELATED WORK

2.1 Introduction

According to the challenges presented in automotive assembly lines, many research works have been conducted to improve the level of automation in the final assembly of automotive manufacturing. Instead of achieving full automation directly, human-robot collaboration is identified as one of the potential solutions for the automation of automotive assembly. Different aspects of HRC in manufacturing, such as collaborative robotic systems, improvement of robot learning capability, and robot motion planning and control, are widely investigated in the past decades. A review of the related work in these three fields is presented in this chapter.

2.2 The State-of-the-art of Collaborative Robotic Systems

The conception of the collaborative robots was introduced by J. Edward Colgate and Michael Peshkin [2], which is intended for direct physical interaction with a human operator. Serval collaborative industrial robots have been marketed since 2004, which are normally with payload range from 0.5 kg to 10 kg. The safety requirements of collaborative robots have been discussed and gradually established since 2011[3]. Collaborative robots allow the human and the machine to work in close and share a common workspace, which release the industrial robots from a fence prohibiting access and give them the opportunities to share some high-strength and low-versatility tasks in the final assembly stage of automotive manufacturing.

To further eliminate the limitation of the workspace, research interest in mobile manipulators (robotic arms mounted on mobile bases) has grown steadily over the past two decades [4]–[6]. Commercial interest has also spiked in recent years due to advances in technology that have enabled the broad use of automation and robotics while simultaneously reducing costs. Mobile manipulators are gradually becoming commercial tools for industrial use [7]–[9]. Among them, heavy payload transport and manipulation have been one of the most popular tasks for the use of mobile manipulators [10], [11]. Overviews of mobile manipulators and applications for manufacturing can be found in [6], [12], which reference examples of commercial-off-the-shelf mobile manipulators that are developed and used for industrial purposes. These works were intended to capture the current state-of-the-art in mobile manipulation (as of 2016); compare the diversity of performance assessment methods available for these systems. Many commercial-off-the-shelf mobile manipulators were developed in a constructionist approach by mounting general-purpose serial arms on various mobile platforms.

2.3 The State-of-the-art of Robot Learning from Demonstrations

As the requirements of high-flexibility tasks and highly customized products with short lifecycle, the conventional robot programming approaches gradually become inefficient for today's smart manufacturing. Robot Learning from Demonstration (RLfD) is one of the techniques that target to reduce the robot programming effort and system setup time cost. RLfD is a wide broad topic ranging from task modeling, machine learning, to human-robot interaction as well. Many related works have been conducted in recent years.

Some studies have been conducted on how to replicate arm trajectories from human demonstrations. Chen et al. [13] developed a method to identify the trajectories and eliminate the noise from human demonstrations based on the statistical regression analysis. Hiratsuka et al. [14] employed Local Procrustes Analysis and Dynamic Movement Primitives to transfer the demonstrated trajectories from the human skeleton model to the robot and then reproduce them on the robot in real-time. Calinon et al. [15] presented an approach based on HMM and Gaussian mixture regression (GMR) to enable the robot to learn new trajectories from humanlike motion data. Jha et al. [16] used incremental inverse kinematics and positional mapping to transfer the demonstrated trajectories from human arm workspace to robot arm workspace. Maeda et al. [17] proposed a cost-function-based approach, which considered the cost of iterations of the inverse kinematics and the cost of task achievement, to enable the robot to mimic human arm motion through stochastic optimization of the embodiment mapping.

In addition to the trajectory-level teaching, some studies have been also conducted for task-level RLfDs. The approaches included motion capture [18], [19], natural language [20], [21], vision system [22], [23], and wearable sensors [24] [25]. Based on these sensing data of human demonstrations, the robot can learn the task-level knowledge and generate its action planning strategies using the integrated learning algorithms. The algorithms include HMM [26], reinforcement learning [27], inverse reinforcement learning [28], and other learning approaches [29], [30]. These approaches improved the robot's ability to adapt to human intentions in more complicated collaborative tasks comparing to the trajectory-level teaching approaches.

2.4 The State-of-the-art of Robot Motion Planning and Control

Optimal trajectory planning for industrial robots is one of the key challenges for manufacturing automation in different applications [31], [32]. In the past two decades, probabilistic sampling-based algorithms have become popular and successful approaches for robotic motion planning problems especially in high-dimensional configuration spaces (e.g. the motion planning of high degree-of-freedom robots) [33]. Probabilistic sampling-based algorithms are commonly classified into two categories: the single-query and the multiple-query. Both the single-query and the multiple-query path planning algorithms aim to explore the configuration space with a search using a probabilistic-based sampling scheme while avoiding explicit construction of the configuration space.

The multiple-query approaches typically generate a roadmap, which is a topological graph that can be utilized by multiple initial-state/goal-state pairs. A classic example of this category is the probabilistic roadmap algorithm (PRM) [34]. The start-state/goal-state pairs are given as initial conditions of the PRM algorithm, the roadmap is established by randomly sampling points in configuration space and connecting nearby points if they can be reached from each other. The path from the start state to the goal state can then be found in the roadmap. The variants of PRM include lazy RPM [35], dynamic PRM [36], and PRM* algorithm [37].

Instead of constructing a roadmap for the free configuration space, the single-query approaches keep searching for a path that connects the given single initial-state/goal-state pair until finding a solution or reporting an early failure. One classical example of this category is the rapidly exploring random trees algorithm (RRT) [38]. The incremental

simulator is used to produce a randomly new state in each step, and the state advancement is determined by the collision detector and the distance between the current states to the goal state. The family of algorithms in this category also includes the rapidly exploring dense trees algorithm [39], RRT* [40], and LQR-RRT [41].

Recently, the Fast Marching Trees (FMT) [42], which combines the features of both PRM and sampling-based roadmap of trees (SRT) [43], is designed to reduce the number of obstacle collision-checks and increase the efficiency in high-dimensional environments. Besides the previous approaches, some other sampling-based motion planning algorithms are also notable, such as the cross-entropy motion planning algorithm [44] and expensive space trees (EST) [45].

2.5 Research Challenges

In this chapter, the related work in collaborative robotic systems, robot learning from demonstrations, and robot motion planning and control are reviewed. From the survey, we can summarize the following research gaps.

First, the lack of suitable robotic systems remains a challenge, especially a compact size collaborative mobile manipulator with high payload capability, flexibility, and reconfigurability. The research to address this challenge is further discussed in Chapter 3.

Secondly, online teaching and learning through human demonstration is not easy to achieve. The intuitive and easy-to-use robot teaching approaches for both expert and non-expert users, and the accurate, fast and affordable online programming approaches are still worth investigating. The research to address this challenge is further discussed in Chapter 4 to Chapter 7.

Thirdly, the lack of suitable human-centered motion planning frameworks, which are safe and friendly to humans in collaborative tasks, for automotive assembly applications, remains a research gap. The research to solve this challenge is further discussed in Chapter 8.

CHAPTER 3

COLLABORATIVE LIFT ASSIST ROBOT FOR AUTOMOTIVE ASSEMBLY

3.1 Introduction

A parallel mobile manipulator, the so-called Smart Companion Robot (SCR), for collaborative lift assistance in automotive assembly is proposed in this chapter. The initial prototype of the SCR is realized by merging a four-wheel-drive (4WD) Mecanum wheel mobile base and a 3-RPS parallel arm, which can achieve a six degree-of-freedom (DoF) movement for the payload attached to its upper platform.

The design requirements of the SCR are described in Section 3.2. The technical details of the mechanical and electronic architectures are presented in Section 3.3. The implementation of the kinematic control, force servoing, and visual servoing are discussed in detail in Section 3.4. In Section 3.5, the technical specifications of the SCR are proposed and compared with other representative commercial mobile manipulators for the manufacturing shop floors. The performance of the SCR is evaluated on a prototype robot with real collaborative assembly tasks in Section 3.6. The chapter is summarized in Section 3.7.

3.2 Design Requirements

Based on the overviews of mobile manipulators and applications for manufacturing [6], [12], which referenced examples of commercial-off-the-shelf mobile manipulators that are

developed and used for industrial purposes as of 2016, and compared the diversity of performance assessment methods available for these systems, we developed a Smart Companion Robot (SCR) to serve as a compact mobile lift-assist for automotive assembly tasks, capable of human-robot collaboration, emerges from the following design requirements:

- Small size: The SCR should be easily handled by a human worker and capable of navigating narrow spaces within automotive assembly lines so that the maximal base dimension is expected to be under 600 x 600 mm.
- High payload: The SCR must be able to transport and manipulate heavy parts in automotive assemblies, such as wheels, brake discs. The maximum payload should be up to 30 kg.
- Flexibility: The SCR should be able to manipulate and gravity-compensate different kinds of parts in a variety of sizes and shapes in the six DoF manipulation, such that the position and orientation of the parts can be adjusted accordingly to match with the assembly position.
- Intuitive user interfaces: The robot should be easy-to-use for human operators, Ease of collaborative interactions with intelligent assistive modes are necessary.

These requirements guided the creation of the SCR prototype and were realized by mounting a parallel manipulator on a mobile base with the capacity to intelligently assist the co-manipulation of heavy payloads for automotive assembly applications. The SCR is intended to track its human partner and facilitate the presentation of "the right part at the right time" by visual servoing to track a marker with a QR code. Moreover, it can be

responsive to the human-associated touch and implement gravity compensation for the delivered part by force servoing.

3.3 Design of The Smart Companion Robot (SCR)

3.3.1 Mechanical Architecture

A custom-designed 4WD omnidirectional mobile base with Mecanum wheels is developed as the foundation of the SCR for adapting to the complicated dynamic environment in the workshop. It consists of a central platform and four Mecanum wheels, and each wheel is independently driven by a gear motor. Unlike most traditional Automated Guided Vehicle which cannot move sideways, the 4WD mobile base has omnidirectional mobility (forward/back, left/right, yaw), which greatly enhances the mobility and maneuverability of the system, especially in a narrow space.

A custom-designed 3-RPS parallel manipulator is built for the SCR, which has three degrees of freedom (up/down, pitch, roll) and significantly improves the payload capacity of the system. It can achieve gravity compensation for a 30 kg part. The overall system has six degrees of freedom in total when the 3-RPS parallel manipulator is mounted on the center platform of the 4WD mobile base with Mecanum wheels.

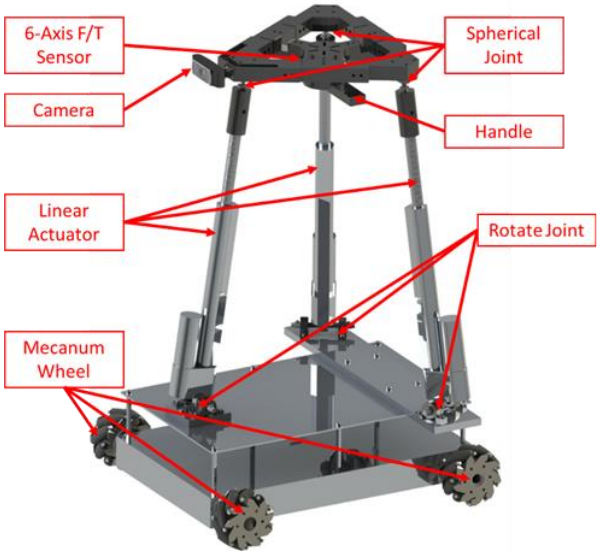


Figure 3.1. Mechanical Architecture of the SCR.

3.3.2 Electronic Architecture

The electronic system diagram of the SCR, including the power supply system, actuators, the perception system, and the controller, is presented in Figure 3.2. The SCR consists of 7 motors, including 4 brushless DC motors for the mobile base and 3 linear actuators for the parallel manipulator, a multi-threading Microcontroller Unit (MCU) to handle all the low-level controls, a PC laptop running Robot Operating System (ROS) for high-level motion planning, a 6-axis F/T sensor, a camera, and battery packs.

For the 3-RPS parallel manipulator, each linear actuator is independently driven by a DC motor amplifier rated at 15A. A PWM signal is used to control the velocity of the linear actuator while a potentiometer provides a 0~5V analog feedback. For the mobile base, each wheel is driven by a gear motor powered by a dual-channel motor driver rated at 25A per channel. The 1024 lines count/revolution encoder on the gear motor provides the position feedback of each wheel.

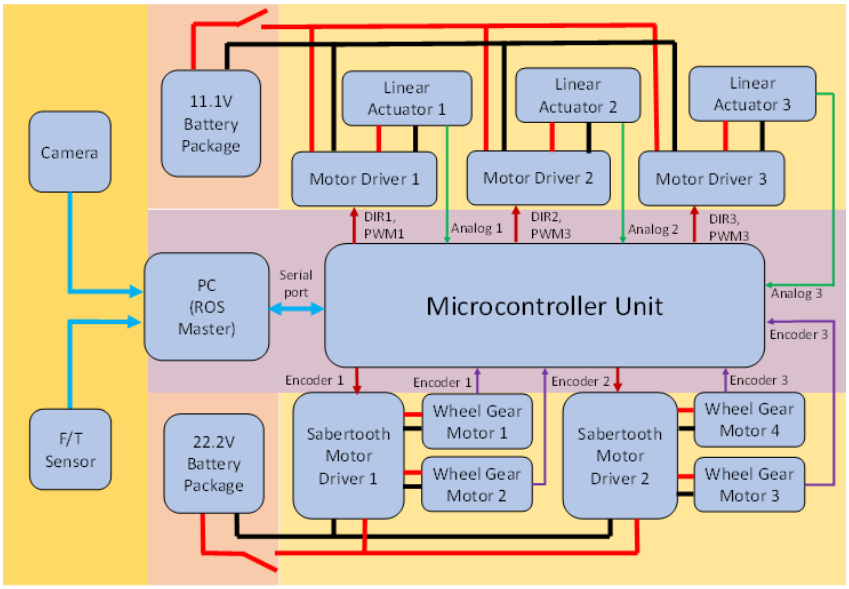


Figure 3.2. Electronic design of the SCR.

For the robot perception, we employ an OptoForce 6 Axis F/T Sensor to support the force-servoing mode (discussed in the next section), which is mounted on the upper platform of the parallel manipulator. The forces and torques are sampled (300 Hz) and post-processed to predict the human intentions of transportation and manipulation. Moreover, a wide-angle webcam (640x480, 30fps) is integrated for deploying the visual servoing mode. All the sensory and feedback data are collected by the PC-based controller for the implementation of motion planning and control strategies. Then the velocity commands are directly sent to and then executed by the MCU.

3.4 Modeling and Control Methodology

3.4.1 Kinematic Modeling

Based on the mechanical design, the kinematic model of the SCR is derived for the velocity control for the end-effector. The modeling approach of the 3-RPS parallel architecture determines the Jacobian matrix of the linear actuator to the end-effector [46].

The definitions of the coordinate system are illustrated in Figure 3.3. The three rotation joints are fixed on the center platform of the mobile base and the centers of the rotation joints are defined as A_1 , A_2 and A_3 . The origin (M) of the coordinate system of the mobile base (as well as the bottom platform of the 3-RPS parallel manipulator) is set at the center of the equilateral triangle $\Delta A_1 A_2 A_3$. The x-axis is in the direction of vector $\overrightarrow{MA_1}$ and z-axis is vertical to the center platform. The centers of the three spherical joints are defined as B_1 , B_2 and B_3 , which construct an equilateral triangle $\Delta B_1 B_2 B_3$. The origin (P) of the upper platform is set at the center of the triangle $\Delta B_1 B_2 B_3$ and the x-axis is in the direction of the vector $\overrightarrow{PB_1}$, and the z-axis is vertical to the upper platform.

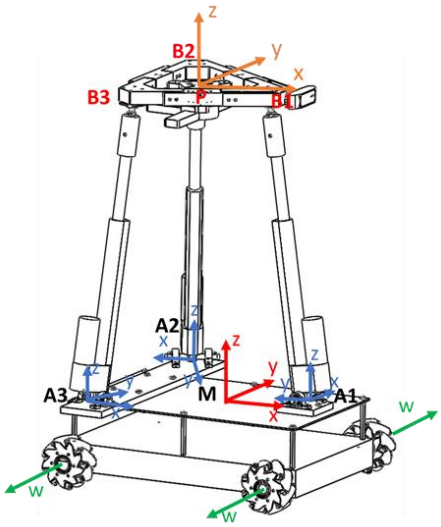


Figure 3.3. The coordinate system of the SCR.

For the velocity kinematics of the 3-RPS parallel manipulator, the velocities of z-direction ($\dot{z}_a$), pitch ($\dot{\beta}_a$) and roll ($\dot{\alpha}_a$) are defined as the inputs of the 3-RPS parallel

manipulator model, and then the velocities of the three linear actuators ($\dot{\mathbf{L}}$), as the outputs of the 3-RPS parallel manipulator model, can be calculated by

$$\dot{\mathbf{L}} = \mathbf{J}^{-1}(x_a, y_a, z_a, \alpha_a, \beta_a, \gamma_a, R_b, r_a)\begin{bmatrix} \dot{z}_a \\ \dot{\alpha}_a \\ \dot{\beta}_a \end{bmatrix} \tag{3.1}$$

Since the 3-RPS parallel manipulator has only 3 degrees of freedom, its velocities in these three degrees of freedom will also cause the velocities in the other three degrees of freedom, i.e., velocities in x-direction ($\dot{x}_a$), y-direction ($\dot{y}_a$), and yaw ($\dot{\gamma}_a$), which can be expressed by

$$\begin{bmatrix} \dot{x}_a \\ \dot{y}_a \\ \dot{\gamma}_a \end{bmatrix} = \mathbf{G}(\mathbf{L}_i, \mathbf{u}_i, \mathbf{r}_i)\begin{bmatrix} \dot{z}_a \\ \dot{\alpha}_a \\ \dot{\beta}_a \end{bmatrix} \tag{3.2}$$

where $\mathbf{G}$ is the velocity mapping function which can be expressed by a function of the following items: $\mathbf{L}_i (i = 1,2,3)$ represent the vector of each linear actuator; $\mathbf{u}_i (i = 1,2,3)$ represent the unit vector of each rotating joint, and $\mathbf{r}_i (i = 1,2,3)$ represent the position of each vertex with respect to the upper platform of the 3-RPS parallel manipulator.

For the 4WD mobile base, its velocity kinematic model can be expressed by

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \mathbf{\Psi}(r_w, l_x, l_y)\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{\gamma}_b \end{bmatrix} \tag{3.3}$$

where $\dot{x}_b$, $\dot{y}_b$ and $\dot{\gamma}_b$ represent the x-direction, y-direction, and yaw velocities of the mobile base, $\dot{q}_i (i = 1,2,3,4)$ represent the angular velocities of the four Mecanum wheels and $\mathbf{\Psi}$ is the inverse Jacobian of the mobile base, which can be expressed as a function of

the outside radius of the Mecanum wheel r_w, the half-length of the wheelbase l_x, and the half-length of the wheel span l_y.

Define the position and orientation of the end-effector (i.e., the center point on the upper platform) in the world frame as $[\dot{z}, \dot{\alpha}, \dot{\beta}, \dot{x}, \dot{y}, \dot{\gamma}]^T$. By combining the kinematic models of the parallel manipulator and mobile base, the inverse velocity kinematic model of the SCR is expressed as

$$\begin{bmatrix} \dot{\mathbf{L}}_{(3\times1)} \\ \dot{\mathbf{Q}}_{(4\times1)} \end{bmatrix}_{(7\times1)} = \begin{bmatrix} \mathbf{J}^{-1}{}_{(3\times3)} & \mathbf{0} \\ -\boldsymbol{\Psi}_{(4\times3)} \cdot \mathbf{G}_{(3\times3)} & \boldsymbol{\Psi}_{(3\times3)} \end{bmatrix} \begin{bmatrix} \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{x} \\ \dot{y} \\ \dot{\gamma} \end{bmatrix} \tag{3.4}$$

Therefore, given desired velocities of the end-effector, the velocities of the three linear actuators of the parallel manipulator and the velocities of the four wheel motors of the mobile platform can be calculated respectively, which will then be implemented by their corresponding low-level motion controllers. In our design, the desired velocities are generated either from force/torque data in force servo mode (i.e., human-robot physical interaction mode) or from visual data in visual servo mode (i.e., human-following mode).

3.4.2 Human-robot Interaction Modeling

In force servo mode, the human and robot hold the part at the same time. When the gravity of the delivered part is always compensated by the robot, the human worker can operate the robot regardless of the weight of the part. The human-robot interaction model used in our design can be expressed as:

$$F_s = m\ddot{X}_d + c\dot{X}_d \tag{3.5}$$

$$T_s = J\ddot{\theta}_d + \xi\dot{\theta}_d \tag{3.6}$$

where F_s is the force input of the force/torque sensor, m is the virtual mass of the part, c is the virtual damping of the part, $\dot{X}_d$ is the desired linear velocity of the end-effector, $\ddot{X}_d$ is the desired linear acceleration of the end-effector, T_s is the torque input of the F/T sensor, J is the virtual movement of inertia, ξ is the virtual rotary damping, $\dot{\theta}_d$ is the desired angular velocity of the upper platform, and $\ddot{\theta}_d$ is the desired angular acceleration the upper platform. Based on the interaction model, the desired linear velocity and angular velocity of the end-effector are calculated by

$$\dot{X}_d(t) = \frac{1}{m} e^{-\frac{c}{m}t} \int F_s(t) e^{\frac{c}{m}t} dt \tag{3.7}$$

$$\dot{\theta}_d(t) = \frac{1}{J} e^{-\frac{\xi}{J}t} \int T_s(t) e^{\frac{\xi}{J}t} dt \tag{3.8}$$

3.4.3 Robot Motion Planning

Targeting human-robot collaborative tasks in automotive assembly, the SCR is designed to work in and intelligently switch between visual servoing and force servoing modes. The visual servoing mode is designed for long-distance delivery of heavy parts which is regarded as the first stage of the assembly task. In this mode, the SCR follows the human worker (by tracking the QR code marker) around the assembly workspace. The force servoing mode is designed for accurately assembling the part to the vehicle, as the second stage of the assembly process. In this mode, the human worker can adjust the position and orientation of the part by directly applying forces/torques on the operating handle. In addition, the range of the roll and pitch angles are limited by the software

thresholds for safety purposes (e.g. the falling of an object) based on the weight and geometric shape of payloads, although the mechanism is capable of a larger range of movements.

The software package for control implementation of the SCR is built based on the Robot Operating System (ROS) and Visual Servo Platform (VISP) [47]. The general control diagram is illustrated in Figure 3.4. Mode-switching from the default safety/rest mode is determined by the detection of the QR code (to visual-servo mode) and/or presence of handle-forces (to force-servo mode). In the visual servoing mode, the SCR seeks to maintain a fixed distance with the detected QR marker. Low-level wheel control commands (for only the mobile base) are determined by the position and orientation of the marker with QR code and the kinematic model of the mobile base. The SCR comes to a safety/rest mode immediately if the detection is lost and resumes when the QR marker is re-detected. The SCR switches to the force servoing mode when a threshold handle-force is exceeded. In this mode, the control commands are determined by the 6-axis F/T sensor inputs and the kinematic model to synchronize all actuators of the SCR. Based on the high-level motion planning, the MCU realizes low-level independent PID control for the 3 linear actuators of the parallel mechanism and 4 wheels of the mobile base.

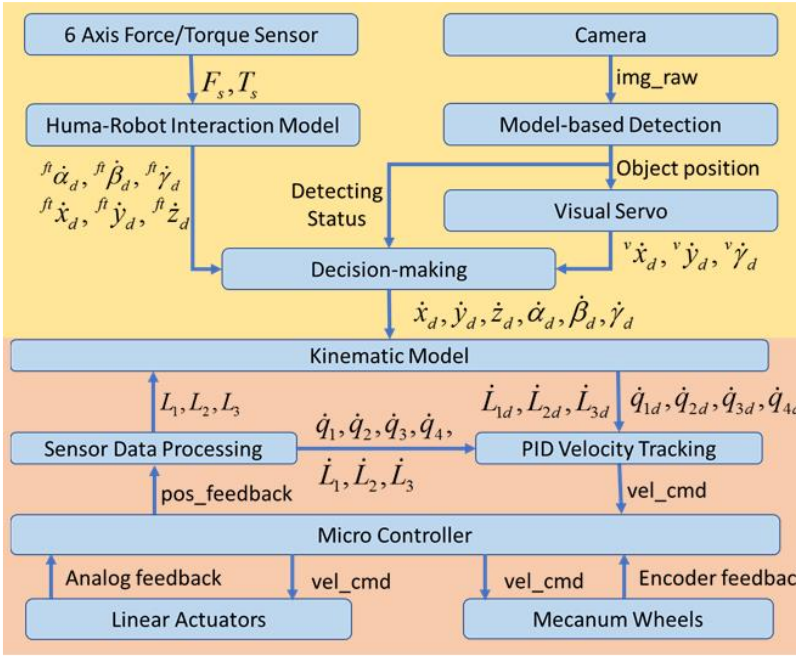


Figure 3.4. The control diagram of the SCR.

3.5 Prototype and Specifications of the SCR

In this section, we first analyze the performance of the SCR and present a comparison between the SCR and other serial-arm mobile manipulators. An experiment representative of an automotive assembly task is implemented to further test the functionality and performance of the SCR.

Figure 3.5. The prototype of the SCR.

3.5.1 Specifications

A real SCR prototype is built for the validation and demonstration, as shown in Figure 3.5. The specifications for its performance are shown in Table 3.1. The current 3D printed realization of structural components limits the rated payload of the SRC to 18 kg (maximum payload is 30 kg). However, the feedback rod linear actuators in the 3-RPS parallel mechanism are capable of each supporting up to 68 kg static payload. Hence, with machined components, our SCR architecture is capable of handling up > 70 kg payload without increasing the dimensions of the robot.

The speed performance and motion ranges of the robot end-effector (i.e., the center point on the upper platform) are determined by the configurations of the robot and capabilities of the actuators. For the 3-RPS parallel architecture of the SCR, the side length of the equilateral triangle, which is formed by the three rotation joints, is 390 mm; the side length of the equilateral triangle, which is formed by the three spherical joints, is 260 mm; and the stoke of the and maximum speed of the linear actuators are 200 mm and 20 mm/s

respectively. For the mobile base, the maximum speed of the gear motor is 75 rpm and the outside diameter of the Mecanum wheel is 122 mm. Based on these configurations, including dimensions and actuator capabilities, we derive the performances of the robot as follows. The maximal translational speeds for X, Y, and Z are 500 mm/s, 500 mm/s, and 20 mm/s respectively. The maximal rotational speeds for Roll, Pitch, and Yaw are 10 degree/s, 10 degree/s, and 30 °/s respectively. The motion ranges for X, Y, and Yaw are unlimited. The motion range for Z is 600-800 mm. The motion ranges for Roll and Pitch are both +/- 30 degrees. We have set software limits to avoid running actuators beyond their acceptable ranges. The limit of tilt (roll and pitch) angles can be appropriately adjusted by controlling angle range thresholds based on the weight and geometric shape of various payloads.

Table 3.1. The performance parameters of the SCR.

Parameter	**Value**
Base Dimension	550 mm (L)*550mm(W)
Rated Payload	18 kg
Maximum Payload	30 kg
Curb Weight	31.8 kg
Maximum Speed	x-axis: 500 mm/s y-axis: 500 mm/s z-axis: 20 mm/s yaw: 30 °/s pitch: 10 °/s roll: 10 °/s
Maximum Motion Range	z-axis: 600-800 mm pitch angle: +/– 30° roll angle: +/– 30°
Battery Type	Lipo
Operating Temperature	0° – 40° C
DOF	6
Software	ROS Kinetic, VISP
Wi-Fi	802.11-2.4/5.0 GHz

3.5.2 Comparison

To illustrate the advantages of the SCR, we compare the robot with two representative commercial mobile manipulators for the manufacturing shop floors: KUKA KMR iiwa 14 [48] (the type with a larger arm and payload) and Yaskawa YMR-12 [49]. The comparison metrics include robot dimensions, curb weight, payload, operation DOFs, number of actuators, and base drive types. The results of the detailed comparisons are listed in Table 3.2. From the results, we can intuitively see that the SCR has the smallest size, highest payload, lightest weight, and a minimum number of motors. Specifically, the rated payload density for SCR, KMR, and YMR are 59.5 kg/m^2, 20.6 kg/m^2, and 5.6 kg/m^2 respectively. We can see that the SCR requires the smallest dimensions to handle the same amount of payload. The payload to curb-weight ratios for SCR, KMR, and YMR are 0.57, 0.03, and 0.01 respectively. We can see that the SCR requires the smallest curb weight to support the same amount of payload. The ratios between the operation DOFs and the required number of motors for SCR, KMR, and YMR are 6/11, 6/11, and 6/8 respectively. We can see that the SCR requires the minimum number of motors to achieve 6 operation DOFs of the end-effector. Moreover, it also has an omnidirectional mobility capability to make the robot much more flexible to operate on crowded manufacturing shop floors in a holonomic manner.

Table 3.2. Comparison: SCR vs KUKA iiwa vs YMR-12

Parameter	SCR	KUKA KMR iiwa 14	Yaskawa YMR-12
Robot Dimensions	L: 550 mm W: 550 mm	L: 1080 mm W: 630 mm	L: 1805 mm W: 1186 mm
Rated Payload	18 kg	14 kg	12 kg
Curb Weight	31.8 kg	419.5 kg	980 kg
Operation DOFs	6	6	6
No. of Actuators	7	11	8
Base Drive Type	Omni-directional	Omni-directional	Differential

3.6 Human-robot Collaborative Assembly Experiment

In order to demonstrate the effectiveness of the SCR, we choose a realistic automotive assembly task which is to install a heavy part into a vehicle by simultaneously inserting multiple bolts into their corresponding fixture holes. The corresponding assembly tasks include, but are not limited to, front differential installation, wheel hub installation, and brake rotor installation. Conventionally, human workers have to carry these parts, transport them to the vehicle, and then hold and manipulate them by appropriately adjusting the positions and orientations in order to install them. In this experiment, we will use the SCR to assist human workers to accomplish the task.

3.6.1 Experimental Setup

The configuration of the realistic assembly task is illustrated in Figure 3.6 (a). Part A is a 10 kg metal part with a size of 254 mm*254 mm and containing 4 bolts (15 mm in diameter) at the corner. Part B is another piece of the metal fixture with 4 holes (20 mm in diameter) at each corner. Part A and Part B can assembly together when their position and orientation are matched with each other so that all the 4 bolts on Part A can pass through the 4 holes on Part B. In this task, Part A is fixed on the upper platform of the 3-RPS parallel manipulator and its position and orientation can be adjusted by operating the SCR.

Meanwhile, Part B is fixed on a workbench by a bench vise. Part A is originally placed to be mismatched with Part B as shown by yellow dashed lines in Figure 3.6 (a). In order to install Part A into Part B, one has to appropriately adjust all 3 positions and 3 orientations. The panorama of the collaborative task is illustrated in Figure 3.6 (b). Three obstacles are set in the central zone of the scene. The trajectory from the start position to the workspace of the assembly task is shown by the yellow dash arrow. The distances between the corresponding nearest point of the obstacles are 85 cm and 95 cm as shown by blue lines in Figure 3.6 (b). The entire process of the collaborative assembly task can be allocated as two steps:

- Step 1: Heavy part transport by visual servoing (Figure 3.6 (b)): the SCR helps the human worker to transport the heavy part to the workspace of the assembly task. During this process, the robot is working in the visual servoing mode. It follows the human worker by tracking a QR marker held by the human. Guided by a human worker, it needs to pass through the narrow working space by leveraging its flexible omnidirectional mobility capability and arrive at the target workspace of the assembly task.
- Step 2: Heavy part assembly by force servoing (Figure 3.6 (a)): the SCR helps the human worker to accomplish the assembly task. During this process, the robot is working in the force servoing mode. The human worker can appropriately adjust the position and orientation of the part on the SCR with the handle and install the part into the fixture part which is fixed on the workbench with a bench vise.

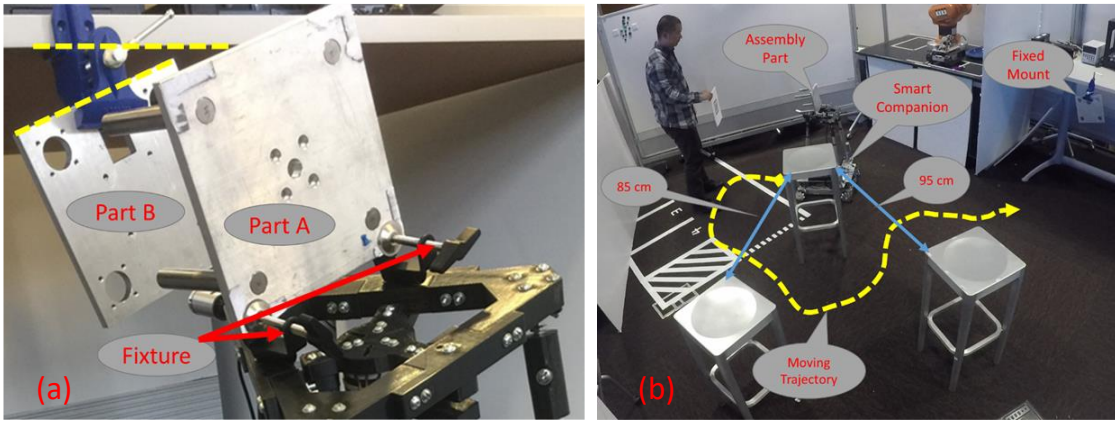


Figure 3.6. Human-robot collaborative assembly experiment setup.

