Robot Learning Assembly Tasks from Human Demonstrations

Zuyuan Zhu

Supervisor: Prof. Huosheng Hu

School of Computer Science and Electronic Engineering

University of Essex

This thesis is submitted for the degree of

Doctor of Philosophy

July 2020

I would like to dedicate this thesis to my loving parents.

Declaration

Acknowledgements

First and foremost, I would like to express my unique appreciation and thanks to my supervisor Professor Huosheng Hu who has been a tremendous mentor for me. I would like to sincerely thank him for encouraging my research, giving instant feedback, and allowing me to grow as a researcher. His constant and attentive guidance on my academic skills has been invaluable.

I especially thank my supervisory panel member, Dr. Vishwanathan Mohan, for his encouragement, support, excellent advice during my panel meetings. My thanks also go to Prof. Dongbing Gu for his useful suggestions on my early research. Besides, I would like to thank Mr. Robin Dowling who supports my PhD research, open day demos and GLA labs warm-hearted. Thanks to Mr. Ian Dukes for helping my English at the early stage.

I am grateful to all my colleagues and friends, namely Yaser N. Ibrahem Alothman, Penelope Roberts, Aakash Soni, Dingtian Yan, Shengnan Li, David Liao, Kai Zhang, Junpeng Shi, Geleite Xu, Wentong Liu, Jiejun Hu, Binxgin Cai, Lili Yan, Mingchen Sun, Tianyu Li, Yufan Lu and many academic visitors of Essex robotics lab during my PhD. My special appreciation goes to Qiang Liu for his help on my GLA teaching and career development, Jianjun Gui for his help on my research and settling down at Essex, Ruihao Li for his help on my research and gym workout.

Last but not least, I sincerely thank my parents, my brother and sister. They always have been loving, supporting and trusting me throughout my studies. Without their support, I wouldn't have made it.

Abstract

The industry robots are widely deployed in the assembly and production lines as they are efficient in performing highly repetitive tasks. They are mainly position-controlled and pre-programmed to work in well-structured environments. However, they cannot deal with dynamical changes and unexpected events in their operations as they do not have sufficient sensing and learning capabilities. It remains a big challenge for robotic assembly operations to be conducted in unstructured environments today.

This thesis research focuses on the development of robot learning from demonstration (LfD) for the robotic assembly task by using visual teaching. Firstly, the human kinesthetic teaching method is adopted for robot to learn an effective grasping skill in unstructured environment. During this teaching process, the robot learns the object's SIFT feature and grasping pose from human demonstrations. Secondly, a novel skeleton-joint mapping framework is proposed for robot learning from human demonstrations. The mapping algorithm transfers the human motion from the human joint space to the robot motor space so that the robot can be taught intuitively in a remote place. Thirdly, a novel visual-mapping demonstration framework is built for robot learning assembly tasks, in which, the demonstrator is able to teach the robot with feedback in real-time. Gaussian Mixture Model and Gaussian Mixture Regression are used to encode the learned skills for the robot. Finally, The effectiveness of the approach is evaluated with practical assembly tasks by the Baxter robot.

The significance of this thesis research is on its comprehensive insight of robot learning from demonstration for assembly tasks. The proposed LfD paradigm has the potential to

effectively transfer human skills to robots both in industrial and domestic environments. It paves the way for general public to use the robots without the need of programming skills.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This thesis focuses on the development of robot learning from demonstration (LfD) for robotic assembly task by using visual teaching. This chapter presents research motivations, questions, objectives, methodologies, thesis contributions and outline.

1.1 Motivation

Traditionally, the robot manipulators are widely used in the assembly lines and production lines in the industry as they are precise and robust, especially in highly repetitive tasks. However, the industrial robots are pre-programmed and operated in a well-structured environment. They are designed for specific tasks and unable to deal with dynamical changes in the production lines autonomously. Their programming processes can be both challenging and time-consuming when the tasks or environments change.

Inspired by the way humans acquire and learn skills from their observation of others, researchers have started to investigate how the robots can learn skills from observing human demonstrators performing complex manipulations. This kind of learning from demonstration (LfD) paradigm allows robots to master manipulating capabilities, like assembly skill in manufacturing. More importantly, LfD or imitation learning allows the general public to teach the robot new skills without tremendous programming. Anyone could buy a robot and teach it by practical demonstration, such as cleaning a room or cutting grass in a garden. LfD is a powerful and practical alternative to reinforcement learning (RL) for learning sequential decision-making policies. Different from RL which means the robot has to learn from the sparse rewards or manually specified reward function, LfD involves a supervisor that provides data to the learner. The robot then tries to learn the optimal policy by following and imitating the demonstrator's decisions. By this way, LfD speeds up the learning process for robots.

In the context of human demonstration, a policy of how to transfer the human skill to the robot is required. That is, the robot should have a way to see or feel the examples shown by the human demonstrator using cameras or other sensors. Furthermore, the demonstrating policy should be simple for the human to perform so that it can be easily deployed at home or in the industry. Therefore, an effective demonstrating strategy must be developed for the LfD paradigm.

Considering the assembly task, the objects to be assembled could be in various kinds of shapes. The picking pose of a robotic manipulator is fundamental in the assembly task, especially when the object is long. As human beings, we can use in-hand manipulation to adjust the pose of picked objects. However, for the robot, in-hand manipulation would introduce very complicated computing and mechanical problems which are challenging to be achieved. Therefore, it is necessary to investigate how to prompt robots to learn an effective and efficient picking strategy before starting the assembly task.

In summary, the motivation for this research is to develop and analyse novel efficient robot learning system for robotic assembly using learning from demonstration paradigm.

1.2 Research Questions

Baxter robot will be used in this thesis research. It is an industrial dual-arm robot, and each arm has 7 degrees of freedom. Rethink Robotics builds the robot in the USA. The robot has an animated face for interacting with human users. It is 3 feet tall and weighs 165 lbs without its pedestal. After fitted with its pedestal, its height is 5’10"–6’3" and its weight is 306 lbs. The Baxter robot can move smoothly and has force feedback to work with humans closely. Each wrist has a camera to monitor and guide the manipulation tasks. Figure 1.1 shows some detailed components and functions.

(1) What kind of demonstration method is suitable with Baxter?

Up to now, three main demonstration methods could be used for the Baxter robot, namely kinesthetic demonstration, motion-sensor demonstration, and tele-operation demonstration. More specifically, the kinesthetic demonstration is the most common method deployed on Baxter robots. In kinesthetic teaching, the robot directly records its movements which are guided by human demonstrator hand-by-hand. Hence, the data of motion trajectories are collected directly in Cartesian space or robotic joint space. In this way, the corresponding problem between the demonstrator and the imitator is avoided.

However, in some situations such as hand-by-hand is not available or not safe to operate, remote teaching would be a better choice to achieve a general and accurate demonstration. The other two demonstration methods have advantages on the point of remote teaching. The first method, tele-operation demonstration, uses a control box or delivers hints to present the examples to the robot but at the cost of additional tele-operation devices or vocal controlling system. Besides, the vocal method has its limitation which relies on the high intelligence of the vocal model. The second method, motion-sensor demonstration uses marker-based tracking devices to capture human motions, but the defects are apparent. The markers can not work alone. They need a complementary detecting device which usually takes big space. Furthermore, the markers are not convenient for the demonstrator to equip themselves.

It should be noted that Baxter robot is used as a benchmark to evaluate the proposed LfD paradigm. The proposed paradigm is not limited to the specific robot. In fact, any robotic

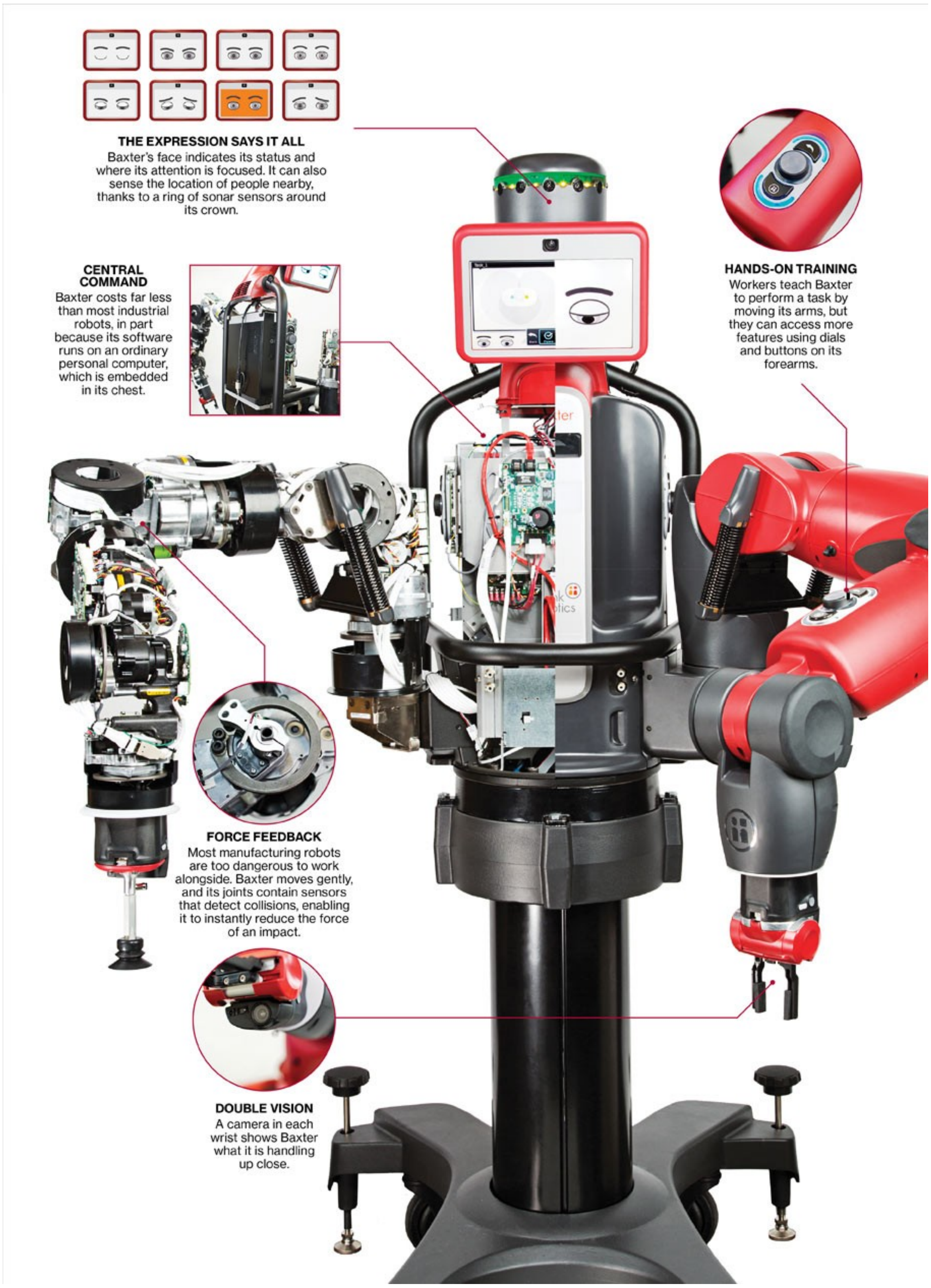

THE EXPRESSION SAYS IT ALL
Baxter's face indicates its status and where its attention is focused. It can also sense the location of people nearby, thanks to a ring of sonar sensors around its crown.
CENTRAL COMMAND
Baxter costs far less than most industrial robots, in part because its software runs on an ordinary personal computer, which is embedded in its chest.
HANDS-ON TRAINING
Workers teach Baxter to perform a task by moving its arms, but they can access more features using dials and buttons on its forearms.
FORCE FEEDBACK
Most manufacturing robots are too dangerous to work alongside. Baxter moves gently, and its joints contain sensors that detect collisions, enabling it to instantly reduce the force of an impact.
DOUBLE VISION
A camera in each wrist shows Baxter what it is handling up close.

Fig. 1.1 The technical details of Baxter robot. Source: [58]

manipulator with multiple joints which enable the robot to explore in their workspace is eligible to use the proposed LfD paradigm.

(2) How to demonstrate without cutting the corresponding mechanism?

As the demonstrator's body differs from the robot's body in architecture, the demonstrator's body scheme does not match with the robot's body scheme, which results in the corresponding problem. This correspondence problem may make the robot fail to understand the demonstrator's motions. Besides, the direct imitation of human motions can also be used for the robot to imitate human dances. To build a map between the human skeleton joints and robot motor joints, a new transformable mechanism need to be proposed.

(3)How to teach the robot effective grasping pose without using additional assistant devices in the robotic assembly task?

In the robotic assembly task, the robot needs to grasp the object from the desk before assembling motion. As adult human beings, we have no difficulties in grasping objects around us, because we have learned how to grasp the object from the right position. For example, to pick up a mug, we choose to grasp the cup handle; to pick up a pen, we choose to grasp the middle of the pen. Similarly, the robot needs to learn a strategy to pick objects from effective poses like the human just use the visual sensor and hand-eye coordination ability. Baxter robot comes with a camera embedded in each wrist. It is worth to investigate how to teach the robot the effective grasping ability for robotic assembly task without introducing additional assistant devices.

(4) How to improve the quality of demonstrating with few demonstration times?

In a robot learning from demonstration system, the human demonstrator performs the skills as examples for robot learning. The number of examples used for training the robot should not be enormous as the LfD requires. Therefore, the quality of the demonstrated examples should be emphasized. A real-time feedback mechanism can be considered to help the demonstrator guide and correct their actions. Then, we consider what kind of sensor and algorithm can be used for real-time tracking and feedback. Besides, we need to consider how to incorporate the sensor information with the learning paradigm.

1.3 Research Objectives

Based on the motivation and research questions described in the previous sections, this thesis research aims to develop a novel robot learning system for robotic assembly tasks. It also provides comprehensive insights on the system's performance by conducting theoretical and experimental analyses of learning from demonstration. The proposed research will focus on the following specific objectives:

- **To develop an effective human demonstration mechanism for robot imitation (In chapter 3 and 5)**.
  In LfD, the way that the human demonstration conducted plays a key role for the robot to learn from the human demonstrator effectively. In the learning of grasping pose research, the thesis focuses on the end position (palm or hand) of the human arm. In this way, the correspondence issue between demonstrator and robot is avoided, only the kinematic information of endpoint joint of demonstrator's hand is taught to the robot. During the human visual demonstration, the human teacher stands in front of the robot, his or her arm joint position, velocity, and acceleration are captured by

Kinect 3D camera. To further improve the efficiency of teaching, the robot repeats the motion following the human demonstrator, the teacher gives feedback immediately and improves the demonstration, such as slow down the motion, restrict the motion within robot's workspace, try a promoted motion trajectory and so on, according to the learning performance of the robot.

- **To develop a demonstrating method via visual teaching (In chapter 4).**
  In order to teach the robot from a remote position without wearing additional assisting tracking devices, a new demonstrating method needs to be developed. The new teaching method does not rely on the controller to deliver the human's motion as the existed tele-operation demonstration does. This kind of simplicity and wireless features improve the efficiency of the teaching process and extend the application scope of the robot learning from human demonstrations. The motion of human upper limbs will be tracked and extracted as features for robot learning. The elbow and shoulder motions will also be captured, not only the hand's motions. The goal is to keep the corresponding relationship between the demonstrator and the robot, while the robot still can learn from the human teacher.

- **To develop a new performance metric to evaluate the autonomous learning system (In chapter 5).**
  The developed autonomous skill learning system needs to be tested on a specific task, such as an assembly task. The learning system has the potential to be used in real industry assembly production, such as assembly a chair. However, this research focuses on developing an autonomous learning system for the Baxter robot to conduct assembly tasks in the world of MEGA blocks. The Baxter robot uses the skills learned from a human demonstrator to complete an assembly task. The skills include approach/retreat, grasp, and insertion. A metric function based on the generalized joint angle trajectories, the generalized hand path will be used to evaluate the performance of the Baxter robot.

1.4 Research Methodology

In order to realize the research objectives outlined in the previous section, the thesis research has adopted the following research methodologies:

- **Kinesthetic guiding**

  Kinesthetic guiding is a simple way to conduct the robot learning from human demonstrations. It maintains the advantages of LfD and also significantly reduces the system cost in terms of both demonstrating and learning. For solving the problem of finding an appropriate picking pose in the robotic assembly task, the proposed method learns the best picking pose by extracting the object's SIFT (scale-invariant feature transform) features from the human kinesthetic demonstration. Differing from the traditional kinesthetic teaching that records the whole demonstrated trajectory, the proposed method learns the key point of the trajectory in a robotic assembly example, i.e. the assembly location of the two piece works. In addition, the assembly process is assured by the force sensor embedded in the robot hand, which is controlled by a force threshold.

- **Visual mapping:**

  Visual mapping is another way to conduct the robot learning from human demonstrations. Visual sensors are deployed to allow a robot to watch and learn the assembly skills demonstrated by a human. Since the obtained visual demonstrating data contains noise and latency, Holt's two parameter exponential smoothing algorithm is used to reduce the noise and latency. As mentioned before, human movements can not be directly transferred to the robot motions. A skeleton-joint mapping algorithm is proposed to transfer the motion from human joint space to robot task space. Moreover, an object tracking algorithm is deployed to segment the demonstrating trajectories and assist in the implementation of robotic assembly tasks.

- **Imitation learning:**

  Imitation learning means the robot imitates the human teacher and learns the specific skill demonstrated by the teacher. The observed examples are converted from the human task space to the robot joint space. In another word, the human motions are transferred to robot trajectories. Gaussian distributions are used to build models of the learned skills, and Gaussian Mixture Regression (GMR) built upon the previous distributions are used for robot motion planning in the reproducing stage.

- **Experimental verification:**

  The development of a novel autonomous skill learning system allows an industrial robot to conduct assembly tasks. This is a potentially fruitful research endeavour with a strong science base for skill learning, as well as wide real-world applications such as assembly or similar manipulative applications. Experiments are carried out in both simulation and real robot in order to validate the proposed approaches. The results could provide a better understanding of the feasibility and performance of the proposed solutions, as well as how they surpass traditional solutions in both industrial and domestic applications.

1.5 Thesis Contributions

The research work presented in this thesis contributes to a number of new solutions for Learning from Demonstration (LfD), with a particular focus on applications of robotic assembly tasks. The following list highlights the most important contributions:

- A LfD paradigm is proposed based on key positions and object features. Instead of traditionally imitating the trajectories demonstrated by a human, the robot learns the effective grasping poses and assembly positions of the Peg-in-Hole (PiH) task through the kinesthetic teaching. Compared with the existing works that use predefined objects for robotic assembly tasks, the proposed method improves the robot's adaptive ability by learning from human demonstration with unstructured objects in PiH task.

- A teaching method is proposed for the robot to observe human demonstrations by a visual sensor so that the assembly skill demonstrated by a human can be effectively mapped from task space to configuration space for robot learning. Compared with existing works that use low-accuracy computer-vision-based human tracking method or marker-based tracking with additional devices, the proposed method uses a Kinect-based skeleton-joint mapping with improved accuracy of human motion tracking and without additional markers attached to human demonstrator.
- A real-time feedback feature is introduced for the robot to learn assembly skill from observing human demonstrating the manipulating skills. Compared with the existing works that requires multiple or enormous demonstrations for robot learning assembly tasks, the proposed LfD paradigm improves the demonstrating quality by using the real-time simulating robot to provide feedback. Furthermore, the proposed method has comprehensive advantages over kinesthetic, motion-sensor, and teleoperated demonstration in robot learning assembly tasks.

1.6 Thesis Outline

This thesis is organised into 6 chapters. Chapter 1 is an overview of the thesis. Chapter 2 is the background of the research work. Chapter 3, 4, and 5 are the main research work conducted in this thesis research. Finally, Chapter 6 concludes the thesis. The details of the individual chapters are outlined here:

- Chapter 2 reviews some previous work related to this research. The main focus is placed on how to demonstrate the example behaviours to the robot in assembly operations, and how to extract the manipulation features for robot learning and the generation of imitative behaviours. Various metrics are analysed to evaluate the performance of robot imitation learning. Specifically, the application of LfD in robotic assembly is a focal point in this chapter.

- Chapter 3 presents a novel algorithm for Peg-in-Hole operations by robot learning from human demonstration. The learning of robotic assembly task is divided into two phases: teaching and reproduction.The human demonstrator teaches a robot to grasp an object from the effective positions and orientations. During the reproduction phase, the robot uses the learned knowledge to reproduce the grasping manipulation autonomously. The robustness of the robotic assembly system is evaluated through a series of grasping trials. A dual-arm Baxter robot is used to perform the Peg-in-Hole tasks, and the results are given to verify the proposed approach.
- Chapter 4 proposes a demonstration method based on human skeleton mapping. A smoothing algorithm based on Holt's two parameter exponential is designed for reducing the latency of human skeletons data. Holt's model uses three separate equations that work together to generate a final forecast. Moreover, a mapping algorithm is designed for demonstration teaching. The human's hands joints, torso joint, and head joint are used to calculate the angles between them. A series of equations are proposed to map these angles into the robot's motors. The mapping algorithm is tested by some basic gestures, such as lateral raise, front up raise, and front low raise.
- Chapter 5 shows the experimental results on the robotic assembly task based on the visual teaching. The experimental environment and testing platforms are introduced. Then the scenario of the robotic assembly experiments is presented. The modelling process of GMM and GMR is investigated and analysed. The visual-based teaching is tested with robotic assembly tasks on the Baxter robot.

- Chapter 6 summarizes the research work conducted in this thesis research, including significant research achievements so far, the discussion of pros and cons on the proposed research, and some brief ideas for the future work.

Chapter 2

Background and Related Research

2.1 Introduction

2.1.1 Robotic assembly

The industrial robots that are currently deployed in assembly lines are position-controlled and programmed to follow desired trajectories for conducting assembly tasks [63, 47]. These position-controlled robots can handle known objects within the well-structured assembly lines very well, achieving highly accurate position and high velocity. However, they cannot deal with any unexpected changes in assembly operations, and need tedious reprogramming to adapt to new assembly tasks.

Knepper et al. investigated a multi-robot coordinated assembly system for furniture assembly [57]. The geometry of individual parts was listed in a table so that a group of robots can conduct parts delivery or parts assembly collaboratively. For the modelling and recognition of the furniture parts, the object's representation was predefined in CAD files so that the correct assembly sequence can be deduced from geometric data. Suárez-Ruiz and Pham proposed a taxonomy of the manipulation primitives for bi-manual pin insertion, which was only one of the critical steps in the autonomous assembly of an IKEA chair [119].

In general, a typical robot-assembly operation involves operating with two or more objects/parts. Each part is a subset of the assembly. Assembly aims to compute an ordering of operations that brings individual parts so that a new product appears. Examples of assembly tasks can be summarised below.

- Peg-in-hole: A robotic gripper grabs the peg and inserts it in a hole. Peg-in-hole is the most essential and representative assembly task that has been widely researched [135, 61, 56, 62, 94, 53, 2, 122, 109, 91, 139].
- Slide-in-the-groove: A robot inserts a bolt fitting inside a groove and slides the bolt to the desired position where the bolt is to be fixed [98].
- Bolt screwing: A robot screws a self-tapping bolt into a material of unknown properties, which requires driving a self-tapping screw into an unstructured environment [98, 72, 71, 84].
- Chair assembly: A robot integrates chair parts together with a fastener [39, 88, 116].
- Pick-and-place: A robot picks up an object as the base and places it down on a fixture [132, 71, 70].
- Pipe connection: A robot picks and places two union nuts on a tube [71].

As typical robot-assembly operations need to contact the workpieces to be assembled, it is crucial to estimate the accompanying force–torque profiles besides position and orientation trajectories. To learn the execution of assembly operations, a robot needs to estimate the pose of the workpieces first, and an assembly sequence is then generated by learning from human demonstration. For some particular objects appearing in the assembly workspace, some specialised grippers should be designed to grab these parts with various shapes and acquire force–torque data. In particular, during a screwing task, the material to be screwed is unstructured, which makes the control of rotating angles more complicated.

Considering these challenges, robotic assembly remains one of the most challenging problems in the field of robotics research, especially in unstructured environments. In contrast, humans have excellent skills to perform assembly tasks that require compliance and force control. This motivates us to review the current research of learning from demonstration (LfD) in robotics assembly and its potential future directions.

2.1.2 Learning from demonstration

Traditional robots require users to have programming skills, which makes robots beyond the reach of the general public. Nowadays, robotics researchers have worked on a new generation of robots that could learn from demonstration and have no need of programming. In other words, these new robots could perceive human movements using their sensors and reproduce the same actions that humans do. They can be used by the general public who have no programming skills at all.

The term of learning from demonstration (LfD), or learning by imitation, was analysed in depth by Bakker and Kuniyoshi [11], who defined what imitation is and what robot imitation should be. From a psychological point of view, Thorndike defined imitation as learning to do an act being witnessed [124]. Based on this, Bakker indicated that imitation takes place when an agent learns a behaviour from observing the execution of that behaviour by a teacher [11]. This was the starting point for establishing the features of robot imitation: (i) adaptation; (ii) efficient communication between the teacher and the learner; (iii) compatibility with other learning algorithms; and (iv) efficient learning in a society of agents [105].

In addition, three processes in robot imitation have been identified, namely sensing, understanding and doing. In other words, they can be redefined as: observe an action, represent the action and reproduce the action. Figure 2.1 shows these three main issues and all the associated current challenges in robot imitation.

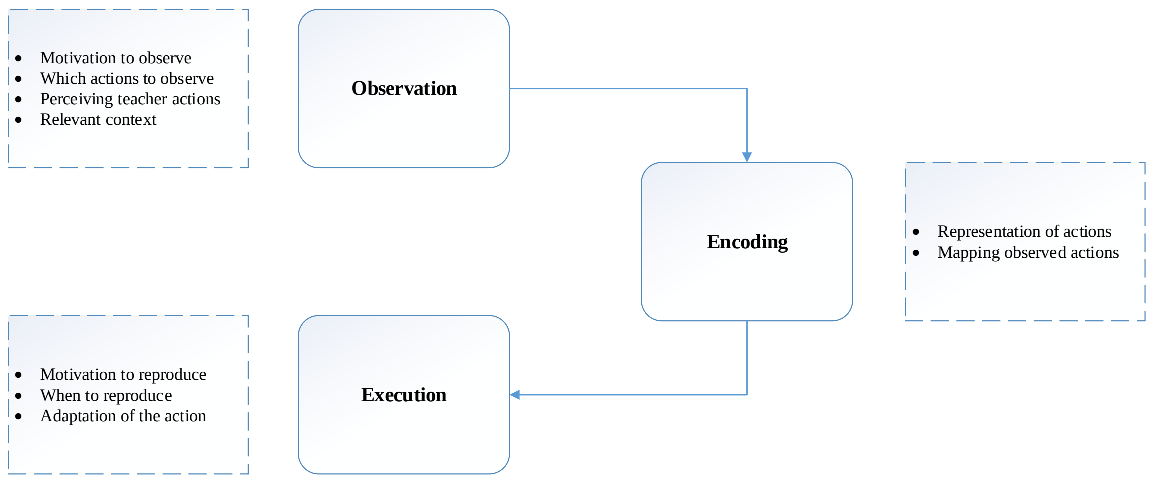


Fig. 2.1 The three main phases in imitation learning according to [11].

Mataric et al. defined the imitation learning from a biological perspective, that is, a behaviour-based control [85]. They indicated that key challenges are how to interpret and understand observed behaviours and how to integrate the perception and motion-control systems to reconstruct what is observed. In other words, there are two essential tasks in imitation learning: (i) to recognise the human behaviour from visual or sensory input; (ii) to find methods for structuring the motor-control system for general movements and imitation learning capabilities. Current approaches to representing a skill can be broadly divided into two trends: (i) trajectories encoding–a low-level representation of the skill, taking the form of a nonlinear mapping between sensory and motor information; (ii) symbolic encoding–a high-level representation of the skill that decomposes the skill in a sequence of action–perception units [12].

In general, to achieve robot learning from demonstration, we need to address three challenges: the correspondence problem, generalisation, and robustness against perturbation [96]. Firstly, the correspondence problem means how to map links and joints from a human to a robot. Secondly, learning by demonstration is feasible only if a demonstrated movement can be generalised, such as different goal positions. Finally, we need robustness against perturbation: exactly replaying an observed movement is unrealistic in a dynamic environment, in which obstacles may appear suddenly.

Most assembly tasks can be represented as a sequence of individual movements with specific goals, which can be modelled as dynamic movement primitives (DMPs, explained in Section 2.4.2), where DMPs are the fundamental blocks of the LfD architecture. Besides, LfD has been suggested recently as an effective means to speed up the programming of learning processes from the low-level control to the high-level assembly planning [64]. Therefore, LfD is a preferable approach for robotic assembly.

Recently, LfD has been applied to robotic assembly [5, 118, 114]. Takamatsu et al. introduced LfD to robotic assembly and proposed a method for recognising assembly tasks by human demonstration [121]. They defined sufficient sub-skills and critical transition-assembly tasks, and implemented a peg-insertion task on a dual-arm robot with a real-time stereo–vision system. The assembly task is completed with two rigid polyhedral objects recognised by a conventional 6-DOF (degree of freedom) object-tracking system. The assembly tasks are encapsulated into chains of two-object relationships, such as maintaining, detaching and constraining. In order to make the process of assembly task smooth, critical transitions are also defined.

The human–robot cooperation in assembly tasks reduces the complexity of impedance control. Rozo et al. worked one step forward to achieve a multimodal LfD framework, in which a robot extracted the impedance-based behaviour of the teacher recording both force patterns and visual information in a collaborative table-assembly task [104]. It should be noted that the experiments did not take into account the independence and autonomy of the robot. For the modelling of assembly task, Dantam et al. transferred the human demonstrations into a sequence of semantically relevant object-connection movements [33]. Then, the sequence of movements is further abstracted as motion grammar, which represents the demonstrated task. It should be noted that the assembly task is implemented in simulation.

Different from the previous survey of learning from demonstration [8], this thesis research is mainly focused on the applications of LfD techniques in robotic assembly. The rest of the chapter is organised as follows. Section 2.2 outlines the major research problems in robotic assembly, which are classified into four categories. The key issue of how to demonstrate the assembly task to a robot is explained in Section 2.3, and how to abstract the features of the assembly task is illustrated in Section 2.4. Then, we examine the question of how to evaluate the performance of the imitator in Section 2.5. Finally, a brief conclusion and discussion on the open research areas in LfD and robotic assembly are presented in Section 2.6.

2.2 Research Problems in Robotic Assembly

Robotic assembly needs a high degree of repeatability, flexibility and reliability to improve the automation performance in assembly lines. Therefore, many specific research problems have to be resolved in order to achieve automated robotic assembly in unstructured environments. The robot software should be able to convert the sequences of assembly tasks into individual movements, estimate the pose of assembly parts, and calculate the required forces and torques. As there are many challenges in robotic assembly, this section will be focused on four categories which are closely related to LfD: pose estimation, force estimation, assembly sequences, and assembly with screwing.

2.2.1 Pose estimation

In assembly lines, it is often indispensable that the position and orientation of workpieces are predetermined with high accuracy. The vision-based pose estimation is a low-cost solution to determine the position and orientation of assembly parts based on point cloud data [135]. The texture projector could also be used to acquire high-density point clouds and help the stereo matching process. Abu-Dakka et al. used a 3D Kinect for capturing 3D scene data, and the pose of the objects could be estimated based on the point cloud data [2].

Before pose estimation, an object should be recognised by using local features, as this is an effective way for matching [81]. Choi et al. developed a family of pose-estimation algorithms that use boundary points with directions and boundary line segments along with the oriented surface points to provide high accuracy for a broad class of industrial parts [30]. However, in the cluttered environments where the target objects are placed with self-occlusions and sensor noise, assembly robots require a robust vision to recognise and locate the objects reliably. Zeng et al. used a fully convolutional neural network to segment and label multiple views of a scene, and then fit predefined 3D object models to the segmentation to obtain the 6D object pose (3D position: x, y, z; and orientation: yaw, pitch, roll) rather than 3D location [137].

However, the vision-based pose estimation has limitations due to the limited resolution of the vision system. Also, in the peg-in-hole task, the peg would usually occlude the hole when the robot approaches the hole. Therefore, vision-based pose estimation is not sufficient for the high-accuracy assembly tasks in which two parts occlude each other. If the camera is mounted on the robotic arm, the occlude problem can be eliminated. However, additional sensory data is needed to estimate the camera pose [111].

To correct the pose of assembly parts, Xiao et al. devised a nominal assembly-motion sequence to collect data from exploratory complaint movements [108]. The collected data are then used to update the subsequent assembly sequence to correct any errors in the nominal assembly operation. Nevertheless, the uncertainty in the pose of the manipulated object should be further addressed in future research.

2.2.2 Force estimation

In assembly tasks, force control could provide stable contact between the manipulator and the workpiece [118, 80, 131, 55, 135, 115, 61, 56, 117]. As human operators perform compliant motions during assembly, the robot should acquire the contact forces that occur in the process of assembly. During the task execution, the robot learns to replicate the learned forces and torques rather than positions and orientations from the trajectory. The force information may also be used to speed up the subsequent operations of assembly tasks [62, 94, 90, 98, 122, 109, 91].

The force applied to the workpiece is usually detected by using the force sensor on the robot end-effector, or using force sensors on each joint of the robot arm, for example, in the DLR (German Aerospace Centre) lightweight arm. The problem of using inside force sensors is that the measured forces must be compensated for disturbance forces (for example, gravity and friction) before use. The feedback force is then introduced to the control system that generates a corresponding transnational/rotational velocity command on the robot manipulator to push the manipulated workpiece.

To enable a robot to interact with the different stiffness, Peternel et al. used the impedance-control interface to teach it some assembly tasks [97]. The human teacher controlled the robot through haptic and impedance-control interfaces. The robot was taught to learn how to perform a slide-in-the-groove assembly task where the robot inserted one part fitted with a bolt into another part. However, the low movement variability did not necessarily correspond to the high impedance force in some assembly tasks such as slide-in-the-groove tasks.

The dedicated force–torque sensors can easily acquire the force, but may not be easily able to mount to the robot hand. Alternatively, Wahrburg et al. deployed motor signals and joint angles to reconstruct the external forces [131]. It should be noted that force/torque estimation is not a problem and has been successful in the traditional robotic assembly in structured environments. However, it is a problem for robotic assembly in unstructured environments. Force/torque estimation is not only about acquiring force/torque data but also about using these data for a robot to accommodate its interaction with the different stiffness.

2.2.3 Assembly sequence

As an appropriate assembly sequence helps minimise the cost of assembly, the assembly sequences are predefined manually in the traditional robotic assembly systems. However, the detailed assembly sequence defined in [115] significantly holds back the automation of next-generation assembly lines. In order to achieve an efficient assembly sequence for a task, an optimisation algorithm is required to find the optimum plan. Bahubalendruni et al. found that assembly predicates, i.e. some sets of constraints, have a significant influence on optimal assembly-sequence generation [10].

Wan and Harada presented an integrated assembly and motion-planning system to search the assembly sequence with the help of a horizontal surface as the supporting fixture [132]. Kramberger et al. proposed two novel algorithms that learned the precedence constraints and relative part-size constraints [62]. The first algorithm used precedence constraints to generate previously unseen assembly sequences and guaranteed the feasibility of the assembly sequences by learning from human demonstration. The second algorithm learned how to mate the parts by exploratory executions, i.e. learning-by-exploration.

Learning assembly sequences from demonstration can be tailored for general assembly tasks [70, 62, 114, 135, 79, 118]. Mollard et al. proposed a method to learn from the demonstration to automatically detect the constraints between pairs of objects, decompose sub-tasks of the demonstration, and learn hierarchical assembly tasks [88]. In addition, the learned sequences were further refined by alternating corrections and executions. It should be noticed that the challenge in defining assembly sequence is how to automatically extract the movement primitives and generate a feasible assembly sequence according to the manipulated workpieces.

2.2.4 Assembly with screwing

Screwing is one of the most challenge sub-tasks of assembly and requires robust force control so that the robot could screw a self-tapping bolt into a material of unknown properties. The self-tapping screw driving task consists of two main steps. The first step is to insert the screwdriver into the head of a bolt. The contact stiffness is kept at a constant value such that the screwdriver keeps touching the head of the bolt. The second step is to fit the screwdriver into the screw head and rotate it for a specific angle to actuate the bolt into the unstructured objects.

To measure the specific force and angle needed for actuating the screwdriver, Peternel et al. used a human demonstrator to rotate the screwdriver first and captured the correspondence angle by a Haptic Master gimbal device [98]. The information was then mapped to the end-effector rotation of the robot. It should be noted that the torque used to compensate the rotational stiffness is unknown, so the demonstrator manually commands high rotational stiffness through the stiffness control to make the robot exactly follow the demonstrated rotation.

There are significant uncertainties that existed in the screwing task, such as the screwdriver may not catch the head of the bolt correctly. In fact, as the complexity of the task increases, it becomes increasingly common that tasks may be executed with errors. Instead of preventing errors from happening, Laursen et al. proposed a system to automatically handle certain categories of errors through automatic reverse execution to a safe status, from where forward execution can be resumed [72]. The adaptation of an assembly action is essential in the execution phase of LfD, as presented in Figure 2.1. Besides, the adaptation of uncertainties in robotic assembly still needs further investigation.