3.6.2 Velocity Kinematics Validation

The velocity kinematics is validated by manipulating the end-effector of the SCR in all six degrees of freedom, i.e., three translations and three rotations. The results are shown in Figure 3.7 (a) ~ (f) respectively. In each result, the first row of blue curves describes the human applied forces and torques; the second-row of blue curves represent the desired end-effector linear and angular velocities generated by the human-robot interaction model based on the human applied forces and torques; the second-row of red curves represent the actual end-effector linear and angular velocities achieved by the robot using the proposed kinematic control model. The desired linear and angular velocities, which respond to a variety of human input forces and torques, demonstrate the effectiveness of the human-robot interaction model. The actual linear and angular velocities, which track their desired values closely, demonstrate the effectiveness of the developed kinematic control model. Therefore, we can see that the kinematic velocity control in six degrees of freedom is well achieved and it enables the human to interact with the robot by using naturally small forces/torques to operate the robot in six degrees of freedom.

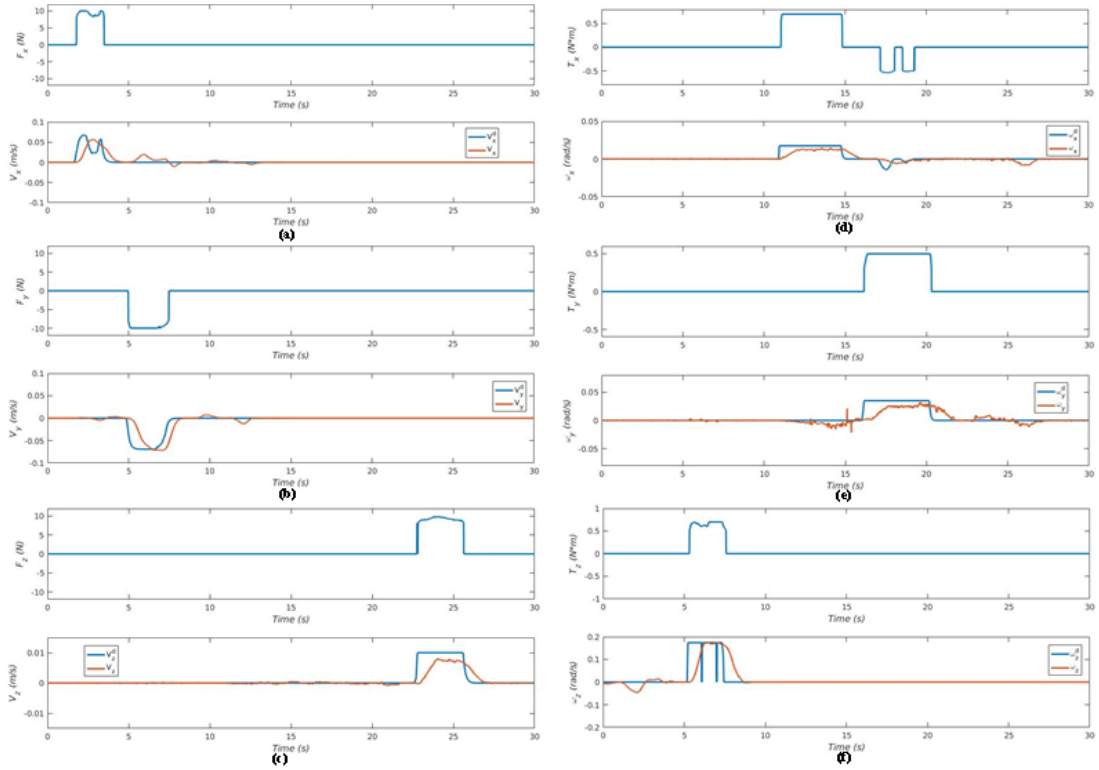


Figure 3.7. The sensor data inputs and velocity kinematics outputs

3.6.3 Part Transport Validation

In the process of human-guided part transport, the SCR can pass through the narrow space without hitting any obstacles and successfully deliver the part to the target workspace for the following assembly task. Figure 3.8 (a) ~ (d) illustrates the reaction of the SCR when the tracking of the QR code is lost. The positions of the obstacles are marked by the red crosses, while the position and orientation of the mobile base are marked by a yellow point and an arrow. The results indicate that the SCR can stop immediately when the QR code is lost in the camera frame in Figure 3.8 (b), and automatically resume following the QR code when it is re-detected in Figure 3.8 (c) and (d). The process of passing through the narrow space is presented in Figure 3.9 (a) ~ (d). The results indicate that the SRC can

make a sharp turn and pass through the channel of 85-95 cm in width. Moreover, the test results also demonstrate that the robot can pivot with respect to the center of the mobile base benefiting from the omni-directional mobility of mobile base.

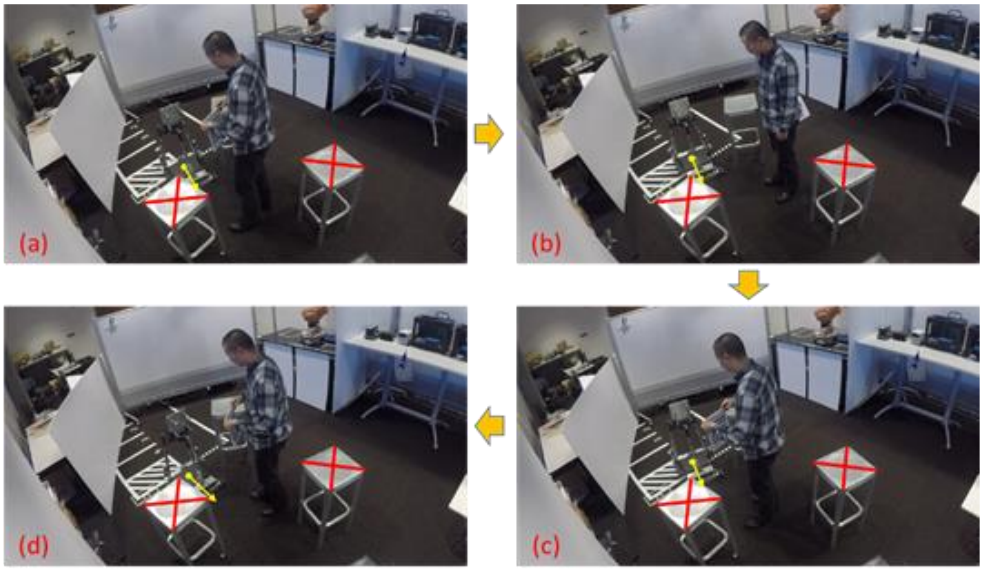


Figure 3.8. Test for lost tracking of the QR code.

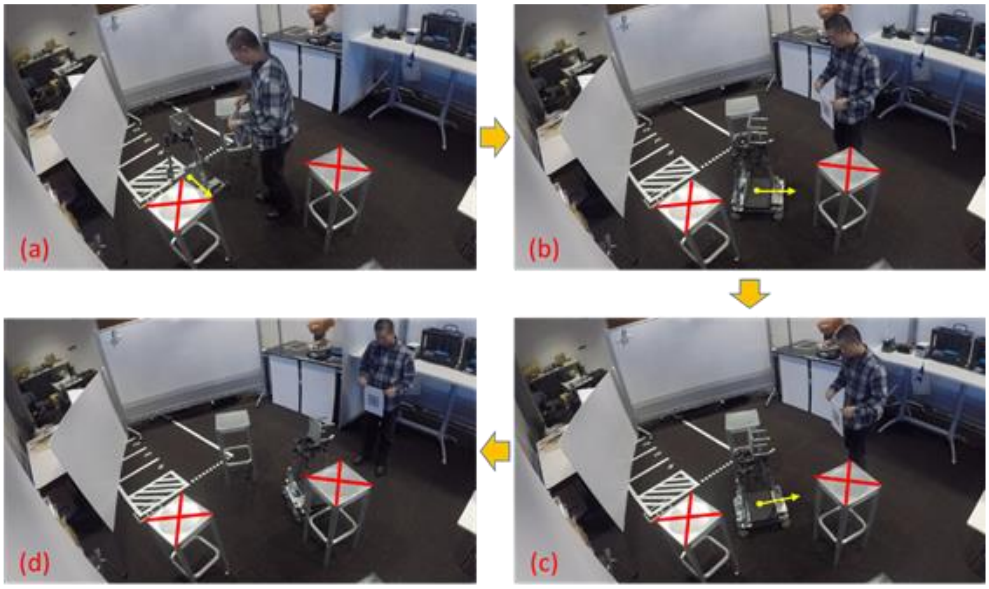


Figure 3.9. Test for passing through the narrow space.

3.6.4 Part Assembly Validation

The close-up shots of the part assembly process are shown in Figure 3.10 (a) ~ (d). To demonstrate the movements of the robot more clearly, we record these close-up shots by a GoPro camera mounted on the workbench nearby the bench vise. The corresponding edges of the parts are marked by the yellow and red lines in each frame. In this process, the robot works in force servoing mode and the human worker operates the robot with the handlebar underneath the 6-axis F/T sensor. First, Figure 3.10 (a) and (b) illustrate that some preparatory maneuvers are conducted so that Part A, whose edges are marked by red lines, is approximately lifted to the same height as Part B whose edges are marked by the yellow lines. Afterward, Figure 3.10 (b) and (d) demonstrate that some fine maneuvers are implemented to adjust the position and orientation of Part A to make the bolts on Part A align with the holes on Part B and then install Part A into Part B.

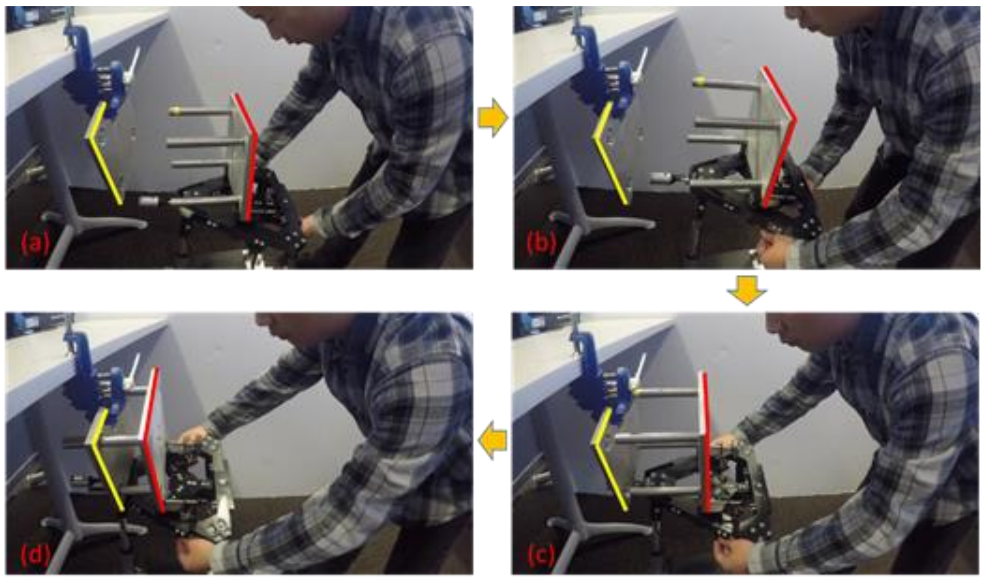


Figure 3.10. Part assembly process.

3.7 Conclusion

A Smart Companion Robot (SCR) is designed, developed, and validated to collaborate with humans in automotive assembly. It can hold a heavy payload for humans and enable humans to use their motions and finger forces to transport and manipulate heavy parts in all six degrees of freedom. By taking advantage of the robot, human workers can be assisted to effectively, flexibly, and conveniently handle heavy parts in automotive assembly, which has great potential benefits in increasing the automotive assembly production efficiency and quality as well as improving ergonomics. The SCR is a representative example of how we can leverage both human and robot capabilities in manufacturing, where the human handles dexterous assembly tasks while the robot handles the heavy payload of automotive parts. The application of such a robot system is clearly not limited to automotive assembly alone. Any manufacturing task that involves heavy-payload transportation and manipulation tasks could be benefited from this type of robot. In addition, the robot could also have a wide range of potential applications in other areas such as assisting the elderly to transport and manipulate heavy objects at home [50], [51], and even assisting soldiers to carry heavy goods like a “robotic mule” in battlefields by leveraging its advantages of compact size, heavy payload, high flexibility, and intuitive user interfaces.

CHAPTER 4

ROBOT LEARNING FROM DEMONSTRATION ON OBJECT PLACING TASKS IN ASSEMBLY

4.1 Introduction

Object pick-and-place is one of the most common manipulations. Especially in collaborative assembly tasks, besides the proper picking capability, how to place the part correctly is also necessary for robots to accomplish the assembly process. Though many studies have been conducted in robot grasping and motion planning for picking, little attention has been paid to object placing tasks. To fulfill this research gap, I propose a vision-based approach to modeling the object placing tasks via contour-based task representation in this chapter. The proposed framework can be used to identify the correction of the task process and the final state of the object placing tasks with a single web camera.

The sequential assembly operations may contain multiple pick-place actions, which pick up an object and place it onto a workpiece with a specific position and orientation. To accomplish the task, correct objects should be placed in the correct position and orientation onto the workpiece in a correct sequence. Inspired by the Semantic Event Chain (SEC) [23] and Generalized Voronoi Diagrams (GVD) [52], we proposed a framework to modeling object-placing tasks, which allows the robot to learn the action sequences and the import intermedia and final states of object placing tasks from human demonstration

videos. In the proposed framework, Rational Scene Dictionary (RSD) is used to present the object-action relations and sequence of the task, while GVD-based contours of the Keyframes of Task (KFT) are used to present the relative position and orientation between the corresponding objects and workpiece in the import intermedia and final state. The object placing tasks can be demonstrated through either a video of human demonstration or a simulation, and then it can be modeled with the proposed framework while a large number of pre-defined features, task states, or primitive skills are not necessary.

The modeling and learning of object placing tasks are presented in Section 4.2. The results of task learning and validation are discussed in Section 4.3. The chapter is summarized in Section 4.4.

4.2 Object Placing Task Modeling and Learning

In this section, we will introduce the approaches and algorithms of the framework for object placing task modeling. Before discussing details about the algorithms, we provide an overview of the framework. Figure 4.1 illustrates the block diagram of the framework. The task demonstration video is regarded as an image sequence. The segmentation and tracking, which is not the key part of our research, is achieved by standard methods based on morphological transformations [53]–[55]. The rational scene dictionary (RSD) presents the general special relations between segment pairs. The key frames of a task (KFT) abstracted from RSD encode the important changes in spatial relations in the task process. According to the object relation changes in KFT, the frames and the corresponding segment-pairs in these frames for the computation of GVD-based contours can be determined. When an object is placed onto a workpiece, the position and

orientation of the object with respect to the workpiece can be represented by the GVD-based contours. Fourier Descriptor (FD) [56], [57] is used to represent the contours. The similarity between two GVD-based contours is estimated by the Hausdorff distance [58]. Therefore, in this framework, the sequence and general special relations for object placing are modeled by KFT, which is a subset abstracted from RSD. For a specific object-pair, the relative accurate position and orientation of corresponding objects are modeled by GVD-based contours, while the FD and shape similarity measurement could potentially be used for fault detection and waypoint searching. In the following sections, we describe the algorithm of each step in detail.

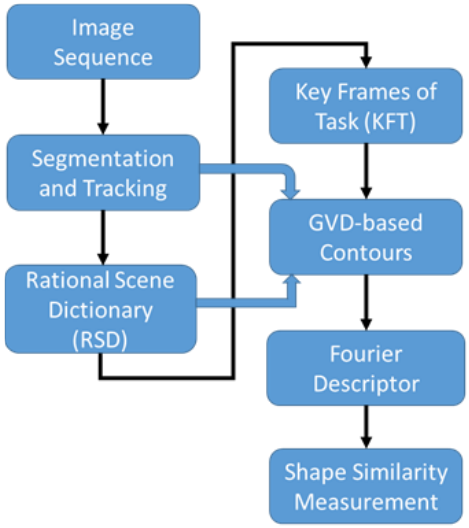


Figure 4.1. Block diagram of the object placing task modeling framework.

4.2.1 Rational Scene Dictionary (RSD)

The RSD is a dictionary that presents the general special relations for all object-pairs in the workspace. The pure background of the workspace is also regarded as a large object as a whole. In our framework, we considered two kinds of relations for object-pair, touching and overlapping, such that the RSD consists of two sub-dictionaries, one is for

touching object-pairs and the other is for overlapping object-pairs. The algorithm of RSD starts with the vertical and horizontal scan of the segmentation image. We assume that all the objects used for object-placing tasks are given, and the set of all the objects is denoted as

$$P = \{p_1, p_2, p_3, ..., p_n\} \tag{4.1}$$

In a video frame, for each vertical or horizontal line in the segmentation image, we can obtain an object sequence according to the pixel values. Moreover, most of these sequences may repeat multiple times in the same frame depending on the shapes and special relations of the objects. The repeated times of all the object sequences in the frame are counted as a reference to determine the general special relations between different objects. Therefore, for each video frame, we have a set of object sequences

$$OS(x) = \{os_{1,}, os_2, os_3, ..., os_m\} \tag{4.2}$$

where x is the frame ID and the elements in $OS(x)$ are

$$\begin{aligned} os_k &= \{Seq_k : C_k\} \ (k = 1, 2, ..., m) \\ Seq_k &= \{q_1, q_2, ..., q_j\} \ (q \in P) \end{aligned} \tag{4.3}$$

where Seq_k is the object sequence and all its elements are belong to the set of objects. C_k is the sum of the lines in the frame corresponding to Seq_k. The touching and overlapping relations are determined according to the results of segmentation image scan by the rules in following.

To determine the touching relation dictionary in this video frame, we need to search for all the object sequences in this frame. In the object sequence os_k, once object p_a and

p_b is adjacent, the object pair ${}^{t}op_{a,b}$ is recorded and its corresponding counter number of touching relation ${}^{t}C_{a,b}$ will increase C_k. After traversing all the object sequences in the frame, a dictionary of touching relations of this frame can be written as

$$\begin{aligned} {}^{x}D_{tch} &= \left\{ {}^{t}op_{a_1,b_1} : {}^{t}C_{a_1,b_1}, {}^{t}op_{a_2,b_2} : {}^{t}C_{a_2,b_2}, ..., {}^{t}op_{a_r,b_r} : {}^{t}C_{a_r,b_r} \right\} \\ &\left(a_i, b_i \in Seq_k \left(i = 1,2,...,r \ and \ k = 1,2,...,m \right) \right) \end{aligned} \tag{4.4}$$

For overlapping relation, since we want eventually present the relative position and orientation of corresponding segment pair through GVD-based contours, we define one object being overlapped by the other object when the later object is fully surrounded by the former object. To identify this kind of overlapping relations in the frame, for specific object sequence Seq_k, we check the pair-triplet and the find the sub-sequence in the format $\{q_a, q_b, q_a\}$, which indicates q_a is potentially overlapped by q_b. Once the sub-sequence is found, the object-pair ${}^{o}op_{a,b}$ is recorded and its corresponding counter number of overlapping relation ${}^{o}C_{a,b}$ will increase C_k. A set of object-pairs and their corresponding counter number are obtained after traversing all the object sequences. In this set of object-pairs, the objects that overlapping multiple objects and the objects crossing the boundary of the workspace are removed. The remaining object-pairs and their corresponding counter numbers formulate the dictionary of overlapping relations, which can be written as

$$\begin{aligned} {}^{x}D_{ovp} &= \left\{ {}^{o}op_{a_1,b_1} : {}^{o}C_{a_1,b_1}, {}^{o}op_{a_2,b_2} : {}^{o}C_{a_2,b_2}, ..., {}^{o}op_{a_s,b_s} : {}^{o}C_{a_s,b_s} \right\} \\ &\left(a_i, b_i \in Seq_k \left(i = 1,2,...,s \ and \ k = 1,2,...,m \right) \right) \end{aligned} \tag{4.5}$$

Figure 4.2 illustrates an example of the vertical and horizontal scan of the segmentation image. Object 2 overlaps Object 4 and Object 4 overlaps Object 6 (the

workspace background). Object 3 is crossing the boundary of Object 4 and Object 6, such that Object 3 is not overlapping either Object 4 or Object 6 but they three are touching with each other.

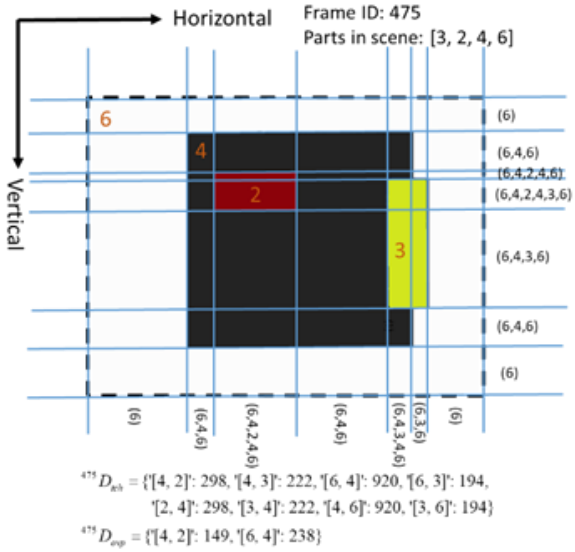


Figure 4.2. Image scan for RSD computation.

The RSD of a demonstration video is obtained by combing the dictionary of touching relations and the dictionary of overlapping relations with their frame ID, which can be expressed as

$$RSD(X) = \{x_1 : [^{x_1}D_{tch}, ^{x_1}D_{ovp}], x_2 : [^{x_2}D_{tch}, ^{x_2}D_{ovp}], ..., x_N : [^{x_N}D_{tch}, ^{x_N}D_{ovp}]\} \quad (4.6)$$

where X represents the specific object placing task and N is the total number of frames in the demonstration video.

4.2.2 Key Frames of Object Placing Tasks

In the object-placing tasks, we mainly concern two kinds of task knowledge, one is the sequence of objects that are placed, and the other is the final state of each object that is placed. Therefore, the keyframes of task (KFT) are needed to be abstracted from the RSD

of the specific task. Based on the algorithm of RSD generation, KFT can be determined by the length change of the overlapping relation dictionaries. If ${}^{x_k}D_{ovp}$ is larger than ${}^{x_{k+1}}D_{ovp}$ then one or more objects are moving and starting to overlap other objects. If the length of ${}^{x_k}D_{ovp}$ is smaller than the length of ${}^{x_{k+1}}D_{ovp}$ then one or more objects are moving and starting to cross the boundaries of other objects. Therefore, if the length of the overlapping relation dictionary decreases at i^{th} frame and it keeps the same from $(i-j)^{th}$ $(j<i)$ frame to i^{th} frame, then the roundness of $(i-j/2)$ is used as the frame number for one of the keyframes. Besides these frames, the last frame of the demonstration is also considered as one of the keyframes. The KFT of a specific task is a subset of the RSD, which can be written as

$$KFT(X)=\{x_{k_1}:[{}^{k_1}D_{tch},{}^{k_1}D_{ovp}],x_{k_2}:[{}^{k_2}D_{tch},{}^{k_2}D_{ovp}],...,x_{k_M}:[{}^{k_M}D_{tch},{}^{k_M}D_{ovp}]\} \quad (k_1,k_2,...,k_M\in[1,N]) \tag{4.7}$$

Figure 4.3 illustrates an example of KFT for a demonstration with 617 frames in total. The task is to place the red-rectangle object on to the dark-rectangle object at its up-right region, and then place the yellow-rectangle object on to the dark rectangle object at its left region. Three keyframes are abstracted based on the RSD of this task. The 128th frame represents one of the middle waypoints before the red-rectangle object being placed onto the dark rectangle object. The 402nd frame represents the final location of the red-rectangle object on the dark-rectangle object and one of the middle waypoints of the yellow-rectangle object before it is placed onto the dark-rectangle object. The 617th frame is the final state of all the objects of this task.

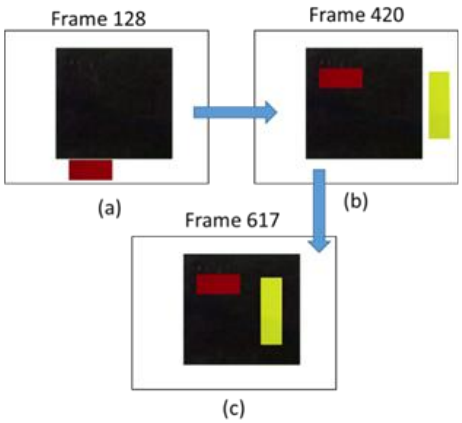


Figure 4.3. Example of KFT for a demonstration video.

4.2.3 Task Representation by GVD-based Contours

From the previous section, we have obtained the keyframes of the task. To describe the relative location between objects in these specific keyframes, GVD-based contours with respect to certain objects are computed for each keyframe. Figure 4.4 illustrates the steps and results of the computation of the GVD-based contours for a specific keyframe in KFT. First, one object and the objects placed onto it can be represented by binary occupancy grids. Based on the values in KFT, the binary occupancy grids for each object who has one or more other object placed on it can be calculated. For example, Figure 4.4 (b) represents the objects that directly on the workspace background in the format of binary occupancy grids, while Figure 4.4 (f) represents the red-rectangle object that is placed onto the dark-rectangle object. Then GVD of each binary occupancy grids is computed through classical Brushfire algorithm [59] such as the results in Figure 4.4 (c) and (e). In order to use the GVD to describe the relative position and orientation between the corresponding object and reduce the influence of the noise in image segmentation and tracking, the external profile of GVD is computed to get the GVD-based contours. These GVD-based

contours are closed shapes showing as the green contours in Figure 4.4 (a) and (d), which contain the information of the relative position and orientation of the object and can be represented by Fourier Descriptor. The GVD-based contours of corresponding frames in KFT can be written as

$$\begin{aligned} GVD(X) = \{ & x_{k1} : \{[{}^{k_1}q_1, {}^{k_1}cnt_1], [{}^{k_1}q_2, {}^{k_1}cnt_2], \ldots [{}^{k_1}q_{m1}, {}^{k_1}cnt_{m1}]\}, \\ & x_{k2} : \{[{}^{k_2}q_1, {}^{k_2}cnt_1], [{}^{k_2}q_2, {}^{k_2}cnt_2], \ldots [{}^{k_2}q_{m2}, {}^{k_2}cnt_{m2}]\}, \ldots \\ & x_{k_M} : \{[{}^{k_M}q_1, {}^{k_M}cnt_1], [{}^{k_M}q_2, {}^{k_M}cnt_2], \ldots [{}^{k_M}q_{mM}, {}^{k_M}cnt_{mM}]\}\} \\ & (k_1, k_2, \ldots, k_M \in [1, N]) \end{aligned} \tag{4.8}$$

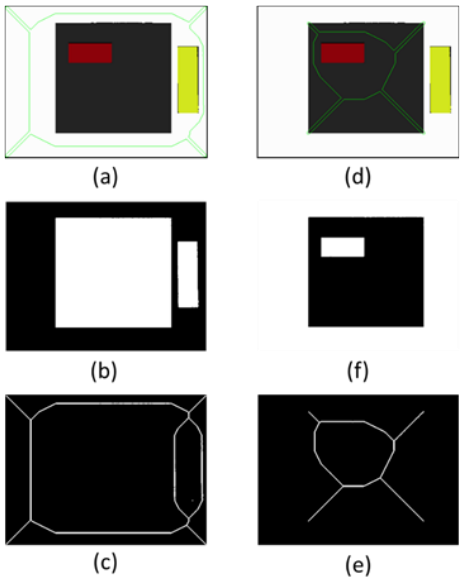


Figure 4.4. GVD-based contours for a specific keyframe.

4.2.4 Task Description and Checking by Fourier Descriptor and Shape Similarity Measurement

Fourier Descriptor is used to describe the GVD-based contours (Figure 4 (a) and (d)). Let *cnt* be one of the GVD-based contours in $GVD(X)$ which is described by U pixels in total. Let $(x_i, y_i)(i = 0,1,\ldots,U-1)$ be the coordinates of the pixels in the image frame. The pixels can be projected to the complex plane by

$$\{z\} = \{x + jy_i\}\left(i = 0,1,...,U-1\right) \tag{4.9}$$

The FD of this contour *cnt* is defined as the discrete Fourier transform for each pixel on the complex plane by

$$\begin{aligned} &\{Z\} = \{Z_0, Z_1, ..., Z_{U-1}\} \\ &Z_k = \sum_{n=0}^{U-1} z_n \cdot e^{-\frac{2\pi j}{U}kn} \left(k = 0,1,...,U-1\right) \end{aligned} \tag{4.10}$$

After the FD is normalized, the Hausdorff distance can be used to compute the similarity between two GVD-based contours. Let Z_{cnt1} and Z_{cnt2} be the normalized FD of two GVD-based contours, the Hausdorff distance between these two contours can be computed by

$$d_H(Z_{cnt1}, Z_{cnt2}) = \max\{\sup_{z_1 \in Z_{cnt1}} \inf_{z_2 \in Z_{cnt2}} d(z_1, z_2), \sup_{z_2 \in Z_{cnt2}} \inf_{z_1 \in Z_{cnt1}} d(z_1, z_2)\} \tag{4.11}$$

4.3 Experimental Results and Validation

In this section, we test and verify the functionality of our framework by comparing the results from different demonstration videos. First, a demonstration of the target task analyzed using the proposed framework. Then, the KFT and corresponding GVD-based contours are used to estimate multiple object-placing operations.

4.3.1 Results of Task Modeling and Learning based on Human Demonstration

The setup of the experiment including the camera, the objects, the corresponding configuration of workspace and the robotic system is illustrated in Figure 4.5. In the experiment, the target object-placing task is to first place the blue-triangle object onto the dark-rectangle object. The blue-rectangle object should finally locate at the top-right region

of the dark-rectangle object and point to the left. Then the yellow-rectangle object is placed onto the dark-rectangle object, who should locate at the right-bottom region of the dark-rectangle object and its long side should be parallel to the right-side edge of the dark-rectangle object. The task process is shown in Figure 4.6. The demonstration video of this task is generated in a simulation environment. The video contains 643 frames in total, and the original frames are with resolution 1920x1080 pixels. The frames are then cropped to 1440x1080 pixels and resized to 480x360 pixels before feeding to the proposed framework.

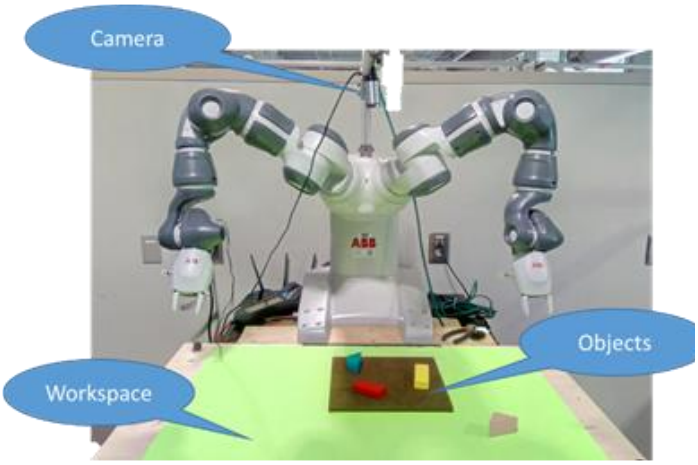


Figure 4.5. Experiment Setup.

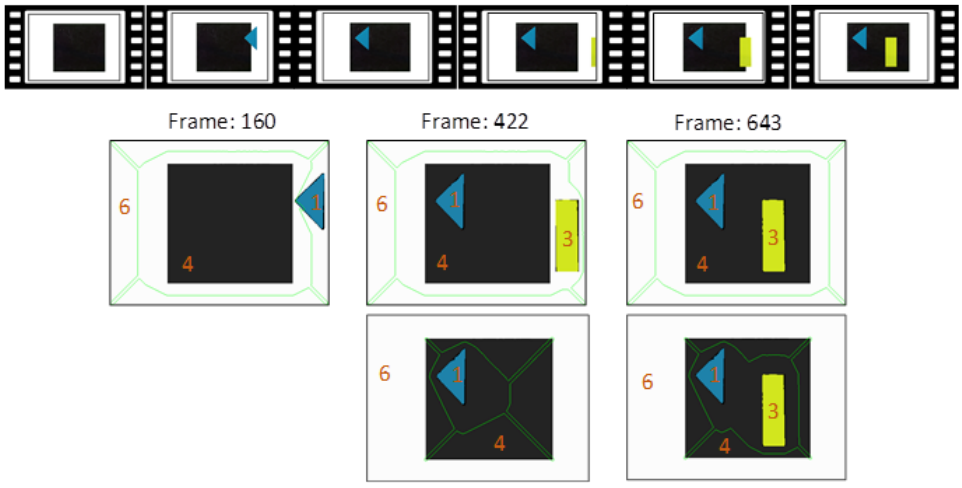


Figure 4.6. Modeling of the target object-placing task.

Figure 4.7 illustrates the process to get the GVD-based contours for a single keyframe. Figure 4.7 (a) is the original image whose background is removed; Figure 4.7 (b) is the visualization of the segmented image. From the segmented images, we can identify that the locations of the two rectangle objects with respect to the workspaces and the location of the triangle object with respect to the dark-rectangle object should be estimated by GVD-based contours. Then their corresponding binary occupancy grids, GVD, and GVD-based contours are generated separately. Figure 4.7 (c) ~ (e) illustrates the binary occupancy grids, GVD and GVD-based contours (in green color) of the two rectangle objects. Those of the triangle object with respect to the dark-rectangle object is shown in Figure 4.7 (f) ~ (h). After all the frames processed by the algorithm steps: segmentation and tracking, RSD, KFT, the keyframes of this task are the 160th, 422^{nd,} and 643rd frames. The overlapping and touching relations between the objects in these keyframes are list as follow:

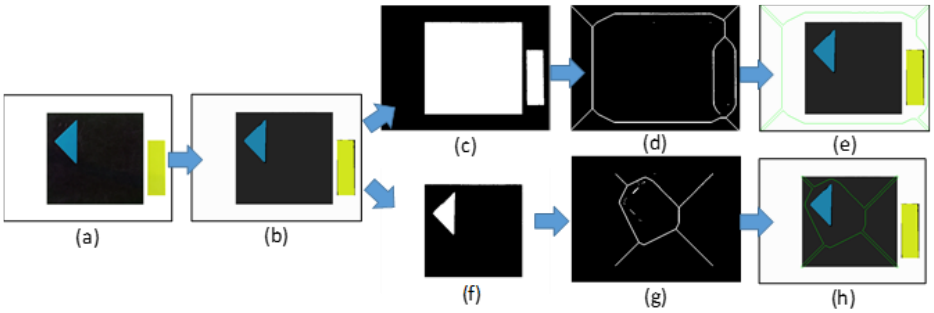


Figure 4.7. Image processing to get GVD-based contours.

$$KFT = \{'160':[{}^{160}D_{tch},{}^{160}D_{ovp}],'422':[{}^{160}D_{tch},{}^{160}D_{ovp}],'643':[{}^{160}D_{tch},{}^{160}D_{ovp}]\} \tag{4.12}$$

The corresponding overlapping dictionaries in KFT are

$$
\begin{aligned}
^{160}D_{ovp} &= \{\text{'[6, 1]': 186, '[6, 4]': 539}\} \\
^{422}D_{ovp} &= \{\text{'[6, 4]': 354, '[6, 3]': 208, '[4, 1]': 185}\} \\
^{643}D_{ovp} &= \{\text{'[4, 3]': 208, '[6, 4]': 210, '[4, 1]': 185}\}
\end{aligned} \tag{4.13}
$$

The (4.12) and (4.13) present the data structure of the KFT and the overlapping dictionaries in the KFT, which are illustrated in (4.7). The keys in KFT are the corresponding frame numbers. Therefore, the sequence of the object-placing task can be obtained by simply sorting the keys in KFT. Each key in KFT is related to two dictionaries, one is for overlapping relations, and the other is for touching relations. (4.13) shows the data structure of the dictionaries for overlapping relations. The keys in the dictionaries are the corresponding object-pairs, whose first element is the object at the bottom. The value of each key is the counter of the lines in the image scan process. The GVD-based contours are visualized in Figure 4.6 as green contours in the corresponding images of keyframes. Based on our test, when the correct object approaching the corresponding target location, the GVD-based contour of the current state will have higher similarity with the corresponding target frame. When the correct object is located at the correct location, the Hausdorff distance between the GVD-contour corresponding to the current workspace state and the target key frame goes to a minimal value, which is around 5~10. In this specific task, the threshold of the Hausdorff distance is set as 11.

4.3.2 Validations of Tasks Models by Checking Incorrect Tasks in Manufacturing

The task modeling approach is verified by checking incorrectly performed tasks with the learned task models. Two types of faults are applied as examples: incorrect object location and incorrect object sequence. Figure 4.8 illustrates the situation that the first part

is not placed at a correct location in the task. From the corresponding keyframes of the demonstration, we can see the overlapping relations are the same in the process of the task, but the GVD-based contours of corresponding frames are different. In this test, the Hausdorff distance between the contours in Figure 4.8 (a) and (c) is 32.846, the Hausdorff distance between the contours in Figure 4.8 (b) and (d) is 32.846. They are both larger than the threshold value in our test. Thus, the incorrect object position can be detected by the difference of GVD-based contours.

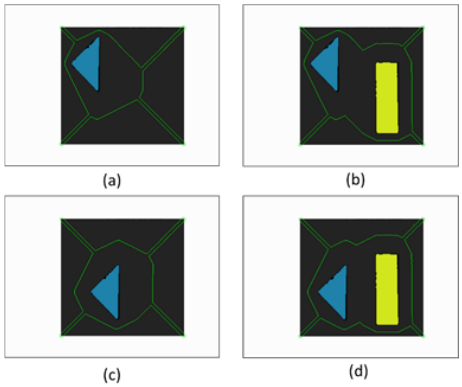


Figure 4.8. Fist part is not placed to correct position.

Similarly, Figure 4.9 illustrates the situation that the second part is not placed to the correct orientation. In this case, the general special of objects in the process of the task is still as same as the target demonstration. However, in the final state, the GVD-based contours in Figure 4.9 (a) and (b) are different, the Hausdorff distance between them is 52.438, which is larger than the threshold in our test.

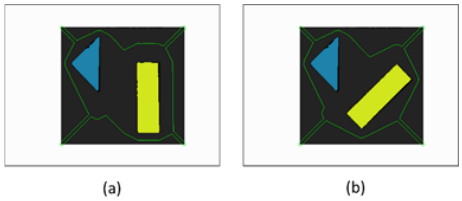


Figure 4.9. The second part is not placed in the correct orientation.

Figure 4.10 illustrates the situation that an undesired part is used as the second part. In this case, both the value of KFT and GVD-based contours show the difference of the current state and the target state. The overlapping relation in Figure 4.10 (a) is ${}^{643}D_{ovp}$ in Eq. (13), while the overlapping relation in Figure 4.10 (b) is

$$ {}^{654}\mathrm{D}_{ovp} = \{\text{'[4, 2]': 149, '[6, 4]': 233, '[4, 1]': 185}\} \tag{4.14} $$

The first object-pair in ${}^{643}D_{ovp}$ and ${}^{654}D_{ovp}$ is different. Therefore, the difference in the dictionary of overlapping relation can tell the undesired part is placed. Further, the size of the undesired object is different from the target object, so the corresponding GVD-based contours are different. In this case, the Hausdorff distance between the GVD-based contours in Figure 4.10 (a) and (b) is 30.406, which is larger than the threshold in our test.

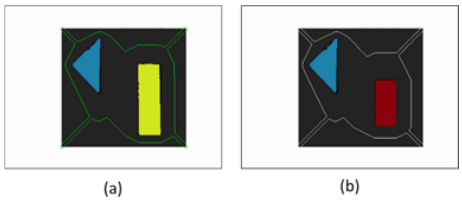


Figure 4.10. Undesired part is placed in the scene.

Figure 4.11 illustrates the situation that the object sequence in the task has an error with respect to the sequence of the target task. Though the final state of these two demonstrations is the same (Figure 4.11 (b) and (d)), the keyframes for the intermediate

state are different (Figure 4.11 (a) and (c)). The object overlapping relations of Figure 4.11 (a) is represented as ${}^{422}D_{ovp}$ in Eq. (13), while the object overlapping relation in Figure 4.11 (c) is

$$ {}^{349}D_{ovp} = \{\text{'[6, 1]': 186, '[4, 3]': 208, '[6, 4]': 331}\} \tag{4.15} $$

The object-pairs in ${}^{422}D_{ovp}$ and ${}^{349}D_{ovp}$ are different from each other. Moreover, the Hausdorff distance between the GVD-based contours in Figure 4.11 (a) and (c) is 30.406, which is larger than the threshold in our test.

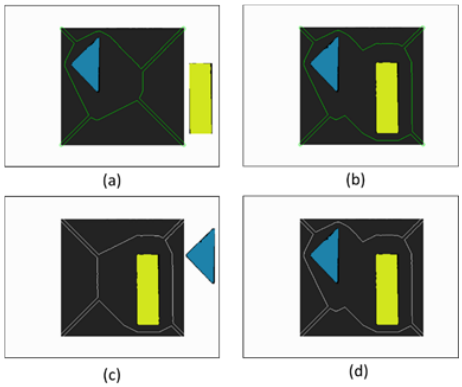


Figure 4.11. The incorrect sequence of object-placing actions.

The results above indicate that it possible to identify the correct task process and final states for object-placing tasks with the proposed framework. The key intermedia states and the final state of object-placing tasks can be abstract from the demonstration videos. The GVD-based contours are possible to be used for relative position description in object-placing tasks.

4.4 Conclusion

The objective of the proposed approach is to model and learn the object-placing task from human demonstration. The experimental results indicate that our framework can

abstract the knowledge of object placing tasks from a human demonstration video in a simulation environment. The object relation sequence can be described by the RSD, while the import intermedia states and final states of the placed object are possible to be represented by the KFT and GVD-based contours. Our approach does not need many pre-defined features or a large-scale dataset for the task modeling. The knowledge of the object-placing tasks is eventually presented by the small scale of data: RSD, KFT matrix, and corresponding FD of GVD-based contours in the frames. One potential future work is to apply the algorithm for more complicated on-line human fault detection in smart manufacturing. In addition, the proposed framework will also be used to guide robots to accomplish or assist humans to accomplish manufacturing tasks after learning from human demonstrations in future work.

CHAPTER 5

ROBOT LEARNING FROM DEMONSTRATION ON ASSEMBLY ASSISTANCE USING CNN

5.1 Introduction

Starting with the relatively simple object-placing tasks in assembly, I stepped further for robot learning of assembly assistance in collaborative assembly tasks. In current automotive assembly applications, professional robot programming and complex system setup process are required to implement most of the autonomous assembly process. The data-driven approaches are potentially eliminating the complexity of the assembly task modeling and system setup. Thus, I proposed a teaching-learning-collaboration (TLC) framework to enable the conventional industrial robot to learn assembly tasks and assist humans in the collaboration process. With the framework, humans can teach robots with simple joystick operations, while the data can be automatically labeled for training. The trained robot can assist humans actively in collaborative assembly tasks. The research was conducted with a custom-defined convolutional neural network with single RGB image input of the human-robot shared workspace. The approach also suggests a potential way by which the robot can be personalized by its users to assist them in their preferred ways in collaborative assembly applications.

An overview of the CNN-based teaching-learning-collaboration (TLC) framework is proposed in Section 5.2. The collaborative assembly tasks, which are modeled as a time

series are analyzed in Section 5.3. The robot learning of assembly tasks from human demonstrations using CNN is discussed in Section 5.4. Section 5.5 gives the experimental results and analysis. The chapter is summarized in Section 5.6.

5.2 An Overview of the CNN-based TLC framework

The existing approaches for human-robot collaborative tasks usually need a set of complex modeling and setup efforts [60], [61], and robots usually need to be programmed by a well-trained expert. This increases the cost of applying collaborative robots in human-robot assembly and also makes the collaborative robots very complicated and very inconvenient for end-users to use. To address this challenge, deep learning is introduced and adapted to the teaching-learning-collaboration (TLC) framework to enable robots to easily learn undefined tasks and workspace situations from human demonstrations and enable the trained robots to actively assist human operators in the human-robot collaborative assembly in real-time.

The system diagram of the TLC framework is shown in Figure 5.1. In the human-phase, the collaborative assembly task is allocated to both the human and the robot. In general, the robot should cooperate with the human operator to handle the lower-precision and higher-strength jobs, while the human operator focuses on the higher-precision and lower-strength assembly operations. To demonstrate the collaborative assembly task, the human operates the robot through intuitively human-robot interactions, such as leading-through, joystick operation when conducting the assembly maneuvers. Therefore, human operators are able to teach the robot to accomplish collaborative assembly tasks through natural demonstrations. In the robot-learning phase, the robot learns the expected behaviors

in the process of collaborative assembly tasks online based on the scene of the shared workspace captured by the version system and the robot operations taught by the human operator. Based on the task knowledge learned from human demonstrations, in the human-robot collaboration phase, the robot makes action decisions and generates proper assistant behaviors by given real-time images of the shared workspace. In this work, by modeling a collaborative assembly task as a time series, CNN is introduced to map the real-time vision of the shared workspace to proper robot assistant behaviors.

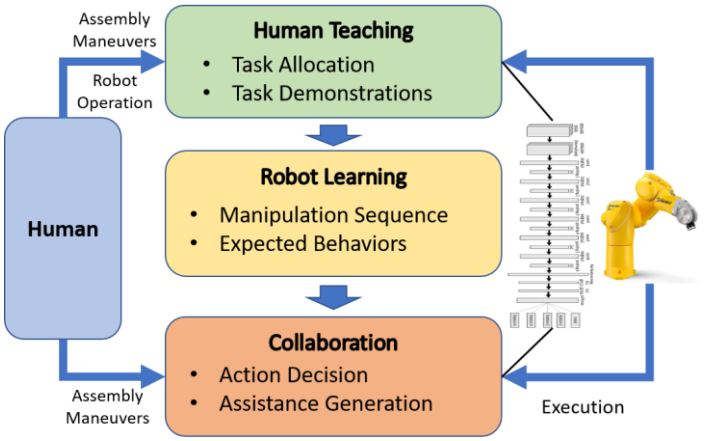


Figure 5.1. The system diagram of the TLC model.

5.3 Time Series Analysis of Collaborative Assembly Tasks

Human-robot collaborative assembly tasks are complex time series, which include massive strict constraints (e.g. force/torque, tolerance, etc.), plentiful flexible manipulations (e.g. personalized preference in gestures, tools, and sequence of maneuvers, etc.) and dynamic environments. These characteristics lead to challenges to pre-define and program every possible state that may happen in the collaborative assembly. However, any specific assembly process must follow a certain sequence of requirements to achieve a successful final assembly.

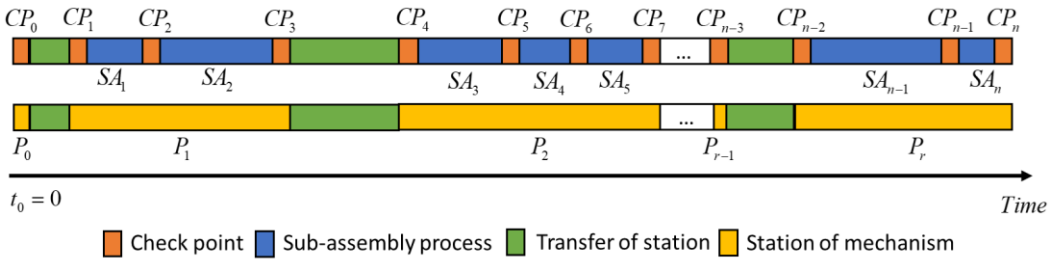


Figure 5.2. The time series analysis of collaborative assembly tasks.

Figure 5.2 illustrates the time series of a collaborative assembly task. From a mechanical perspective, *checkpoints* CP_i represent a series of discrete states, which are necessary and order-sensitive for the mechanism targeting for a successful final assembly. A *sub-assembly* SA_i is defined as a set of maneuvers, which make the state of the mechanism transfer from the current checkpoint to the next checkpoint. The sub-assembly processes often include plenty of flexible operations accomplished by human operators which are not order-sensitive and highly based on the humans' personal preferences. A *station* P_i represents a position and orientation of the semi-assembled machine that leads to the comfortable and convenient installation of new parts for human operators corresponding to the current sub-assembly section. In summary, an entire assembly task can be regarded as a time series consists of checkpoints, sub-assembly processes, hold and transfer of stations of the semi-assembled machine. The robot must generate proper behaviors to move from one station to another in real-time based on the state of the assembly task and the behavior of the human operator.

For an arbitrary human-robot collaborative assembly task, the configurations of stations are always discrete and finite though their values are unknown in advance. The

configurations of stations are mainly determined by the design of the mechanism and humans' personal performances. Therefore, the problem is formulated as two steps:

- Step 1: the robot learns the applicable station set $\{P\}$ for the collaborative assembly. In this process, the information of the tasks, such as the operations of the human operator and the states of semi-assembled mechanism, are represented through sequences of camera frames. Meanwhile, the applicable station set $\{P\}$ is abstracted from the position and speed feedback of the robot in human demonstrations.
- Step 2: the robot generates proper behavior to assist the human in the collaborative assembly. In this human-robot collaboration process, the robot should generate proper behavior based on the real-time images of the shared workspace. This is achieved by a trained CNN, which maps the situation of the shared workspace to a set of proper behaviors learned in the human demonstrations.

5.4 Learning from Demonstrations using CNN

5.4.1 Robot System Configuration

The robot system configuration for the human-teaching process is shown in Figure 5.3. The robot holds the semi-assembled mechanism in its gripper, the state of the workspace is captured by a camera. In the process of human demonstrations, the human operator uses a joystick to control the motion of the robot. After moving the robot to a station that is convenient for him/her to conduct the following sub-assembly maneuvers, the human operator accomplishes the desired sub-assembly maneuvers by selecting correct parts and assembling them to the semi-assembled mechanism with hand tools. For each

round of demonstration, we can obtain four datasets of time series data: a series of timestamped images, which include the information of mechanism status and human operations; the moments when the robot speed turns to zero, which indicate the human intends to hold the position of the end-effect; the moments when the robot starts to move, which indicate the human wants to move the robot to the next station; and the positions of the end-effector when the robot speed turns to zero, which indicate the set of stations selected by the human operator for the assembly task. Based on these four datasets, the images can be automatically and effectively labeled according to their timestamps.

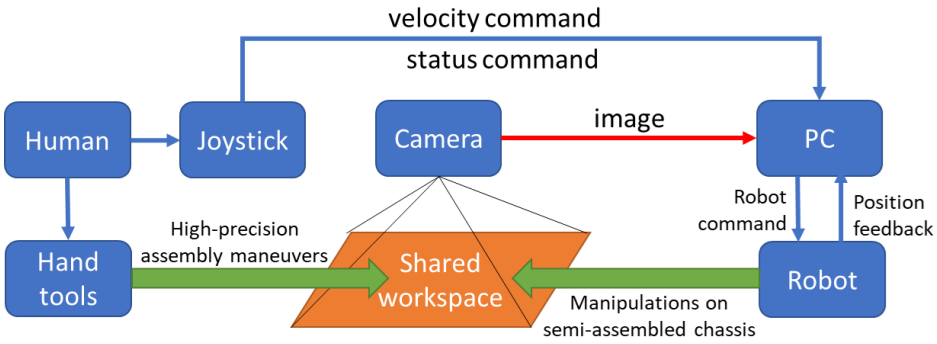


Figure 5.3. Robot system configuration for learning from demonstration.

5.4.2 Automatically Image Labeling

Unlike many deep learning cases, whose datasets are manually labeled, the image frames of the workspace in the collaborative assembly process are automatically labeled based on the timestamps. Figure 5.4 illustrates the timing sequence to label the image time series data sampled in the human demonstration. The algorithm of automatic image labeling is shown in Algorithm 5.1. When the robot is stopped at a station, the position of the robot end-effector is recorded through the robot feedback. When the human operator is conducting the assembly maneuvers, the real-time images are sampled and saved to the

computer, and all the images are timestamped. All these timestamped images are mapped to a robot behavior, which is the robot should stop at this specific station to wait until the sub-assembly is accomplished.

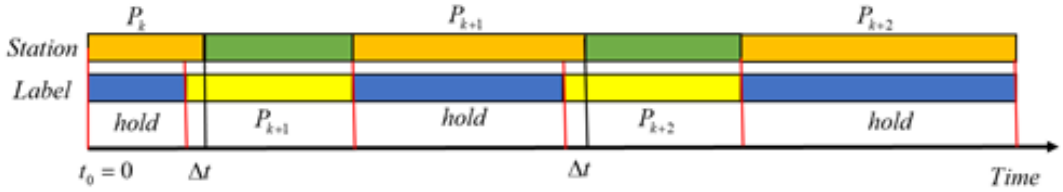


Figure 5.4. Automatic image labeling based on time series.

After the current sub-assembly process being finished, the human operator uses a joystick to move the robot to the next station. The real-time images of the process that the robot transfers from the current station to the next station are also captured by the camera and all the images are timestamped as well. When the robot arrived at the next proper station for the human operator to conduct the following assembly maneuvers, the robot is stopped by the human via the joystick. The images in this period should be mapped to a robot motion, which is the robot end-effector transfers from the previous station to the current station. The current station is also recorded via robot position feedback.

Since the human operator uses a joystick to operate the robot and uses hand tools to conduct the assembly maneuvers, there is a time interval Δt when the human operator switches his/her hand between the joystick and hand tools. In this time interval, the last sub-assembly has been finished and the state of the assembly has already arrived at the checkpoint, therefore, the images in this period should be mapped to the next robot motion. In our experiment, we found that the time interval Δt is generally kept consistent in serval rounds of human demonstrations and is affected by the level of proficiency of the human

operator. By selecting a proper value of the time interval, the error rate of image labeling can be controlled from 1% to 2%, which is normally acceptable for CNN training.

Algorithm 5.1. Data Acquisition and Automatic Image Labeling.

Algorithm: Data Acquisition and Automatic Image Labeling

Initialization
Initialize the list of "robot moving moment"
Initialize the list of "robot stopping moment"
Initialize the list of "robot stopping pose"
Initialize the robot position and velocity
Initialize the folder for temporary image storage

Human demonstration
While the current round of demonstration is not finished, **do**
Save the timestamped image of workspace
Read current robot speed
Read current robot position

If the robot is stopping, **then**
Append the current robot position to "robot stopping pose"
Append the current timestamp to "robot stopping moment"

If the robot is starting to move, **then**
Append the current timestamp to "robot moving moment"

If the current round of demonstration is finished, **then**
Break the while loop

Automatic image labeling
Set the proper time interval Δt
For image in temporary image storage **do**
Get the timestamp of the image t_i
Find the nearest timestamp t_s in the list of "robot stopping moment"
Find the nearest timestamp t_m in the list of "robot moving moment"
If t_i is earlier than t_s and t_i is later than $t_m - \Delta t$, **then**
Get the corresponding robot stopping pose P at t_s
Label the image as "Moving to P"

If t_i is later than t_s and t_i is earlier than $t_m - \Delta t$, **then**
Label the image as "Stop at current position"

5.4.3 Structure of CNN

The structure of CNN and system diagram we used in this work is shown in Figure 5.5. The detail of the structure of CNN is given in Table 5.1. The input size is notated by image width, image height, and number of channels. The filter shape of convolutional layers is noted by filter height, filter width, filter height, number of channels, and number of filters. The image of the workspace is captured by a webcam and the sampling frequency is 2Hz. This sampling frequency is selected based on the normal operating speed of the human operator, which can obtain enough data to present the assembly process and avoid too many repetitive images. The original RGB images obtained by the camera are first cropped and resized to 800 x 300 x 3. Then the image is pixel-wise normalized in each channel by

$$\sigma_x = \sqrt{\frac{1}{H \times W} \sum_{i=1}^{H \times W} \left(x_i - \mu_x \right)^2} \tag{5.1}$$

$$\hat{x}_i = \frac{x_i - \mu_x}{\sigma_x} \tag{5.2}$$

$$\mu_x = \frac{1}{H \times W} \sum_{i=1}^{H \times W} x_i \tag{5.3}$$

where H is the height of the image, W is the width of the image, and $x_i \in [0, 255]$ is the pixel value at a specific position in one channel of the RGB image. The parameters of the CNN applied in this work are illustrated in Table 5.1. It includes six convolution-pool sections after the image normalization. The number of filters is variant corresponding to the convolutional layers. The filter size and the stride of the max pool are set as 2 x 2 and 1. After the convolution-pool sections, the output is flattened as an array with 4160

elements. Then, there are two fully connected layers, which have 512 and 256 neurons respectively. A Softmax classifier is used to calculate the loss function and map the probability to each robot's behavior. The probability of each potential robot behavior can be written as

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \left(\forall i \in 1,2,...,K\right) \tag{5.4}$$

where K is the total number of robot behaviors to predict in the collaborative assembly task, which is learned from the human demonstrations. The sparse cross-entropy loss is applied for the measurement of classification measurement. The Adam optimizer is implemented for the training of the CNN. The initial learning rate is set as 0.002 with an exponential decay rate of 0.98. Considering the limitation of the memory on our workstation, the batch size of training, validation, and testing are set as 100 images. After every 500 iterations, the updated CNN is validated throughout the overall validation dataset. If the validation result is better than the previous validation, then the current parameters of the CNN are saved to a file. The maximum training epochs are set as 100 rounds. Once the training is finished, the CNN parameters with the minimum validation error are selected for robot assistance generation.

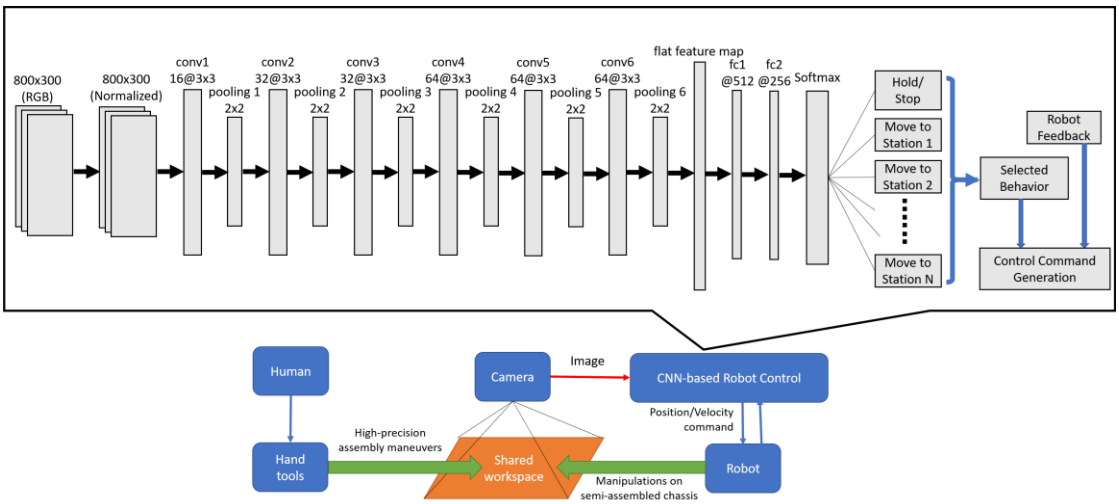


Figure 5.5. The structure of CNN and robot configuration for CNN implementation.

Table 5.1. Structure of CNN

Type	Stride	Input Size	Filter Shape
Conv	1	800 x 300 x 3	3 x 3 x 3 x 16
Max Pool	2	800 x 300 x 3	Pool 2 x 2
Conv	1	400 x 150 x3	3 x 3 x 3 x 32
Max Pool	2	400 x 150 x3	Pool 2 x 2
Conv	1	200 x 75 x3	3 x 3 x 3 x 32
Max Pool	2	200 x 75 x3	Pool 2 x 2
Conv	1	100 x 38 x 3	3 x 3 x 3 x 64
Max Pool	2	100 x 38 x 3	Pool 2 x 2
Conv	1	50 x 19 x 3	3 x 3 x 3 x 64
Max Pool	2	50 x 19 x 3	Pool 2 x 2
Conv	1	25 x 10 x 3	3 x 3 x 3 x 64
Max Pool	2	25 x 10 x 3	Pool 2 x 2
Flat	N/A	13 x 5 x 64	N/A
FC	1	1x1x4160	4160 x 512
FC	1	1 x 1 x 512	512 x 256
Softmax	1	1 x 1 x 256	Classifier

* Conv is the convolutional layer
* FC is the fully connected layer
* Filter shape is noted by width, height, channel, and number of filters

5.4.4 CNN-based Robot Assistance Generation

The system configuration of the robot system for CNN implementation is illustrated in Figure 5.5. In this case, the human does not use the joystick to control the robot behavior. The diagram of the robot control logic is illustrated in Figure 5.6. The scenario of the shared workspace is sampled at a frequency of 10Hz, which is a normal frequency used in real-time control. The same pre-process including cropping, resizing and normalization is conducted for the image as the human demonstration before feeding to the CNN. The trained CNN can generate the probability of each potential robot behaviors, which are learned from human demonstration. The robot behavior with the highest probability is selected to generate the robot assistant manipulation.

If the selected behavior is to move the robot end-effector to a pose $P = \{x, y, z, \alpha, \beta, \gamma\}$, the desired pose with assigned robot speed will be sent to the low-level controller. The low-level controller compares the received desired position with the real-time robot pose feedback. If the current robot pose is different from the desired robot pose, then a joint motion trajectory is generated based on the inverse kinematics of the robot in the low-level controller. The robot executes the joint trajectory to move to the desired robot pose with the assigned robot speed. If the selected robot behavior is to stop and hold the robot at the current pose, a zero robot velocity command will send to the low-level robot controller, which makes the robot stop immediately.

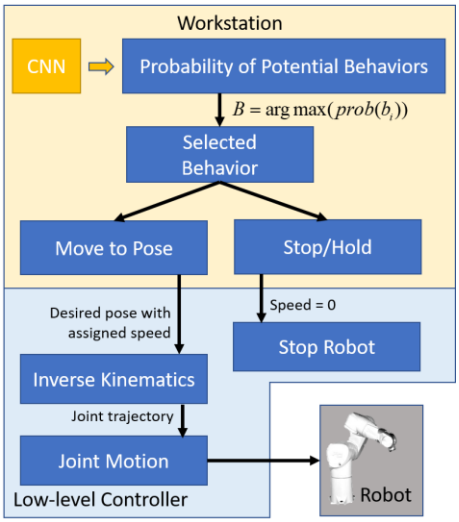


Figure 5.6. Robot control diagram for CNN implementation.

5.5 Experimental Results and Analysis

5.5.1 Experimental Setup

The real robot system setup for the experiment is shown in Figure 5.7 (a). The experiment is conducted based on a Staubli TX40 industrial robot. A web camera is used to record the state in the human-robot shared workspace of the collaborative assembly. The system integration for robot control, joystick operation, computer version are based on Robot Operating System (ROS). The CNN is built, trained and deployed with TensorFlow. The low-level motion planning and robot speed control is implemented by the joint motion function of Staubli TX40 controller.

The vehicle model (Figure 5.7 (c)) is disassembled as four wheels, the front bumper, the rear bumper, the semi-assembled chassis and the corresponding screws and washers for each component (Figure 5.7 (b)). In the human demonstrations, human uses the joystick to operate the robot to move the semi-assembled chassis to a proper location, which is comfortable and convenient for him/her to conduct the following sub-assembly

maneuvers. The image of the workspace and the position of the robot are recorded for the CNN training. In the process of the CNN implementation, the real-time images are acquired by the webcam at the same location and the robot behavior is triggered automatically based on the image input of the CNN, which enables the robot to move to a position that is comfortable and convenient for the human worker to accomplish the assembly maneuvers.

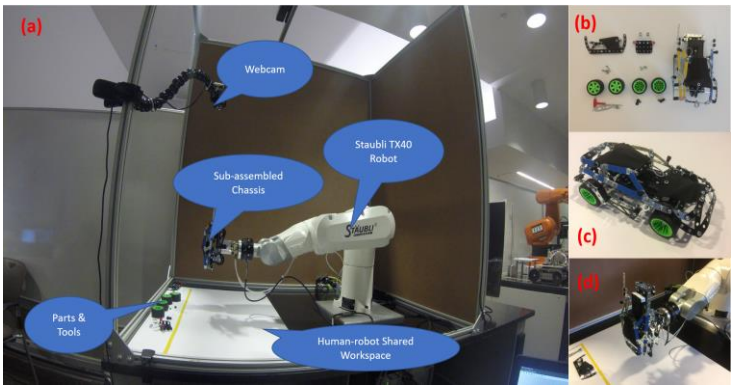


Figure 5.7. Robot setup for human-robot collaborative assembly.

5.5.2 CNN Validation and Test Results

In the learning efficiency perspective, the robot should accomplish the learning process in a few demonstrations of specific tasks for the collaborative assembly applications. In our experiment, we have created four datasets (D1- D4) from various human demonstrations of accomplishing the assembly tasks. Based on these datasets we have conducted the training, validation, and testing with two different configurations.