In summary, pose estimation, force estimation, assembly sequence, and assembly with screwing have been partially resolved in limited conditions, but are still far away from the industrial application. Most of the current assembly systems are tested in relatively simple tasks, like peg-in-hole. In addition, a more robust and efficient control strategy is needed to deal with complicated assembly tasks in an unstructured environment.

2.3 Demonstration Approach

Robot learning from demonstration requires the acquisition of example trajectories, which can be captured in various ways. Alternatively, a robot can be physically guided through the desired trajectory by its operator, and the learned trajectory is recorded proprioceptively for demonstration. This method requires that the robot is back drivable [44, 99] or can compensate for the influences of external forces [86, 4, 49]. The following subsections discuss various works that utilise these demonstration techniques.

2.3.1 Kinesthetic demonstration

The advantage of kinesthetic guiding is that the movements are recorded directly on the learning robot and do not need to be first transferred from a system with different kinematics and dynamics. During the demonstration movement, the robot's hands are guided by a human demonstrator [26, 22, 136, 84]. It should be noted that kinesthetic teaching might affect the acquired forces and torques, especially when joint force sensors are used to estimate forces and torques for controlling of assembly tasks on real robots. Besides, if the manipulated objects are large, far apart, or dangerous to deal with, kinesthetic guiding can be problematic.

Figure 2.2 shows that the robot was taught through kinesthetics in gravity-compensation mode, i.e. by the demonstrator moving its arm through each step of the task. To achieve this, the robot motors were set in a passive mode so that the human demonstrator could move each limb. The kinematics of each joint motion were recorded at a rate by proprioception during the demonstration. The robot was provided with motor encoders for every degree-of-freedom. By moving its limb, the robot "sensed" its motion by registering the joint-angle data provided by the motor encoders. The interaction with the robot was more playful than using a graphical simulation, as the interacting enabled the user to feel the robot's limitation in its real-world environment implicitly.

In [78], example tasks were provided to the robot via kinesthetic demonstration, in which the teacher physically moved the robot's arm in the zero-gravity mode to perform the task, and used the button on the cuff to set the closure of the grippers. By pushing the button on the arm, the recording began, and the teacher started to move the same arm to perform manipulation. When the manipulation was done, the teacher pressed the button again to pause the recording. The teacher simply repeated the steps to continue the manipulation with another hand and the recording.

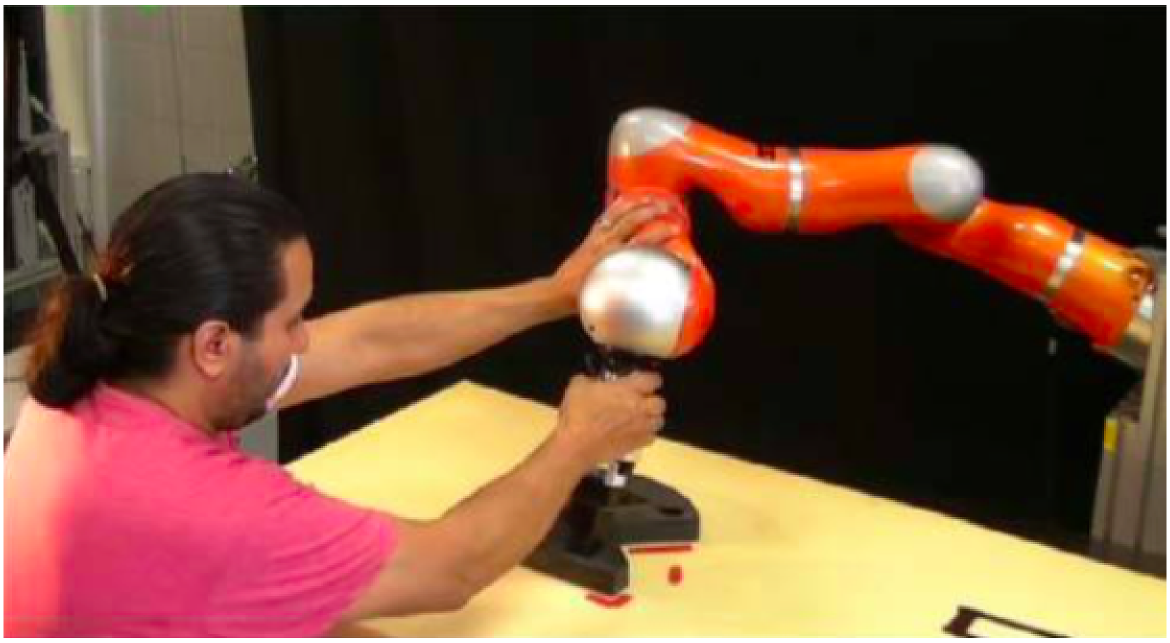


Fig. 2.2 Human demonstrator teaches the Kuka LWR arm to learn peg-in-hole operations by kinesthetic guiding [2].

Figure 2.2 shows that a human demonstrator teaches the Kuka LWR arm to learn peg-in-hole operations by kinesthetic guiding [2]. The signals of arm activation and the grippers' state during the demonstration were recorded to segment the tool-use process into sequential manipulation primitives. Each primitive was characterised by using a starting pose, an ending pose of the actuated end-effector, and the sequence of the poses. The primitives are learned via the DMPs framework. These primitives and the sequencing of the primitives constitute the model for tool use. The decomposed primitives are used to infer the underlying sequential structure of the demonstrated task [84]. The related sensory data of the end-effector, like position, orientation and force, are used for sequencing. By linking DMPs with expected sensory data and comparing current sensory values with expected ones, the best match will be chosen.

2.3.2 Motion-sensor demonstration

The limb movements of a human demonstrator are very complex and challenging to capture. Computer vision could be used in capturing human-demonstrator motion with a low accuracy[87]. In contrast, the optical or magnetic marker-based tracking systems can achieve high accuracy and avoid visual overlapping of computer vision [103, 101, 128, 106]. Therefore, marker-based tracking devices are deployed to track the manipulation movements of a human demonstrator for assembly tasks.

Skoglund et al. presented a method for imitation learning based on fuzzy modelling and a next-state-planner in the learning-from-demonstration framework [113]. A glove with LEDs at its back and a few tactile sensors on each fingertip was used in the impulse motion-capturing system. The LEDs were used to compute the orientation of the wrist, and the tactile sensors were to detect contact with objects. Alternatively, a motion sensor can be used for tracking instead of LEDs.

Colour markers are a simple and effective motion tracking technique used in the motion-sensor demonstration. Acosta-Calderon and Hu proposed a robot-imitation system, in which the robot imitator observed a human demonstrator performing arm movements [3]. The colour markers on the human demonstrator were extracted and tracked by the colour-tracking system. The obtained information was then used to solve the correspondence problem as described in the previous section. The reference point used to achieve this correspondence is the shoulder of the demonstrator, which corresponded to the base of the robotic arm.

To capture human movement with whole-body motion data, a total of 34 markers were used in a motion-capture setup [67]. During the data-collection process, a sequence of continuous movement data was obtained, for example, a variety of human walking motions, a squat motion, kicking motions and raising arm motions. Some of the motions are discrete, and others are continuous. Therefore, the learning system should segment the motions automatically. The segmentation of full-body human-motion patterns was studied in [66]. The human also observed the motion sequence, manually segmented it and then labelled these motions. Note that the motions segmented by the human were set as ground truth, and no further criteria were used.

In Figure 2.3, a motion sensor mounted in the glove is used for tracking the 6D pose (the position and orientation) relative to the transmitter; the robot then receives a transformed pose on a 1:1 movement scale. The peg-in-hole experiments show that the data glove is inefficient compared with using an external device during the teleoperation process. In Figure 2.4, both hand-pose and contact forces are measured by a tactile glove. In the robotic assembly, the force sensor is essential as the task requires accurate control of force. Therefore, the motion sensor and force sensor are usually combined.

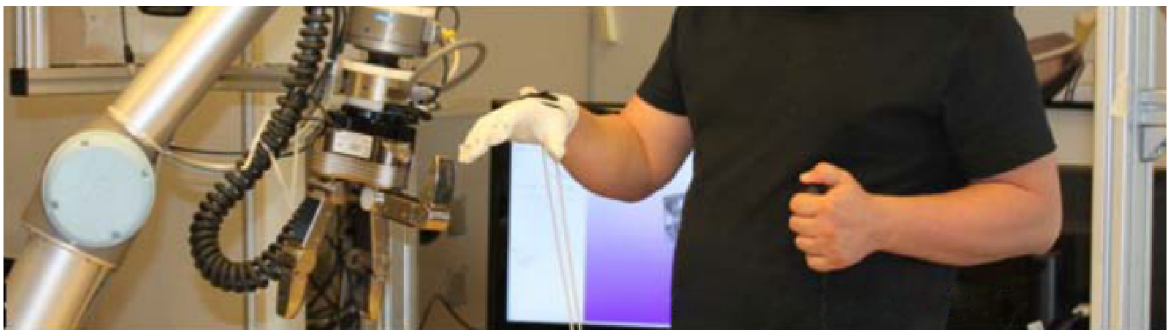


Fig. 2.3 The motion sensor is integrated in the glove at the back of the hand [65].

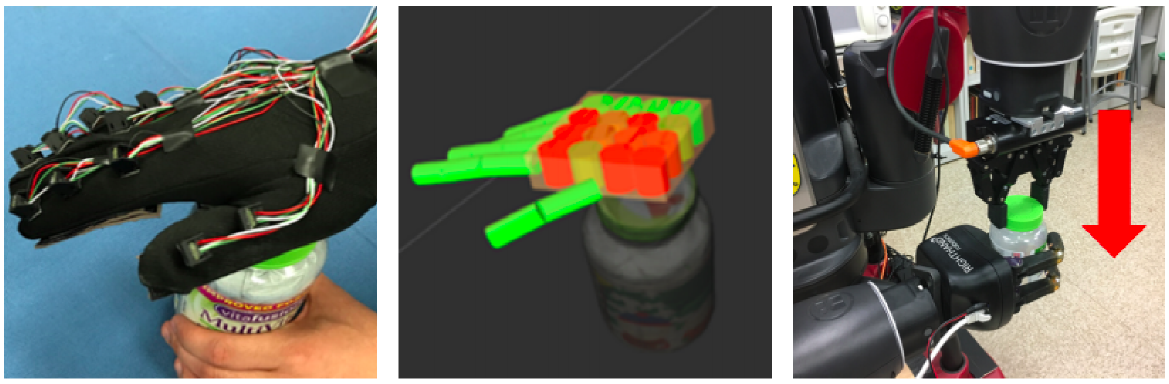


Fig. 2.4 A tactile glove is used to reconstruct both forces and poses from human demonstrations, enabling the robot to directly observe forces used in demonstrations so that the robot can successfully perform a screwing task: opening a medicine bottle [41].

2.3.3 Teleoperated demonstration

During the teleoperated demonstration, a human operator uses a control box or delivers hints to control a robot to execute assembly tasks, and the robot keeps recording data from its own sensors, see Figure 2.5. Similar to the kinesthetic demonstration, the movements performed by the robot are recorded directly on the robot, that is, the mapping is direct, and no corresponding issue exists. Learning from teleoperated demonstrations is an attractive approach to controlling complex robots.

Teleportation has the advantage of establishing an efficient communication and operation strategy between humans and robots. It has been applied in various applications, including remote control of a mobile robotic assistant [127, 31, 83], performing an assembly task [65, 28, 16], performing a spatial-positioning task [7], demonstrating grasp pre-shapes to the robot [120], transmitting both dynamic and communicative information on a collaborative task [24], and picking and moving tasks [40].

In assembly tasks, when a human demonstrator performs the assembly motions, the pose information is fed into a real-time tracking system so that the robot can copy the movements of the human demonstrator [109]. The Robonaut, a space-capable humanoid robot from NASA, was controlled by a human through full-immersion teleoperation [100]. Its stereo

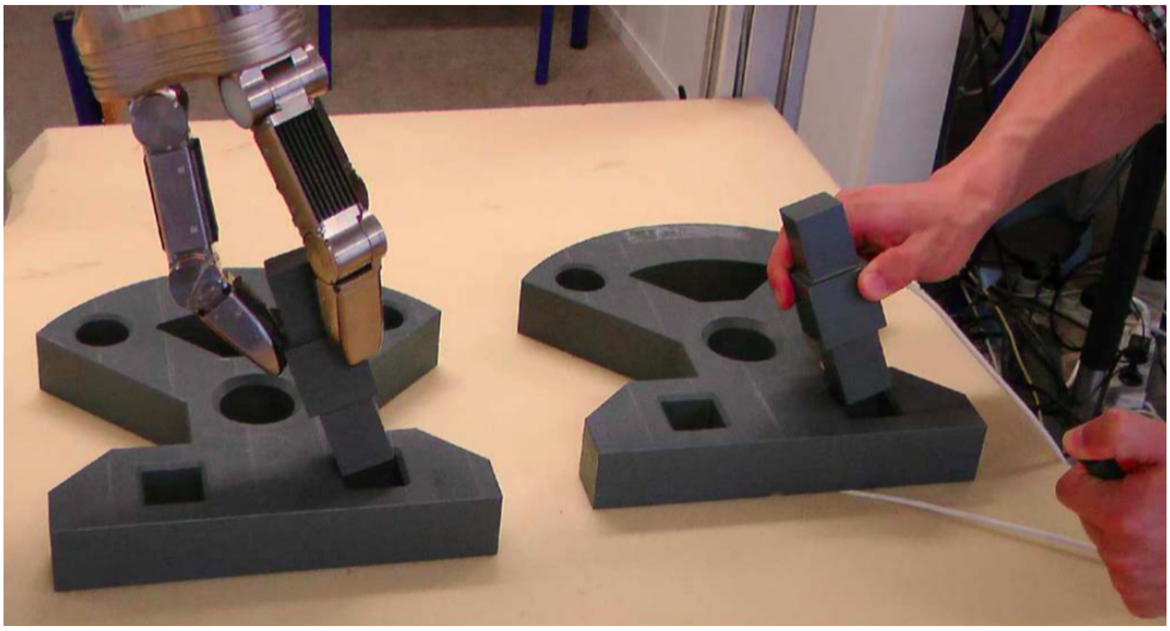
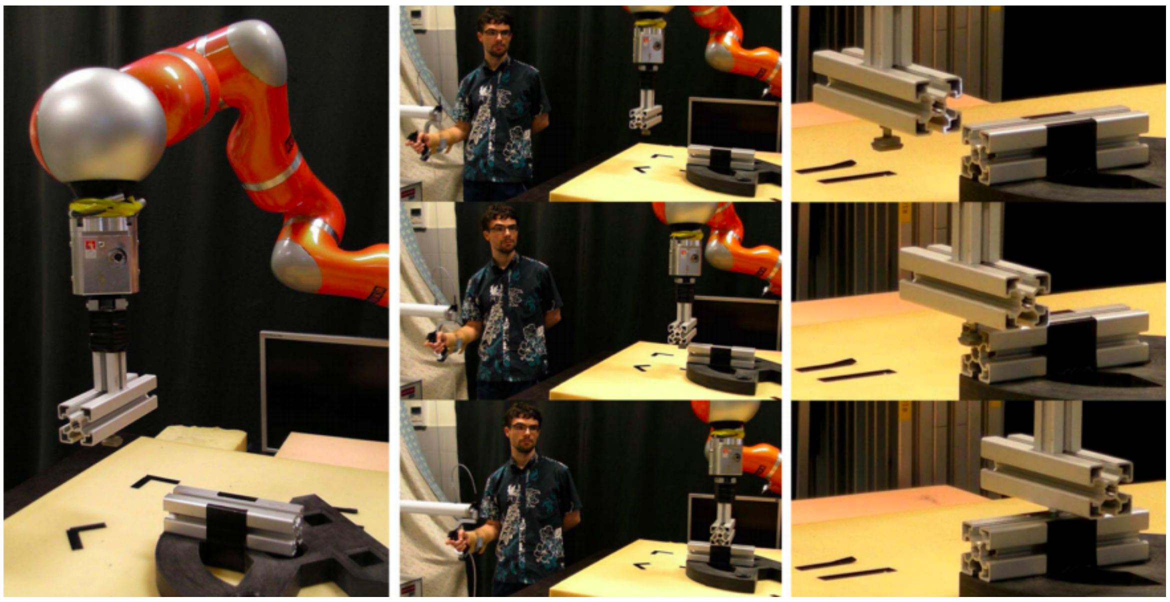


Fig. 2.5 Human demonstrator performs the peg-in-hole task, in teleoperation mode, as the robot copies the movements of the human [109].

camera and sound sensor transmit vision and auditory information to the teleoperator through a helmet worn on his or her head. Although the full-immersion teleoperation is a good strategy for the Robonaut, its dexterous control is very tedious and tiring.

During teleoperation, the human operator usually manipulates the robot through a controller, standing far away from the robot. Figure 2.6 shows that the human demonstrator teaches the robot to perform the slide-in-the-groove task, as shown in the middle of Figure 2.6a, and then the robot autonomously repeats the learned skill, as shown in the right of Figure 2.6a. For the bolt-screwing task, as shown in Figure 2.6b, DMPs are used to encode the trajectories after demonstrations. Sometimes the controller can be a part of the robot itself.

Tanwani et al. applied teleoperation in the robot- learning process where the robot is required to do the tasks of opening/closing a valve and pick–place an object [123]. The human operator held the left arm of the robot and controlled its right arm to receive visual feedback


(a) Experimental setups for the slide-in-the-groove assembly task, the human demonstrator teleoperates using a remote controller (left and middle) and the robot learns to repeat the slide-in-the-groove task (right).

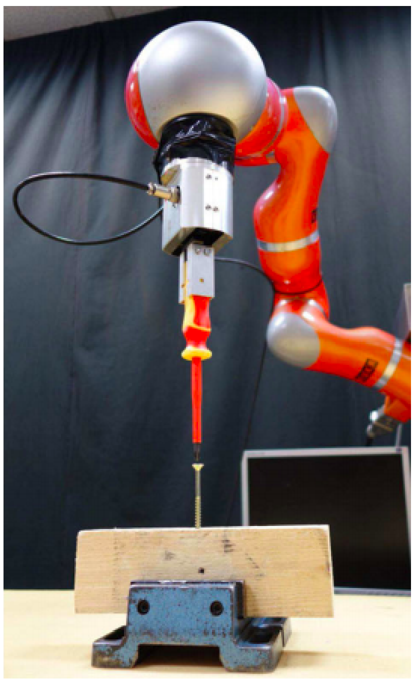

(b) For the bolt-screwing task, DMPs are used to encode the trajectories after demonstrations.

Fig. 2.6 Experimental setups for the slide-in-the-groove assembly task and bolt-screwing task [98].

from the camera mounted on the end of the right arm. In the teleoperation demonstration, the left arm of the robot plays the role of the controller, and the right arm plays as an effector.