Table 5.2. Dataset of Each Demonstration Process

Class	**D1**	**D2**	**D3**	**D4**
Images of -80°	38	39	34	39
Images of 0°	38	39	37	39
Images of 90°	39	37	39	38
Images of -178°	39	41	38	37
Images of Hold	510	514	486	462
Total	664	670	634	615

Table 5.3. Training, Validation, and Testing Results

Training Dataset	Validation Dataset	Test Dataset	Validation (%)	Test (%)
D1	D3	D4	96.21	98.34
D1 & D2	D3	D4	98.26	98.49

Firstly, we used only the dataset D1 to train and the dataset D3 to determine the parameters of the neural networks. The fourth demonstration was used as a test dataset for the trained neural networks (Figure 5.8). The training process was totally run for 100 epochs, meanwhile, the entire images of the 3rd demonstration as the validation set were fed to the neural networks for every 500 iterations. The parameters of the neural networks were saved whenever we get a better result in the validation. The best validation result is 96.21% correct prediction which was achieved in 54000 iterations. The trained neural networks with the parameter corresponding to the minimum validation error get an average prediction accuracy as 98.34% with the test dataset.

Secondly, we used the dataset D1 and D2 to train the neural networks so that the training dataset increased to 1334 images in total. The dataset D3 and D4 demonstrations were still used for validation and testing (Figure 5.9). The best validation result is 98.26% correct prediction which was achieved in 120000 iterations. The trained neural networks with the parameter corresponding to the minimum validation error get an average prediction accuracy as 98.49% with the test dataset.

According to the results of the two different training-validation-testing configurations, the prediction accuracies on the validation dataset are both higher than 95%. Through the CNN trained by the former configuration has a lower validation accuracy, the average test accuracies of both configurations are similar to each other, which

are 98.34% and 98.49% respectively. In our experiment, both CNNs successfully assisted the human operator in accomplishing the model vehicle assembly task.

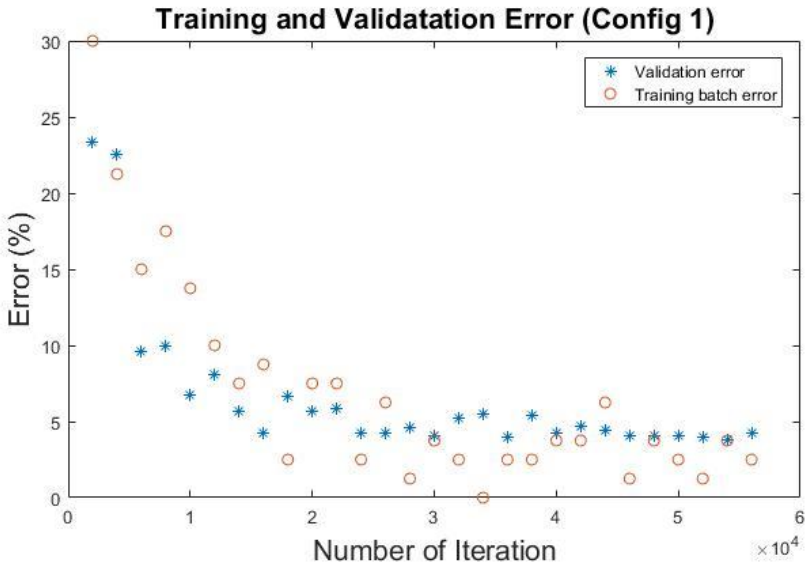


Figure 5.8. The training process of one-demonstration training configuration.

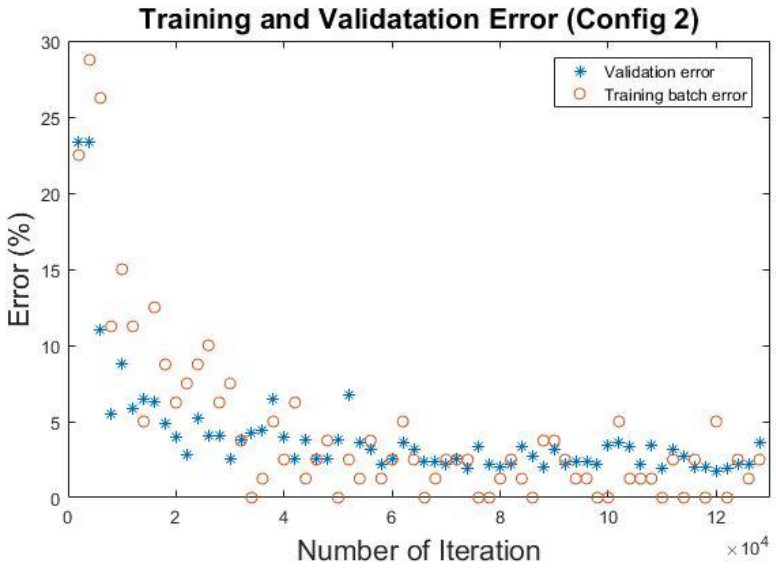


Figure 5.9. The training process of two-demonstration training configuratioin.

The human-robot collaboration in the process of the two wheels and front bumper assembly is shown in Figure 5.10. The robot can hold or turn the proper station to make the human operator to install the parts easily. The scenario of an intelligent emergency stop function of the robot is shown in Figure 5.11. Once the features of human hands were detected, the control commands to stop the robot were generated by the neural networks. The robot stopped immediately before collision when the human operator's hand suddenly approached the moving chassis.

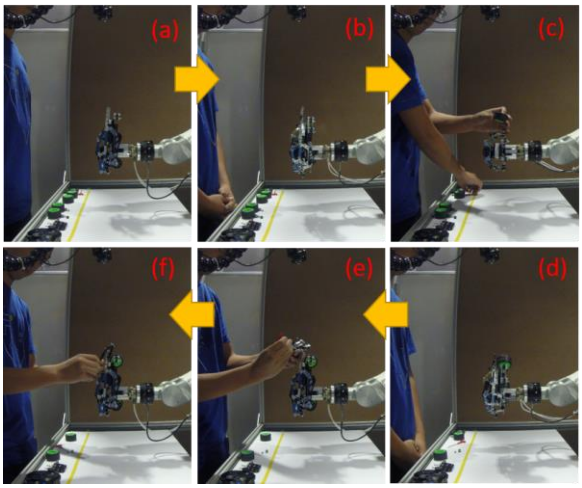


Figure 5.10. The robot supportive behaviors for two wheels and front bumper assembly.

These experimental results demonstrated the trained CNN can generate the proper supportive behaviors automatically in the human-robot collaboration to help the human operator in the assembly of the model vehicle. In the process of human demonstrations, when the human hands are working on assembly maneuvers in the shared workspace, the robot is always stopping and holding on a specific station. The feature of human hands is successfully abstracted by the CNN, which enables the robot to stop immediately in many

other states besides the learned stations when the human hands are approaching the moving robot arm.

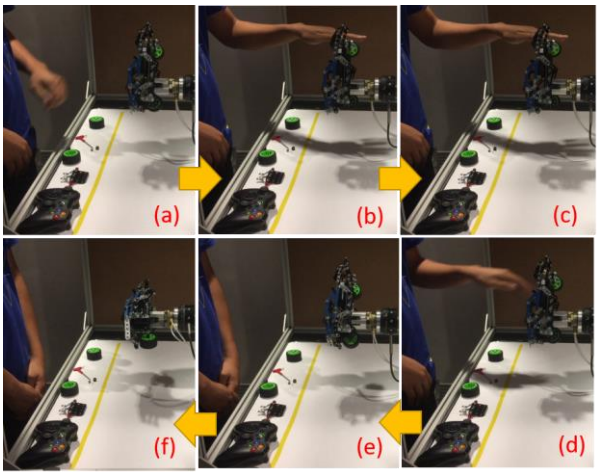


Figure 5.11. The robot stops immediately when human hands approaching.

5.6 Conclusion

In this chapter, a CNN-based approach is proposed to learn and assist humans in assembly tasks from human demonstrations. Experimental results show that CNN is effective in robot learning during collaborative assembly and the robot can be trained to actively assist humans in the human-robot collaborative assembly process in real-time. The datasets for training, validation, and testing can be created and labeled online from human demonstrations. The approach can help alleviate the need for complex modeling and setup compared to the existing approaches. Our approach also suggests a potential way by which the robot can be personalized by its users to assist them in their preferred ways in collaborative assembly applications.

CHAPTER 6

ROBOT LEARNING FROM DEMONSTRATION ON ASSEMBLY TASKS USING TC-IRL

6.1 Introduction

Existing robot learning approaches mainly focus on making robots repeat the tasks that humans have demonstrated and lack scalability to new tasks. Recently, some studies have applied inverse reinforcement learning (IRL) on task learning from human demonstrations, which use reward functions to capture human working patterns [62]–[65]. However, due to the large state and action space of IRL, it usually requires a large amount of training data and computational efforts. To address the above challenges of existing approaches, this section proposes a new teaching-learning-collaboration (TLC) framework to make collaborative robots learn the tasks from human teaching demonstrations and then assist humans to collaboratively accomplish the tasks including new tasks with larger geometric scales instead of repeating the learned tasks. The TLC model enables collaborative robots to learn from human demonstrations using a new task constraint-guided inverse reinforcement learning (TC-IRL) approach. Compared to conventional IRL, it can significantly reduce the state and action space and computational efforts, and therefore lead to less training data requirement and better real-time performance. Furthermore, a robot assistance generation approach with task extension is then proposed to generate assistive robot actions to collaborate with humans to accomplish not only the

demonstrated tasks but also new tasks with larger geometric scales. The proposed approaches potentially allow humans to teach the robot by just a few small-scale demonstrations and then the robot can assist humans to accomplish a series of larger-scale tasks in the human-robot collaboration process.

An overview of the TLC framework using TC-IRL is presented in Section 6.2. The representation of collaborative assembly tasks is discussed in Section 6.3. The robot learning of collaborative assembly tasks using TC-IRL is proposed in Section 6.4. The generation of the robot assistance which is guided by TC-IRL is presented in 6.5. The experimental results and analysis is discussed in Section 6.6. Finally, the chapter is summarized in Section 6.7.

6.2 An Overview of TLC Framework Using TC-IRL

In the conventional inverse reinforcement learning (IRL) approach [62], a collaborative assembly task can be modeled by a Markov decision process (MDP), which can be described by a tuple as

$$M = (S, A, T, \gamma, R) \tag{6.1}$$

where S represents the state space, A represents the action space, $T = P(s' \mid s, a)$ is the state transition probability, $\gamma \in [0,1)$ is the discount factor, and R is the reward function. In this IRL formulation, task constraints are connotative in the definitions of task states. Generally, all the potential states and actions that satisfy the task constraints must be defined in the model. For collaborative assembly applications, the actions and states in the MDP depend on parts, tools, assembly locations, sequences, etc. Therefore, the size of the action and state space will increase dramatically when the options of parts, tools, and task

scales are slightly increased. As known, the size of the state and action space is proportional to the number of unknown parameters that need to be learned through IRL, and the number of unknown parameters indicates the required amount of data for training of IRL. In addition, the size of the state and action space is also proportional to the computational costs of implementing IRL. Therefore, when applying conventional IRL to learning assembly tasks, due to the wide variety of the parts, tools, and assembly task variations, it will require a very large state and action space and therefore require a large amount of data for training and significant computational efforts for implementation. This makes it difficult to be applied to realistic assembly scenarios. To address this issue, we propose the TLC model based on the TC-IRL approach. An overview of the TLC model is illustrated in Figure 6.1.

The goal of the TLC model is to enable the robot to learn the task constraints and human preference from human demonstrations, to assist the human to accomplish assembly tasks collaboratively by delivering proper parts and tools to proper human hands at the correct moment. In the human-teaching phase, a human demonstrates and allocates the collaborative tasks. In the robot-learning perspective, the robot learns the task constraints and human working styles from human demonstrations. In the human-robot collaboration phase, the robot generates supportive behaviors based on updated task parameters and the results of reward calculation and policy optimization.

In the proposed approach, object-based constraints, location-based constraints, and human hand-based constraints are considered in the task constraint perspective. The object-based constraints refer to the constraints between object pairs, for example, a certain part

must be assembled with a specific tool. The location-based constraints refer to the constraints that a certain part must be assembled to a specific location of the final product. Moreover, studies [66] indicated that about 10% of people are left-handedness. The human hand capabilities are considered as the human hand-based task constraints in the robot handover process. These types of constraints are all learned through the human demonstration in the proposed approaches. In addition to task constraints, the process of assembly also varies for different humans based on their working styles. According to the learned task constraints, we proposed TC-IRL to model the assembly process by a task constraint-guided Markov decision process (MDP) and extract the human preference in the assembly process via learning.

In human-robot collaboration, the task can be new scalable tasks and the robot first uses natural language to guide the human worker to respond to the required task parameters for the new scalable task. Based on the learned task knowledge in human demonstrations and the new task parameters from the human response, the robot extends the learned knowledge from human demonstrations to the new task and then generate appropriate actions for both arms to appropriately assist the human during the new assembly process.

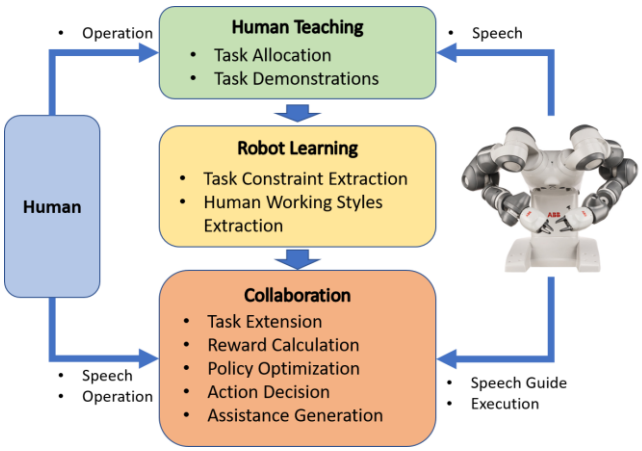


Figure 6.1. The framework of TLC model using TC-IRL.

6.3 Collaborative Assembly Task Representation

In the TLC model, the human worker and the robot work collaboratively in a shared workspace. The human worker uses his/her hands to assemble correct parts to correct locations with correct tools in sequence to formulate a final product in the teaching process. Then, the goal is to make the robot use both left and right arms hand over the correct parts and correct tools to correct human hands based on the knowledge learned from human demonstrations. To formulate the human-robot collaborative assembly task, we introduce the definitions of task resources.

First, the set of tools can be defined as

$$Ts = \{ts_1, ts_2, ..., ts_{N_{ts}}\} \tag{6.2}$$

where N_{ts} is the total number of tools. The set of parts to be assembled is defined as

$$Ar = \{ar_1, ar_2, ..., ar_{N_{ar}}\} \tag{6.3}$$

where N_{pt} is the total number of parts. The parts are distinguished from each other by a set of attributes, such as shape, color, mass, etc. The set of the attributes that are used to describe different parts is defined as

$$V_{ar_i} = \{{}^{ar_i}v_1, {}^{ar_i}v_2, ..., {}^{ar_i}v_{N_{ar_i}}\} \tag{6.4}$$

where $ar_i \in Ar$ is an attribute, N_{ar_i} is the total number of values corresponding to the attribute ar_i. The set of assembly locations can be written as

$$Lc = \{lc_1, lc_2, ..., lc_{N_{lc}}\} \tag{6.5}$$

where N_{lc} is the total number of assembly locations. The number of locations in the human-robot collaboration phase can be variant and different from that in the human teaching phase. The former location set depends on how the human worker wants to extend the task scale, while the later location set is generally the minimal scale of human demonstrations, which are enough to teach all the task constraints and human preferences. In this research, we present the derivation of the proposed approaches in two-dimensional assembly scenarios.

6.4 Learning Collaborative Tasks via TC-IRL

In this section, we present the detail of the task-constraint-guided inverse reinforcement learning (TC-IRL). A general form of the MDP in the TC-IRL can be written as

$$M_t = (S_t, A_t, T_t, \gamma, R_t) \tag{6.6}$$

where S_t is the task constraint-guided state space, A_t is the task constraint-guided action space, $T_t = P(s_t' \mid s_t, a_t)$ is the state transition probability, $\gamma \in [0,1)$ is the discount factor, and R_t is the task constraint-guided reward function.

In TC-IRL, the robot first learns the task constraints from human demonstrations, and the learned task constraints are then used to construct task constraint-guided state and action space which is much smaller than the original space. The task constraint-guided reward is then defined based on this constrained space and learning is conducted to learn the unknown parameters for assembly tasks from human demonstrations. In the following, we will first introduce the learning of task constraints including object-based task constraints, location-based task constraints, and human hand-based task constraints which are used to limit the size of the state and action space and then introduce the learning algorithm to learn how the human accomplishes the task.

6.4.1 Learning of Task Constraints

6.4.1.1 Object-based Task Constraints

The object-based constraints refer to the constraints between object pairs, for example, a certain part must be assembled with a specific tool. Based on the definitions in the previous section, the object-based constraints can be represented by a matrix

$$C_{obj} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N_{pt}} \\ \vdots & \vdots & \cdots & \vdots \\ c_{N_{ts},1} & c_{N_{ts},2} & \cdots & c_{N_{ts},N_{pt}} \end{bmatrix}_{N_{ts} \times N_{pt}} \tag{6.7}$$

where the i^{th} row of C_{obj} corresponds to the tool ts_i in the set Ts, while the j^{th} column of C_{obj} corresponds to the part pt_j in the set Pt. To indicate the object-based constraints, the element $c_{i,j} = 1$ if the part pt_j should be assembled with the tool ts_i, otherwise, $c_{i,j} = 0$

To learn the object-based constraints, the states of parts/tools and human hands are tracked in the demonstration process. The combined state at any given time can be represented as

$$s_h(t) = [obj_L(t), obj_R(t)]^T \tag{6.8}$$

where $obj_L(t)$ and $obj_R(t)$ indicate the object in the human's left and right hand at the moment $T = t$. Each hand can be empty and can also with either a part or a tool in hand. Based on human operations in the demonstrations, we developed a statistic-based approach to learn the object-based constraints. For the k^{th} demonstration, let ${}^d L_k$ be the overall length of the state sequence in the demonstration, A_{1k} be the times of appearances of $s_h = [ts_i, pt_j]^T$, A_{2k} be the times of appearances of $s_h = [pt_j, ts_i]^T$, and N_D be the total number of demonstrations. The probability that the part pt_j should be assembled with tool ts_i can be written as

$$P(ts_i, pt_j) = \frac{1}{N_D} \sum_{k=1}^{N_d} \frac{A_{1k} + A_{2k}}{L_i} \tag{6.9}$$

This probability is then used to update each element in C_{obj} where for each part pt_j in the set Pt, the tool with the highest probability is considered the object-based constraint, and the corresponding element is set as 1.

6.4.1.2 Location-based Tasks Constraints

The location-based constraints refer to the constraints that a certain part must be assembled to a specific location of the final product. Each location-based constraint reveals which part is correct for a specific assembly location. The location-based constraint can be written in matrix format as

$$C_{loc}=\begin{bmatrix} o_{1,1} & o_{1,2} & \cdots & o_{1,W} \\ \vdots & \vdots & \cdots & \vdots \\ o_{H,1} & o_{H,2} & \cdots & o_{H,W} \end{bmatrix}_{H\times W} \tag{6.10}$$

where H and W are the height and width for the demonstrated tasks and $o_{i,j}$ is a vector, which indicates the values of attributes of the object assembled at the corresponding assembly location.

For each assembly location in the task, we consider all the parts that have been installed at this assembly location throughout multiple human demonstrations. The probability distribution for a specific attribute ar_i at an assembly location (x, y) can be calculated by

$$P(ar_i = v_j \mid X = x, Y = y) = \frac{count(ar_i = v_j)}{N_{\mathrm{D}}} \tag{6.11}$$

where $count(\bullet)$ function means to count the times of appearance of the given condition. N_D means the total number of rounds of human demonstrations. As we mentioned in the previous section, a two-dimensional assembly scenario is used for the derivation.

Let p_{he} be the probability of human demonstration errors. For the part at assembly location (x, y), the attribute ar_i should have the value v_j as the constraint if it satisfies

$$P(ar_i = v_j \mid X = x, Y = y) > 1 - p_{he} \tag{6.12}$$

This probability is then used to update the element in C_{loc} where the attributes whose probabilities satisfy (12) are all considered as constraints in $o_{i,j}$.

6.4.1.3 Human-Hand-based Task Constraints

In dual-hand assembly operation scenarios, human workers may have different hand preferences to accomplish assembly operations. To enable the robot to deliver parts and corresponding tools to the proper hand of the human worker, we proposed a statistic approach to learn the human hand preference. The human hand-based task constraint with respect to part-tool pairs can be represented by a matrix

$$C_{hand} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,N_{pt}} \\ \vdots & \vdots & \cdots & \vdots \\ h_{N_{ts},1} & h_{N_{ts},2} & \cdots & h_{N_{ts},N_{pt}} \end{bmatrix}_{N_{ts} \times N_{pt}} \tag{6.13}$$

The i^{th} row corresponds to the tool ts_i in the set Ts, while the j^{th} column corresponds to the part pt_j in the set Pt. The element $h_{i,j} = 1$ if the part pt_j should be delivered to the right hand, while the tool should be delivered to the left hand. The element $h_{i,j} = -1$ represents the opposite hand preference. For those part-tool pairs, which are not satisfied with the object-based constraints or have never been appeared in any of human demonstrations, the corresponding element is set as $h_{i,j} = 0$.

With the definitions of variables in (6.9), for each specific part-tool pair (ts_i, pt_j), the probability of different hand over methods among all the human demonstrations is computed by

$$P(L = ts_i, R = pt_j) = \frac{1}{N_D}\sum_{k=1}^{N_d}\frac{A_{1k}}{L_i}$$
$$P(L = pt_j, R = ts_i) = \frac{1}{N_D}\sum_{k=1}^{N_d}\frac{A_{2k}}{L_i} \quad (6.14)$$

The higher probability in (6.14) is regarded as the human hand preference on this specific part-tool pair (ts_i, pt_j). The values, -1 or 1, are signed to the corresponding elements in C_{hand} based on the probability.

6.4.2 IRL Learning of Assembly Tasks with Task Constraints

In order to learn how human accomplishes the assembly task from his/her demonstration, we first need to construct the state and action space in (6.6). The learned object-based task constraints will help limit the size of the state space S_t because only the constrained part-tool pairs $c_{i,j}$ learned in C_{obj} are considered in the state definition. The learned location-based task constraint will further limit the state space S_t because only parts whose attributes satisfy the constraint vector $o_{i,j}$ in C_{loc} are considered in the state definition for this location. The human hand-based task constraint will help limit the action space A_t because for a specific part or tool, which hand to use in the action will be specified in the constraint C_{hand}. With constrained state space definition and constrained action space definition, the size of the entire state and action space (i.e., different action options at different states) will then be significantly reduced.

The learning of assembly tasks is not only to learn the final assembly state but also to learn the process of how the human conduct the assembly during demonstrations. This process can be captured by a set of feature functions defined by

$$f = [f_1, f_2, ..., f_k, ...]^T \tag{6.15}$$

where each f_k is defined in the state and action space $f_k(s) \in \{0,1\}$ to specify a special feature of the human assembly process. Each different value of the vector f is corresponding to a human working style in the assembly, such as assembly the part row by row, left to right, from far to near with respect to his/her body position, etc.

The overall human assembly process can be then reflected by a task constraint-guided reward function which is defined as a weighted sum of the feature functions

$$R_t(s) = W^T f(s,a) = \sum_k w_k f_k(s,a) \tag{6.16}$$

where $W = [w_1, w_2, ...]^T$ is a set of weights to determine the preferences of the human on different features during assembly. The weights together will determine how the human would like to accomplish the assembly tasks.

In this research, we propose to apply the maximum entropy inverse reinforcement learning (MaxEnt-IRL) [67] to learn the weights in the reward function from human demonstrations. We assume that the MDP is deterministic in this work. Therefore, according to the MaxEnt-IRL principle, the distribution over assembly strategy under deterministic transitions can be defined as

$$P(\zeta \mid M_t, W) = \frac{1}{Z(W)} \exp\left(\sum_k {w_k}^T f_k(s,a) \right) \tag{6.17}$$

where ζ is the assembly strategy in human demonstrations, $Z(W)$ is the partition function. The weights of features in the reward function can then be optimized by maximizing the entropy through

$$W^* = \arg\max_W \log P(s \mid M, W)$$
$$= \arg\max_W \left(\sum_k {w_k}^T f_k(s) - \log Z(W) \right) \tag{6.18}$$

6.5 TC-IRL Guided Robot Assistance

In this section, we will introduce how to generate assistive robot actions to collaborate with humans to accomplish not only the demonstrated tasks but also new tasks with larger geometric scales. The robot will ask the human through natural language about the dimensions of the new scalable tasks and the human will respond through natural language before the collaboration starts.

6.5.1 Extension of TC-IRL

When the collaborative assembly task is extended to a larger geometric scale, the object-based task constraints and the human hand-based task constraints should usually remain the same. However, the location-based task constraints for the extended task must be updated to fit the extended tasks. In the two-dimensional assembly scenarios, each assembly location in the human-demonstrated tasks is regarded as a center of a cluster and then each assembly location in human-demonstrated tasks can be mapped to a new center of a cluster in the extend tasks throughout a linear scaling transformation, which can be written as

$$x' = \frac{W_{ex}}{W_d} x, \quad y' = \frac{W_{ex}}{W_d} y \tag{6.19}$$

where W_{ex} and H_{ex} are the width and height of the enlarged assembly process. W_d and H_d are the width and height of the demonstrated assembly. The location-based task constraint

at each assembly location (x, y) in human-demonstrated tasks formats a center of location-based constraint cluster at (x', y') in the enlarged assembly task. Afterward, the constraint of each assembly location in the extended assembly task is determined by the k-nearest neighbor (KNN) classifier, which can be written as

$$c_{ex}(x_{ex}, y_{ex}) = C^{KNN}_{H_d \times W_d}(x', y') \tag{6.20}$$

where $x_{ex} \in [1, W_{ex}]$ and $y_{ex} \in [1, H_{ex}]$ gives a specific assembly location in the extended assembly task, c_{ex} is the location-based task constraint corresponding to the given assembly location. The right side of the equation means selecting the same location-based task constraint corresponding to the center of the closest cluster among all the nearby clusters which are centered at different (x', y'). The minimal Euler distance is used as the criterion for the KNN to select the closest cluster in the enlarged assembly task.

When the collaborative assembly task is extended to a larger scale, the MDP model in TC-IRL must also be updated to fit the extended tasks. After knowing the dimensions (height and the width) of the extended task from the human, the task constraints will be first updated for the new task. Based on the new task constraints, the task constraint-guided states, actions, and rewards can be defined in the same way as the originally demonstrated task in the previous sections. to construct a new task constraint-guided MDP and therefore result in a new TC-IRL. The new MDP and TC-IRL will also retain the advantage of small size for state and action space because the learned task constraint has been fully extended to the new scalable tasks to guide the definition and state and action space.

Because the human does not change for the new scalable task, his or her preference on how to accomplish the task giving the state and action space should retain the same. Therefore, the feature functions will remain the same as the originally TC-IRL, and more importantly, the optimized weights W^* which are previously learned from human demonstrations can also be used for the new task. This means that we do not need to re-train the TC-IRL at all although the tasks have been extended to a larger scale. We only need to use the learned reward function to update the reward map for the new tasks with newly updated state and action space and then use the assistance action generation approach which is introduced in the following section to generate appropriate robot assistance for the human.

6.5.2 Robot Assistance Generation

After the reward map is updated based on the reward function for the new task, the value function of the extended MDP can be determined through value iterations:

$$V_{i+1}(s) = \max_a \left(\sum_{s'} R(s) + \gamma V_i(s') \right) \quad s.t. \quad s \xrightarrow{a} s' \tag{6.21}$$

where s' is the next state of the system after the action a is executed at the state s. The converged value function with respect to the state s is noted as $V(s)$. Since we aim to make the robot assist human to accomplish the assembly task, we will, therefore, require the human to initialize the task. The human first needs to accomplish two assembly actions based on his/her preference to establish an initial condition for the robot. Then, in order to generate appropriate assistance, the robot will first recognize the current state of the task

via its sensing system in real-time and then determines which action to choose from the action space of the extended MDP model.

$$a^* = \arg\max_{a} \left(R(s) + \gamma V(s') \right) \quad s.t. \quad s \xrightarrow{a} s' \tag{6.22}$$

The optimized action a^* can also infers the assembly location that the human should be working on, the robot can then search all the available parts that can be assembled to this assembly location based on the location-based constraints. To determine which robot arm should be used to pick which available part, the Euclidean distances from the work home positions of both robot arms to each available part are calculated. The arm and the part corresponding to the minimal Euclidean distance are paired. According to the selected part, the robot then generates the list of all the available tools based on the object-based constraints. Similarly, the arm and the tool corresponding to the minimal Euclidean distance are paired. Based on the selected part-tool pair, the delivery targets are determined with the human hand preference. Afterward, the robot will execute the pick and handover actions to use different arms to pick up the correct part and tool and deliver them to the correct hands of the human to assist him/her to accomplish the assembly task.

6.6 Experimental Results and Analysis

6.6.1 Experimental Setup

The proposed approaches are verified and evaluated on a multi-model human-robot collaborative assembly test platform. The hardware setup of the test platform is illustrated in Figure 6.2. The ABB Yumi is a dual-arm collaborative robot. The human stands face to face with the robot to work in a shared workspace. A six-camera VICON motion capture

system is set up on the roof surrounding the workspace to capture the motion of human hands. The Kinect RGB-D sensor offers a top-view point cloud of the workspace for part recognition and tracking. Therefore, we can identify which part is operated by which human hand in the realistic operations in both human demonstrations and human-robot collaborations. The software of the test platform is developed based on robot operating system (ROS) and visualized through Rviz [68]. The trajectory-level motion planning from point to point is accomplished based on the open motion planning library (OMPL) [69] via MoveIt! motion planning framework [70].

In our experiments, the configuration of the workspace is illustrated in Figure 6.3. The shared workspace consists of the part/tool stack zone and the assembly zone. The part/tool stack zone is close to the robot, and the assembly zone is near to the human in the shared workspace. Initially, all the parts and tools are sorted in the stack zone and the assembly zone is empty. The 15 tools have three different types: square (S), hex (H) and crisscross (C), and all of them are in white color (W). The 18 parts have three different shapes: square (S), hex (H) and crisscross (C), and three different colors: red (R), yellow (Y) and blue (B). Mechanically, a part can be assembled with a tool if and only if they have the same shape. In the following sections, we use "color/shape" to present a part or a tool for convenience, for example, "B/S" means the blue square part, and "W/H" means the white hex tool.

In the human demonstration process, the human manipulates the parts and fasteners in the workspace by his/her both hands directly. The human operations are tracked by hand motion capture. Since the object-based constraints and the human hand preference are

learned via statistic-based approaches, the sample size is critical for the human teaching and robot learning phase. The human starts the demonstration according to the natural language introduction of the robot. After the human finishes all the object manipulations through naturally pick-assembly-place operations, the human should put both hands at the work home position to indicate the robot that the demonstration is accomplished.

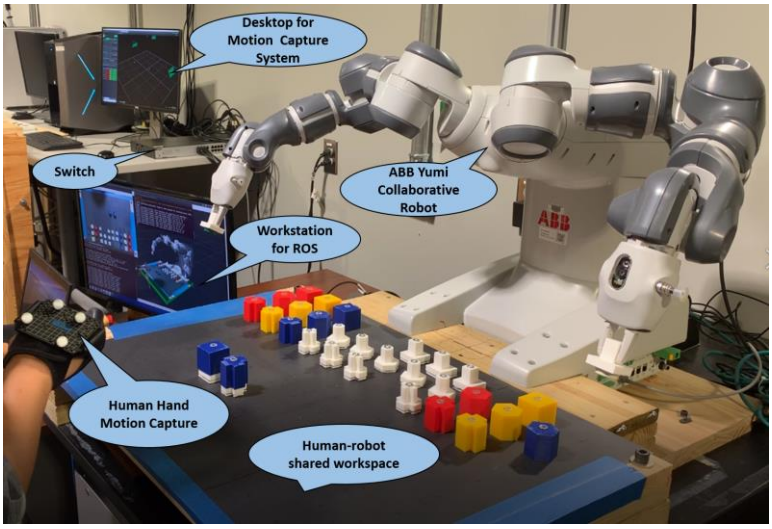


Figure 6.2. The hardware setup of the test platform.

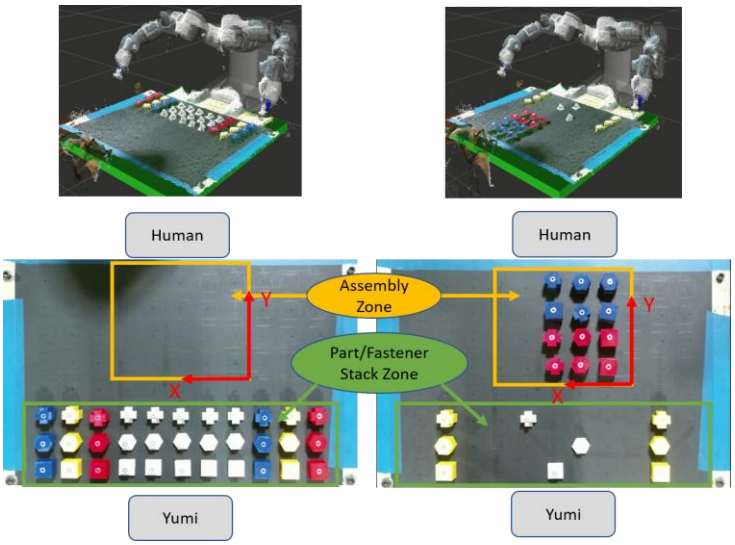


Figure 6.3. The workspace configuration and collaborative assembly task.