Delivering hints to robots is another way of teleoperation. Human operators deliver hints to the robot by repeating the desired tasks many times, or by pointing out the important elements of the skill. The hints can be addressed in various ways throughout the whole learning procedure. One of the hints is vocal deixis, that is, an instruction from a human operator. Pardowitz et al. used vocal comments given by a demonstrator while demonstrating the task to speed up the learning of robots [95]. The vocal elucidations are integrated into the weight function, which determines the relevance of features consisted in manipulation segments.

Generally speaking, the acoustic of speech consists of three main bits information: the identification of the speaker, the linguistic content of the speech, and the way of speaking. Instead of focusing on the linguistic content of the vocal information, Breazeal et al. proposed an approach to teach the robot how to understand the speaker's affective communicative intent [17]. In the LfD paradigm, to understand the intention of the human demonstrator through HRI (Human-Robot Interaction) is the critical point for the robot to learn. Demonstrations are goal-directed, and the robot is expected to understand the human's intention and extract the goal of the demonstrated examples [25].

The vocal or visual pattern could be used in non-anthropomorphic robots [92]. The understanding of the human intention could be transferred from the standpoint of movement matching [35, 89] to joint motion replication [9, 20, 50, 14, 96, 110, 129, 22]. Recent research works believe that robots will need to understand the human's intention as socially cognitive learners, even if the examples are not perfectly demonstrated [18, 36, 19]. However, to keep track of intentions to complete desired goals, the imitating robots need a learning method. The solution could be building a cognitive model of the human teacher [54, 27], or using simulations of perspective [125].

2.4 Feature Extraction

When a dataset of demonstration trajectories, i.e. state-action examples, has been collected by using the demonstration approaches mentioned above, we need to consider how to map these data into a mathematical model. There are thousands of position data points distributing along a demonstration trajectory, and no need to record every point as the movement trajectories are hardly repeatable. Also, direct replication of the demonstrated trajectories may lead to poor performance because of the limited accuracy of the vision system and uncertainties in the gripping pose. Therefore, learning a policy to extract and generalise the key features of the assembly movements is the fundamental part of LfD.

Hidden Markov models (HMMs) are a popular methodology to encode and generalise the demonstration examples [102, 133, 46, 73, 126, 134, 107, 130, 52, 38, 51, 74, 75, 23, 76, 93]. HMM is a robust probabilistic method to encapsulate human motion, which contains spatial and temporal variables, through numerous demonstrations [68]. Initially, the training of HMMs is learned offline, and the sample data are manually classified into groups before learning. To make HMMs become online, Kulic et al. developed adaptive hidden Markov chains for incremental and autonomous learning of the motion patterns, which were extracted into a dynamic stochastic model [67].

The probabilistic approach can also be integrated with other methods to learn robust models of human motion through imitation. Calinon et al. combined HMM with Gaussian mixture regression (GMR) and dynamical systems to extract redundancies from a set of examples [23]. The original HMMs depend on a fixed number of hidden states and model the observation as an independent state when segmenting continuous motion. To fix the two major drawbacks of HMMs, Niekum et al. proposed the beta process auto-regressive hidden Markov model, in which the modes are easily shared among all models [93].

Based on Gaussian mixture models (GMMs), Chernova and Veloso proposed an interactive policy-learning strategy that reduces the size of training sets by allowing an agent to actively request from expert and effectively represent the most relevant training data [29]. Calinon et al. encoded the motion examples by estimating the optimal GMM, and then the trajectories were generalised through GMR [22]. Tanwani and Calinon extended the semi-tied GMMs for robust learning and adaptation of robot-manipulation tasks by reusing the synergies in various parts of the task sharing similar coordination patterns [123].

Dynamic movement primitives (DMPs) represent a fundamentally different approach to motion representation based on nonlinear dynamic systems. DMP is robust to spatial perturbation and suitable for following a specific movement path. Calinon et al. [23] used DMPs to reproduce the smoothest movement, and the learning process was faster than HMM, TGMR (time-dependent Gaussian mixture regression), LWR (locally weighted regression), and LWPR (locally weighted projection regression). Li and Fritz [78] extended the original DMP formulation by adding a function to define the shape of the movement, which could adapt the movement better to a novel goal position through adjusting the corresponding goal parameter. Ude et al. [129] utilised the available training movements and the task goal to enable the generalisation of DMPs to new situations, and able to produce a demonstrated periodic trajectory.

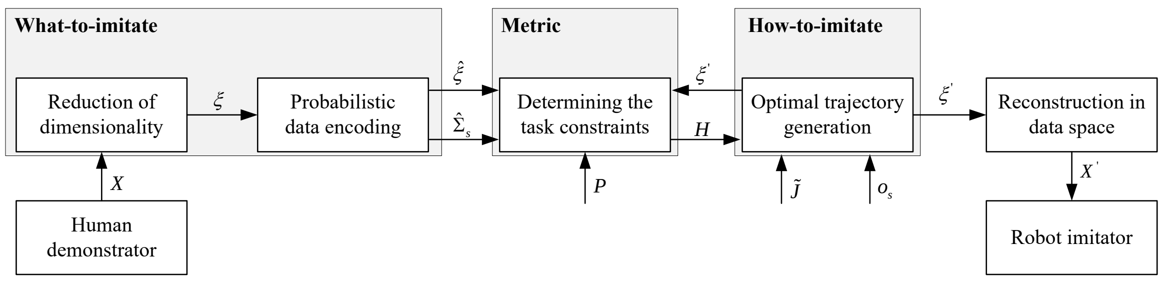

What-to-imitate
Metric
How-to-imitate
Reduction of dimensionality
ξ
Probabilistic data encoding
$\hat{\xi}$
$\hat{\Sigma}_s$
Determining the task constraints
ξ'
H
Optimal trajectory generation
ξ'
Reconstruction in data space
X
P
$\tilde{J}$
o_s
X'
Human demonstrator
Robot imitator

Fig. 2.7 An example of showing how to extract the features of the demonstration and how the features are used for robot learning [22]. The robot learns the latent features from the human demonstration X and reproduces the demonstrated trajectory with X'. The latent features of X are first lower-dimensioned to ξ, then generalised as $\hat{\xi}$, and optimised to ξ' before reconstruction.

Figure 2.7 gives an overview of the input–output flow through the complete model. It shows how to extract the feature of the demonstration (X) and how to carry on the whole imitation-learning process. Here, the feature means the trajectory data of human demonstrator and the robot. More specifically, individual modules have their role to play in this framework.

- Firstly, the human demonstrator performed demonstration motions X. The *What-to-imitate* module shows how to extract the feature of X. Then the motions were projected to a latent space using principal component analysis (PCA), in the *what-to-imitate* module. After being reduced of dimensionality, signals ξ were temporally aligned through the dynamic time warping (DTW) method. The Gaussian mixture model (GMM) and Bernoulli mixture model (BMM) were then optimised to encode the motion as generalised signals $\hat{\xi}$ with the associated time-dependent covariance matrix $\hat{\Sigma}_s$.

- In the *metric* module, a time-dependent similarity-measurement function was defined, considering the relative weight of each variable and dependencies through the variables included in the optional prior matrix P.

- After that, the trajectory of the imitating motion is computed in the *how-to-imitate* module, aiming at optimising the metric H. The trajectory was generated by using the robot's architecture Jacobian matrix $\widetilde{J}$, and the initial position of the object O_s within the workspace taken into consideration.

- Finally, the processed data ξ' in the latent space is reconstructed to the original data space X' before the robot imitation. Three main models used in the feature extraction of LfD will be analysed in the following subsections, namely the hidden Markov model (HMM), dynamic movement primitives (DMPs), and Gaussian mixture model (GMM).

2.4.1 Hidden Markov models

The hidden Markov model (HMM) is a statistical model used to describe a Markov process with unobserved states. An HMM can be presented as the most straightforward dynamic Bayesian network. The critical problem in HMM is to resolve the hidden parameters that are used to do further analysing (for example, pattern recognition) from the visible parameters. In the standard Markov model, the state is directly visible to the observer. Therefore, the transformation probabilities between states are the whole parameters. While the states are not visible in the hidden Markov model, some variables that are influenced by the states are visible.

Every state has a probability distribution on the token of the possible output, and therefore the sequence of the output token reveals the information of the state's sequence. HMM can be described by five elements, namely two state sets and three probability matrices:

- Hidden state S

  The states (for example, $S_1, S_2, S_3 \dots$) are the actual hidden states in HMM which satisfy the Markov characteristics and cannot be directly observed.

- Observable state O

  The observable state O is associated with the hidden state and can be directly observed. (For example, O_1, O_2, O_3 and so on, the number of observable states is not necessarily the same as hidden states.)

- Initial state probability matrix Π

  Π is the probability matrix of hidden state at the initial moment $t = 1$. (For example, given $t = 1$, $P(S_1) = p_1$, $P(S_2) = p_2$,$P(S_3) = p_3$, then the initial state probability matrix $\Pi = \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}$.)

- Hidden state transition probability matrix A

  Matrix A defines the transition probabilities between different states of HMM, where

$A_{ij} = P(S_j|S_i), 1 \leq i, j \geq N$, which means given the time t and state S_i, the state is S_j with probability P at time $t+1$.

- Observable state transition probability matrix B

  Assume that N is the number of hidden states, and M is the number of observable states, then: $B_{ij} = P(O_i|S_j), 1 \leq i \leq M, 1 \leq j \leq N$, which means that given the time t and hidden state S_j, the observed state is O_i with probability P.

In general, HMM can be simply defined with three tuples as $\lambda = (A, B, \pi)$. It is the extension of the standard Markov model in which the model is updated with observable state sets and the probabilities between the observable states and hidden states. In addition, the temporal variation of the latent representation of the robot's motion can be encoded in an HMM [26]. Trajectories demonstrated by human operators consist of a set of positions x and velocities $\dot{x}$. The joint distribution $P(x,\dot{x})$ is represented as a continuous HMM of K states, which are encoded by Gaussian mixture regression (GMR). For each Gaussian distribution of HMM, the centre and covariance matrix are defined by the equation based on positions x and velocities $\dot{x}$.

The influence of different Gaussian distributions is defined by the corresponding weight coefficients $h_{i,t}$ by taking into consideration the spatial information and the sequential information in the HMM. In a basic HMM, the states may be unstable and have a poor solution. For this reason, Calinon et al. extended the basic control model by integrating an acceleration-based controller to keep the robot following the learned nonlinear dynamic movements [23]. Kuli'c et al. used HMM to extract the motion sequences tracked by reflective markers located on various joints of the human body [67]. HMM was chosen to model motion movements due to its ability to encapsulate both spatial and temporal variability. Furthermore, HMMs can be used to recognise labelled motions and generate new motions at the same time as it is a generative model.

2.4.2 Dynamic movement primitives

Dynamic movement primitives (DMPs) was originally proposed by Ijspeert et al. [50], and further extended in [96]. The main question of DMPs is how to formulate dynamic movement equations to flexibly represent the dynamic performance of robotic joint motors, without the need of manually tuning parameters and ensuring the system stability. The desired movement primitives are built on variables representing the kinematic state of the dynamic system, such as positions, velocities, and accelerations. DMP is a kind of high-dimensional control policy and has many desirable features.

- Firstly, the architecture can be applied to any general movements within the joint's limitation.
- Secondly, the dynamic model is time-invariant.
- Thirdly, the system is convergent at the end.
- Lastly, the temporal and spatial variables can be decoupled. Each DMP has localised generalisation of the movement modelled.

DMP is a favourable way to deal with the complex assembly motion by decoupling the motion into a DMP sequence [61, 62]. Nemec et al. encoded the desired peg-insertion trajectories into DMPs, where each position/orientation dimension was encoded as an individual DMP [91]. In the self-tapping screw driving task, Peternel et al. collected the commanded position, rotation and stiffness variables, which were used to represent the phase-normalised trajectories [98].

In a basic point movement system, the discrete DMP motion can be represented by the following formulations [96]:

$$\tau \dot{v} = K(g-x) - Dv - K(g-x_0)s + Kf(s), \tag{2.1}$$

$$\tau \dot{x} = v, \tag{2.2}$$

where g is the desired goal position; x_0 is the initial start position; x and v are position and velocity of the DMP motion; τ is the temporal scaling factor which determines the duration of the movement; K is the spring constant; and D is a damping term.

The nonlinear function f, which changes the rather trivial exponential and monotonic convergence of the position x towards goal position g, is defined as [96]:

$$f(s) = \frac{\Sigma_i \psi_i(s)\theta_i s}{\Sigma_i \psi_i(s)}, \tag{2.3}$$

where $\psi_i(s) = exp(-h_i(s-c_i)^2)$ are Gaussian basis functions characterised by a centre c_i and bandwidth h_i; θ_i is the adjustable parameter which differs one DMP from another. s is a 'phase' value which monotonically decreases from the start '1' to the end '0' of the movement and is generated by a canonical system which is among the most basic dynamic systems used to formulate a point attractor [96]:

$$\tau \dot{s} = -\alpha s, \tag{2.4}$$

where α is a known time constant; τ is the temporal scaling factor as above.

To encapsulate a primitive movement into a DMP, the kinematic variables such as position, velocity, and acceleration are computed for each time-step using the recorded movement trajectory. Then, the corresponding coefficients are deduced according to the above Equations (2.1)–(2.4) and represent different DMPs. The discrete DMP then generates various motions with specific sequence of primitives. Any demonstrated movements observed by the robot are encoded by a set of nonlinear differential equations to represent and reproduce these movements, i.e. movement primitives.

Based on this framework, a library of movement primitives is built by labelling each encoded movement primitive according to the task classification (for example, approaching, picking, placing, moving, and releasing). To address the correspondence problem, Pastor et al. [96] used resolved motion rate inverse kinematics to map the end-effector position and gripper orientation onto the corresponding motor angles. To pick up an object from a table, the sequence of movement primitives recalled from the library is approaching–picking–moving–placing.

To increase the end-effector's range, Li and Fritz proposed a hierarchical architecture to embed the tool use from demonstrations by modelling different sub-tasks as individual DMPs [78]. Figure 2.8 shows its hierarchical architecture of teaching robots the use of human tools in an LfD framework [78], in which the temporal order for dual-arm coordination is learned at a higher level, and primitives are learned by constructing DMPs from exemplars at a lower level. Note that the dashed arrow is for the process of repeating the learned skill on the novel task.

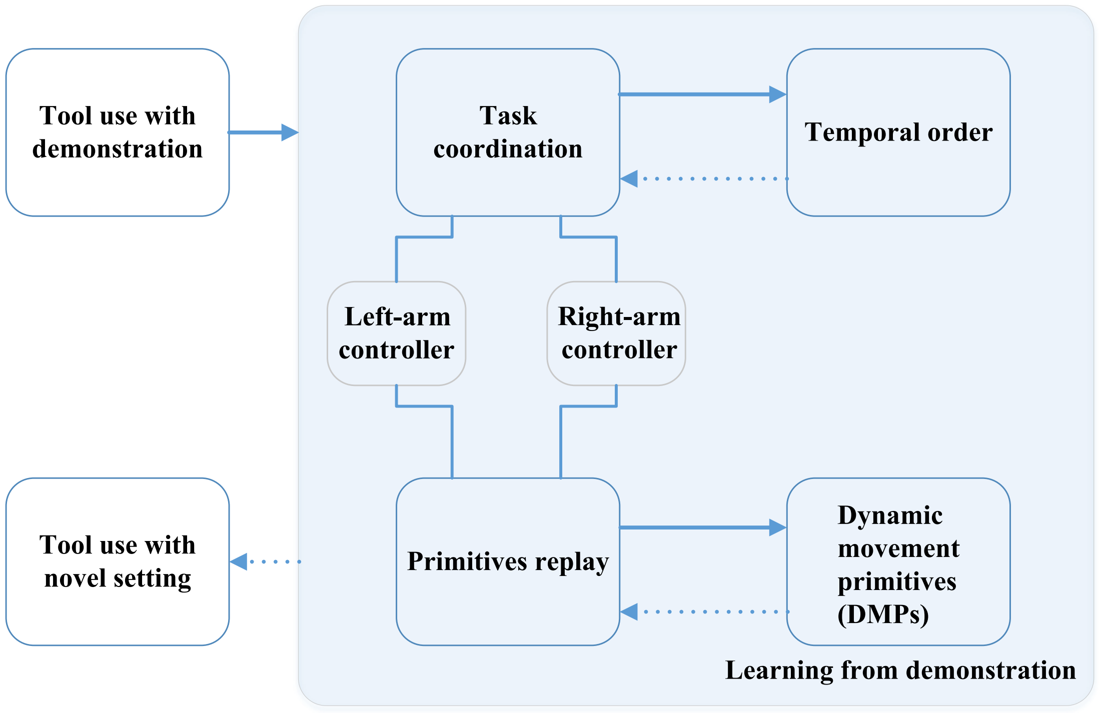

Tool use with demonstration
Task coordination
Temporal order
Left-arm controller
Right-arm controller
Tool use with novel setting
Primitives replay
Dynamic movement primitives (DMPs)
Learning from demonstration

Fig. 2.8 The hierarchical architecture of teaching robots the use of human tools in an LfD framework [78]. The solid arrow means the demonstration flow and the dashed arrow means the reproducing flow.

Compared to an end-to-end trajectory model that queries the inverse kinematic solver for the start and end positions, applying DMPs to every movement primitive has significantly improved the success rate for using tools [78]. Although the end-to-end model provides a motor-position solver, the physical constraint of the tool is neglected, which may lead to a failure manipulation.

2.4.3 Gaussian mixture models

Gaussian mixture models (GMMs) are probabilistic models for clustering and density estimation [32]. As mixture models, they do not need to know which classification a data point belongs to. Therefore, the model can learn the classification automatically. A Gaussian mixture model is constituted by three categories of parameters, namely the mixture component weights, and the component means and variances/covariances. GMMs are powerful tools for robotic motion modelling, as the approaches are robust to noise.

Combined with expectation maximization (EM), GMM is outperforming many assembly modelling approaches, like gravitational search–fuzzy clustering algorithm (GS–FCA), stochastic gradient boosting (SGB), and classical fuzzy classifier [55]. For a dataset of N datapoints $\{\boldsymbol{\xi}_j\}_{j=1}^N$ with $\boldsymbol{\xi}_j \in \mathbb{R}^D$, which can be either joint angles, hand paths, or hand–object distance vectors, a Gaussian mixture model with a weighted sum of M components $\{\boldsymbol{C}_i\}_{i=1}^M$ is defined by a probability density function [32]:

$$p(\boldsymbol{\xi}_j|\lambda) = \sum_{i=1}^{M} \omega_i g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{2.5}$$

where $\boldsymbol{\xi}_j$ is a D-dimensional data vector, $\omega_i, i = 1, \ldots, M$, are the mixture component weights of component $\boldsymbol{C}_i$, with the constraint that $\Sigma_{i=1}^M \omega_i = 1$, and $g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, ..., M, j = 1, ..., N$ are the component Gaussian densities.

For each component $\boldsymbol{C}_i$, the Gaussian density is defined as [32]:

$$g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{\xi}_j-\boldsymbol{\mu}_i)'\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{\xi}_j-\boldsymbol{\mu}_i)\}, \tag{2.6}$$

where $\boldsymbol{\mu}_i$ are the component means, $\boldsymbol{\Sigma}_i$ are the covariance matrices. $\lambda = \{\omega_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{M}$ are the set of parameters to be estimated in the density function; these parameters define a Gaussian mixture model. λ is usually estimated by maximum likelihood estimation using the standard expectation maximization (EM) algorithm [37].

Gaussian mixture components with full covariance matrices are confronted with the problem of over-fitting the sample data points when the sample data is noisy, or the sample amount is not enough. By decomposing the covariance into two parts—a common latent feature matrix and a component-specific diagonal matrix—the mixture components are then forced to align along with a set of common coordination patterns. The semi-tied GMM yields favourable characteristics in the latent space that can be reused in other parts of the learning skill [123].

In summary, HMMs are suitable for modelling motions with both spatial and temporal variability, while DMPs are time-invariant and can be applied to any general movement within the joint's limitation. GMMs are robust to noise but may lead to over-fitting when in high-dimensional spaces or when the sample data is not good enough. For better performance, HMMs, DMPs, and GMMs are usually combined with other optimisation algorithms as presented above.

2.5 Metric of Imitation Performance

Determining a metric of imitation performance is a key element for evaluating LfD. In the execution phase (see Figure 2.1), the metric is the motivation for the robot to reproduce. Once the metric is set, an optimal controller could be found by minimising this metric (for example, by evaluating several reproductive attempts or by deriving the metric to find an optimum). The imitation metric plays the role of the cost function or the reward function for reproducing the skill [43]. In other words, the metric of imitation quantitatively translates human intentions during the demonstrations and evaluates the similarity of the robot-repeating performance.

In the robotic assembly, the evaluation of the robot's performance is intuitive, accomplishing the assembly sequence as demonstrated and assembling the individual parts. However, a specific built-in metric is indispensable if we want to drive and optimize the learning process. Figure 2.9 shows an example of illustrating how to use three different displacements (relative displacement, absolute position, and relative position) to evaluate a reproduction attempt and find an optimal controller for the reproduction of a task [6]. Relative displacement: comparing to the demonstrated square, the imitating square moves same displacement to the right direction, no matter where the circle and rectangle locate. Absolute position: comparing to the demonstrated square, the imitating square moves to the absolute position where the demonstrated square stays. Relative position: comparing to the demonstrated square, the imitating square moves to the same position related to the rectangle and the circle. These three different imitations show the reproducing results from three different metrics of imitation, i.e., three different displacements.

LfD is a relatively young but rapidly growing research field in which a wide variety of challenges have been addressed. The most intuitive metric of evaluation is to minimise the difference between the observed motion repeated by the robot and the teaching motion demonstrated by the human teacher [69, 13, 82]. However, there exists little direct comparison between different feature-extracting models of LfD currently, since the evaluation is constrained to the specific learning task and robotic platform. LfD needs a set of unified evaluation metrics to compare the different imitation systems. The existing approaches mainly consider the variance and correlation information of joint angles, trajectory paths, and object–hand relation.

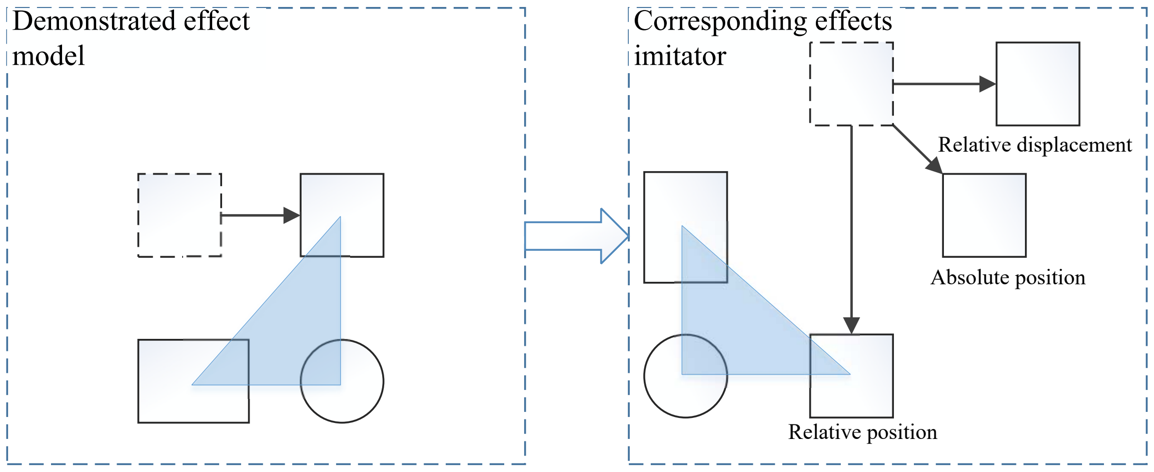


Fig. 2.9 An example of illustrating how to use three different displacements (relative displacement, absolute position, and relative position) to evaluate a reproduction attempt and find an optimal controller for the reproduction of a task [6]. Relative displacement: comparing to the demonstrated square, the imitating square moves same displacement to the right direction, no matter where the circle and rectangle locate. Absolute position: comparing to the demonstrated square, the imitating square moves to the absolute position where the demonstrated square stays. Relative position: comparing to the demonstrated square, the imitating square moves to the same position related to the rectangle and the circle.

2.5.1 Weighted similarity measure

Taking only the position of the trajectory into consideration, a Euclidean distance measure can be defined as [22]:

$$\sum_{i=1}^{D-1} \omega_i (\xi_{s,i} - \hat{\xi}_{s,i})^2 = (\xi_s - \hat{\xi}_s)^T W (\xi_s - \hat{\xi}_s), \tag{2.7}$$

where ξ_s is the candidate position of the trajectory reproduced by the robot, $\hat{\xi}_s$ is the desired position of the optimum trajectory, and both position points have the same dimensionality $(D-1)$.

It should be noted that the optimum trajectory is not equal to the demonstrated trajectory, as the body schema of the human and robot is very different. W is the time-dependent matrix with dimensionality of $(D-1) \times (D-1)$. The matrix's diagonal variables ω_i are the weights defining the influence of each corresponding point. Generally, W is a full covariance matrix, which represents the correlations across the different variables.

2.5.2 Generic similarity measure

Generic similarity measure H is a general formalism for evaluating the reproduction of a task, proposed in [15]. Compared to the weighted similarity measure of Euclidean distance defined in Equation (2.7), the similarity measure H takes into account more variables, such as the variations of constraints and the dependencies across the variables. It should be noted that the matrix is continuous, positive, and is estimable at any point along the trajectory.

In the latent joint space, given the generalised joint angle trajectories $\hat{\xi}_s^\theta$, the generalised hand paths $\hat{\xi}_s^x$, and the generalised hands–object distance vectors $\hat{\xi}_s^y$, which are obtained from the demonstrated examples, the generic similarity measure H is defined as [15]:

$$H = (\xi_s^\theta - \hat{\xi}_s^\theta)^T W^\theta (\xi_s^\theta - \hat{\xi}_s^\theta) + (\xi_s^x - \hat{\xi}_s^x)^T W^x (\xi_s^x - \hat{\xi}_s^x) + (\xi_s^y - \hat{\xi}_s^y)^T W^y (\xi_s^y - \hat{\xi}_s^y), \quad (2.8)$$

where $\{\xi_s^\theta, \xi_s^x, \xi_s^y\}$ represent the candidate trajectories for repeating the movements.

2.5.3 Combination of metrics

Calinon et al. used five metrics to evaluate a reproduction attempt $x' \in \mathbb{R}^{(D\times T)}$ reproduced from the demonstrated example set $x \in \mathbb{R}^{(D\times M\times T)}$ [23]. The last two metrics consider the computation time of the learning and retrieval process, which partially depends on the performance of the central processing unit, so here, only the other three metrics will be introduced.

$\mathscr{M}_1$: The metric evaluates the spatial and temporal information of the reproduced motion, where a root-mean-square (RMS) error is calculated based on the position difference [23]:

$$\mathscr{M}_1 = \frac{1}{MT}\Sigma_{m=1}^{M}\Sigma_{t=1}^{T}\|x_t' - x_{m,t}\|, \tag{2.9}$$

where M is the number of demonstrations and T is the moment along the demonstrating processes.

$\mathscr{M}_2$: In this metric, the imitated motion is first temporally aligned with the demonstrations through dynamic time warping (DTW) [68], and then an RMS based on the position difference similar to $\mathscr{M}_1$ is calculated. However, not like $\mathscr{M}_1$, spatial information has more priorities here, that is, the metric compares the whole path instead of the exact trajectory along time.

$\mathscr{M}_3$: This metric considers the smoothness of the imitated motion by calculating the derivation of the acceleration extracted from the motion [23]:

$$\mathscr{M}_3 = \frac{1}{T}\Sigma_{t=1}^{T}\|\dddot{x}_t'\|. \tag{2.10}$$

The smoothness is very useful, especially for evaluating the transition between different movement primitives.

Calinon et al. also used the above three metrics to evaluate the stability of the imitation system by superposing random force along with the motion. The combination of the metrics provides a comprehensive evaluation of the imitation learning system. Most of the current imitative systems are evaluated through completing particular tasks as demonstrated by the teachers. Pastor et al. demonstrated the utility of DMPs in a robot demonstration of water-serving [96], as shown in Figure 2.10. The top row shows that the robot executes a pouring task by pouring water into the inner cup. The bottom row is the reproduction of a pouring task with a new goal, the outer cup [96].

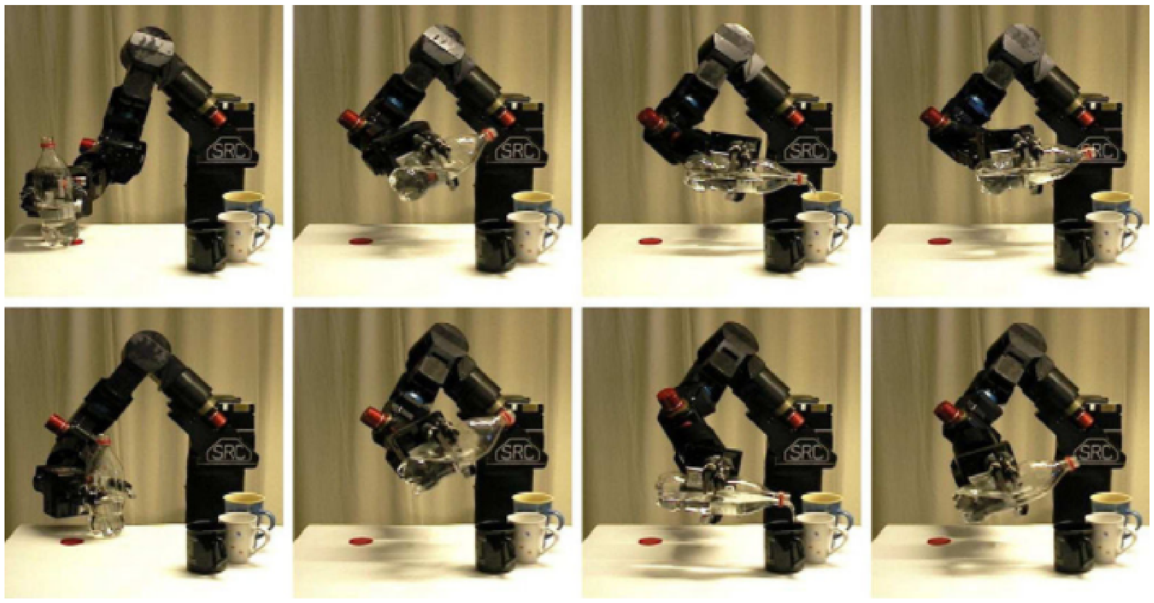


Fig. 2.10 Example of the movement reproduction and generalisation in a new environment with the Sarcos Slave Arm. The top row shows that the robot executes a pouring task by pouring water into the inner cup. The bottom row is the reproduction of a pouring task with a new goal, the outer cup [96].

Firstly, a human operator demonstrates the pouring task, including grasping, pouring, retreating bottle and releasing movements. Secondly, the robot extracts the movement primitives from the observed demonstrations and adds the primitives to the motion library. Thirdly, the experimental environment (water and cups) is prepared. Fourthly, the sequence of movement primitives is manually determined. Fifthly, appropriate goals are assigned to each DMP. Finally, the robot reproduces the demonstrated motion with the determined sequence of primitives and learns to apply the skill to new goal positions by adjusting the goal of the pouring movement.

To demonstrate the framework's ability to adapt online to new goals, as well as to avoid obstacles, Pastor et al. extended the experimental setup with a stereo camera system. The robot needed to adapt movements to goals that changed their position during the robot's movement, as shown in Figure 2.11. The top row shows that the robot places the cup in a fixed position. The middle row shows that the robot places the cup on a goal position which

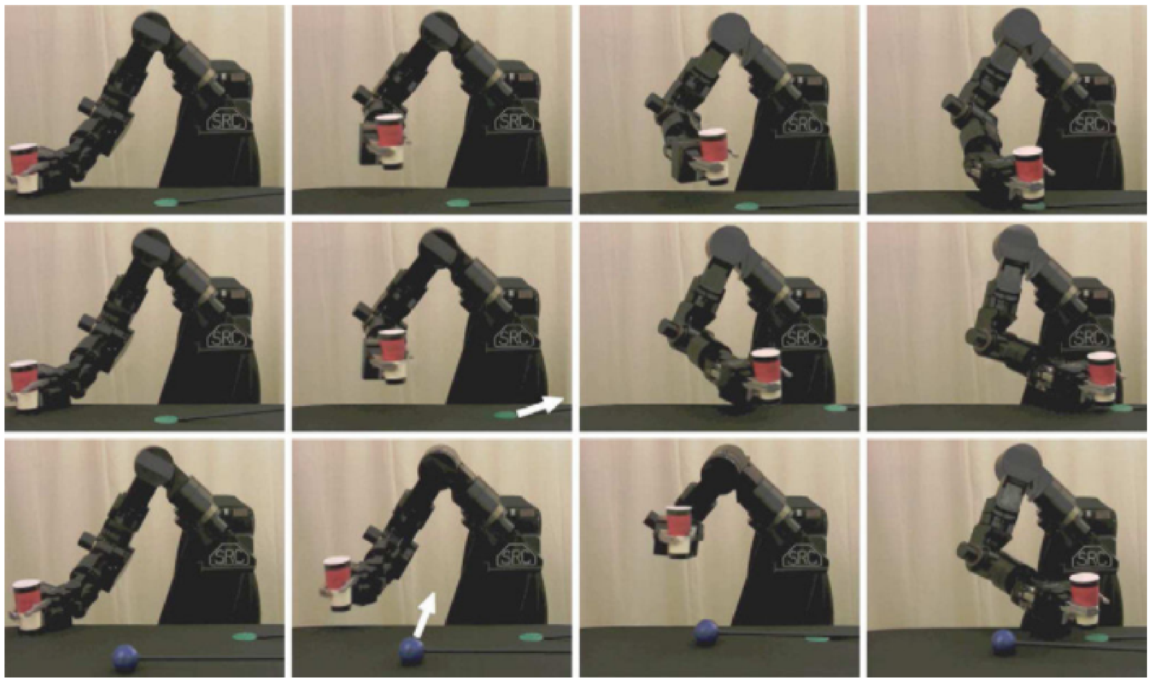


Fig. 2.11 Example of placing a red cup on a green coaster. The top row shows that the robot places the cup in a fixed position. The middle row shows that the robot places the cup on a goal position which changes location during placing. The bottom row shows that the robot places the cup with the same goal as the middle row, while accounting for the interference of a blue ball [96].

changes location during placing. The bottom row shows that the robot places the cup with the same goal as the middle row, while accounting for the interference of a blue ball [96].

2.6 Summary

This chapter presents a comprehensive survey of learning-from-demonstration (LfD) approaches, with a focus on their applications in robotic assembly. LfD has the unique advantage of enabling the general public to use the robot without the need of learning programming skills. Additionally, kinesthetic demonstration of LfD solves the correspondence problem between human demonstrators and robots. Consequently, LfD is an effective learning algorithm for robots and has been used in many robotic-learning systems.

LfD has several promising areas to be further investigated in the future, ranging from insufficient data to incremental learning and effective demonstration. More specifically, a summary can be presented below

- Learning from insufficient data. LfD aims at providing non-experts with an easy way to teach robots practical skills, although usually, the quantity of demonstrations is not numerous. However, the demonstration in the robotic assembly may contain noise. Due to the lack of some movement features and the intuitive nature of interacting with human demonstrators, it becomes hard for non-expert users to use LfD. Requiring non-experts to demonstrate one movement in a repetitive way is not a good solution. Future research work on how to generalise through a limited number of feature samples is needed.

- Incremental learning. The robot can learn a skill from a demonstrator, or learn several skills from different demonstrations. The study on incremental learning is still very limited during past research. The skills that the robot has learned are parallel, not progressive or incremental. DMPs are fundamental learning blocks that can be used to learn more advanced skills, while these skills cannot be used to learn more complicated skills in a novel task. For example, the DMPs learned in Peg-in-Hole task can be used for chair-assembly task but not for cooking task. Incremental learning features should be given more attention to the robotic assembly in future research.

- Effective demonstration. When the demonstrator executes any assembly actions, the robot tries to extract the features from the demonstration. In most cases, the learning process is unidirectional, lacking timely revision, leading to less effective learning. The most popular approaches adopted in LfD systems are reward functions. However, the reward functions only give the evaluation of the given state and no desirable information on which demonstration can be selected as the best example. One promising solution is that the human demonstrator provides timely feedback, e.g. through a GUI [88]) on the robot's actions. More research on how to provide such effective feedback information is another aspect of future work.

- Fine assembly. Robotics assembly aims at enormously promoting industry productivity and helping workers on highly repeated tasks, especially in 'light' industries, such as the assembly of small parts. The robots have to be sophisticated enough to handle more complicated and more advanced tasks instead of being limited to the individual sub-skills of assembly, such as inserting, rotating, screwing and so on. Future research work on how to combine the sub-skills into smooth assembly skills is desired.

- Improved evaluation. A standardised set of evaluation metrics is a fundamentally important research area for future work. Furthermore, improved evaluation metrics help the learning process of imitation by providing a more accurate and effective goal in LfD. The formalisation of evaluation criteria would also facilitate the research and development of the extended general-purpose learning systems in robotic assembly.

There are many promising areas to be investigated, however, this thesis will only focus on one of these areas: effective demonstration. The thesis explores from the effective grasping by kinesthetic demonstration (See Chapter 3) to an effective demonstration by human-robot skeleton visual mapping (See Chapter 4), then effective teaching of the robotic assembly task by real-time feedback (See Chapter 5).