6.6.2 Results of TC-IRL in Collaborative Assembly

In the experiment, the proposed approaches are verified on a designed assembly task that the upper half of the assembly locations are expected to be red parts but no specific constraint in shape; the lower half of the assembly locations are expected to be blue parts but no constraint on the shape. Three rounds of 2 x 2 human demonstrations are given. In the demonstrations, the parts in red color but with different shapes are installed to the assembly locations in the upper half plane, while the parts in blue color but with different shapes are installed to the assembly location in the lower half plane. Meanwhile, human always use the left hand for tool operations and manipulate the parts with the right hand. All the three demonstrations are accomplished raw by raw, right to left, and from near to far with respect to the human's body position.

The result of a collaborative assembly case for an extended 4 x 3 assembly task based on the human demonstrations of the previous section is shown in Table 6.1. In the human-robot collaboration phase, the human first picked up the blue/crisscross with his right hand and a corresponding white/cross tool with his left hand and placed the assembled part at location 11 (3, 4). Then, the human picked up a blue/hex part with his right hand and a corresponding white/hex tool with his left hand and placed the assembled part at location 10 (2, 4). After this, the task state was successfully initialized by the human, and the robot started to assist the human in the following steps of the task. The column of process prediction in Table 6.1 gives the assembly locations predicted by the robot at different task states. The columns of robot assistance illustrate the robot's decisions on

using which robot arm to pick up which part or tool and delivering to which human hand based on the real-time task state and the learned task constraints.

Table 6.1. Results of Robot Assistance in 4 x 3 Assembly.

State	Process Prediction	Robot Assistance		
		Arm	Part/Tool	Hand
[10, "000000000011"]	9 (1, 4)	L	Part: B/H	R
		L	Tool: W/H	L
[9, "000000000111"]	8 (3, 3)	L	Part: B/S	R
		R	Tool: W/S	L
[8, "000000001111"]	7 (2, 3)	R	Part: B/C	R
		L	Tool: W/C	L
[7, "000000011111"]	6 (1, 3)	R	Part: B/S	R
		L	Tool: W/S	L
[6, "000000111111"]	5 (3, 2)	R	Part: R/C	R
		R	Tool: W/C	L
[5, "000001111111"]	4 (2, 2)	R	Part: R/H	R
		R	Tool: H	L
[4, "000011111111"]	3 (1, 2)	R	Part: R/S	R
		L	Tool: W/S	L
[3, "000111111111"]	2 (3, 1)	L	Part: R/C	R
		R	Tool: W/C	L
[2, "001111111111"]	1 (2, 1)	L	Part: R/H	R
		L	Tool: W/H	L
[1, "011111111111"]	0 (1, 1)	L	Part: R/S	R
		R	Tool: W/S	L
[0, "111111111111"]	-1 (end)	Stop	N/A	N/A

6.6.3 Quantitative Evaluations

To evaluate the proposed model, we tested the model with different assembly processes, human hand preferences, and extended tasks with larger dimensions in assembly with 9 different kinds of parts and 3 different kinds of tools. Each participant is first introduced about how to demonstrate the object-based constraint, location-based constraint, human hand-based constraint, and assembly process by giving three human demonstrations with 2 x 2 dimensions based on his/her preferences. After the robot learning

phase, the 2 x 5 and 4 x 3 collaborative assembly tasks are accomplished with robot assistance. The results in Table 6.2 show that with correctly calibrated the motion capture system, the robot tool center points, and the location of the objects, the proposed approach can obtain 100% accuracy in assembly sequence prediction, pick-delivery actions.

Based on the same workspace configuration and the same collaborative assembly task, the comparison on the action space size, the state space size, the transition map size, and computation effort are shown in Table 6.3. For the TC-IRL, the corresponding results are automatically generated online by setting the proper parameters of the task size. For conventional IRL, we assume that the robot can use either left or right arm to pick up a part or a tool then deliver to either left or right human hand in each manipulation. The sizes of state and action spaces are then calculated for conventional IRL respectively.

We can see that with TC-IRL, the size of the action space, state space, and state transition matrix of the MDP process in the model are significantly reduced compared with the model without the constraint extractions. Based on our task configuration, the size of the state and action space for TC-IRL is significantly smaller than conventional IRL and the advantages become more obvious when the task dimension increase, which leads to a dramatic increase of the state and action space for IRL. Based on the principle of IRL, the reduced size also implies reduced requirement on the training data, which is why our proposed approach only requires several human demonstrations. At the same time, because of the reduced size, the computational cost is also significantly reduced, which leads to better real-time performance.

Table 6.2. Statistic Results of Robot Assistance.

Assembly Process	Hand Preference	No. of Prediction (2x5)	No. of Pick (2x5)	No. of Handover (2x5)	No. of Prediction (4x3)	No. of Pick (4x3)	No. of Handover (4x3)	Accuracy
1	L	8/8	16/16	16/16	10/10	20/20	20/20	100%
	R	8/8	16/16	16/16	10/10	20/20	20/20	100%
2	L	8/8	16/16	16/16	10/10	20/20	20/20	100%
	R	8/8	16/16	16/16	10/10	20/20	20/20	100%
3	L	8/8	16/16	16/16	10/10	20/20	20/20	100%
	R	8/8	16/16	16/16	10/10	20/20	20/20	100%
4	L	8/8	16/16	16/16	10/10	20/20	20/20	100%
	R	8/8	16/16	16/16	10/10	20/20	20/20	100%

* The elements from the 2nd column to the 7th column are represented by "number of correct actions / numbers of total actions"

Table 6.3. TC-IRL vs Conventional IRL.

Task Size	Size of Action Set		Size of State Set		Size of Transition Map		Computation Effort	
	TC-IRL	IRL	TC-IRL	IRL	TC-IRL	IRL	TC-IRL	IRL
2 x 2	4	432	28	1.2754e7	3136	7.0277e16	0.00076 s	>10 min
2 x 3	6	648	186	2.7894e11	207576	5.0420e25	0.02174 s	>10 min
2 x 4	8	864	1016	1.1387e16	8258048	1.1204e35	0.4286 s	>10 min
2 x 5	10	1080	5110	7.4713e20	261121000	6.0287e44	11.10 s	>10 min
4 x 3	12	1296	24564	7.1895e25	7240681152	6.6989e54	292.2 s	>10 min

* The 2 x 2 tasks are used in human demonstrations on the real robot
* The 2 x 5 and 4 x 3 tasks are tested on the real robot
* The computation effort includes the time cost of the extended MDP generation and training of IRL

6.6.4 Subjective Evaluations

To evaluate the general acceptability and suitability of the proposed approaches, the robot is trained by a knowledgeable user via 2x2 demonstrations to adapt to two hand preferences (lefthanded and righthanded) and eight process preferences shown in Table 6.4. We asked nine non-expert subjects to accomplish both 3x3 and 2x5 collaborative assembly tasks with the robot. Based on the training set, the parts in the first row of the 3x3 assembly must be in red color, and the second and the third row must be in blue color. For

the 2x5 assembly, the first row of the final assembly are red parts, and the second row are blue parts. Before starting the task, we spend 3-5 minutes to let them watch the attached video and give them a brief introduction simultaneously about how we train the robot, how the robot will assist them, and what kind of final assembly they are expected to accomplish with the robot. In the two assembly tasks, the subjects can choose different process preferences for different assembly tasks, and accomplish the tasks based on their own hand preference. Once both collaborative assembly tasks are completed, the subjects are asked to finish a questionnaire, which contains 9 questions (Q1 ~ Q9) to assess his/her personal feeling about the human robot collaboration process.

Table 6.4. Human Working Process Preferences for Subjective Evaluation

Preference	Description
1	Row by row, left to right, far to near with respect to the body position.
2	Row by row, left to right, near to far with respect to the body position.
3	Row by row, right to left, far to near with respect to the body position.
4	Row by row, right to left, near to far with respect to the body position.
5	Column by column, left to right, far to near with respect to the body position.
6	Column by column, left to right, near to far with respect to the body position.
7	Column by column, right to left, far to near with respect to the body position.
8	Column by column, right to left, near to far with respect to the body position.

Table 6.5. Participants' hand preferences and selected process preferences

Subject	Hand Preference	Process Preference	
		3x3 Assembly	2x5 Assembly
1	Righthanded	6	1
2	Lefthanded	8	5
3	Righthanded	5	1
4	Righthanded	4	1
5	Righthanded	1	5
6	Lefthanded	2	5
7	Righthanded	3	1
8	Lefthanded	2	1
9	Righthanded	4	1

Table 6.6. Items in the questionnaire for the subjective evaluation

Item	Description
Q1	What do you think of the fluency [71] to naturally collaborate with the robot by this approach?
Q2	What do you think the robot response speed [72] in human-robot collaboration by this approach?
Q3	What do you think the safety [73] in the human robot collaboration by this approach?
Q4	What do you think the sociability [74] in human-robot collaboration by this approach?
Q5	What do you think the task efficiency improvement [75] by this approach in human-robot collaboration?
Q6	What do you think the hand preference matching in the collaboration process by this approach?
Q7	What do you think the process preference matching in the collaboration process by this approach?
Q8	What do you think the constraints matching in the collaboration process by this approach?
Q9	What do you think the overall comfort [76] in the collaboration process by this approach?

The participants' hand preference and the selected process preferences for the assembly tasks are shown in Table 6.5. There are three lefthanded participants and six righthanded participants in the subjective evaluation experiment. Different assembly processes are employed in the collaborations based on their own preferences. The items in the questionnaire are shown in Table 6.6. The evaluation indicators normally employed by previous studies on human-robot interaction [71]–[76]. Additionally, Question 6 to Question 8 is specific designed as the performance indicators for the proposed TC-IRL approach. In this research, the participants use the Likert scale [77] to rate their feeling. The Likert scale is divided into five levels, excellent, very good, good, fair, and poor, by nine points. Since we have nine participants, the full score for each evaluation indicator is 45. The scores of each evaluation indicator by different participants are shown in Figure 6.4. The results of the total score, the average score, and the standard. deviation (SD) of each performance indicator is shown in Table 6.7. The collaboration fluency and the robot response speed are between very good to good, which indicates the trajectory-level motion

planning can be further improved, such as enable the robot to handover both part and tool with both arms simultaneously. The results show that all the participants give full score for the safety of the collaboration due to the predictable handover manipulations generated by the collaborative robot. The assessments of the hand preference matching, process preference matching, and task constraint matching are either excellent or very good, which proves the effectiveness of the proposed TC-IRL approach.

Table 6.7. The Total, Average, Standard Deviation of Scores of Evaluation Indicators

Indicator	Total	Average	SD
Collaboration fluency	39.5	4.39	0.57
Robot response speed	33.5	3.72	0.85
Collaboration safety	45	5.00	0.00
Collaboration sociability	40.5	4.49	0.33
Task efficiency improvement	39	4.33	0.33
Hand preference matching	43	4.78	0.34
Process preference matching	43	4.78	0.42
Task constraint matching	42.5	4.72	0.34
Overall comfort	41.5	4.61	0.31

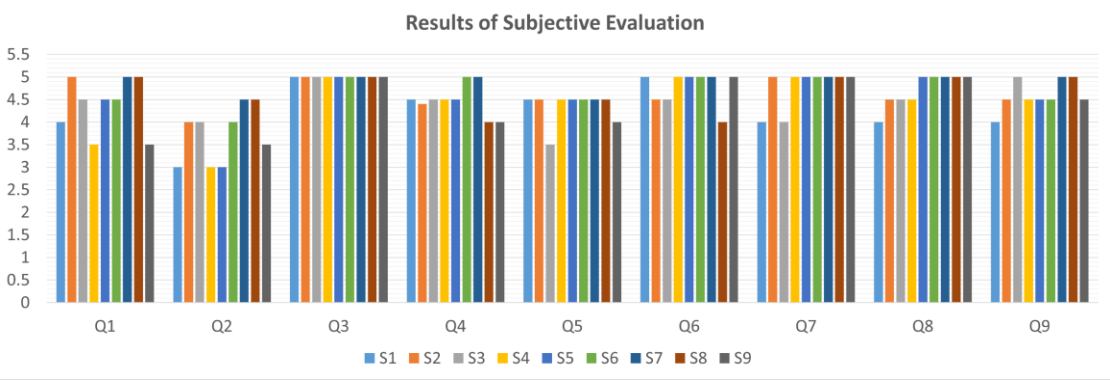


Figure 6.4. Scores of each subjective evaluation indicator.

6.7 Conclusion

In this chapter, we proposed a new learn-to-collaboration approach with the TC-IRL method that generates robot assistance to assist humans in human-robot collaborative

assembly. The TC-IRL approach can significantly reduce the size of the action and state space and lead to a reduced requirement of training data and computational cost compared to traditional IRL. The proposed approach can also allow humans to teach the robot to accomplish new larger-scale tasks by learning from several small-scale demonstrations. The experiment results demonstrated the effectiveness and advantages of the proposed approach

CHAPTER 7

ROBOT LEARNING OF ASSEMBLY TASKS FROM NON-EXPERT HUMAN DEMONSTRATIONS

7.1 Introduction

Most of the existing approaches, as well as the research we have conducted in previous chapters, usually assume that the demonstrations are performed by human experts who can conduct the task in an efficient way in order to achieve efficient robot executions through learning in RLfD, e.g., in assembly tasks, the demonstrations must be conducted by an expert worker in the assembly domain.

There are several recent works [78]–[80] which use reinforcement learning to learn from imperfect human demonstrations. However, such works mainly aim to learn the lower-level control policies which are very different from higher-level assembly tasks. Also, these approaches usually require a significantly large number of demonstrations and usually cannot always guarantee the best solution compared to model-based approaches because of their data-driven heuristic nature.

In this chapter, we aim to advance the robot learning from demonstration by reducing the requirement of demonstrations with human experts. Our major motivation is to make robots learn just like humans who can usually learn tasks from others from different perspectives and then synthesize the best way to accomplish the task although the others may not always demonstrate the tasks in efficient ways. Therefore, we propose a new

FOON-based approach to address robot learning from non-expert demonstrations in the robotic assembly contexts. We have previously introduced FOON as a graphical knowledge representation of human cooking tasks [81], [82]. In this research, we extend FOON to learning assembly tasks from non-expert demonstrations. We reconstruct some features of Functional Object-Oriented Network (FOON) to make it suitable for assembly tasks and also develop automatic subgraph creation and merging algorithms for FOON construction from multiple non-expert assembly demonstrations. Furthermore, we also propose an assembly task tree retrieving algorithm with the robot execution optimization process to enable robots to learn and generate the best possible task execution based on the constructed FOON. Because our approach employs models, it requires just a few demonstrations, which is significantly fewer compared to existing data-driven approaches.

The weighted FOON for assembly tasks is introduced in Section 7.2. The approaches to construct FOON from non-expert demonstrations is presented in Section 7.3. The algorithms for the assembly tasks retrieval and optimization based on FOON are introduced in Section 7.4. The experimental results and analysis is discussed in Section 7.5. Finally, the chapter is summarized in Section 7.6.

7.2 Weighted FOON for Assembly Tasks

7.2.1 Structure of FOON

The FOON proposed in our paper is a graphical task representation that includes robot motions, physical interactions between robots and objects in the workspace, and overall assembly task state descriptions. It provides a more intuitive way to represent, analyze, and visualize human demonstrations. More importantly, FOON decouples objects

and motion from a holistic view of action. This decoupling allows FOON nodes to have a more granular representation and gives FOON more flexibility than traditional task-step representations. The flexibility created by the motion nodes and object nodes enables more integrated task-tree merging and can generate more optimal task trees.

The constructed FOON for assembly tasks is a bipartite network that contains motion nodes and object state nodes. To make it suitable for assembly task representations, as opposed to the original FOON for cooking tasks, a specific type of object node, the so-called assembly state node, is introduced into the FOON to keep track of the assembly states in human demonstrations. Mathematically, an assembly state node can be either an input node or an output node of a motion node, and each motion node can have at most one assembly state node in its input nodes and at most one assembly node in its output nodes. FOON would only allow the object state nodes and assembly state nodes to be connected to motion nodes, and the motion nodes to be connected to object state nodes and assembly state nodes, which form a bipartite network.

7.2.1.1 Nodes

The nodes in FOON for assembly tasks have three types: object state O, motion M, and assembly state A. An object state node N_O represents a state of an object, which includes the object's identifier, name, and attributes. The attributes of an object include but are not limited to, its position, mass, size, color, and so on. For assembly tasks, a single part in a specific state can be defined as an object state node. When the state of the part is changed, a new object state node is generated. When more than one part is assembled as a component, a new object state node is also generated for this component. An assembly

node N_A has the same mathematical properties as an object node. It does not represent the state of a part or component, but rather it represents the state variations of the final assembly product. A motion N_M represents a manipulation related to specific objects. The information in a motion node includes, but is not limited to, the type of action, manipulated objects, start position, goal position, and so on. In a FOON for assembly, each object node and assembly node are unique in their name and attributes. Motion nodes with exactly the same attributes can appear at different locations in the FOON graph.

7.2.1.2 Edges

FOON for assembly is a directed graph. An edge can be drawn from an object state node or an assembly state node to a motion node, or vice-versa. In general, the object state nodes with edges directed to a motion node are regarded as task constraints of the manipulation in the motion node. The object state nodes that have edges that come from a motion node are regarded as the manipulation outcomes. These are analogous to input and output nodes coined in [81].

If there is an assembly node that has an edge directed to a motion node, this assembly node would indicate the state of the final assembly product before the manipulation corresponding to the motion node. If a motion node has an edge directed to an assembly node, this assembly node would indicate the state of the final assembly product after the manipulation corresponding to the motion node. Some motion nodes may not lead to a change in the final assembly state so that they do not have an assembly node as input or output.

7.2.1.3 Functional Units

A functional unit in a FOON for assembly tasks consists of a motion node and multiple object state nodes. Some functional units also contain one assembly state node as input and/or output. As shown in Figure 7.1, the object state nodes and the assembly node with edges directed to the motion node are the input nodes of the functional unit, and the object state nodes and the assembly state node with edges pointing from the motion node are the output of the functional unit. In assembly tasks, the input assembly state node represents the state of the final assembly product before the manipulation of the motion node. Together, the object state nodes form the task constraints of the manipulation. A functional unit is defined as a minimum learning unit in a FOON for assembly.

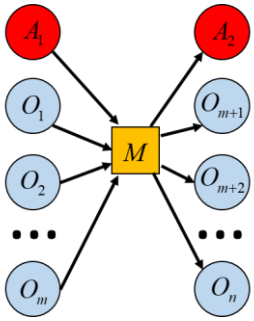


Figure 7.1. A functional unit with *m* object state nodes and one assembly state node as input, (*n*-*m*) object state nodes and one assembly state node as output.

7.2.2 Integrating Weights into FOON

The weights of FOON reflect both the success rate W_{SR} and average efficiency W_e of a given manipulation. The success rate is mainly related to the capabilities of robots, which corresponds to the accuracy required for executing manipulations. For example, robots have payload limitations for their arms and object size limitations for their grippers. It is nearly impossible for robots to handle manipulations beyond those limitations. Low-

accuracy manipulations, such as handover, usually have a relatively higher success rate. On the contrary, the high-accuracy manipulations, such as placing a bolt into a hole, usually have a relatively lower success rate. In terms of average efficiency, it mainly relates to safety and the complexity of manipulations. For instance, robots can move at a higher speed when simply picking up a single part from stock and handing it over to humans. On the contrary, robots may have to run at a slower speed when moving a component, which contains some loose parts on it, from one location to another. Moreover, some manipulations, such as switching the location of two parts, may contain multiple motion steps and require two robot arms to cooperate with each other, which means relatively higher complexities and lower efficiency. The weight of manipulation for the corresponding functional unit can be computed by:

$$W_{FU} = W_{SR} \times W_e \tag{7.1}$$

where W_{SR} is the success rate of the manipulation and W_e is the weight of efficiency for the manipulation.

The representative success rates of manipulations can be determined empirically. The average efficiency of a specific type of manipulation can be identified from the average time cost of those in human demonstrations via hand tracking and object tracking with the optical tracking system. However, these are not trivial tasks to perform. To simplify this, we assign estimated weights based on our experiences and results of waypoint teaching and trajectory execution time. Manipulations that cannot be executed by a robot were assigned a success rate W_{SR} of 0.01, while other motions would be assigned higher values which varies between 0.8 to 0.95. In addition, single-arm-single-part manipulations were

assigned a higher average efficiency weight which varies between 0.9 to 0.95, single-arm-multi-parts manipulations (moving a component) were assigned a medium average efficiency weight which varies between 0.75 to 0.85, and dual-arm-single-component manipulations, which require both robot arms to work on different parts of a single component at the same time, assign a lower average efficiency weight of 0.1.

7.3 FOON Construction from Non-expert Demonstrations

In assembly tasks, parts and their corresponding attributes, such as mass, color, shape, etc. are usually well-defined. Using object tracking and human hand tracking, the velocities and locations of parts and the sequence of human manipulations can be obtained through human demonstrations. Therefore, FOON for assembly tasks can be learned from human demonstrations. However, the human demonstrations of non-expert end-users can be inefficient. For example, humans may first place a part at an incorrect location and then fix it, which introduces unnecessary manipulations into the demonstration. Similarly, humans may accomplish assembly tasks with an unoptimized manipulation sequence, which increases the usage of manipulations with lower success rates or efficiency. To eventually get an efficient solution, we first need to learn the assembly task representation using FOON based on the non-expert demonstration

7.3.1 Creating Subgraphs

For each assembly task, multiple rounds of human demonstrations are performed by different non-expert users. Each round of human demonstration automatically generates a list of functional units based on object tracking and human hand tracking. The process is also recorded as an instructional video online. The corresponding object state nodes,

assembly state nodes for both input and output, and the motion node of each functional unit are manually verified according to the instructional video. These functional units are then connected and combined into a subgraph automatically. The subgraph of FOON is then visualized and verified manually. Each subgraph represents the structured knowledge of an overall process of an assembly task. However, each process may be inefficient since it is demonstrated by a non-expert user.

7.3.2 Merging Subgraphs

The FOON for assembly can be expanded by merging new subgraphs generated by different human demonstrations of different assembly tasks. The merging algorithm is described in Algorithm 7.1. Two functional units are regarded as equal if and only if the set of nodes in one functional unit is exactly the same as the other functional unit. The FOON is first empty; the subgraph of each round of human demonstration for each assembly task is merged into the universal FOON in sequence. For each functional unit in the subgraph, if it does not exist in the universal FOON, it will be added to the universal FOON. The merged FOON contains the structured knowledge from multiple rounds of non-expert demonstrations for multiple assembly tasks, which gives the potential to robots to find the optimized efficient solution for each assembly task. The functional units in the universal FOON can be further connected according to their input and output nodes. For each output node of a functional unit, it might connect to the same input node of other functional units. The connections are rebuilt in the following task retrieval process.

Algorithm 7.1. Merging New Subgraph to Universal FOON

Algorithm: Merging New Subgraph to Universal FOON

```
for all functional unit FU_i in new subgraph:
  for all existed functional unit FU_j in the universal FOON do
    if FU_i is equal to FU_j then
      FU_i is already existed in the universal FOON.
      continue to search next functional unit in new subgraph
    else
      Add FU_i to the universal FOON
      Add input object state nodes of FU_i to node list
      Add input assembly state node nodes of FU_i to node list
      Add output object state nodes of FU_i to node list
      Add output assembly state node of FU_i to node list
(The connections are rebuilt in the task retrieval process)
```

7.4 Assembly Task Retrieval and Optimization

In addition to FOON being a knowledge representation obtained or learned from human demonstrations, a FOON can also be used by robots for problem-solving and process optimization for assembly tasks. Given the goal of an assembly task, robots can search for all possible solutions and then choose the most efficient solution for the assembly task based on all the assembly tasks learned from human demonstrations. As mentioned in the previous section, each subgraph of FOON corresponds to a single round of human demonstration of an assembly task. Robots can at least choose the most efficient subgraph from the original non-expert demonstrations. Additionally, when multiple rounds of demonstrations of multiple assembly tasks are merged as a universal FOON, it is possible to find more efficient subgraphs other than the original demonstrations for assembly tasks.

Algorithm 7.2. Retrieval of Assembly Processes

Algorithm: Retrieval of Assembly Processes

Let N_{goal} be the goal assembly state node.
Let R be the list of root tree nodes.
Let P_{all} be the list of all possible assembly process.
Initialize R and P_{all} as empty list.
for all functional units FU_i in the universal FOON **do**
  if N_{goal} is the output assembly state node of FU_i **then**
    Add FU_i to R.
  end if
end for
for all root nodes r_i in R **do**
  Initialize a tree stack TS.
  Initialize a prelim tree node list L_p
  Append the root node r_i to TS.
  Append the root node r_i to L_p.
  while the tree stack TS is not empty **do**
    Pop the functional unit FU_h from the right side of TS.
    Set the head of search h to FU_h.
    for all input object state nodes N_{input} **do**
      Initialize a list of candidate functional units L_c
      for all functional units FU_i in the universal FOON **do**
        if N_{input} is one of the output object state nodes **and** FU_i *is* not an ancestor *of the head* h **then**
          Set FU_i as a child of the head h
          Set h as the parent of the functional unit FU_i
          Append FU_i to candidate list L_c
        end if
      end for
      Append candidate list L_c to prelim tree node list L_p
    end for
  end while
  Let P_d to be the cartesian product of L_p.
  for each dependent tree path d_p in P_d **do**
    for all task paths p found by breath-first-search $BFS(r_i)$ **do**
      Append path p *to* P_{all}.
    end for
  end for
for all path in P_{all} **do**
  Calculate the integrated weight of the path
return optimized task path p^*

7.4.1 Retrieving Assembly Task Trees

Once multiple non-expert demonstrations for multiple assembly tasks have been conducted, a universal FOON can be established by merging subgraphs generated by each round of human demonstrations using Algorithm 7.1. In order to find the optimized

solution for a task, we need to find all the possible assembly processes based on the universal FOON. Most existing symbolic planning algorithms focus on solving the planning problems that are represented using planning domain definition languages. Since the proposed FOON for assembly tasks uses a different representation, we will need to develop a corresponding planning algorithm for it. Each assembly process is a combination of functional units, which gives the path from an initial condition to the goal of an assembly task. The algorithm to retrieval all possible assembly task processes is shown in Algorithm 7.2.

First, we give a goal assembly state node N_{goal} to the robot. All the nodes, which contain N_{goal} as an output assembly state node, in the universal FOON are appended to a list of root tree nodes R. Over each root node $r_{i\,i}$ in R, it is possible to generate multiple task tree paths, which can accomplish the given assembly task. Starting from the root tree node, we iterate for each input object state node of the corresponding functional unit and search for the functional units which can produce it. When we search for the dependencies of a functional unit, we define this functional unit as the *head.* For an input object state node N_{input} of the head, if a functional unit contains it as an output object state node and it is not an ancestor of the head, then this functional unit is regarded as a dependency of the head for the input object node N_{input}. All of the functional units that produce N_{input} are regarded as candidate functional units and are added to the list of candidate functional units L_c, which is then appended to a list of the preliminary tree nodes L_p. This step proceeds until the tree stack is empty; at this point, the list L_p covers the functional units for the

dependencies for all the object state inputs. To accomplish the given assembly task, we only need one functional unit to meet the dependency for each input object state node. Thus, we compute the Cartesian product of the list L_p. Each product set of functional units will contain a whole path that meets object state input requirements of the corresponding root. By conducting a breadth-first search $BFS(r_i)$ with respect to the root r_i, we can obtain one assembly process to accomplish the assembly task. By iterating for each product set, we can obtain all possible assembly processes for the given assembly goal from the universal FOON.

7.4.2 Robot Execution Optimization

Once all possible assembly processes are determined, the optimal solution for the given assembly task is determined by the integrated weight of the assembly process. For an assembly process consisting of N functional units and weights $W_i \left(i = 1, 2, ..., N\right)$ for each function unit, the integrated weight of the assembly process can be written as

$$W_I = \prod_{i=1}^{N} W_i \tag{7.2}$$

The optimal assembly process can be determined by

$$p^* = \arg\max_{W_I} \left(P_{all}\right) \tag{7.3}$$

where P_{all} is the set of all possible assembly processes of the given assembly task. Once the optimized task-level assembly process p^* is determined using FOON and the corresponding task retrieval algorithm, for each motion in p^*, the robot will search for the

trajectory with minimum execution time among all the taught trajectories of the motion based on the situation of the workspace.

7.5 Experimental Results and Analysis

To evaluate the proposed approaches, different non-expert demonstrations for different assembly tasks are conducted. In this section, we first present the experimental setup. The assembly tasks and the corresponding non-expert demonstrations used in our experiments are then explained respectively. Based on the demonstrations, the results of robot learning and task retrieval and optimization are discussed.

7.5.1 Experimental Setup

The proposed approaches are verified and evaluated on a multi-model human-robot collaborative assembly test platform. The hardware setup of the test platform is illustrated in Figure 7.2 (a). In our experiments, we use the ABB Yumi, which is a dual-arm collaborative robot. The human stands face to face with the robot to work in a shared workspace. The Kinect RGB-D sensor offers a top-view point cloud of the workspace for part recognition and tracking. The software of the test platform is developed based on the Robot Operating System (ROS) and visualized through Rviz. The trajectory-level motion planning from point to point is accomplished based on the Open Motion Planning Library (OMPL) via MoveIt! motion planning framework [70].

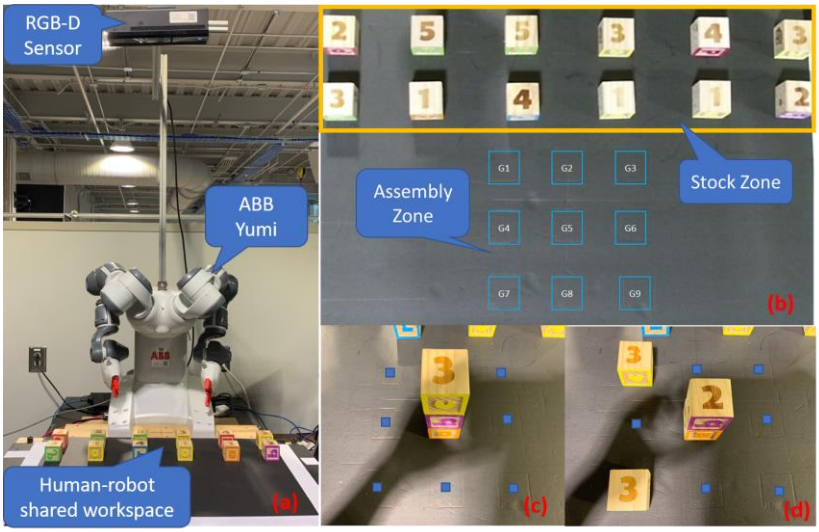


Figure 7.2. Experimental setup for FOON for Assembly.
(a) The configuration of the robot and RGB-D sensor. (b) The top view of the human-robot shared workspace captured by the RGB-D sensor. (c) The final state of the stacking task. (d) The final state of the shape constructing task.

7.5.2 Subgraphs of Non-expert Demonstrations

In our experiments, we use blocks with different numbers to represent different types of parts (O1 to O5). Initially, there are 12 parts in total located at different given locations in the stock zone. A 3 x 3 grid (G1 to G9) is defined as an assembly zone on the workbench. The start positions and the goal positions of pick-place and stacking robot actions between different grids are defined by assembly task in advance and the motion trajectories for the actions are generated via sampling-based motion planning algorithms in MoveIt! motion planning framework [70]. The motions and corresponding object states are learned via human demonstrations. The observed motions are mapped to corresponding elements in the action set according to the corresponding grid locations. The corresponding robot action assignment based on the observed motions is executed by sending the corresponding start position and the goal position and calling the sampling-based motion

planner to realize the action. Two types of assembly tasks are performed: the stacking task and the shape constructing task. The final assembly states of both tasks and the indexes of the grid locations are shown in Figure 7.2 (c) and Figure 7.2 (d). Three non-expert demonstrations are conducted for each task.

For the stacking task, the goal is to build a 3-level stack (O3O2O1) at G5. The process of the first demonstration (Figure 7.3) is picking an O1 from stock and placing it at G1; then picking an O2 from stock and placing it onto the O1 at G1; then picking O2O1 together and moving them from G1 to G5; picking an O3 from stock and stacking it on the O2O1 at G5. This demonstration is not efficient because the human stacked the first two parts at an improper location and then fixed it by moving the component to the correct location. The process of the second non-expert demonstration (Figure 7.4) is picking an O2 from stock and placing it to G5; then picking an O1 from stock and stacking it on the O2; then switching the positions of O1 and O2; finally picking an O3 from stock and stacking it on the O2 at G5. This demonstration is inefficient since an extra dual-arm manipulation was conducted to fix the order of the stack. The process of the third demonstration (Figure 7.5) is picking an O3 from stock and placing it to G8; then picking an O2 from stock and stacking it on the O2 at G8; then switching the positions of O2 and O3; then picking an O1 from stock and placing it to G5; finally picking the component (O3O2) from G8 and stacks it on the O1 at G5. This demonstration is less efficient than the previous two demonstrations since it contains five manipulations, including one dual-arm manipulation, to accomplish a three-parts stacking task.

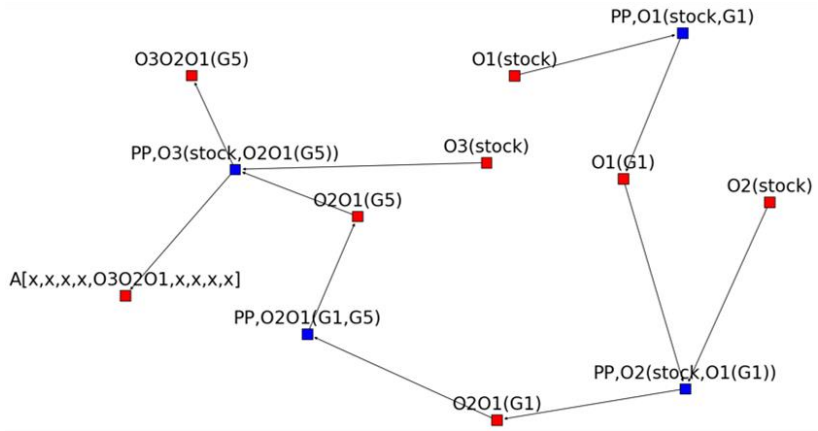


Figure 7.3. The subgraph of the first-round demonstration of the stacking task.

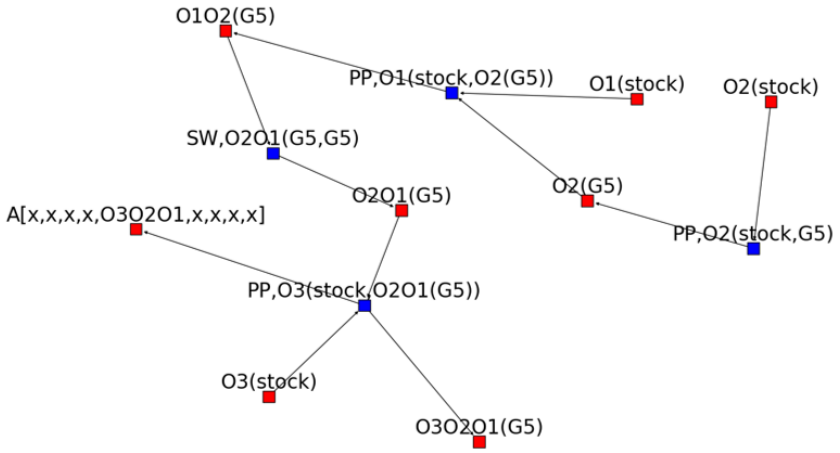


Figure 7.4. The subgraph of the second-round demonstration of the stacking task.

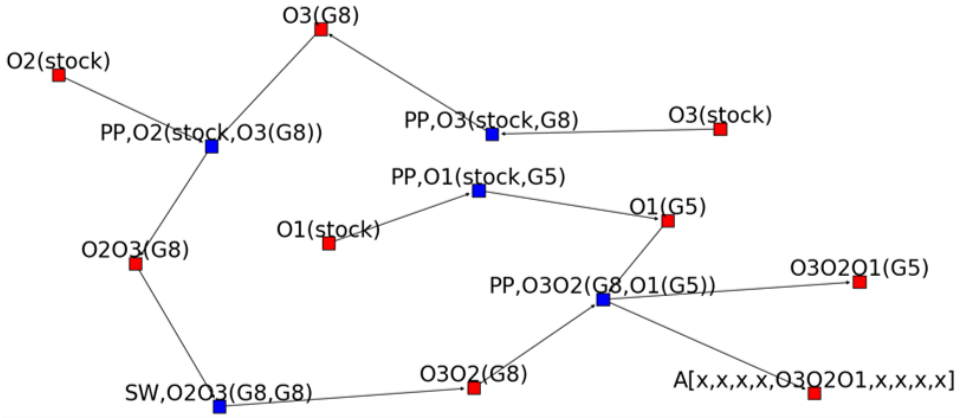


Figure 7.5. The subgraph of the third-round demonstration of the stacking task.

The shape constructing is to build a specific 3D shape in the 3 x 3 grid. The goal is to have a block O3 at both G1 and G7 and have a block O2 on the top of a block O1 at G5. Similar to the stacking task, we also conducted three non-expert demonstrations for the shape constructing task. The process of the first demonstration (Figure 7.6) is stacking two O3 at G1; then picking an O1 from stock and placing it to G5; then picking an O2 from stock and stacking it onto the O1 at G5; finally moving the O3 at the top of the component O3O3 to G7. This demonstration is inefficient because of the error pick-place action in the second step, which is fixed in the last step. The process of the second non-expert demonstration (Figure 7.7) is picking an O3 and placing it to G1; then building the component O2O1 via two pick-place actions at G4; then picking another O3 from stock and placing it to G7; finally moving the O2O1 from G4 to G5. The process of the third non-expert demonstration (Figure 7.8) is picking an O3 from stock and placing it to G9; then picking another O3 from stock and placing it to G8; then moving them to G1 and G7 respectively; afterward, a block O1 and a block O2 are picked from stock and stack at G5 to accomplish the task.

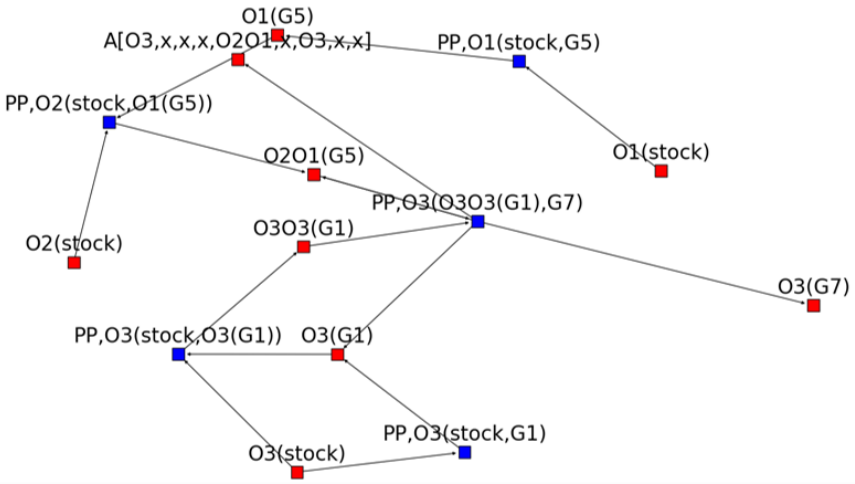


Figure 7.6. The subgraph of the first-round demonstration of the shape constructing task.

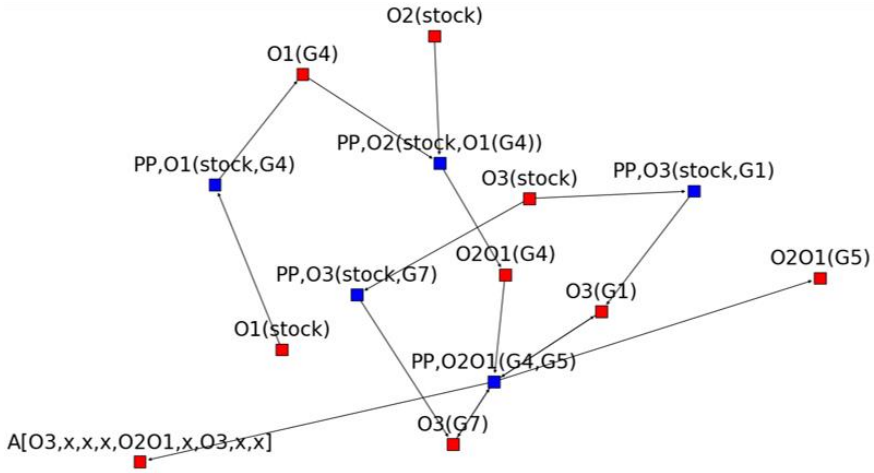


Figure 7.7. The subgraph of the second-round demonstration of the shape constructing task.

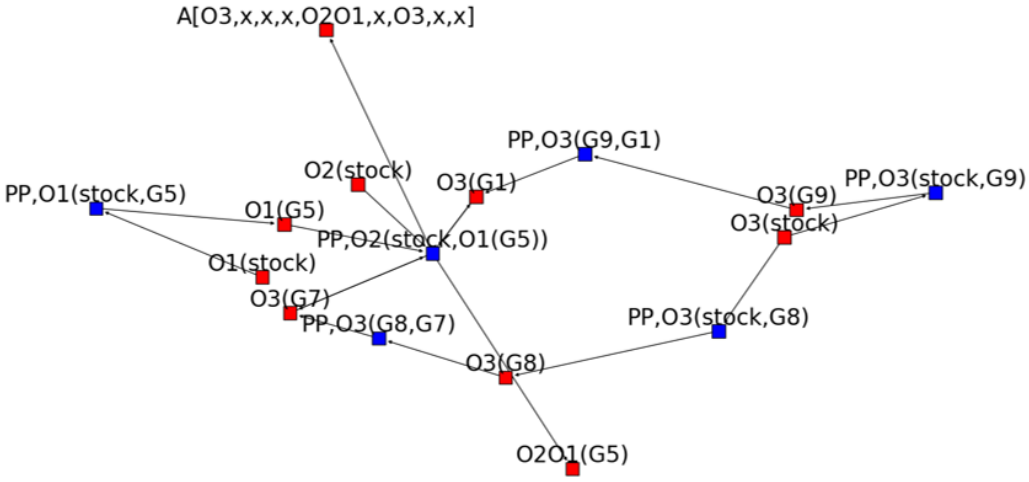


Figure 7.8. The subgraph of the third-round demonstration of the shape constructing task.

7.5.3 Results of FOON Generation and Optimal Assembly Task Tree Retrieving

To verify the proposed approaches, we merge the subgraphs generated by the non-expert demonstrations progressively to establish the universal FOON. For each stage, we compare the optimal assembly task process we can obtain based on the present situation of the robot knowledge.

In the first stage, the universal FOON was only built based on the three non-expert demonstrations of the stacking task is shown in Figure 7.9. It contains 12 functional units. Based on this merged FOON, the assembly process retrieval algorithm finds the same as the three non-expert demonstrations. The integrated weights of these three demonstrations are 0.478, 0.050, and 0.032 respectively. The results indicate that the robot will implement the stacking task by repeating the first human demonstration at this stage of merged FOON. The real robot execution of the stacking task is shown in Figure 7.10. Also, the robot cannot find a solution for the shape formatting task since none of its demonstrations has been integrated into the merged FOON yet.

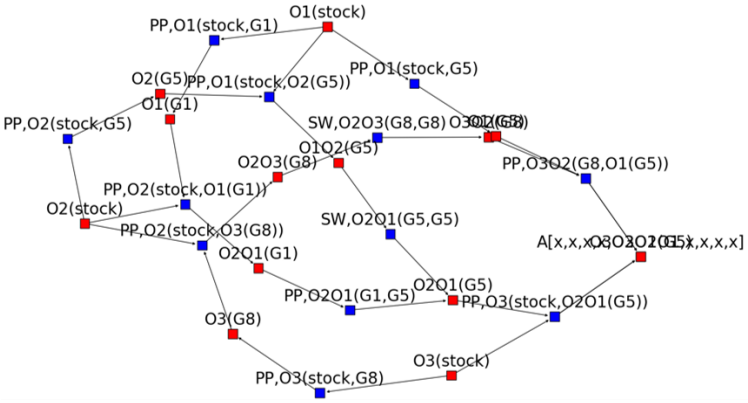


Figure 7.9. The merged graph of the three demonstrations of the stacking task.

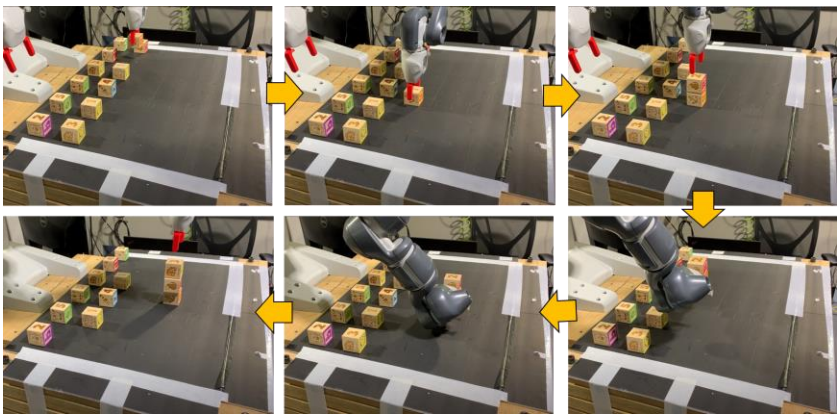


Figure 7.10. The real robot execution of the stacking task at the first stage.

In the second stage, the universal FOON was built based on the three non-expert demonstrations of the stacking task and the first non-expert demonstration of the shape constructing task. For the stacking task, the robot can find an optimized assembly process with three single-arm-single-part manipulations as shown in Figure 7.11, which is to pick up a block O1, a block O2, and a block O3 from stock and stack them at G5 in sequence. The corresponding integrated weight of this assembly process is 0.625, which is higher than all the non-expert demonstrations of the stacking task. The real robot execution of the optimized strategy on the stacking task is shown in Figure 7.12. For the shape constructing task, the first-round human demonstration is successfully reconstructed as the solution of the task at this stage of the merged FOON.

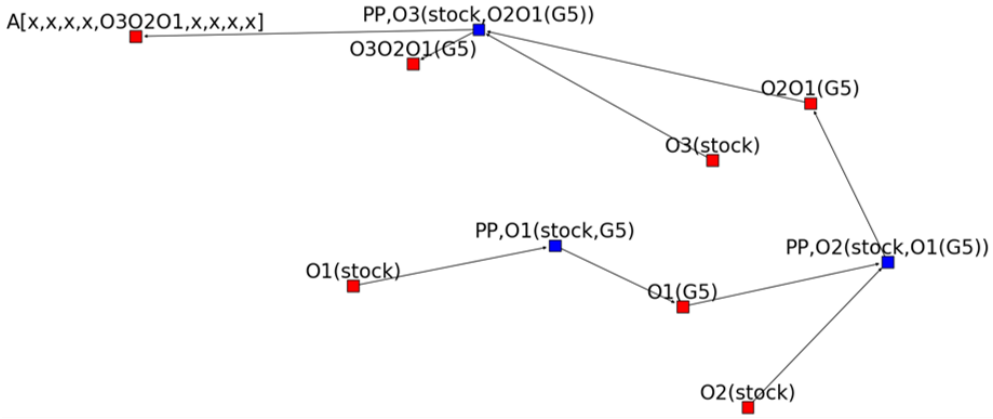


Figure 7.11. The optimized result of the stacking task.

In the third stage, we merged all the six non-expert demonstrations for both tasks to construct a universal FOON. The result of the universal FOON contains all six non-expert demonstrations of both the stacking task and shape constructing task. At this stage, the optimized result for the stacking task is also found as same as the three-motions solution shown in Figure 7.11. For the shape constructing task, an optimized solution with four

single-arm-single-part manipulations is found, which is to directly pick up a block O1, a block O2, and two O3 blocks and place them to their corresponding locations in sequence. The subgraph of the optimized solution of the shape constructing task is shown in Figure 7.13, and the robot execution of this optimized strategy is shown in Figure 7.14.

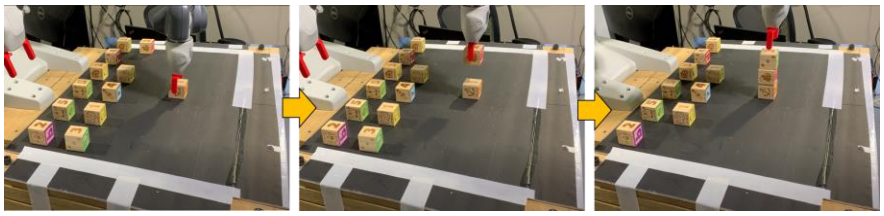


Figure 7.12. The real robot execution of the optimized solution of the stacking task.

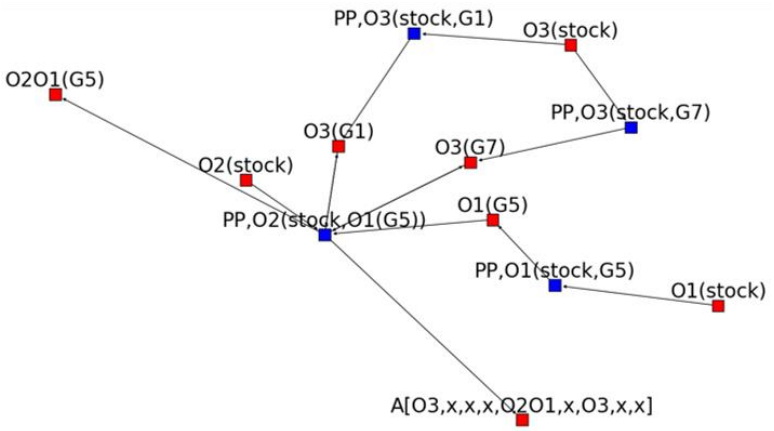


Figure 7.13. The optimized solution for the shape constructing task.

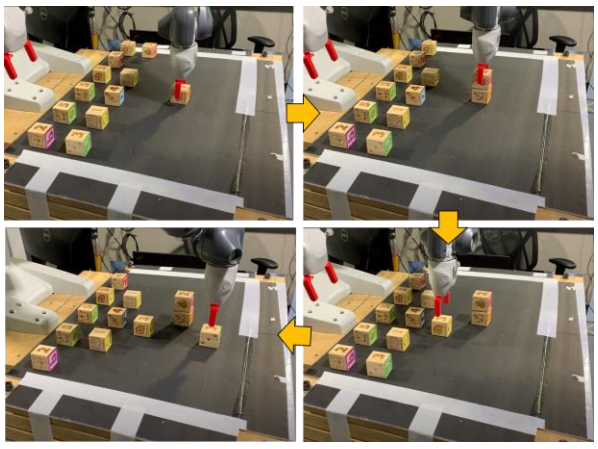


Figure 7.14. The real robot execution of the optimized shape constructing task.

7.5.4 Evaluations

The summary of the assembly process optimization of the three stages of the universal FOON is illustrated in Table 7.1. The results of the raw non-expert demonstrations and the optimized results of the stacking and the shape constructing task are shown in Table 7.2and Table 7.3 respectively. The universal FOON built based on the three demonstrations of the stacking task contains 12 functional units and 13 object state nodes in total. Based on this universal FOON, we can find the optimized solution for the stacking task with an overall success rate of 0.693 and an overall efficient weight of 0.690. By adding a non-expert demonstration of the shape constructing task, the universal FOON contains 16 functional units and 16 object state nodes. With the updated universal FOON with more knowledge of assembly tasks, we can obtain a better-optimized solution for the stacking task with an overall success rate of 0.770 and an overall efficient weight of 0.812. According to the task design, this is already the best solution for the stacking task. Also, the robot successfully learned the shape constructing task via the additional one-round human demonstration, though it is not very efficient. After merging two additional non-expert demonstrations of the shape constructing task, a better solution for the shape constructing task is found from the universal FOON, which is built based on six in-efficient human demonstrations. Based on the proposed task, the optimal solution simply contains four single-arm-single-part manipulations, which should be the most efficient solution based on our experimental setup. Moreover, the most efficient solution for the stacking task, which has already been found in the earlier stage, is also found in this universal FOON.

Furthermore, since we use a FOON model-based approach, we only require several rounds of non-expert demonstrations to derive the best possible solution for the task. In contrast, the existing data-driven approaches [78]–[80] usually require humans to demonstrate the task thousands of times to obtain enough training data to learn the task. Therefore, comparing to the existing data-driven approaches, the time cost and the teaching effort of human demonstrations can be significantly reduced using our proposed approach.

Table 7.1. Summary of Assembly Process Optimization of Universal FOON.

Graph	**Items**	**Stage**		
		1	2	3
Universal FOON	No. of Object State Node	13	16	18
	No. of Motion Node	12	16	24
	No. of Functional Unit	12	16	24
Optimized Stacking Assembly Process	No. of Functional Unit	4	3	3
	Overall Success Rate	0.693	0.770	0.770
	Overall Efficient Weight	0.690	0.812	0.812
	Integrated Weight	0.478	0.625	0.625
Optimized Shape Constructing Process	No. of Functional Unit	N/A	5	4
	Overall Success Rate	N/A	0.694	0.772
	Overall Efficient Weight	N/A	0.621	0.772
	Integrated Weight	N/A	0.432	0.595

Table 7.2. Raw Demonstrations vs Optimized Solution of Stacking Task.

Item	**Raw Demonstrations**				**Optimized**
	1st	2nd	3rd	Average	
No. of Actions	4	4	5	4.33	3
Success Rate	0.693	0.616	0.520	0.609	0.770
Efficient Weight	0.690	0.081	0.061	0.280	0.812
Integrated Weight	0.478	0.050	0.032	0.187	0.625

Table 7.3. Raw Demonstrations vs Optimized Solution of Shape Constructing Task.

Item	**Raw Demonstrations**				**Optimized**
	1st	2nd	3rd	Average	
No. of Actions	5	5	6	5.33	4
Success Rate	0.694	0.694	0.696	0.695	0.772
Efficient Weight	0.621	0.656	0.696	0.658	0.772
Integrated Weight	0.432	0.456	0.485	0.457	0.595

7.6 Conclusion

This chapter introduces a graph-based task representation approach based on the Functional Object-Oriented Network (FOON) to represent the knowledge of assembly tasks. It creates algorithms to create a FOON from multiple non-expert assembly demonstrations and also develops an assembly task tree retrieving approach with a robot execution optimization process to generate the best possible task execution plan from the FOON. The results indicate that robots can find the best possible assembly process among multiple rounds of non-expert demonstrations. The evaluation also indicates the effectiveness and advantages of the proposed approach compared to other existing approaches

CHAPTER 8
MULTI-MODEL SAMPLING-BASED MOTION PLANNING FOR TRAJECTORY OPTIMIZATION WITH EXECUTION CONSISTENCY

8.1 Introduction

Sampling-based motion planners are commonly used to generate collision-free trajectories in real time for industrial robots However, the trajectories generated by such approaches tend to be inconsistent (with large variations in path length, execution time and average manipulability) across multiple trials. Such unpredictability of robot motions makes them inappropriate for many robotic (and especially co-robotic) applications that demand consistency and predictability to promote human trust. In this chapter, we propose an optimization-based multi-model motion planning framework – to first leverage existing sampling-based motion planning algorithms to create alternate choices; downselect amongst these choices using a multi-criteria performance measure, and finally execute selected motion plan to ensure execution consistency. The simulation and experimental results validate that our approach can achieve the optimized collision-free trajectory with predictable and consistent executions in manipulation tasks.

The research problem is described in Section 8.2. The proposed multi-model sampling-based motion planning framework is presented in Section 8.3. The proposed approaches are evaluated the real automotive assembly tasks by simulation and real robot

experiments in Section 8.4 and Section 8.5. Finally, the chapter is summarized in Section 8.6.

8.2 Problem Statement

In the implementations of the sampling-based motion planners in such applications, we find that by given the same pair of start and goal positions in the same plan scene, the trajectories may have large variations between each motion planning and execution, even if the planner and its parameter configuration are unchanged. These uncertainties may lead to efficiency and safety issues in industrial applications.

For example, the torsion bar assembly with a mobile manipulator is illustrated in Figure 8.1. When the mobile base is parked at the stage, the manipulator needs to pick up the torsion bar, which is mounted on the mobile base, and place it to the correct assembly location underneath the vehicle chassis (represented by the overhead structure). In this case, the pick-up position (Figure 8.1 (a)) and the assembly position (Figure 8.1 (b)) of the torsion bar are known. The joint states of the manipulator at the pick-up and the assembly positions are used for motion planning. The sampling-based motion planning algorithms may achieve predictable and safe collision-free trajectories. For example, a straightforward collision-free trajectory of the torsion bar assembly manipulation is shown in Figure 8.2 (a). When there is an obstacle in the previous trajectories, a relatively optimized trajectory of the same task can also be obtained as shown in Figure 8.2 (b).

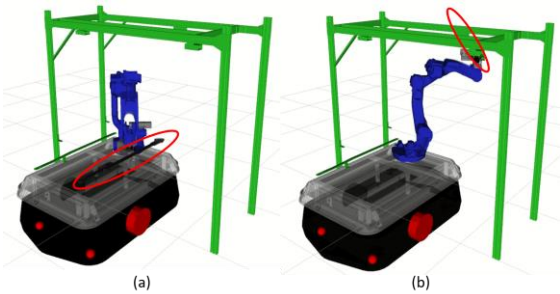


Figure 8.1. Start and goal position of the torsion bar assembly. manipulation.

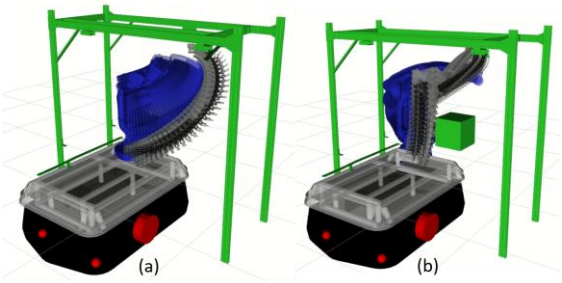


Figure 8.2. Examples of relatively predictable and safe trajectory for torsion bar assembly manipulation.

However, the optimized trajectory is not guaranteed in multiple executions. Tremendous differences are found between executions when the planning scene, planner, and its corresponding configurations are unchanged. For the trajectory shown in Figure 8.3 (a), the manipulator tends to move the torsion bar around the obstacle through a circle for collision-avoidance and finally lift the torsion bar to the assembly position, which makes the torsion bar rotate like a propeller. For the trajectories shown in Figure 8.3 (b), the manipulator moves the torsion bar to the backward of the mobile base to create some clearance to the obstacle.

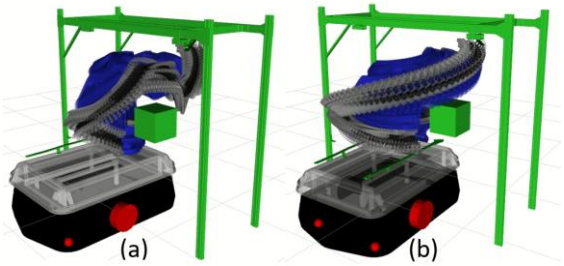


Figure 8.3. Examples of relatively unpredictable and unsafe trajectories for torsion bar assembly manipulation in continuous planning and executions.

Although it is possible to get the optimized trajectory by sampling-based motion planning algorithms, the trajectory generated from a single plan can also be unpredictable, inefficient, or even unsafe. These insufficiencies discourage the application of sampling-based motion planning algorithms in human-robot collaboration (HRC) in smart manufacturing contexts. To address the challenges, a multi-model sampling-based motion planning framework is proposed in this research.

8.3 Multi-model Sampling-based Motion Planning

In this section, the multi-model sampling-based motion planning framework is discussed in detail. The existing sampling-based motion planners used in the framework are introduced at first. Then the overview of the framework and the algorithms in trajectory optimization and motion planning for consistency are presented.

8.3.1 Multi-model Motion Planning Framework

The diagram of the multi-model motion planning framework is illustrated in Figure 8.4. The framework consists of three phases: multi-model preplanning, trajectory optimization, and planning and execution for consistency. In the multi-model preplanning phase, the planning scene, and the joint states of the robot arm at the start position and the

goal position are given, multiple sampling-based motion planners are called to generate the collision-free trajectories. Multiple unoptimizable collision-free trajectories with variations can be obtained in this phase.

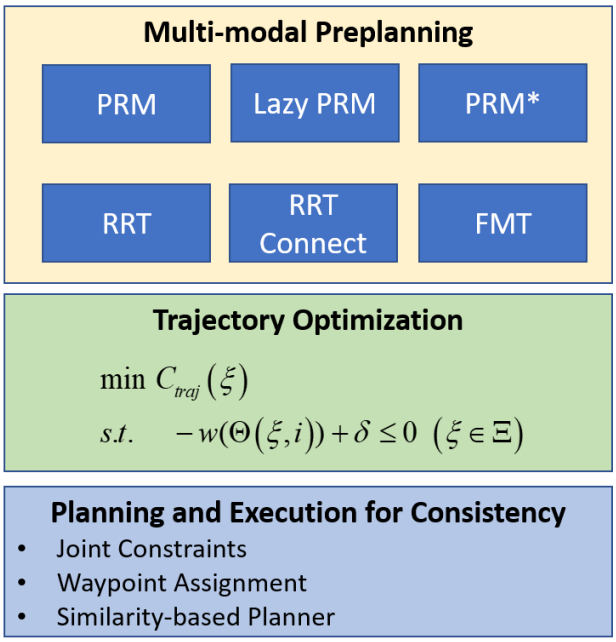


Figure 8.4. The diagram of the multi-modal motion planning framework.

In the trajectory optimization phase, a cost-function-based approach is developed to select an optimized trajectory from the trajectories generated in the multi-model preplanning phase. In the proposed cost function, predictability, efficiency, manipulability, and safety are considered. The selected trajectory and its corresponding motion planner are delivered to the third phase, planning, and execution for consistency, to generate consistent robot motions.

In the phase of planning and execution for consistency, the corresponding sampling-based motion planner is first specified according to the optimized trajectory. Then the optimized trajectory is divided into two to three segments based on the distance

and curvature, and the corresponding intermediate waypoints are assigned between the original start and goal state. For each segment of the robot motion, the joint constraints are added based on the boundary of each joint value in the corresponding segment of the optimized trajectory. Afterward, the longest common subsequence (LCS) is applied to measure the similarity between the corresponding segment of the new trajectory and the optimized trajectory. Given the tuned threshold of similarity and the maximum number of planning attempts, the similarity-based planner returns a relatively consistent trajectory with respect to the optimized trajectory for robot motion executions. The algorithms of each phase are discussed in more detail in the following sections.

8.3.2 Multi-model Preplanning

A list of sampling-based motion planners supported by the open motion planning library (OMPL) [69] is shown in Table 8.1. In our applications, two typical planners are commonly used for robot motion planning: RRT (multi-query) and PRM (single query).

For the RRT motion planner, the corresponding joint state of the manipulator at the start position is given as the initial configuration. Given the number of vertices, incremental distance, and the joint state of the robot arm at the goal position, the RRT planner grows a tree rooted at the initial configuration by randomly sampling from the configuration space of the robot arm. For each sample, a connection is attempted between it and the nearest state in the tree. Once a joint trajectory that connects the initial configuration and the goal configuration is found, the joint trajectory is returned for a motion execution.

Table 8.1 Sampling-based motion planners in OMPL

Planner Categories	
Single-query	**Multiple-query**
PRM	RRT
Lazy PRM	RRT Connect
PRM*	RRT*
Lazy PRM	Lower Bound Tree RRT
SPArse Roadmap Spanner	EST
SPARS2	FMT

For the PRM motion planner, given the joint state of the robot arm at the start and the goal positions, a roadmap of the free space is built via multiple random sampling processes of the configuration space of the robot arm in the construction phase. After each sampling process, all neighbors less than the predefined distance in the free space are connected until the road map is dense enough. Afterward, the collision-free path is obtained by Dijkstra's shortest path query [83].

In general, let Θ_{start} and Θ_{goal} be the joint state of the manipulator at the given start position and the goal position, a set of collision-free trajectories $\Xi = \{\xi_1, \xi_2, ..., \xi_N\}$ and the corresponding motion planners $\mathrm{I} = \{\eta_1, \eta_2, ..., \eta_N\}$ can be obtained in the multi-model preplanning phase.

8.3.3 Trajectory Optimization

The trajectory optimization phase targets to select an optimized trajectory from the set of trajectories generated by the multi-model preplanning. To solve this problem, a constraint of manipulability and cost-function-based trajectory evaluation metrics are proposed.

8.3.3.1 Manipulability Constraint

Since the trajectory candidates are collision-free and satisfy joint limits and velocity constraints, the main constraint for trajectory optimization in the proposed approach is singularity. For any waypoints in the optimized trajectory, the corresponding joint state of the manipulator Θ must not be close to a singularity of the manipulator. For each sampled joint state Θ in trajectory candidates, the manipulability index is calculated by [84]–[86]

$$w = \sqrt{\det\left(J\left(\Theta\right)J^{T}\left(\Theta\right)\right)} \tag{8.1}$$

where $J\left(\Theta\right)$ is the Jacobian matrix of the manipulator with respect to the joint state Θ. To avoid the singularity, for all the sampled waypoints in the trajectory, the corresponding manipulability index must satisfy

$$-w + \delta \leq 0 \tag{8.2}$$

where δ is a small constant.