Chapter 3

Peg-in-Hole Assembly

This chapter presents a novel approach for a robot to conduct assembly tasks, namely robot learning from human demonstrations. In Section 3.1, a brief introduction of robotic assembly is presented. Section 3.2 overviews the existing work related to Peg-in-Hole Assembly, including the introduction of the robot operating system (ROS) which is used in this thesis research. In Section 3.3, the methods used to solve the assembly problem are explained in terms of two aspects: (i) how the human demonstrator teaches the robot and (ii) how the robot reproduces the learned skills. Then, experiments are conducted in Section 3.4 to demonstrate the feasibility and performance of the proposed approach. Finally, a brief conclusion and further improvement are given in Section 3.5.

3.1 Introduction

The robotic assembly needs a high degree of repeatability, flexibility, and reliability to improve the automation performance in assembly lines. Traditionally, the robotic assembly operation is programmed or hard-coded by human operators with a good knowledge of all geometrical characteristics of individual parts. This assembly operation is usually the position-based control and designed to follow desired trajectories with an extremely tight position accurately[47]. Similar to human assembly operations that are based on force and tactile feedback, the robots will use their sensors to gather contact and force data in order to implement the assembly procedures.

In general, the current robotic assembly systems can handle known objects within the well-structured assembly lines very well. Knepper et al. introduced a multi-robot coordinated assembly system for furniture assembly [57]. The furniture parts were predefined in CAD files for the modelling and recognition purpose so that the correct assembly sequence can be deduced from geometric data. They listed the geometry of individual parts in a table so that a group of robots can conduct parts delivery or parts assembly collaboratively. On the other hand, Suarez-Ruiz and Pham proposed a taxonomy of the manipulation primitives for bi-manual pin insertion, which was only one of the critical steps in the autonomous assembly of an IKEA chair [119].

However, when the assembly tasks change, the current robotic assembly needs tedious reprogramming for every new assembly task before the operation. In contrast, Learning from Demonstration (LfD) paradigm enables robots to learn the involved forces and trajectories for assembling tasks from human demonstrations. There is no programming involved in the robot learning and reproducing stages. The LfD allows for creating a connection between perception and action for the robot. Recently, LfD has been suggested as an effective way to accelerate the robot learning processes from the low-level control to the high-level assembly planning [64]. Therefore, LfD is a preferable approach for robotic assembly tasks [138].

This chapter proposes a LfD paradigm for solving one of the fundamental robotic assembly tasks, Peg-in-Hole (PiH) task. PiH task requires the robot to grasp an object placed on the desktop and assembles it with another object. For grasping an object, there are infinite grasping poses, but only few of the poses are effective. The effective grasping poses are especially important in the robotic assembly task as the following assembly movement requires the manipulated object is secure and easy for force control. This chapter shows a method that the robot learns effective grasping poses from human demonstrations for robotic PiH task.

3.2 Background

3.2.1 Related work

PiH is one of the most essential and representative assembly tasks and has been widely researched [135, 56, 94]. It is a process that a robotic gripper grabs the peg and inserts it in a hole. The positioning inaccuracies and tight tolerances between the peg and the hole involved in PiH operations require some degree of online adaptation of the programmed trajectories. Up to now, several robotic assembly systems were proposed to solve the PiH problem, and most of them use additional specialized force sensors, markers or cameras. Nemec et al.[91] proposed an approach to acquire not only trajectories but also forces and torques occurred during the task demonstration. During the human demonstration phase, the Cartesian space trajectory and the associated force/torque profile of the human motion in the PiH task are recorded. In the reproduction phase, the robot uses admittance or impedance control law to adapt to the desired forces and reduce the force/torque error.

Tang et al. introduced Gaussian Mixture Regression (GMR) to learn the state-varying admittance directly from human demonstration data [122]. The demonstration data is collected by a specially designed device in which force/torque sensor is used to collect the wrench information, and active markers are placed to record the corrective velocity that human applies on the peg. Instead of learning from human demonstrators, Kramberger et al. proposed an algorithm to learn geometrical constraints between the parts and their final locations from the experiments executed by a real robot [62]. The robot inserted the available pegs into different holes. The judgment of success or failure is accomplished by using force/torque data and poses extracted by vision. If the action is executed successfully, the robot learns that the peg fits in the hole. Besides, the PiH skill can also be learned through self-supervised learning that based on convolutional neural network (CNN) and multilayer perceptron (MLP), but the learning process takes dozens of hours [77].

In addition, the peg would usually occlude the hole when the robot approaches the hole during the peg-in-hole operation. Therefore, vision-based pose estimation may not be suitable for the high-accuracy assembly tasks in which two parts occlude each other. However, if a camera is mounted on the robotic arm, the occlude problem can be eliminated, but additional sensory data is needed to estimate the camera pose [111]. To correct the pose of assembly parts, Xiao et al. devised a nominal assembly-motion sequence to collect data from exploratory complaint movements [108]. The data were then used to update the subsequent assembly sequence to correct errors in the nominal assembly operation. Nevertheless, the uncertainty in the pose of the manipulated object should be further addressed in future research.

The above research works avoid the effective grasping problem or use predefined objects in the PiH task, which weakens the robot's adaptive ability. This chapter proposes a new approach to solve one of the assembly tasks, Peg-in-Hole (PiH) problem, by using the LfD paradigm. The primary contribution of this chapter is:

- A LfD paradigm is proposed based on key positions and object features. Instead of traditionally imitating the trajectories demonstrated by a human, the robot learns the effective grasping poses and assembly positions of the Peg-in-Hole (PiH) task through the kinesthetic teaching. Compared with the existing works that use predefined objects for robotic assembly tasks, the proposed method improves the robot's adaptive ability by learning from human demonstration with unstructured objects in PiH task.

The object to be assembled is not limited to any predefined objects. The geometrical characteristics of the parts are not necessarily known a prior. In addition, the objects can be placed in arbitrary poses and positions within the workspace for the robotic arms to manipulate. The robot learns assembly skills through LfD paradigm, which allows non-experts to teach the robot how to assemble. Instead of imitating the trajectories demonstrated by a human, the robot learns the effective grasping poses for a PiH task through the kinesthetic teaching.

3.2.2 Robot operating system

In the proposed LfD system, Baxter robot is used for object manipulating, and the Kinect sensor is responsible for visual information inputting. These two individual devices liaise with the Robot Operating System(ROS) installed on a workstation PC. ROS is a set of software libraries and tools that help to build robot applications. From drivers to state-of-the-art artificial intelligence algorithms, and with powerful developer tools, ROS is open source and popular with researchers from academia and industry.

The ROS works alongside a traditional Linux operating system, like Ubuntu. As described by the ROS official website [34]:

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

ROS is intended to ease some of the difficulties for software developers, such as:

- Distributed computation. Many robots rely on programs that run on multiple computers, ROS can satisfy the need for communication between these devices over the network.
- Software reuse. There is a growing collection of useful algorithms for common tasks like navigation, motion planning, mapping, and many others. ROS provides standard packages for these robotics algorithms without the need to reimplement for each new robot system.

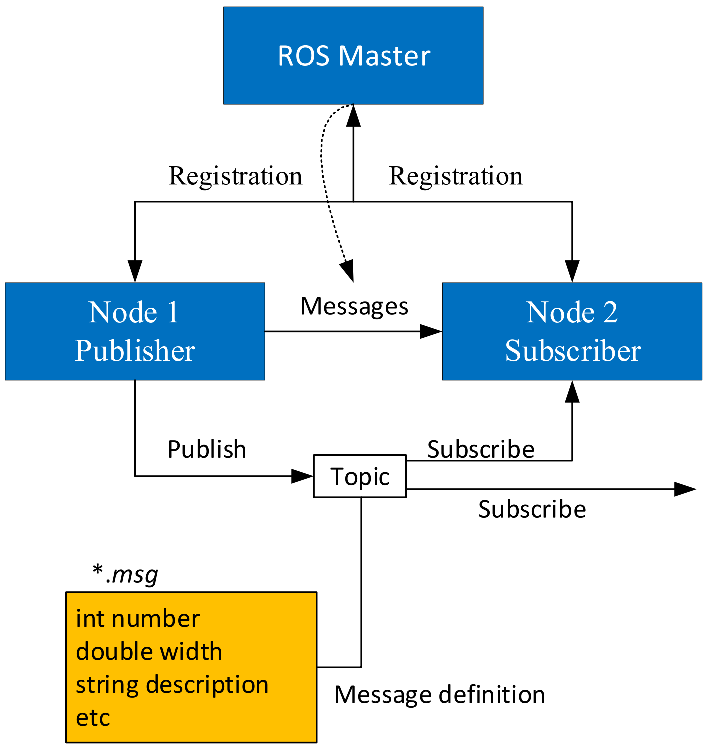


Fig. 3.1 The structure of ROS messages. The ROS Master coordinates all other nodes in the ROS system. Node 1 and 2 registered with ROS Master so they can locate and then communicate with each other. Node 1 publishes messages to a topic which has a unique name and node 2 subscribe to this topic through its name. The content published by node 1 is a ROS message which consists of a set of data, like 'int number', 'double width', 'string description' etc.

- Rapid testing. When working with a real robot, the testing process is often time-consuming and error-prone. Sometimes, the robot is not even available to work with the operator. ROS solves this issue with simulators that can run almost all of the modern commercial robots at any time and without any worries on breaking the robot.

ROS concepts

ROS works as a platform for developing and running robotic applications. It consists of some core concepts: ROS Master, node, service, topic, and message, see Figure 3.1. The ROS Master coordinates all other nodes in the ROS system. Node 1 and 2 registered with ROS Master so they can locate and then communicate with each other. Node 1 publishes messages to a topic which has a unique name and node 2 subscribe to this topic through its name. The content published by node 1 is a ROS message which consists of a set of data, like 'int number', 'double width', 'string description' etc.

- **Master** The ROS Master manages all the nodes, provides naming and registration services to the rest of the nodes in the ROS system. The Master coordinates the communication of all the nodes, tracks publishers and subscribers to topics. All the nodes talks and swap messages peer-to-peer after locating each other through the Master. The Master also provides the Parameter Server that allows data to be stored by key in a central location.
- **Node** The *node* is a process that performs computation of messages. Nodes are combined together into a graph and usually communicate with one another using streaming topics or through system services. A node can be regarded as a single module of the ROS system, and each node can represent a device or processing component of the robot system. For example, one node controls joint motors; one node controls torque sensor; one node controls Kinect sensor; one node controls motion planning, and so on.
- **Message** Messages are the data transferred between nodes. A message can be any type of data structure, including standard primitive types like integer, floating point, boolean, string, array, etc. Any other user-defined data structures similar as C structures are also commonly used in ROS.
- **Topic** A *topic* is a place that different nodes exchange messages via a transport system with publish/subscribe semantics. A node sends out its data encapsulated into a message to a given topic. Each topic has a name that is used to identify the place where the messages are subscribed and published. A node that is interested in a certain kind of message subscribes to the corresponding topic through the topic name. There

may be multiple publishers and subscribers communicating through a single topic, and a single topic can publish or subscribe to multiple topics. Publishers and subscribers are independent of each other, they do not know each other, but they can communicate through topics that they are interested.

- **Service** The feature of publish/subscribe model is a very convenient communication, but it is also many-to-many and one-way transport which does not work for the request/reply interactions of a distributed system. The solution of the ROS is to create services defined by a name and a pair of message structures: one for the request and one for the reply. ROS services only for computations and quick actions, it offers a nice complementarity with topics. Topics will be used for unidirectional data streams, and services will be used for a client/server architecture.

3.2.3 ROS for controlling a Baxter robot

Figure 3.2 shows the software architecture that runs within the Robot Operating System (ROS) contains several nodes. These nodes run simultaneously and can be explained as follows:

- At the lowest level, there are hardware-specific nodes to control the robotic arm and base through ROS wrappers for the Baxter driver. These, in turn, are given commands by the planner, which directs the overall flow of the assembly process.
- The planner is in constant communication with the blueprint node, which maintains the state of the assembly process and the goal structure.
- The blueprint node coordinates with the inside partitioner node, where the heart of the algorithm exists.
- The sensors connected with the robot publishes perceptional data to the ROS controller for decision making.

- Finally, the controller plans the motion for imitation learning. The Gazebo simulator or the real robot performs the learned skills.

Our system takes advantage of the distribution and communication infrastructure in ROS. All nodes are run in a decentralized manner on the appropriate robot. Communication is performed through ROS channels or “topics”. All nodes are designed to function with an arbitrary number of robotic sensors, although the experiments described here will only focus on two or four in order to demonstrate the specifics of the system.

3.3 Learning from Kinesthetic Teaching

This section begins with the analysis of object mapping. Two problems are addressed, namely how the object is mapped and how the robot learns effective grasping pose. Then, the investigation is focused on how the human demonstrator teaches the robot to learn the assembly skill by kinesthetic teaching.

Kinesthetic teaching can be combined with other learning methods. A method that allows a robotic manipulator to learn to perform tasks was presented in [59], which required imposing forces on target objects. The required positional and force profiles were learned from individual demonstrations via kinesthetic teaching and a haptic device, respectively. In this thesis research, the robot does not learn the complete trajectory from kinesthetic guidance. Instead, only essential information is used in the following assembly task. The proposed method combines kinesthetic teaching with object recognition which helps the robot learn an effective picking pose.

3.3.1 Teaching

The object detection runs on a Baxter robot with a development workstation. The Baxter’s wrist camera is used to map one side of an object, and then the learned model is used to

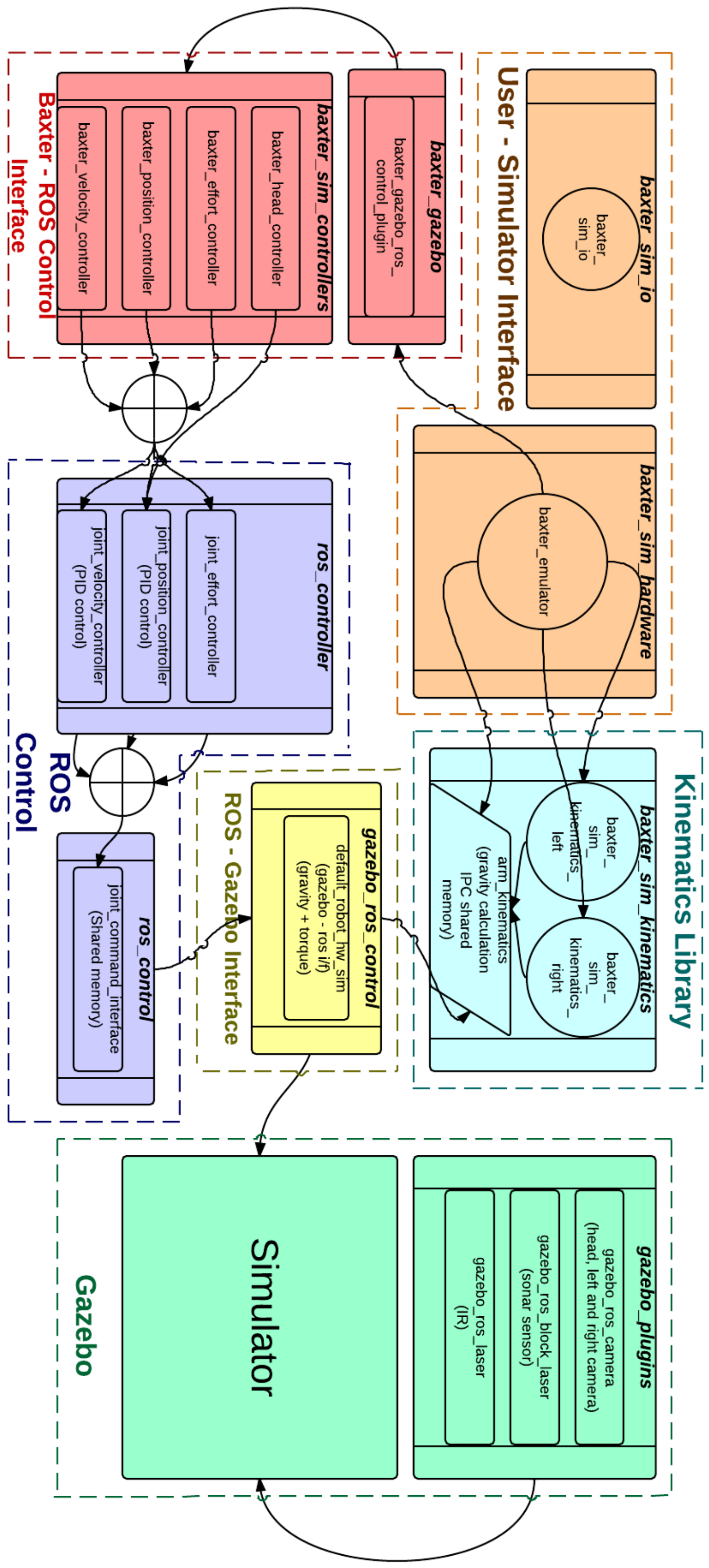

User - Simulator Interface
baxter_sim_io
baxter_sim_io
baxter_sim_hardware
baxter_emulator
Baxter - ROS Control Interface
baxter_gazebo
baxter_gazebo_ros_control_plugin
baxter_sim_controllers
baxter_head_controller
baxter_effort_controller
baxter_position_controller
baxter_velocity_controller
ROS Control
ros_controller
joint_effort_controller
joint_position_controller (PID control)
joint_velocity_controller (PID control)
ros_control
joint_command_interface (Shared memory)
ROS - Gazebo Interface
gazebo_ros_control
default_robot_hw_sim (gazebo - ros i/f) (gravity + torque)
Kinematics Library
baxter_sim_kinematics
baxter_sim_kinematics_left
baxter_sim_kinematics_right
arm_kinematics (gravity calculation IPC shared memory)
Gazebo
Simulator
gazebo_plugins
gazebo_ros_camera (head, left and right camera)
gazebo_ros_block_laser (sonar sensor)
gazebo_ros_laser (IR)

Fig. 3.2 The ROS flowchart for robot learning from demonstration. Credit to Rethink Robotics.

detect the object, localize it and pick it up [48]. The object detection and pose estimation uses conventional computer vision algorithms like SIFT and kNN for feature abstraction and object classification. The object classes are based on the specific object rather than general categories. Each object has a unique name labelled by the human operator.

Figure 3.3 presents the human teaching and robot learning process. Firstly, the wrist camera captures images from an workpiece and extract its bounding boxes. Then the features of the bounding boxes are used to represent an object class. Lastly, the human demonstrator teaches the robot how to grasp the object from an effective pose by kinesthetic guiding.