8.3.3.2 Trajectory Evaluation Metrics

To identify the optimized trajectory in a set of trajectory candidates, multiple factors, such as predictability, efficiency, manipulability, and safety, should be taken into account comprehensively.

In human-robot collaboration, the human can predict the trajectory that the robot might execute according to his/her knowledge about the goal of robot motion. If the trajectories executed by the robot in each duty cycle are constant and match with the prediction, the human is comfortable in the shared workspace [87]. Human usually expects

the robot to be efficient and the trajectory length $L(\xi)$ can be utilized as a proxy of predictability cost [88]. Therefore, the cost of predictability of a trajectory can be defined as

$$C_p(\xi) = L(\xi) \Big/ \sum_{k=1}^{N} L(\xi_k) \ (\xi \in \Xi) \tag{8.3}$$

where k is the index of the trajectories, L is the path length of the trajectory.

For task efficiency, the normalized total execution time of the trajectory is used as the factor of efficiency score, which can be written as

$$C_e(\xi) = T(\xi) \Big/ \sum_{k=1}^{N} T(\xi_k) \ (\xi \in \Xi) \tag{8.4}$$

where T is the execution time of the trajectory.

From the manipulability perspective, we propose a cost of trajectory manipulability as

$$C_m(\xi) = \frac{1}{n(\xi)} \sum_{i=1}^{n(\xi)} w(\Theta_{\xi,i}) \ (\xi \in \Xi) \tag{8.5}$$

where $n(\xi)$ is the number of the sampled waypoint in the trajectory ξ, $w(\Theta_{\xi,i})$ is the manipulability index corresponding to the manipulator configuration at the i^{th} waypoint of trajectory ξ.

The safety of the manipulation not only depends on the robot arm but also relates to the object that detached on the gripper, especially for large-dimension and unsymmetrical objects. In the torsion bar assembly case, we propose the overall

displacement of the wrist joint $D_t(\xi)$ as the cost of the add-on safety factor. Thus, the cost of the add-on safety factor can be written as

$$C_s(\xi) = D_t(\xi) \Big/ \sum_{k=1}^{N} D_t(\xi_k) \ (\xi \in \Xi) \tag{8.6}$$

where the overall displacement of the wrist joint for the trajectory ξ is computed by

$$D_t(\xi) = \int_0^{T(\xi)} |\omega_t| \, dt \tag{8.7}$$

where ω_t is the angular velocity of the wrist joint.

Based on the scores of predictability, efficiency, manipulability, and add-on safety factor from (8.3) to (8.7), we propose the score of a candidate trajectory as the weighted sum of all the factors above, which can be written as

$$C_{traj}(\xi) = \alpha_p \cdot C_p(\xi) + \alpha_e \cdot C_e(\xi) + \alpha_m \cdot C_m(\xi) + \alpha_s \cdot C_s(\xi) \tag{8.8}$$

where the weights are all positive constants. Based on the cost function in (8.8) and the constraint in (8.2), the trajectory optimization can be written into the general form as

$$\begin{aligned} &\xi^* = \arg\min C_{traj}(\xi) \\ &s.t. \quad -w(\Theta(\xi, i)) + \delta \le 0 \ (\xi \in \Xi) \end{aligned} \tag{8.9}$$

For each collision-free generated by the probabilistic sampling-based planners, the constraint of manipulability is checked, and the value of cost function is calculated. The output of the trajectory optimization phase is the optimized trajectory $\xi^* \in \Xi$, which satisfies the constraint of manipulability and with the minimal value of cost function among all candidates, and its corresponding planner $\eta^* \in \mathrm{I}$.

8.3.4 Planning and Execution for Consistency

Based on the results of the optimized trajectory and the corresponding motion planner, a similarity-based motion planning algorithm is developed to obtain relatively consistent robot motions.

8.3.4.1 Intermediate Waypoint Extraction

Firstly, the motion planner that generated the optimized trajectory is set as the motion planner for the following planning and execution. Secondly, based on the joint trajectory of the optimized path, intermediate waypoints are calculated. As the optimized path is presented by a list of joint states, the halved waypoint Θ_m is determined based on the index of the joint state in the optimized path.

$$\Theta_m = \xi^* \left(\left[l^* / 2 + 0.5 \right] \right) \tag{8.10}$$

where l^* is the total number of the joint states recorded in the optimized trajectory ξ^*.

Moreover, the max-curvature waypoint is determined by the following steps. For each joint state Θ_i of the optimized trajectory, the position of the TCP $p_i = (x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i)$ is calculated by the forward kinematics of the robot arm. Then the curvature of the optimized trajectory at each recorded joint state can be calculated by

$$\kappa(\Theta_i) = \frac{\sqrt{(z_i" y_i' - y_i" z_i')^2 + (x_i" z_i' - z_i" x_i')^2 + (y_i" x_i' - x_i" y_i')^2}}{(x_i'^2 + y_i'^2 + z_i'^2)^{\frac{3}{2}}} \tag{8.11}$$

the waypoint Θ_c corresponding to the maximum curvature value in the optimized trajectory can be determined by

$$\Theta_c = \arg\max \kappa\left(\Theta_i\right)\left(i = 1, 2, ..., l^*\right) \tag{8.12}$$

In general, the optimized trajectory ξ^* can be divided into three segments by given the joint state at the start position Θ_{start}, the halved waypoint Θ_m, the max-curvature waypoint Θ_c, and the goal position Θ_{goal}. If the halved waypoint and the max-curvature waypoint are very closed to each other, the halved waypoint is dropped so that the optimized trajectory is divided into two segments.

8.3.4.2 Joint Constraint Extraction

To consistently generate the new trajectories, which are similar to the optimized trajectory, with sampling-based motion planner, and reduce the planning time, the joint constraints are abstract from the optimized trajectory. For each joint in each segment of the optimized trajectory, the median value of the joint value can be written as

$$\theta_{i,med} = \frac{\theta_{i,\max} + \theta_{i,\min}}{2}\left(i = 1, 2, ..., N_j\right) \tag{8.13}$$

where $\theta_{i,\min}$ and $\theta_{i,\max}$ are the minimum and maximum values of the i^{th} joint appeared in the specific trajectory segment. N_j is the total number of joints of the robot arm. By given a factor of tolerance k_{tol}, the lower and the upper tolerances of each joint can be written as

$$\theta_{i,low} = \left(1 - k_{tol}\right)\theta_{i,med} + k_{tol}\theta_{i,\min}\ (i = 1, 2, ..., N_j) \tag{8.14}$$

$$\theta_{i,high} = \left(1 - k_{tol}\right)\theta_{i,med} + k_{tol}\theta_{i,\max}\ (i = 1, 2, ..., N_j) \tag{8.15}$$

therefore, the constraint of each joint corresponding to the optimized trajectory segment can be represented by

$$c_i = \left\{\theta_{i,med}, \theta_{i,low}, \theta_{i,high}\right\} (i = 1, 2, ..., N_j) \tag{8.16}$$

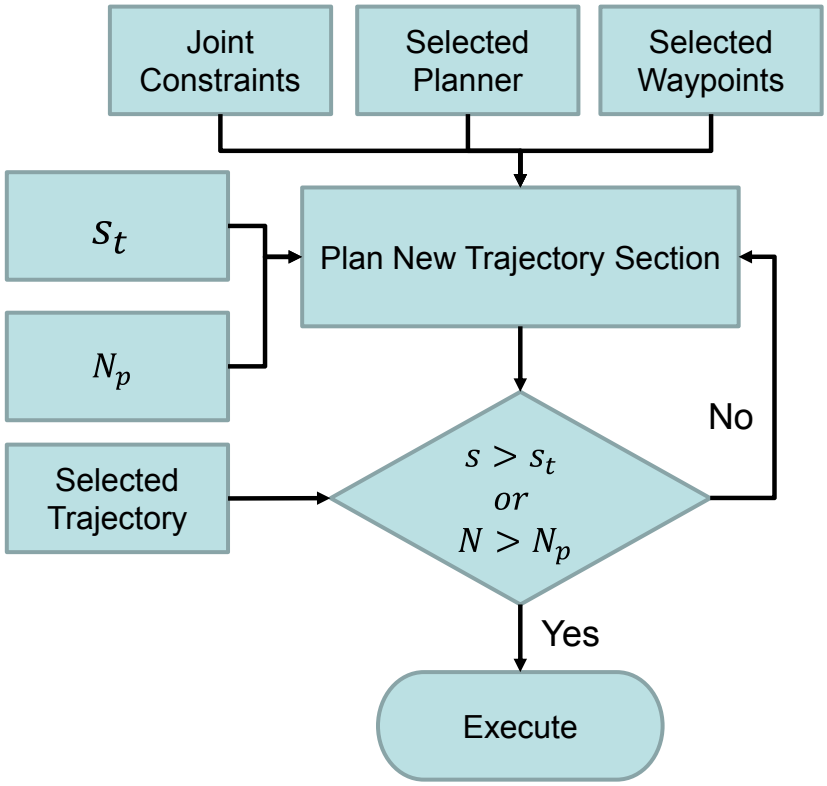


Figure 8.5. The flowchart of the similarity-based motion planning.

8.3.4.3 Similarity-based Motion Planning

With the previous results in optimal trajectory, intermediate waypoints, and joint constraints, a similarity-based motion planning algorithm is proposed to generate the consistent robot motions. As shown in Figure 8.5, for each segment ${}^{c}\xi_j$ of the motion planning for consistency, the corresponding segment ξ_j^* of the optimized trajectory is used as a reference. Give the maximum tolerance of the joint value θ_t in radius, and the maximum time of planning attempt N_p, and the target similarity $s_t \in [0,1]$, the sampling-based motion planner η^* is called to generate the new collision-free trajectory segment.

After each time of motion planning, the similarity between the new trajectory segment and the corresponding optimized trajectory segment is calculated based on the LCS algorithm. If the similarity is larger than the threshold s_t, the new trajectory segment is stored and then moves forward to plan the next segment. Otherwise, the new trajectory segment with the highest similarity will be stored once the maximum time of planning attempt is achieved, and then move forward to plan the next segment. The above procedure keeps working until all the segments between the start position and the position are successfully generated. Afterward, the overall updated trajectory is output for robot motion execution.

8.4 Simulation Evaluations

The proposed multi-model motion planning framework is evaluated via a torsion bar assembly task. In this section, the proposed framework is evaluated by simulation with three different setups in the planning scene, including different positions and orientations of the assembly positions, different sizes, positions, and orientations of the obstacle. For each planning scene, different intermediate waypoint assigning strategies are validated for comparison. The comparison in planning attempt, path length, path execution time, path similarity is presented.

8.4.1 Simulation Environment Setup

The setup of the simulation environment is illustrated in Figure 8.6. The robot model of the Yaskawa YMR12 mobile manipulator, which consists of a Clearpath OTTO 1500 mobile base and a Yaskawa MH12 manipulator is constructed. The 3D model of the overhead structure and the torsion bar are also developed and import into the planning

scene. The motion planning and robot control is implemented based on ROS [68] and MoveIt! [70].

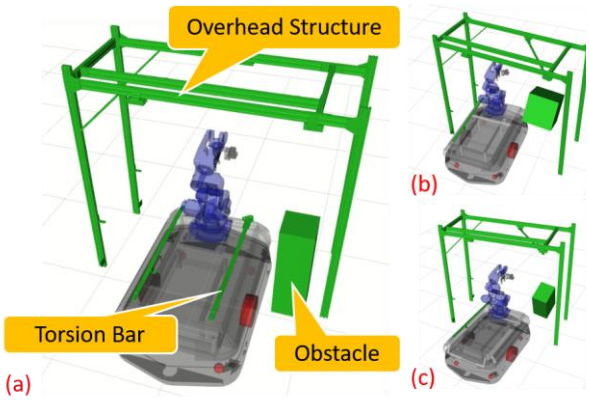


Figure 8.6. The setup of the simulation environment.

The torsion bar assembly tasks with the three different plane scenes in Figure 8.6 (a) ~ (c) are simulated to evaluate the proposed framework. The mobile base remains static in the plane scene, while the position and orientation of the overhead structure with respect to the mobile base are variant in different planning scenes. For each plan scene, the pick-up position and the assembly position are manually tuned for the torsion bar assembly manipulation.

8.4.2 Simulation Results and Analysis

To evaluate the proposed framework, for each plan scene shown in Figure 8.6, the start position is defined by corresponding joint state where the torsion bar is lifted along Z-axis around 5 cm from the mount on the mobile base; the goal position is defined by the corresponding joint state where the torsion bar is moved down along Z-axis around 5 cm with respect to the correct final assembly position. In the multi-model pre-planning phase, ten trajectories are generated by giving the start position and the goal position and

randomly choosing planners among RRT, FMT, and PRM. The joint trajectory, the trajectory of the tool center point (TCP), and the execution time are recorded as the baseline. In the phase of trajectory optimization, the optimal trajectory is selected from the ten trajectories with the proposed approach. Once the optimal trajectory is selected, the joint constraints, the halved waypoint, and the waypoint with maximum curvature are abstracted via the proposed approach. In each plan scene, the consistency of the execution is evaluated by considering the metrics including LCS-based trajectory similarity, the path length, the execution time of the trajectory, and the number of planning attempts. For each factor in the metrics, different waypoint assignment strategies are compared, which are "Constraint Only", "Constraint + Halved Waypoint", "Constraint + Max Curvature Waypoint", and "Constraint + Halved and Max Curvature Waypoints", respectively. Each waypoint assignment strategy is executed repeatedly ten times to evaluate the performance of the proposed approaches.

The results of the average joint trajectory similarity calculated by the LCS-based algorithm are shown in Figure 8.7. As the baseline, the average trajectory similarities are less than 9.0% comparing to the corresponding selected optimal trajectories in all the three simulation setups by only given the start position and the goal position. In the LCS-base similarity calculation, the maximum tolerance of the joint value θ_t is set to 0.03 rad. When the absolute values of the differences between all the corresponding joints of any two joint states are less than the threshold, these two joint states are considered as close to each other. The average trajectory similarities of the three simulation setups are increased to 47.1%, 11.8%, and 36.0%, respectively, after adding the joint constraints extracted from the

optimal trajectory and implementing the similarity-based motion planning algorithm. In Setup 1, both the "Constraint + Max Curvature Waypoint" and the "Constraint + Halved and Max Curvature Waypoint" strategies are with an average trajectory similarity 100%. In Setup 2, the "Constraint + Halved and Max Curvature Waypoints" strategy is with the highest average trajectory similarity 55.9%. In Setup 3, the "Constraint + Max Curvature Waypoints" strategy is with the highest similarity 69.2%, while the "Constraint + Halved and Max Curvature Waypoints" strategy is slightly less similarity 67.2%. The results indicate that adding intermediate waypoints can increase the average trajectory similarity, and the "Constraint + Max Curvature Waypoints" strategy generates good trajectory similarity in all three simulation setups.

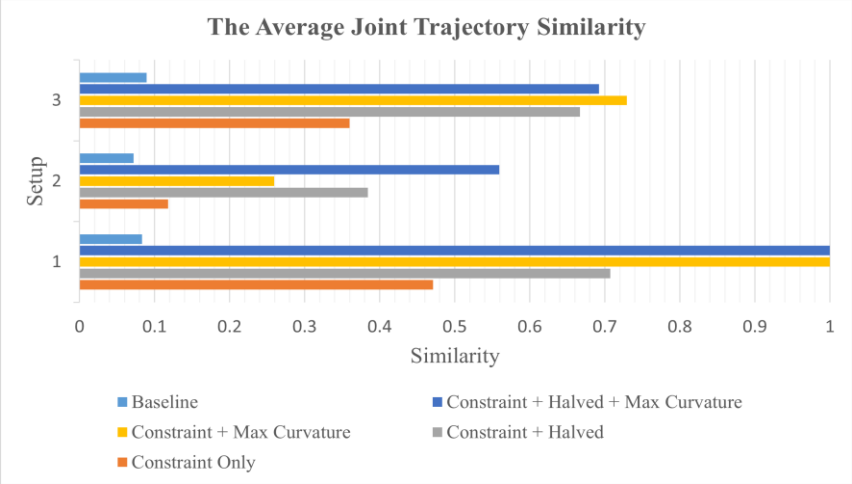


Figure 8.7. The average joint trajectory similarity calculated by the LCS-base algorithm.

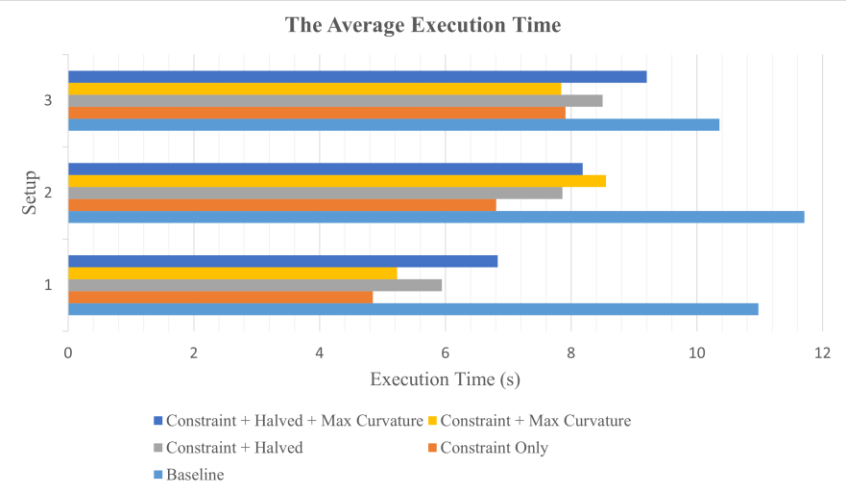


Figure 8.8. The average execution time of the different waypoint assignment strategies. The original execution time of the selected optimal trajectory of each setup is 5.663s, 8.007s, and 7.702s, respectively.

The results of the average execution time of the three simulation setups are shown in Figure 8.8. The execution time includes the planning time of each trajectory section and the robot execution time of the overall trajectory when the accelerate scale and the velocity scale are set as 1.0 in MoveIt. After the multi-model pre-planning and the trajectory optimization, the original execution time of the selected optimal trajectories is 5.663 s, 8.007 s, and 7.702 s, respectively, for the three simulation setups. Since the consistency of the robot executions is significantly increased and the trajectories that are similar to the corresponding optimal trajectories are repeatedly executed, the execution time can be reduced on average when the constraint and intermediate waypoints are added. The standard deviation of the execution time is shown in Figure 8.9. The results show that the standard deviation of the execution time can be significantly reduced by only adding the joint constraint extracted from the optimal trajectory, though in Setup 2, the standard deviation of the execution time is 1.156 seconds that is larger than 1second. Moreover, the standard deviation of the execution time can be further decreased when adding intermediate

waypoints. All three different waypoint assignment strategies achieve a standard deviation less than 0.574 s. The "Constraint + Max Curvature Waypoints" strategy obtains the smallest standard deviation of execution time in all the three simulation setups, which are 0.0006 s, 0.176 s, and 0.172 s, respectively.

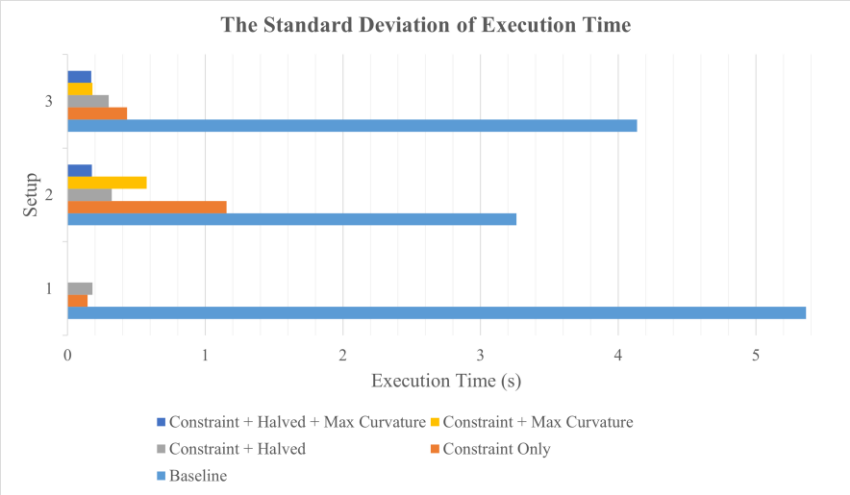


Figure 8.9. The standard deviation of execution time of the different waypoint assignment strategies. The original execution time of the selected optimal trajectory of each setup is 5.663s, 8.007s, and 7.702, respectively.

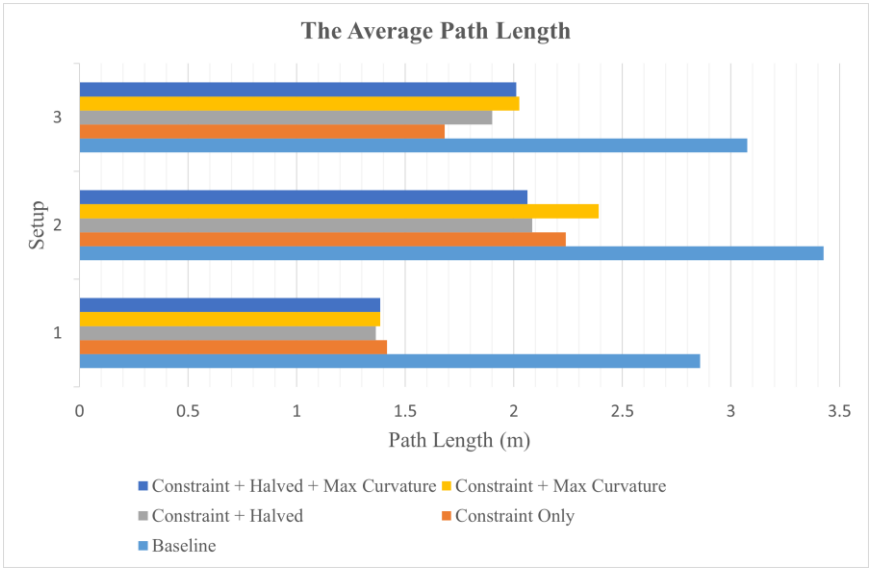


Figure 8.10. The average path length of the different waypoint assignment strategies. The original path length of the selected optimal trajectory of each simulation setup is 1.384m, 2.032m, and 1.948m, respectively.

Moreover, the results of the average path length of the different waypoint assignment strategies are illustrated in Figure 8.10. After the multi-model pre-planning and the trajectory optimization, the original path length of the selected optimal trajectories is 1.384m, 2.032m, and 1.948m, respectively. The standard deviation of the path length of the different waypoint assignment strategies for the three simulation setups is shown in Figure 8.11. The results indicate that, as a baseline, when only given the start and goal position for motion planning, the average path length is much higher than the optimal trajectory and the variance of the path length is larger than 1.4m in all three simulation setups. By adding the joint constraint and the intermediate waypoints, both the average value and the variance of the path length can be significantly reduced, which means better efficiency and consistency. In Setup 1, the "Constraint + Max Curvature Waypoints" strategy and the "Constraint + Halved and Max Curvature Waypoints" strategy obtain very good consistency with the average path length 1.384m and standard deviation less than 10^{-4}m. In Setup 2, the "Constraint + Max Curvature Waypoints" strategy is with the smallest standard deviation of 0.085 m but with a relatively longer average path length of 2.391 m comparing to the selected optimal trajectory. The "Constraint + Halved and Max Curvature Waypoints" strategy obtains a better trade-off between the path length and the standard deviation, which is with an average path length of 2.062 m and a standard deviation of 0.098 m. In Setup 3, the "Constraint + Max Curvature Waypoint" strategy obtains the smallest standard deviation of 0.020 m with 0.034 m difference on average path length with respect to the optimal trajectory. The "Constraint + Halved and Max Curvature

Waypoints" obtains an average path length of 2.011 m that is closest to the optimal trajectory with a standard deviation of 0.038 m.

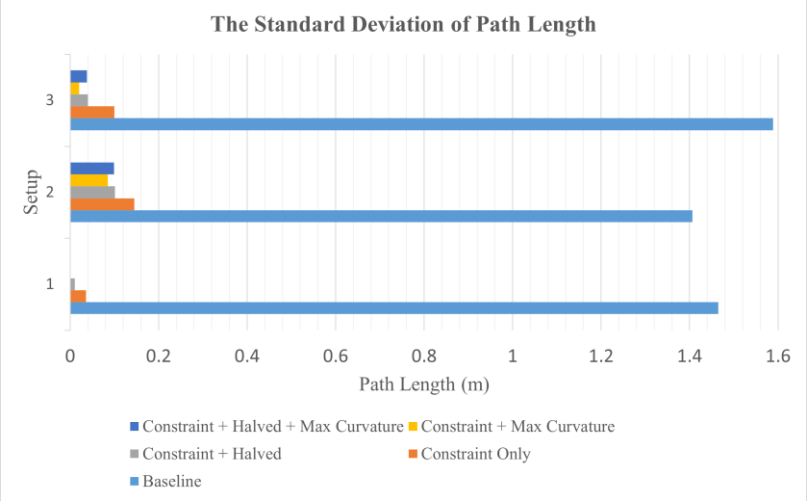


Figure 8.11. The standard deviation of path length of the different waypoint assignment strategies. The original path length of the selected optimal trajectory of each simulation setup is 1.384m, 2.032m, and 1.948m, respectively.

Furthermore, the results of the average plan attempts are illustrated in Figure 8.12. In the simulation, the maximum plan attempts of each trajectory segment are set to 20. For the "Constraint + Halved and Max Curvature Waypoints" strategy, the minimal plan attempts to generate the whole trajectory is 3, while for the "Constraint + Max Curvature Waypoint" strategy or the "Constraint + Halved Waypoint" strategy, the minimal plan attempts to generate the whole trajectory is 2. The results indicate that adding intermediate waypoints can reduce the plan attempts in all three simulation setups. The "Constraint + Max Curvature Waypoint" strategy obtains the best results in both Setup 1 and Setup 2, but not performs very well in Setup 2. The "Constraint + Halved Waypoint" and the "Constraint + Max Curvature Waypoint" strategy obtain a relatively very good result in all three simulation setups.

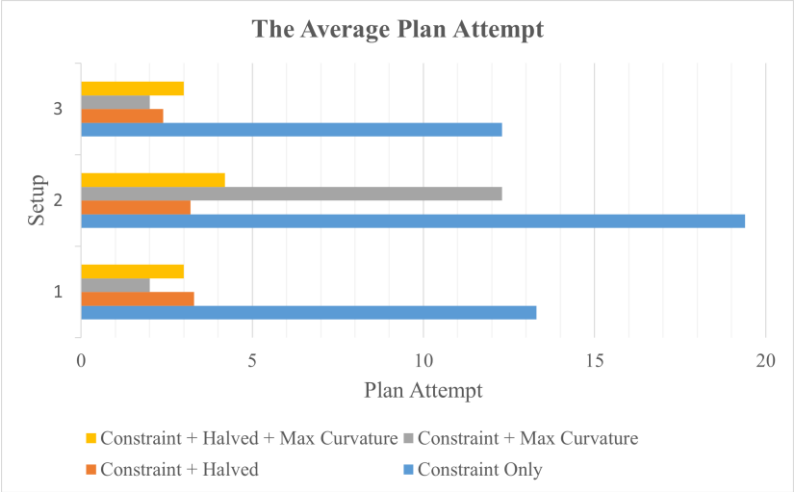


Figure 8.12. The average plan attempts of different waypoint strategies of the three simulation setups.

In summary, the proposed framework can significantly increase the efficiency and the consistency of the robot motions in different simulation setups. Both the joint constraint and intermediate waypoint assignment are necessary in order to generate consistent trajectories. Among the tested waypoint assignment strategies, the "Constraint + Halved and Max Curvature Waypoints" strategy obtains the best performance in line balancing with a stable solution founding and a good trade-off between consistency.

8.5 Experimental Evaluations

To further evaluate the proposed framework, the experiments are conducted on hardware-in-the-loop physical-testing tasks. In this section, we first present the experimental setup of the torsion bar assembly task in the real manufacturing contexts, then discuss the results of both simulation and real robot execution in the experiment.

8.5.1 *Experimental Setup*

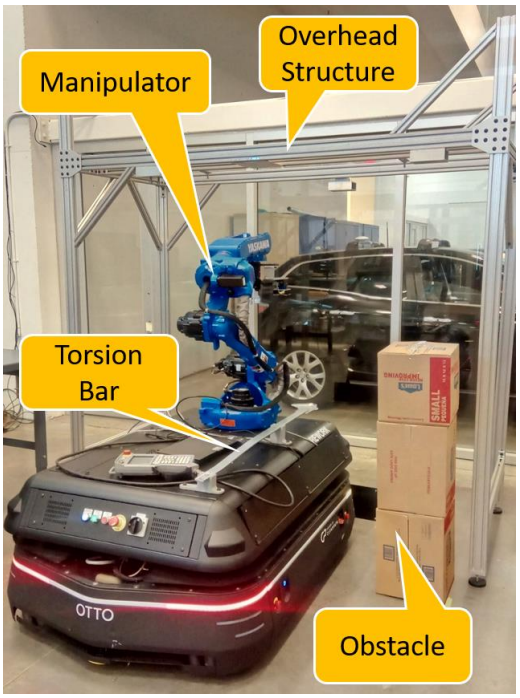


Figure 8.13. Hardware setup for the experiment.

The hardware setup of the YMR12 mobile manipulator, overhead structure, and the obstacle is illustrated in Figure 8.13. The relative positions of the robot, overhead structure, and the obstacle are similar to Setup 1 in the previous section. The obstacle is combined with three cartons, the overall dimensions are 0.47 m in length, 0.47 m in width, and 1.23 m tall. The distance from the center of the obstacle to the right and front edge of the mobile base is 0.38 m.

8.5.2 *Experimental Results and Analysis*

Based on the results of the simulations in the previous section, the "Constraint + Halved and Max Curvature Waypoints" strategy is selected for the hardware-in-the-loop testing. The pick-up position and the assembly position of the torsion bar, the start position,

and the goal position of the testing trajectory are first programmed with a teach pendant in real situation. Based on the start position and the goal position of the testing trajectory, the optimal trajectory and the corresponding joint constraints and intermediate waypoints are generated by the proposed approaches. In the execution of the trajectories, the robot is driven directly through MoveIt! motion planning framework with a laptop running ROS. The gripper is controlled independently through a Python API based on RS485 communication. The action of the gripper is trigged manually through the keyboard.

The process of the torsion bar assembly tasks is illustrated in Figure 8.14 (a) ~ (h). The torsion bar is first located at the mount of the mobile base and the robot arm goes to grasp the torsion bar at the pick-up location (Figure 8.14 (a)). Then the torsion bar is lifted horizontally to the start position of the testing trajectory (Figure 8.14 (b)). The two intermediate waypoints obtained by the proposed algorithms are shown in Figure 8.14 (c) and Figure 8.14 (d). The goal position of the testing trajectory is illustrated in Figure 8.14 (f). Figure 8.14 (e) illustrates an intermediate state when the robot is moving from the second intermediate waypoint to the goal position of the testing trajectory. After arrived the goal position, the torsion bar is lifted horizontally for about 5 cm to its accurate assembly position. As shown in Figure 8.14 (g), the robot holds the torsion bar at the assembly position until the human screws it up to the overhead structure. In the last step, the robot releases the torsion bar and moves back to its home position (Figure 8.14 (h)).

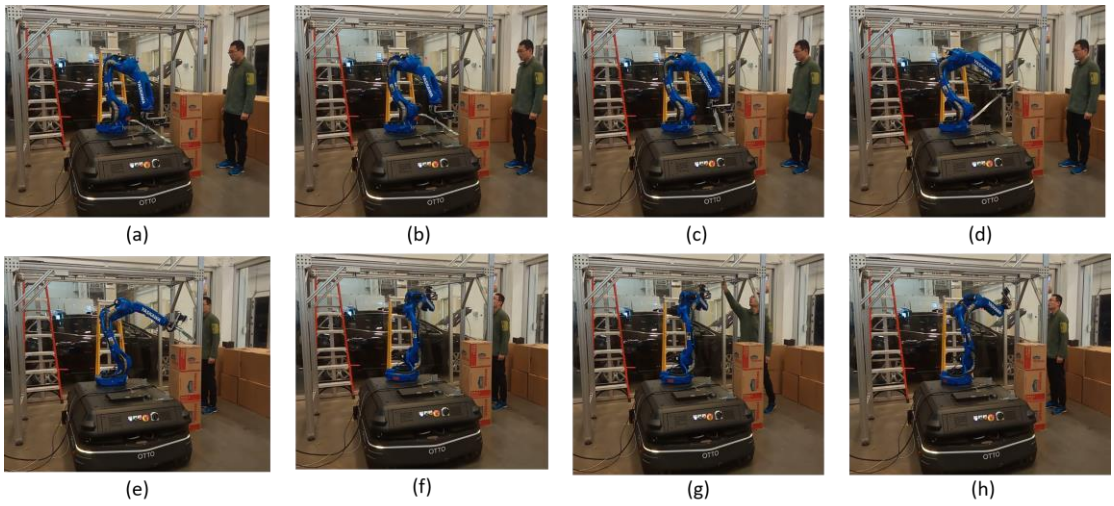


Figure 8.14. The process of the torsion bar assembly.