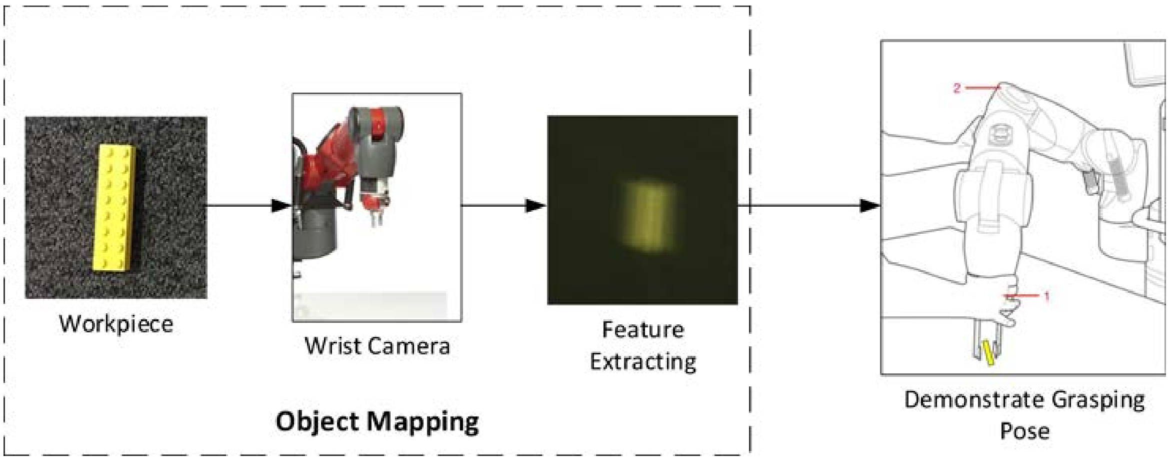


Fig. 3.3 The flowchart of human teaching stage from the operating view.

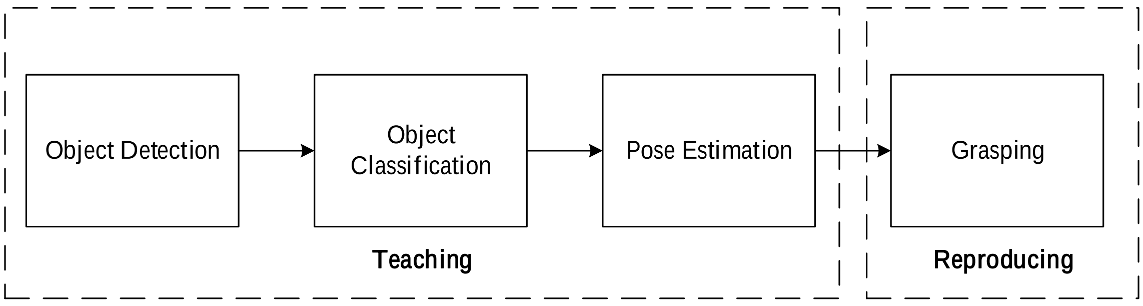


Fig. 3.4 The mapping of the object, from teaching to reproducing, under the information-flow view.

The key steps of the proposed experiment scenario is presented in Figure 3.4, including *Object Detection*, *Object Classification*, *Pose Estimation*, and *Grasping*. The first three steps are parts of human teaching, the last step is the reproducing from the robot.

Object detection

We use the grey background to reduce the reflective light which introducing noise to the detection of objects. During detection, the wrist camera moves along a line over an object at a fixed height. For an input image, the robot extracts the boundary boxes for the object. The smallest bounding box which contains the object is selected. Figure 3.5 and Figure 3.6 show the object detection of YellowLegoBlock and RedBlueLegoBlock in different views: the discrepancy view shows differences between the observed scene and the background, i.e., the object; the standard deviation view of the object shows the edges of the object; the predicted views are the learned example found by the KNN. Each predicted view corresponds to multiple effective grasping poses that learned from human teaching.

Object classification

In the object classification module, the bounding boxes are further abstracted with SIFT features. The k-means algorithm is used to extract a visual vocabulary of the SIFT feature. Then a Bag of Words feature is constructed for each image. Next, the Bag is augmented with a histogram of colours included in the image. The augmented feature vector is learned by the robot and labelled by the human with an intuitive name, like RedLego Block. The KNN algorithm is used for object classification, taking the augmented feature as input. The KNN is robust enough to classify the expected object from many candidates thanks to the high-dimensional input feature.

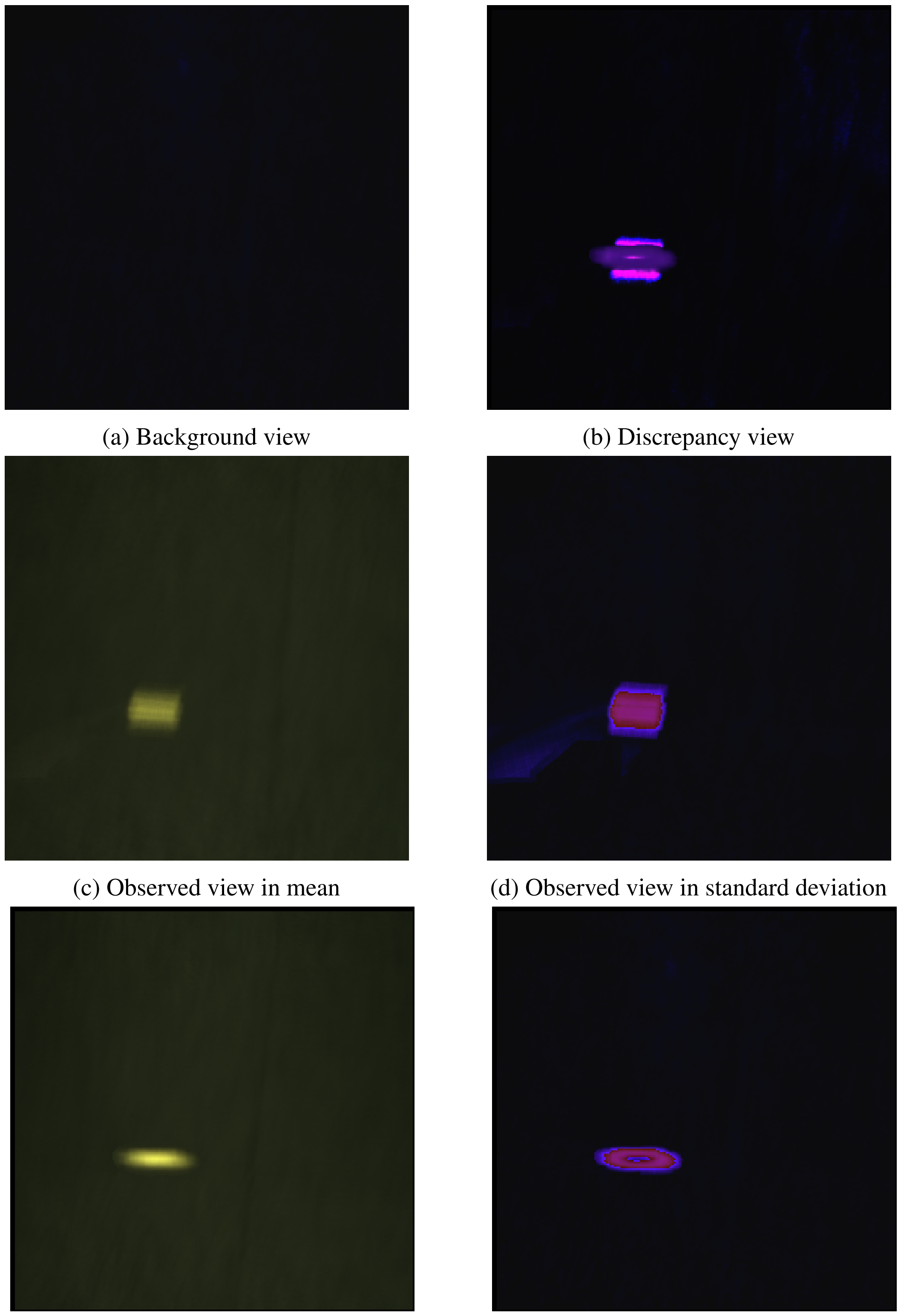
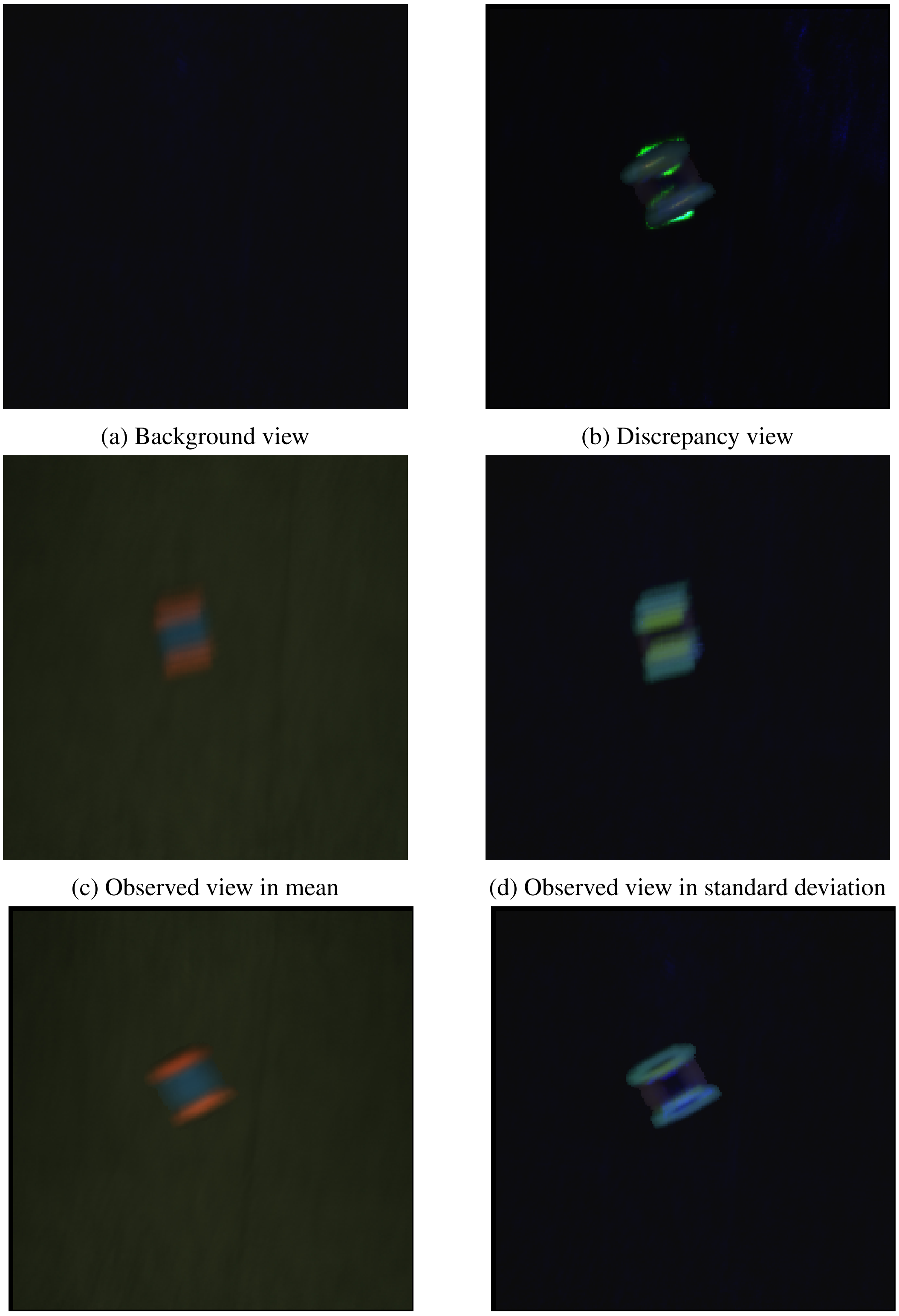

(a) Background view
(b) Discrepancy view
(c) Observed view in mean
(d) Observed view in standard deviation
(e) Predicted view in mean
(f) Predicted view in standard deviation

Fig. 3.5 Object detection of YellowLegoBlock in different views: the discrepancy view shows differences between the observed scene and the background, i.e., the object; the standard deviation view of the object shows the edges of the object; the predicted views are the learned example found by the KNN. Each predicted view corresponds to multiple effective grasping poses that learned from human teaching.



(a) Background view

(b) Discrepancy view

(c) Observed view in mean

(d) Observed view in standard deviation

(e) Predicted view in mean

(f) Predicted view in standard deviation

Fig. 3.6 Object detection of RedBlueLegoBlock in different views: the discrepancy view shows differences between the observed scene and the background, i.e., the object; the standard deviation view of the object shows the edges of the object; the predicted views are the learned example found by the KNN. Each predicted view corresponds to multiple effective grasping poses that learned from human teaching.

Pose estimation

After the robot has learned to detect and localize the object above, it could reach and grasp the object. To optimize the grasping pose for high grasping efficiency, the human demonstrator guides the robot's arm to the grasping pose (see Figure 3.12a). As an object can be grasped from more than one pose, the human demonstrator teaches the robot many poses to ensure the robot has more options if it fails the first time. Therefore, each object corresponds to multiple effective grasping poses. The robot always select the grasping pose that best match with the learned example predicted by the KNN (K-nearest neighbour) classifier. For estimating effective poses for an object, KNN recommends a second best match of effective pose if the first one fails in the grasping.

3.3.2 Reproduction

In the reproduction phase, the robot uses the learned knowledge to reproduce the grasping operation autonomously, which consists of three phases as follows:

- First, the robot uses the wrist camera to scan and detect the object in the workbench. From the input images, the robot extracts boundary boxes of an object. Then, the robot uses the boundary boxes to extract the augmented feature vector as described in Section 3.3.1. Next, the vector is incorporated into a k-nearest-neighbours model which can classify objects and output the label. This label is used to identify the object and refer to other information about the object for grasping.

- Next, to estimate the pose, the robot requires a crop of the image gradient of the object at a specific and known pose. The robot rotates the training image and finds the closest match to the image currently learned in Section 3.3.1.

- Last, once the grasping pose is determined, the robot need to identify the grasping point. The grasping module is a linear model that estimates the grasping success.

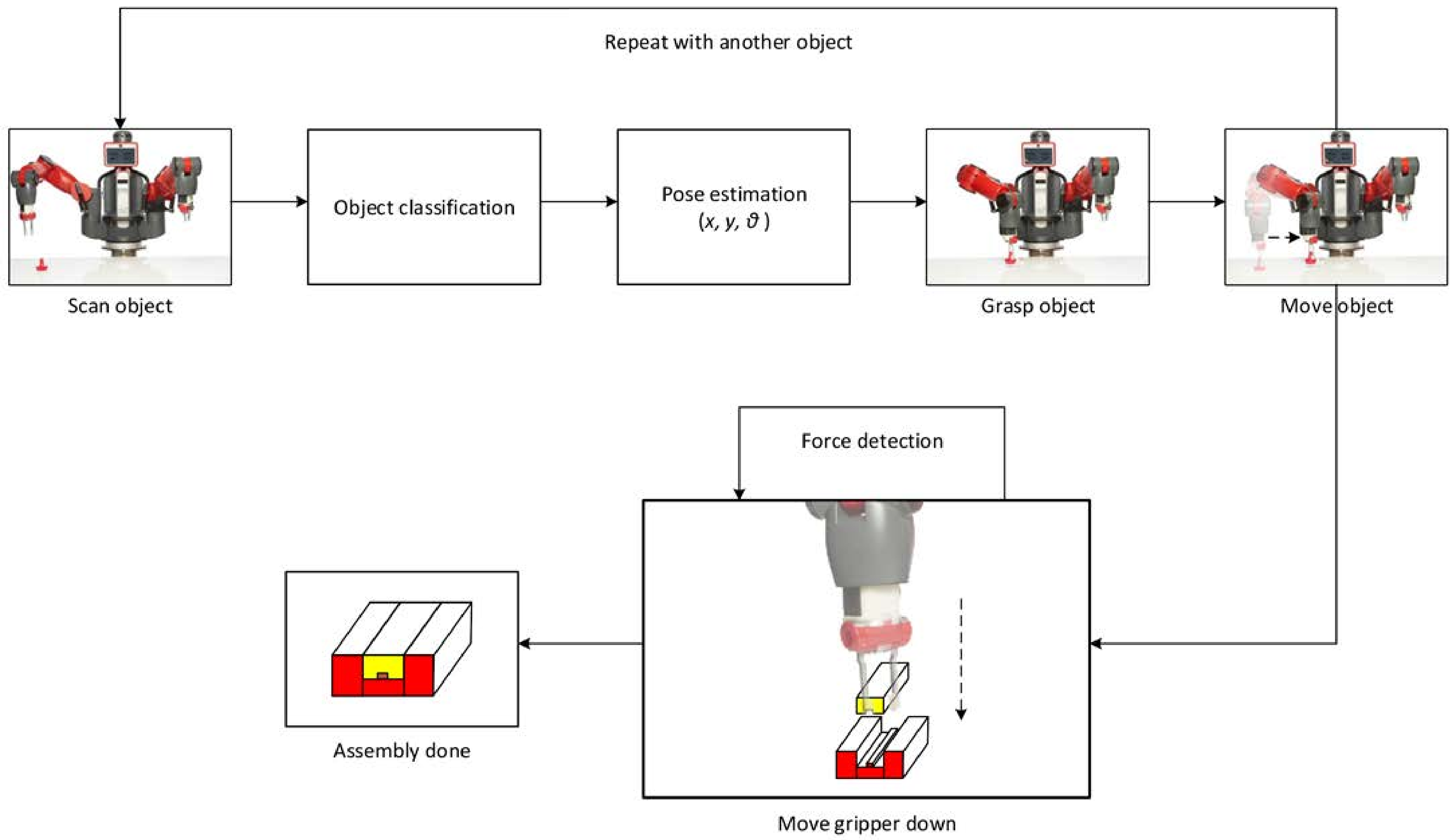


Fig. 3.7 The flowchart of robot reproducing stage.

The module takes the 3D pose of the object as input and outputs the grasping point (x, y, θ). The (x, y) is the 2D position in the plane of the table. The accurate height of the gripper does not matter, as the gripper always start from 38cm over the workbench and approaches gradually until hitting it, triggering the grasping. The θ is the angle which the gripper assumes for grasping.

3.3.3 Pseudocode

This section gives a brief outline of the robot program to implement the proposed approach described above. It is presented here in the format of pseudocode, see **Algorithm** 1. In the grasping part of the pseudocode, O_i is the new object to be mapped; Z_c is the "components zone". During the teaching of grasping skill, it should be noted that the pose $P(x, y, \theta)$ is relative to the camera's orientation, which is recorded in θ. When the robot reproduces the grasping, it rotates the camera to find the closest match to the learned image and pose. $\mathbf{V}_{f,i}$ is the feature extracted in the step of 3.3.1 *Object Classification*.