In the pre-planning phase, ten trajectories are first generated by simulation, which is intuitively presented in Figure 8.15. The overall execution time of these trajectories variants from 10.301 s to 19.001 s, the path length of these trajectories variant from 1.30 m to 5.55 m. As a baseline, the result of the pre-planning indicates that the robot motion is inconsistent and unpredictable by planning while only given the original start position and the original goal position of the testing trajectory. The unpredictable robot motions may lead to safety issues in the human-robot collaboration in the proposed manufacturing contexts.

In the trajectory optimization phase, one optimal trajectory is selected from the above candidate trajectories via the proposed const-function-based approach. The original path length of the selected optimal trajectory is 1.569 m, and the corresponding execution time in full speed in the simulation is 10.301s. Afterward, the joint constraints, the halved waypoint, and the max curvature waypoint are extracted from the optimal trajectory. Based on this information of the optimal trajectory, the “Constraint + Halved and Max Curvature

Waypoints" strategy is implemented in the following similarity-based motion planning in both simulation and hardware-in-the-loop testing.

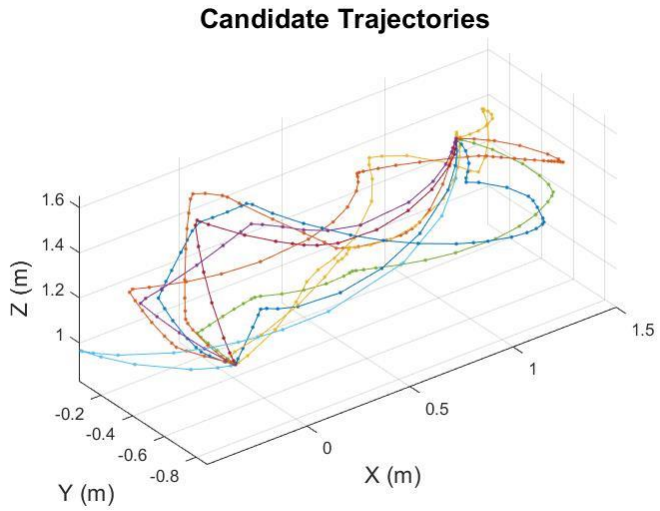


Figure 8.15. The candidate trajectories generated in the pre-planning phase by simulation.

In the hardware-in-the-loop experiment, the robot runs at a reduced speed for safety purposes since the manipulator of the YMR12 mobile robot is not a collaborative manipulator. The maximum velocity scale and the maximum acceleration scale are set as 0.02. To evaluate the consistency of the robot executions, the torsion bar assembly task described in the previous section is repeated 25 times. Another 25 times of full-speed executions in simulation, while the maximum velocity scale and the maximum acceleration scale are set as 1.0 are also conducted, for the reference of ideal execution time. As an intuitive representation of the trajectory consistency, the trajectories of the 25 executions, which are recorded from the feedback of the real robot, are plotted in Figure 8.16. The plot indicates that the proposed approaches can generate consistent trajectories in the hardware test.

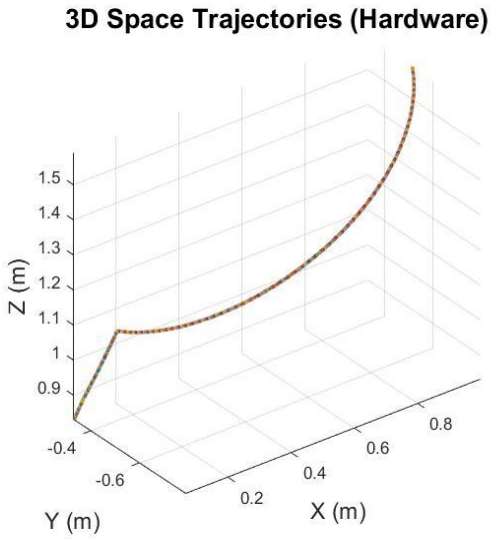


Figure 8.16. The trajectories of 25 executions in robot experiment.

For further quantitative results, the trajectory similarity of the simulation and the hardware test, which is calculated based on the proposed LCS-base algorithm, is shown in Figure 8.17. The average trajectory similarity of the 25 executions in the simulation and the hardware test is 98.4% and 97.4%, respectively. The standard deviation of the trajectory similarity in the simulation and the hardware test is 2.38% and 2.93%, respectively. Though there are some differences between the original optimal trajectory and each execution, the trajectory is still consistent visually as shown in. Figure 8.16.

The overall execution time of the 25 executions of the testing trajectory in the hardware tests and the simulations are illustrated in Figure 8.18. The average execution time of the testing trajectory is 30.28s with a slow speed limit on the hardware and is 11.98s with full speed in the simulation. The standard deviation of the execution time is 0.302 s and 0.297 s for the simulation and the hardware test, respectively.

Figure 8.17. The trajectory similarity in the simulation and the hardware test.

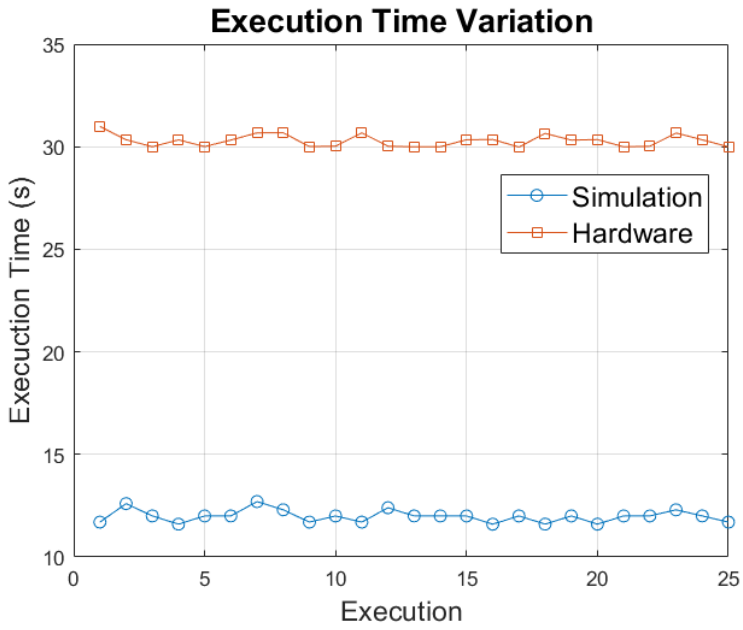


Figure 8.18. The execution time in the simulation and the hardware test.

The path length of 25 executions of the testing trajectory in both the hardware and the simulation are shown in Figure 8.19. The average path length of the hardware test is 1.5733 m and that of the simulation is 1.5675 m. The error between the hardware test and

the simulation result is less than 0.4%. The hardware executions achieve a standard deviation of 0.16 mm in trajectories length, while the simulation is with a standard deviation of 1.59 mm.

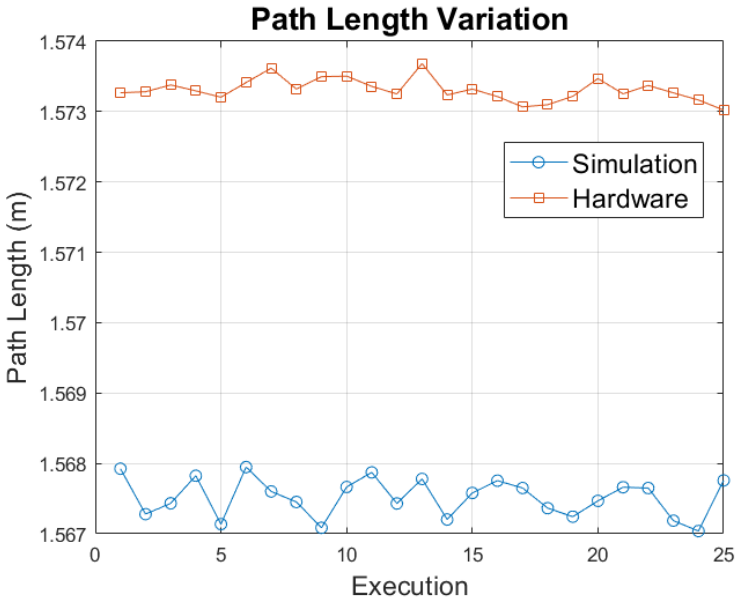


Figure 8.19. Trajectory length in simulation and hardware test.

The experimental results indicate that the proposed approach can obtain a better trajectory consistency in real industrial contexts. In the human-robot collaboration, the optimized robot motions can avoid collisions and singularities, and more predictable to the human in the human-robot collaboration.

8.6 Conclusion

In this chapter, a multi-model sampling-based motion planning framework for trajectory optimization and execution consistency in smart manufacturing contexts is proposed. By selecting the optimized trajectory, extracting the joint constraints, and assigning the intermediate waypoints between the start and goal positions via the proposed

approaches, robots can achieve consistent, predictable, and safe manipulations in human-robot collaboration in real manufacturing contexts. The torsion bar assembly task is demonstrated to validate and evaluate the proposed approach in real-world industrial contexts. The results of the simulation and the hardware-in-the-loop testing revealed the effectiveness and the advantages of the proposed approaches.

The objective of this research is to extend the application of sampling-based motion planning to productive industrial usage and human-robot collaborations. The current results demonstrated a trajectory section of the torsion bar assembly process with a well-constructed environment and static obstacles. More complex tasks and dynamic environment will be conducted as our future work.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

The dissertation aims to improve the level of automation in automotive assembly via human-robot collaboration.

In Chapter 3, the design of a Smart Companion Robot (SCR) is proposed. The robot prototype is developed and validated with real collaborative assembly tasks in manufacturing. The SCR is a representative example of how we can leverage both human and robot capabilities in manufacturing, where the human handles dexterous assembly tasks while the robot handles the heavy payload of automotive parts. The application of such a robot system is clearly not limited to automotive assembly alone. Any manufacturing task that involves heavy-payload transportation and manipulation tasks could be benefited from this type of robot. In addition, the robot may also have a wide range of potential applications in other areas., such as domestic services.

In Chapter 4, a vision-based approach is proposed to enable the robot to learn object placing tasks in assembly from human demonstrations. The experimental results indicate that our framework can abstract the knowledge of object-placing tasks from a human demonstration video in a simulation environment. The proposed framework can be used to modeling the object placing tasks and detecting the error of tasks online. Also, the approach is potential to be used in more completed assembly tasks.

In Chapter 5, a CNN-based approach is proposed to learn and assist humans in assembly tasks from human demonstrations. Experimental results show that CNN is

effective in robot learning during collaborative assembly and the robot can be trained to actively assist humans in the human-robot collaborative assembly process in real-time. The training data can be obtained in a few rounds of demonstrations, which alleviates the need for complex modeling and setup compared to the existing approaches. It also suggests a potential way by which the robot can be personalized by its users to assist them in their preferred ways in collaborative assembly applications.

In Chapter 6, a learn-to-collaboration approach with the TC-IRL method is proposed to generate robot assistance to assist humans in human-robot collaborative assembly. The TC-IRL approach can significantly reduce the size of the action and state space and lead to a reduced requirement of training data and computational cost compared to traditional IRL. The proposed approach can also allow humans to teach the robot to accomplish new tasks with larger geometry scales by learning from several small-scale demonstrations.

In Chapter 7, a graph-based approach is proposed for robot learning of assembly tasks from non-expert human demonstrations. The proposed approaches adapted the FOON to assembly tasks, and the results indicate that robots can find the best possible assembly process among multiple rounds of non-expert demonstrations. The evaluation also indicates the effectiveness and advantages of the proposed approach compared to other existing approaches.

In Chapter 8, a multi-model sampling-based motion planning framework for trajectory optimization and execution consistency in smart manufacturing contexts is proposed. The experimental results in the simulation and the hardware-in-the-loop testing

demonstrate the proposed approach can extend the application of sampling-based motion planning to productive industrial usage and human-robot collaborations.

9.2 Future Work

In this dissertation, research on collaborative robotic systems, especially the development and application of the parallel mobile manipulator, has been conducted. Further exploration can be conducted to facilitate the design and performance of the parallel mobile manipulators for realistic smart manufacturing applications. For example, different combinations of the mobile bases and the parallel manipulators are worth testing in real manufacturing applications.

From the RLfD perspective, novel approaches to improve the safety, efficiency, and human comfort in multi-human multi-robots (a hybrid of both parallel and serial mobile manipulators) collaboration are worth further investigation. Also, the proper human-robot interactions in the process of robot teaching and learning are worth further improvement, especially the proper hints generated by the robots to guide the non-expert human partners to facilitate the effectiveness.

APPENDICES

Appendix A

List of Publications

Published in Peer-reviewed Journals and Conference

2021

- W. Wang, R. Li, **Y. Chen**, Y. Sun, and Y. Jia* *, "Predicting Human Intentions by Learning from Demonstrations in Human-Robot Hand-Over Tasks," *IEEE Transactions on Automation Science and Engineering,* 2021.

2020

- **Y. Chen**, W. Wang, V. Krovi and Y. Jia*, "Enabling Robot to Assist Human in Collaborative Assembly using Convolutional Neural Networks," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- S. Zhang, **Y. Chen**, J. Zhang and Y. Jia*, "Real-Time Adaptive Assembly Scheduling in Multi-Robot-Human Collaboration According to Human Capability," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- J. Gill, **Y. Chen**, F. Niaki, M. Tomaszewski, W. Wang, L. Mears, P. Pisu, Y. Jia and V. Krovi, "A Smart Companion Robot for Automotive Assembly," *Recent Advances in Industrial Robotics, World Scientific,* 2020. (*Book Chapter*)
- W. Wang, **Y. Chen**, R. Li, Z. Zhang, V. Krovi and Y. Jia*, "Human-Robot Collaboration for Advanced Manufacturing by Learning from Multi-Modal Human Demonstrations," *Recent Advances in Industrial Robotics, World Scientific,* 2020. (*Book Chapter*)
- W. Wang, **Y. Chen**, and Y. Jia*, "Evaluation and Optimization of Dual-Arm Robot Path Planning for Human-Robot Collaborative Tasks in Smart Manufacturing Contexts," *ASME Letters in Dynamic Systems and Control*, vol. 1, no. 1, pp. 1-7, 2020.
- Y. Sun, W. Wang, **Y. Chen** and Y. Jia*, "Learn How to Assist Humans through Human Teaching and Robot Learning in Human-Robot Collaborative Assembly," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 2020.

2019

- **Y. Chen**, W. Wang, Z. Abdollahi, Z. Wang, J. Schulte, V. Krovi and Y. Jia*, "A Smart Companion Robot for Heavy Payload Transport and Manipulation in Automotive Assembly," *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. (*Invited Presentation*)

- **Y. Chen**, W. Wang, V. Krovi and Y. Jia*, "Modeling and Learning of Object Placing Tasks from Human Demonstrations in Smart Manufacturing," *SAE Technical Paper*, 2019.
- W. Wang, **Y. Chen**, R. Li and Y. Jia*, "Learning and Comfort in Human–Robot Interaction: A Review," *Applied Sciences*, vol. 9, no. 23, pp. 1-20, 2019.
- W. Wang, **Y. Chen**, and Y. Jia*, "Which Way is the Best? Evaluation and Optimization of Dual-Arm Robot Path Planning for Human-Robot Collaborative Tasks in Smart Manufacturing Contexts," *ASME Dynamic Systems and Control Conference (DSCC)*, 2019.
- Z. Zhang, **Y. Chen**, W. Wang and Y. Jia*, "Prediction of Human Actions in Assembly Process by a Spatial-Temporal End-to-End Learning Model," *SAE Technical Paper*, 2019.
- W. Wang, R. Li, Z. Diekel, **Y. Chen**, Z. Zhang and Y. Jia*, "Controlling Object Hand-Over in Human-Robot Collaboration via Natural Wearable Sensing," *IEEE Transactions on Human-Machine Systems*, vol. 49, no. 1, pp. 59-71, 2019.
- W. Wang, R. Li, **Y. Chen**, Z. Diekel, and Y. Jia*, "Facilitating Human-Robot Collaborative Tasks by Teaching-Learning-Collaboration from Human Demonstrations," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 640-653, 2019.
- R. Li, W. Wang, **Y. Chen**, and Y. Jia*, "Natural Language and Gesture Perception based Robot Companion Teaching for Assisting Human Workers in Assembly Contexts," *ASME Dynamic Systems and Control Conference (DSCC)*, 2019.
- S. Baskaran, F. Niaki, M. Tomaszewski, J. Gill, **Y. Chen**, Y. Jia, L. Mears and V. Krovi, "Digital Human and Robot Simulation in Automotive Assembly using Siemens Process Simulate: A Feasibility Study," *Procedia Manufacturing*, vol. 34, pp. 986-994, 2019. (*47th SME North American Manufacturing Research Conference* ***Best Student Research Presentation Award***)

2018

- **Y. Chen**, W. Wang, Z. Abdollahi, Z. Wang, J. Schulte, V. Krovi and Y. Jia*, "A Robotic Lift Assister: A Smart Companion Robot for Heavy Payload Transport and Manipulation in Automotive Assembly," *IEEE Robotics and Automation Magazine*, vol. 25, no. 2, pp. 107-119, 2018.
- W. Wang, **Y. Chen**, and Y. Jia*, "Predicting Human Intentions by Learning from Multi-modal Human Demonstrations in Human-Robot Hand-over Tasks," *ACM/IEEE International Conference on Human Robot Interaction (HRI)*, 2018
- W. Wang, **Y. Chen**, and Y. Jia*, "Cost Functions based Dynamic Optimization for Robot Action Planning in Human-Robot Collaborative Tasks," *ACM/IEEE International Conference on Human Robot Interaction (HRI)*, 2018.
- L. Jiang, W. Wang, **Y. Chen** and Y. Jia*, "Personalize Vison-based Human Following for Mobile Robots by Learning from Human-driven

Demonstrations," *IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2018.

- Li, R., Wang, W., **Chen, Y**., Srinivasan, S. and Krovi, V.N., 2018, September. An end-to-end fully automatic bay parking approach for autonomous vehicles. In *Dynamic Systems and Control Conference* (Vol. 51906, p. V002T15A004). American Society of Mechanical Engineers. **(Best Robotics Paper Award)**
- W. Wang, N. Liu, R. Li, **Y. Chen** and Y. Jia*, "HuCoM: A Model for Human Comfort Estimation in Personalized Human-Robot Collaboration," *ASME Dynamic Systems and Control Conference (DSCC)*, 2018.

Submitted Manuscripts

- **Y. Chen**, David Paulius, Yu Sun, and Y. Jia*, " Learn Assembly Tasks from Non-expert Demonstrations via Functional Object-Oriented Network," International Conference on Intelligent Robots and Systems (IROS), 2021. (Under review)
- **Y. Chen,** W. Wang, M. Tomaszewski, V. Krovi, and Y. Jia*, " Multi-Model Sampling-based Motion Planning for Manipulation Trajectory Optimization with Execution Consistency," *IEEE Transactions on Automation Science and Engineering,* 2021. (Under review)
- **Y. Chen**, W. Wang and Y. Jia*, " Teaching-Learning-Collaboration Model for Task Constraint and Human Working Style Learning in Collaborative Assembly Tasks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020. (Under review)

REFERENCES

[1] M. Akamatsu, P. Green, K. Bengler, M. Akamatsu, P. Green, and K. Bengler, "Automotive Technology and Human Factors Research: Past, Present, and Future," *Int. J. Veh. Technol.*, vol. 2013, pp. 1–27, 2013.

[2] M. Peshkin and J. E. Colgate, "Cobots," *Ind. Robot An Int. J.*, 1999.

[3] J. Fryman and B. Matthias, "Safety of Industrial Robots: From Conventional to Collaborative Applications," *Robot. Proc. Robot. 2012; 7th Ger. Conf.*, pp. 1–5, 2012.

[4] S. Aguilera-Marinovic, M. Torres-Torriti, and F. Auat-Cheein, "General dynamic model for skid-steer mobile manipulators with wheel-ground interactions," *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 1, pp. 433–444, 2017.

[5] H. N. Rahimi and M. Nazemizadeh, "Dynamic analysis and intelligent control techniques for flexible manipulators: A review," *Adv. Robot.*, vol. 28, no. 2, pp. 63–76, 2014.

[6] M. Shneier and R. Bostelman, "Literature Review of Mobile Robots for Manufacturing Literature Review of Mobile Robots for Manufacturing," 2015.

[7] J. Iqbal, R. U. Islam, S. Z. Abbas, A. A. Khan, and S. A. Ajwad, "Automating industrial tasks through mechatronic systems – a review of robotics in industrial perspective," *Teh. Vjesn. - Tech. Gaz.*, vol. 23, no. 3, pp. 917–924, 2016.

[8] A. Huber and A. Weiss, "Developing Human-Robot Interaction for an Industry 4 . 0 Robot," *Proc. Companion 2017 ACM/IEEE Int. Conf. Human-Robot Interact. - HRI '17*, no. March 6-9, pp. 137–138, 2017.

[9] B. Zhang, J. Wang, and T. Fuhlbrigge, "A review of the commercial brain-computer interface technology from perspective of industrial robotics," *2010 IEEE Int. Conf. Autom. Logist. ICAL 2010*, pp. 379–384, 2010.

[10] F. Ohashi *et al.*, "Realization of heavy object transportation by mobile robots using handcarts and outrigger," *ROBOMECH J.*, vol. 3, no. 1, p. 27, Nov. 2016.

[11] C. P. Tang and V. N. Krovi, "Manipulability-based configuration evaluation of cooperative payload transport by mobile manipulator collectives," *Robotica*, vol.

25, no. 1, pp. 29–42, 2007.

[12] R. Bostelman, T. Hong, and J. Marvel, “Survey of Research for Performance Measurement of Mobile Manipulators,” vol. 121, pp. 342–366, 2016.

[13] J. Chen and A. Zelinsky, “Programing by demonstration: Coping with suboptimal teaching actions,” *Int. J. Rob. Res.*, vol. 22, no. 5, pp. 299–319, 2003.

[14] M. Hiratsuka, N. Makondo, B. Rosman, and O. Hasegawa, “Trajectory learning from human demonstrations via manifold mapping,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3935–3940.

[15] S. Calinon, F. D’halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, “Learning and reproduction of gestures by imitation,” *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 44–54, 2010.

[16] A. Jha, S. S. Chiddarwar, and M. V Andulkar, “An integrated approach for robot training using Kinect and human arm kinematics,” in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 216–221.

[17] G. Maeda, M. Ewerton, D. Koert, and J. Peters, “Acquiring and generalizing the embodiment mapping from human observations to robot skills,” *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 784–791, 2016.

[18] Y. Sun, S. Ren, and Y. Lin, “Object–object interaction affordance learning,” *Rob. Auton. Syst.*, vol. 62, no. 4, pp. 487–496, 2014.

[19] J. Mainprice and D. Berenson, “Human-robot collaborative manipulation planning using early prediction of human motion,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 299–306.

[20] Y. Jia, L. She, Y. Cheng, J. Bao, J. Y. Chai, and N. Xi, “Program robots manufacturing tasks by natural language instructions,” in *Automation Science and Engineering (CASE), 2016 IEEE International Conference on*, 2016, pp. 633–638.

[21] Y. Jia, N. Xi, J. Y. Chai, Y. Cheng, R. Fang, and L. She, “Perceptive feedback for natural language control of robotic operations,” in *2014 IEEE International*

Conference on Robotics and Automation (ICRA), 2014, pp. 6673–6678.

[22] L. Bodenhagen *et al.*, "An adaptable robot vision system performing manipulation actions with flexible objects," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 749–765, 2014.

[23] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object – action," vol. 30, no. 10, pp. 1229–1249, 2011.

[24] M. Javaid, M. Žefran, and A. Yavolovsky, "Using pressure sensors to identify manipulation actions during human physical interaction," in *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2015, pp. 670–675.

[25] W. Wang, R. Li, Y. Chen, and Y. Jia, "Human intention prediction in human-robot collaborative tasks," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 279–280.

[26] Z. Wang, A. Peer, and M. Buss, "An HMM approach to realistic haptic human-robot interaction," in *World Haptics 2009-Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 2009, pp. 374–379.

[27] Y. Gu, A. Thobbi, and W. Sheng, "Human-robot collaborative manipulation through imitation and reinforcement learning," in *2011 IEEE International Conference on Information and Automation*, 2011, pp. 151–156.

[28] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," in *Advances in Neural Information Processing Systems*, 2011, pp. 19–27.

[29] Y.-H. Yoo and J.-H. Kim, "Procedural memory learning from demonstration for task performance," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 2435–2440.

[30] E. L. Sauser and A. G. Billard, "Biologically inspired multimodal integration: Interferences in a human-robot interaction game," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5619–5624.

[31] C. Martinez, R. Boca, B. Zhang, H. Chen, and S. Nidamarthi, "Automated bin picking system for randomly located industrial parts," in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, 2015, pp. 1–6.

[32] H. Chen, B. Zhang, and T. Fuhlbrigge, "Robot throwing trajectory planning for solid waste handling," in *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2019, pp. 1372–1375.

[33] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[34] L. E. Kavraki *et al.*, "Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces," *Robot. Autom. IEEE Trans.*, vol. 12, no. 4, pp. 566–580, 1996.

[35] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 2000, vol. 1, pp. 521–528.

[36] L. Jaillet and T. Siméon, "A PRM-based motion planner for dynamically changing environments," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 2004, vol. 2, pp. 1606–1611.

[37] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[38] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[39] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[40] P. Chaudhari, S. Karaman, D. Hsu, and E. Frazzoli, "Sampling-based algorithms for continuous-time POMDPs," in *2013 American Control Conference*, 2013, pp. 4604–4610.

[41] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*,

2012, pp. 2537–2542.

[42] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *Int. J. Rob. Res.*, vol. 34, no. 7, pp. 883–921, 2015.

[43] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, “Sampling-based roadmap of trees for parallel motion planning,” *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 597–608, 2005.

[44] M. Kobilarov, “Cross-entropy motion planning,” *Int. J. Rob. Res.*, vol. 31, no. 7, pp. 855–871, 2012.

[45] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, “Guided expansive spaces trees: A search strategy for motion-and cost-constrained state spaces,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, 2004, vol. 4, pp. 3968–3973.

[46] Y. Fang and Z. Huang, “Kinematics of a three-degree-of-freedom in-parallel actuated manipulator mechanism,” *Mech. Mach. Theory*, vol. 32, no. 7, pp. 789–796, 1997.

[47] A. T. Vijayan and S. Ashok, “Integrating visual guidance and feedback for an industrial robot,” *2017 2nd Int. Conf. Control Robot. Eng. ICCRE 2017*, pp. 18–22, 2017.

[48] “LBR iiwa A feel for the production world of tomorrow.”

[49] D. Industry, “Mobile robot ymr12.”

[50] Y. Jia, Y. Liu, N. Xi, H. Wang, and P. Stürmer, “Design of Robotic Human Assistance Systems Using a Mobile Manipulator,” *Int. J. Adv. Robot. Syst.*, vol. 9, no. 5, p. 165, 2012.

[51] S. S. Srinivasa *et al.*, “Herb 2.0: Lessons learned from developing a mobile manipulator for the home,” *Proc. IEEE*, vol. 100, no. 8, pp. 2410–2428, 2012.

[52] T. On, R. Avid, and A. Vol, “Motion Planning in a Plane Using Generalized oroncpi Diagrams,” vol. 5, no. 2, pp. 143–150, 1989.

[53] R. M. Haralick, S. R. Sternberg, and X. Zhuang, “Image analysis using

mathematical morphology," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 4, pp. 532–550, 1987.

[54] P. Salembier and J. C. Serra, "Morphological multiscale image segmentation," in *Visual Communications and Image Processing'92*, 1992, vol. 1818, pp. 620–632.

[55] P. Salembier, "Morphological multiscale segmentation for image coding," *Signal Processing*, vol. 38, no. 3, pp. 359–386, 1994.

[56] D. Zhang and G. Lu, "A COMPARISON OF SHAPE RETRIEVAL USING FOURIER DESCRIPTORS AND SHORT-TIME FOURIER DESCRIPTORS," pp. 1–4.

[57] C. T. Zahn and R. Z. Roskies, "Fourier Descriptors ror Plane Closed Curves," vol. c, no. 3, 1972.

[58] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing Images Using the Hausdorff Distance," vol. 15, no. 9, pp. 850–863, 1993.

[59] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Rob. Res.*, vol. 10, no. 6, pp. 628–649, 1991.

[60] L. Johannsmeier and S. Haddadin, "A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes," *IEEE Robot. Autom. Lett.*, vol. 2, no. 1, pp. 41–48, 2017.

[61] L. Wang, B. Schmidt, M. Givehchi, and G. Adamson, "Robotic assembly planning and control with enhanced adaptability through function blocks," *Int. J. Adv. Manuf. Technol.*, vol. 77, no. 1–4, pp. 705–715, 2015.

[62] W. Wang, R. Li, Y. Chen, Z. M. Diekel, and Y. Jia, "Facilitating Human-Robot Collaborative Tasks by Teaching-Learning-Collaboration From Human Demonstrations," *IEEE Trans. Autom. Sci. Eng.*, no. 99, pp. 1–14, 2018.

[63] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient model learning from joint-action demonstrations for human-robot collaborative tasks," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2015, pp. 189–196.

[64] D. Malik, M. Palaniappan, J. F. Fisac, D. Hadfield-Menell, S. Russell, and A. D.

Dragan, “An efficient, generalized bellman update for cooperative inverse reinforcement learning,” *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 8, pp. 5435–5443, 2018.

[65] U. Kartoun, H. Stern, and Y. Edan, “A Human-Robot Collaborative Reinforcement Learning Algorithm,” *J. Intell. Robot. Syst.*, vol. 60, no. 2, pp. 217–239, 2010.

[66] M. K. Holder, “Why are more people right-handed,” *Sci. Am. Inc*, 1997.

[67] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum Entropy Deep Inverse Reinforcement Learning,” 2015.

[68] M. Quigley *et al.*, “ROS : an open-source Robot Operating System,” no. Figure 1.

[69] I. A. Şucan, M. Moll, and L. Kavraki, “The open motion planning library,” *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, 2012.

[70] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ROS topics],” *IEEE Robot. Autom. Mag.*, vol. 19, no. 1, pp. 18–19, 2012.

[71] G. Hoffman, “Evaluating fluency in human–robot collaboration,” *IEEE Trans. Human-Machine Syst.*, vol. 49, no. 3, pp. 209–218, 2019.

[72] K. Dautenhahn *et al.*, “How may I serve you?: a robot companion approaching a seated person in a helping context,” in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 172–179.

[73] L. Takayama and C. Pantofaru, “Influences on proxemic behaviors in human-robot interaction,” *2009 IEEE/RSJ Int. Conf. Intell. Robot. Syst. IROS 2009*, pp. 5495–5502, 2009.

[74] M. L. Walters *et al.*, “The influence of subjects’ personality traits on personal spatial zones in a human-robot interaction experiment,” in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, 2005, pp. 347–352.

[75] D. R. Olsen and M. A. Goodrich, “Metrics for evaluating human-robot interactions,” in *Proceedings of PERMIS*, 2003, vol. 2003, p. 4.

[76] F. Dehais, E. A. Sisbot, R. Alami, and M. Causse, “Physiological and subjective evaluation of a human–robot object hand-over task,” *Appl. Ergon.*, vol. 42, no. 6,

pp. 785–791, 2011.

[77] I. E. Allen and C. A. Seaman, “Likert scales and data analyses,” *Qual. Prog.*, vol. 40, no. 7, pp. 64–65, 2007.

[78] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, “Imitation learning from imperfect demonstration,” *arXiv Prepr. arXiv1901.09387*, 2019.

[79] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, “Reinforcement learning from imperfect demonstrations,” *arXiv Prepr. arXiv1802.05313*, 2018.

[80] A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov, and A. I. Panov, “Hierarchical deep q-network from imperfect demonstrations in minecraft,” *Cogn. Syst. Res.*, vol. 65, pp. 74–78, 2019.

[81] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun, “Functional object-oriented network for manipulation learning,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2655–2662.

[82] D. Paulius, A. B. Jelodar, and Y. Sun, “Functional object-oriented network: Construction & expansion,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–7.

[83] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[84] T. Yoshikawa, “Analysis and control of robot manipulators with redundancy,” in *Robotics research: the first international symposium*, 1984, pp. 735–747.

[85] T. Yoshikawa, “Manipulability of robotic mechanisms,” *Int. J. Rob. Res.*, vol. 4, no. 2, pp. 3–9, 1985.

[86] T. Yoshikawa, *Foundations of robotics: analysis and control*. Mit Press, 1990.

[87] S. S. Srinivasa, “Familiarization to Robot Motion,” pp. 366–373, 2014.

[88] A. D. Dragan and K. C. T. Lee, “Legibility and Predictability of Robot Motion,” 2013.