In the assembly part of the pseudocode, S_k is the sequence motion of assembly task demonstrated by the human. In the reproduction part of the pseudocode, O_i' is the learned object; Z_a is the "assembly zone"; O_m' is the former detected object to be grasped; O_n' is the later detected object to be assembled; F_p is the pressing force that the arm applies on the two objects; F_0 is the threshold force, which controls the insertion movement.

3.4 Experimental Results

In this section, the object recognition and LfD-based robotic assembly systems are both evaluated.

Algorithm 1 Pseudo code of the robotic assembly using LfD

```
1: initialize
2: /* line 3 - 10: learning from demonstration: grasping */
3: for $O_i$ in $Z_c$; $i \in [1,N]$ do
4:     detect the bounding box $B_i$;
5:     extract feature vector $\mathbf{V}_{f,i}$ from $B_i$;
6:     for all demonstration $D_j$; $j \in [1,M]$ do
7:         human demonstrates the grasping pose $P(x,y,\theta)$;
8:         robot maps pose $P(x,y,\theta)$ and feature $\mathbf{V}_{f,i}$;
9:     end for
10: end for
11: /* line 12 - 13: learning from demonstration: assembly*/
12: human demonstrates the assembly;
13: robot learns the motions and sequence $S_k, k \in [1,3]$;
14: /* line 15 - 32: robot reproduces assembly task*/
15: for $O'_i$ in $Z_c$; $i \in [i,N]$ do
16:     detect and classify the object $O'_m$;
17:     grasp object $O'_m$;
18:     break;
19: end for
20: assembly sequence $S_1$: move object $O'_m$ to zone $Z_a$;
21: for $O'_i$ in $Z_c$; $i \in [i,N]$ do
22:     detect and classify the object $O'_n$;
23:     grasp object $O'_n$;
24:     break;
25: end for
26: assembly sequence $S_2$: move object $O'_n$ to zone $Z_a$;
27: assembly sequence $S_3$: assemble object $O'_n$ with object $O'_m$;
28: while $F_p < F_0$ do
29:     robot gripper moves down;
30:     $F_p$++;
31: end while
32: assembly done;
```

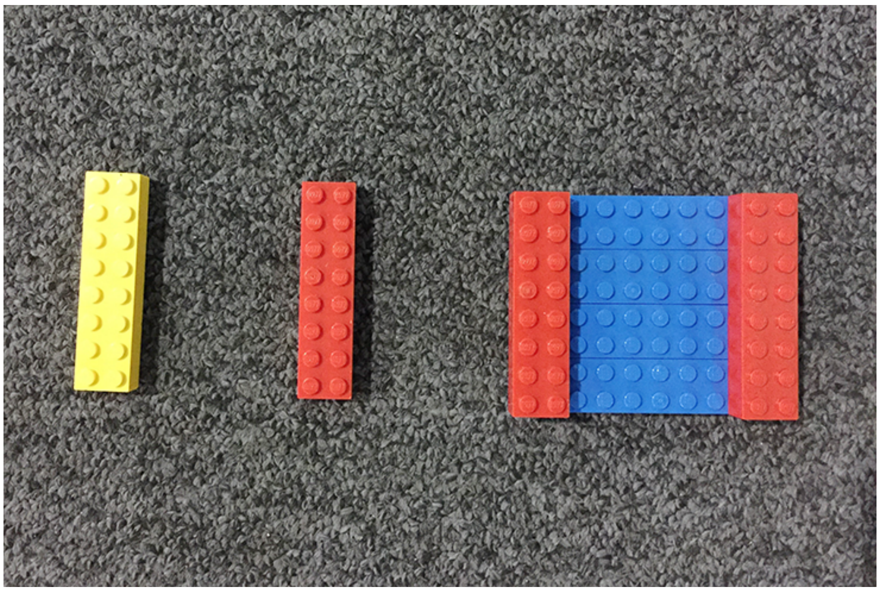


Fig. 3.8 Evaluation objects from left to right: Yellow Lego block, Red Lego block, and RedBlue Lego block.

3.4.1 Experimental setup

Figure 3.8 shows three kinds of Lego blocks used for the evaluation of object recognition and grasping performance of the system: namely Yellow Lego block, Red Lego block, and RedBlue Lego block. The Red Lego block and the Yellow Lego block are same in dimension, i.e. $2\frac{1}{2} \times \frac{1}{2} \times \frac{7}{16}$ *inch*. The RedBlue Lego block is composed of a group of red and blue Lego blocks. The dimension is $2\frac{1}{2} \times 3\frac{1}{8} \times \frac{13}{16}$ *inch*.

Figure 3.9 shows the dual-arm Baxter research robot used to conduct the experiments for all the research work in this thesis research. The camera that built-in the wrist of the Baxter robot can capture images at the maximum resolution of 1280×800. However, an effective image resolution of 640×400 is adopted here. Baxter's arms are also loaded with Infrared Range (IR) Sensors which have a maximum range of 0.4m and a minimum range of 0.04m. The arm of Baxter always keeps crane pose to capture consistent views of the object and makes the picking problem simple.

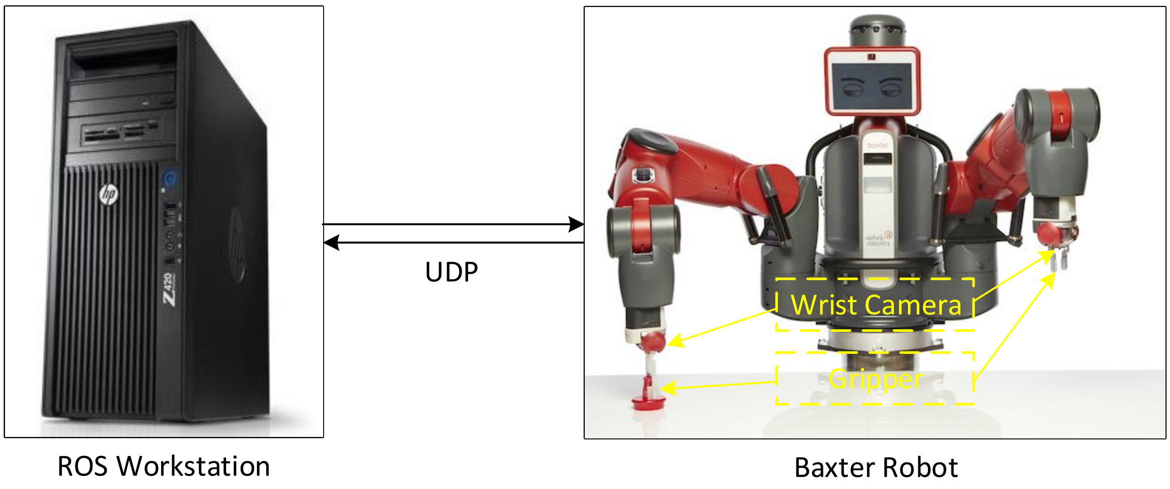


Fig. 3.9 The experiment hardware setup.

3.4.2 Object recognition and picking task

The object recognition and picking task can assess how the Baxter robot is able to learn efficient picking pose from human demonstrations. The robot arm is set at crane pose and kept this pose during the whole experiment. In the beginning, the robot arm located at the height of 38cm above the workbench. The object to be recognized is placed under the robot's wrist camera for scanning. The object's features are abstracted by Line Scan. The robot moves its arm 28cm back and forth above the object to make a synthetic photograph during the line scan. Next, the object's position is estimated by using image matching in the synthetic photograph.

Then, the object is labelled by the human, like RedLegoBlock. After the object position is found, the robot could plan a grasp trajectory using inverse kinematics solver. However, for some objects, the best grasp point is not the geometry centre. For example, the RedBlue Lego block (see Figure 3.8) can only be gripped from the edge as the object is too big for Baxter's gripper to grip around the object's centre. Figure 3.12 shows how the LfD was implemented during the learning of the picking task. The human demonstrator taught the robot to grip the RedBlue Lego block from the edge by kinesthetic guiding. The robot learnt the successful picking pose thereafter.

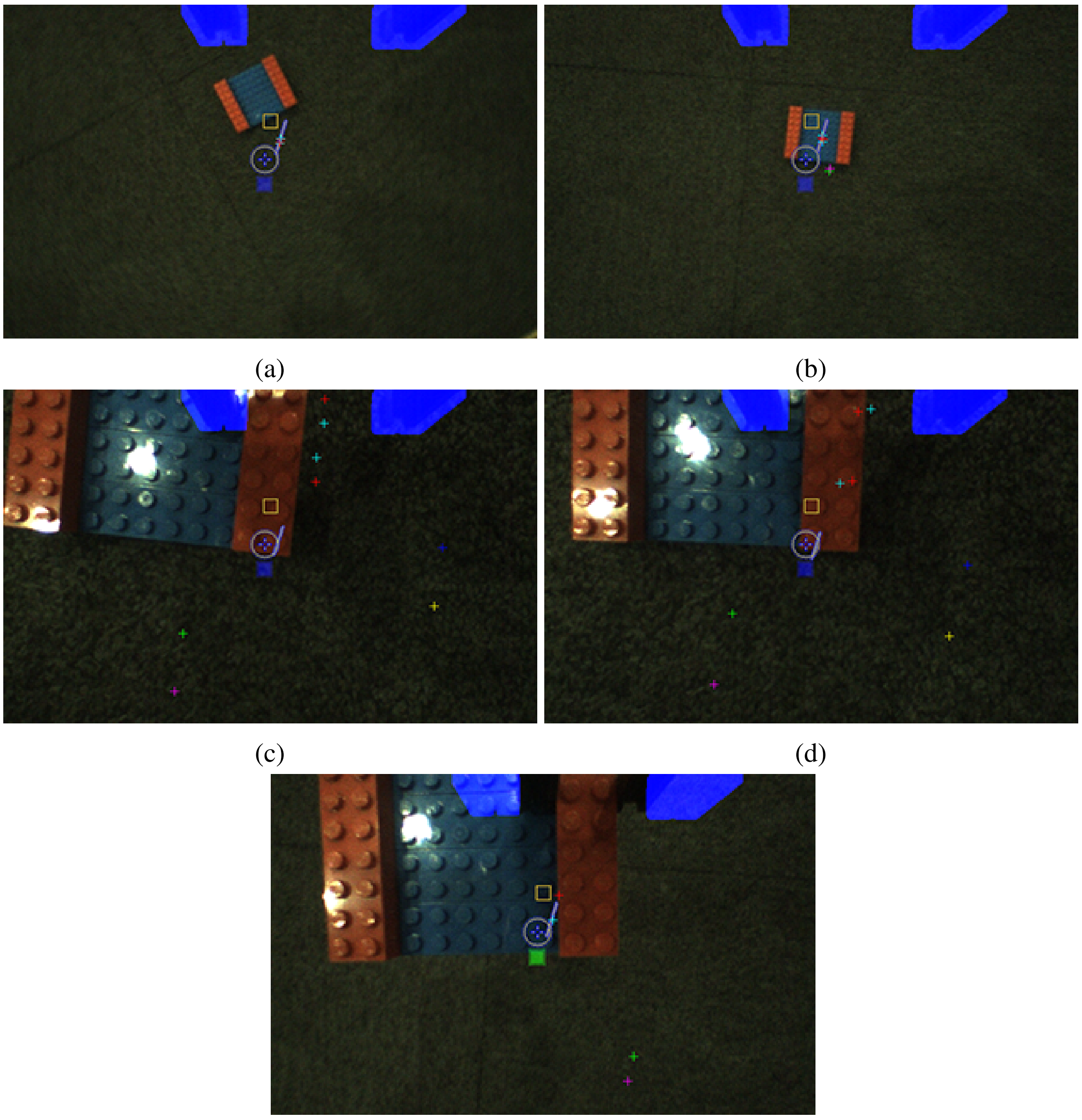
For each trial, the object was placed at a random location on the workbench within approximately 25 cm from the wrist camera's view centre. Three different objects were used to evaluate the recognition ability of the proposed approach, as can be seen in Figure 3.8. Each object was tested for 30 times, and the result is listed in Table 3.1, which clearly shows the excellent performance of the picking ability. The four failures of the Red and RedBlue Lego block was due to the gripper's motor noise during grasping. The RedBlue is more significant than the other two kinds due to the object's complex shape which makes the two-finger gripper hard to grasp.

Table 3.1 Object Recognition and Picking Experiments

Picking Objects	**Picking Performance**		
	Picking Times	***Successful Times***	***Successful Rate***
Red Lego Block	30	29	96.7%
Yellow Lego Block	30	30	100%
RedBlue Lego Block	30	27	90%

3.4.3 Lego blocks assembly task

During the operation of the assembly task, the same Lego blocks, both the Red Lego block and the RedBlue Lego block, were used in the recognition and picking experiments. Figure 3.10 shows the picking process of RedBlueLegoBlock from the robot's wrist camera view. The wrist camera starts scanning the object from the air with the crane pose. Then the bounding box of the object is detected, and the features are extracted. Next, the object is classified based on its features. Last, the robot finds the best picking poses learned from human demonstrations with the previous object classification. Figure 3.11 show the picking process of RedLegoBlock from the robot's wrist camera view, using the same picking strategy as the RedBlueLegoBlock.



(a)

(b)

(c)

(d)

(e)

Fig. 3.10 The picking process of RedBlueLegoBlock from the robot's wrist camera view.

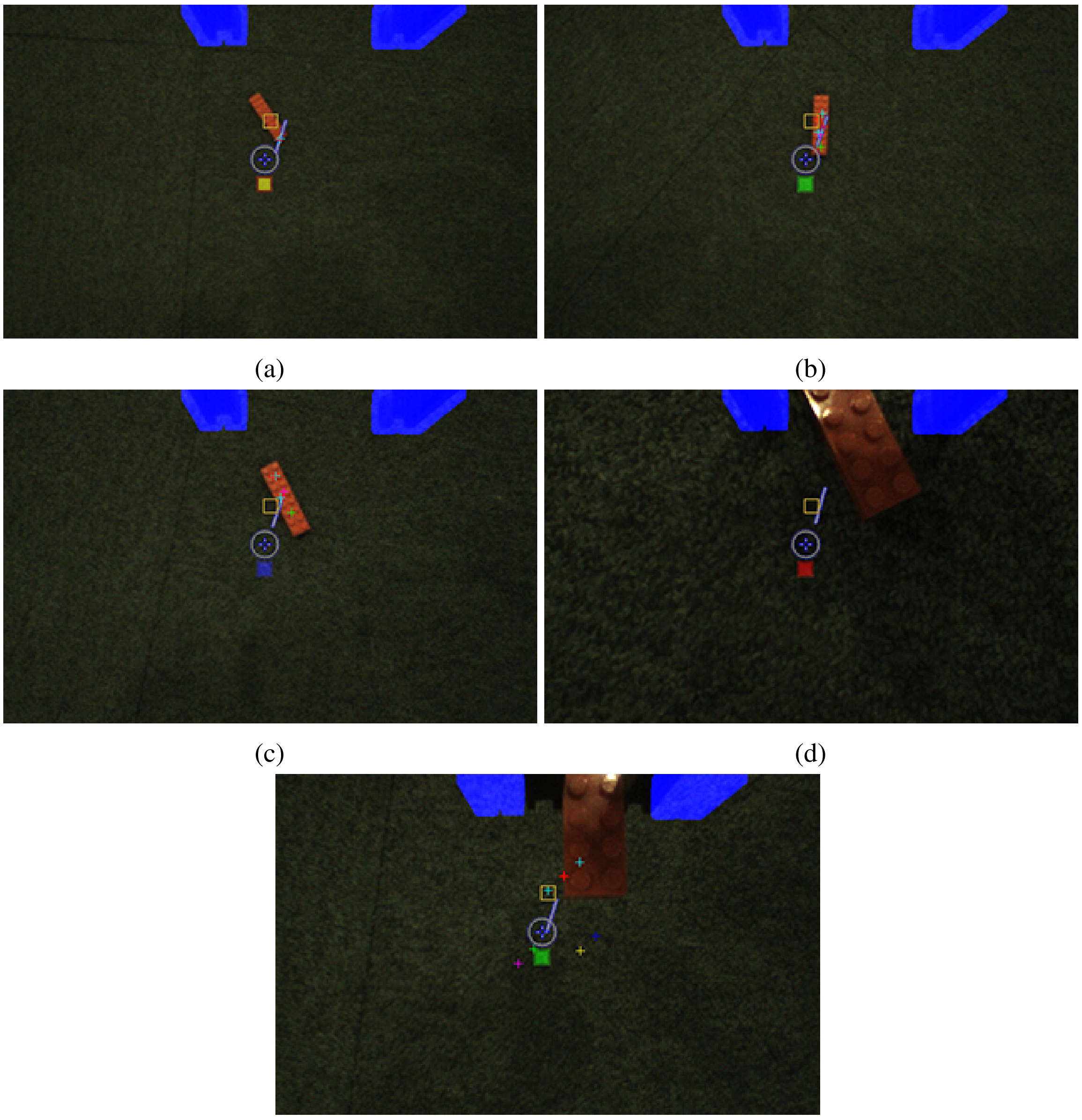

(a)
(b)
(c)
(d)
(e)

Fig. 3.11 The picking process of RedLegoBlock from the robot's wrist camera view.

The Lego block has multiple pegs on one side and multi holes on the opposite side. Therefore, the assembly task is the Peg-in-Hole task, i.e., insert the pegs into the holes. Figure 3.12 shows how a robot learns the skill of picking. More specifically, Figure 3.12a shows that a human demonstrates how to pick a Lego block from an effective position and orientation by kinesthetic guiding. Figure 3.12b shows that the robot reproduces picking skill with the learned object in arbitrary positions.

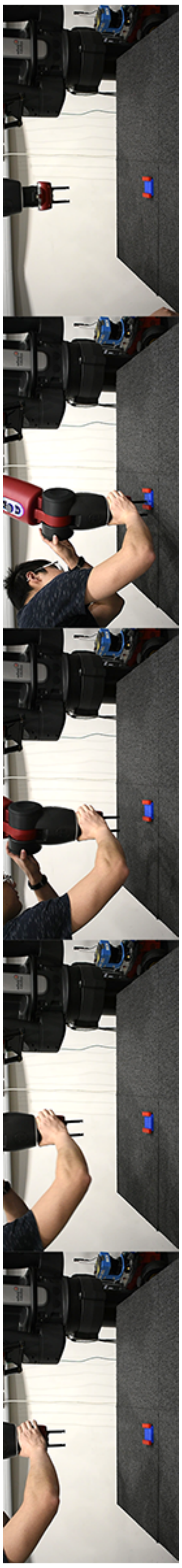
Figure 3.13a shows how the human demonstrator taught the robot to pick up the RedBlue Lego block from the "components zone" to the "assembly zone" by kinesthetic guiding. Then the human demonstrator taught the robot to pick up and move another object (Red Lego block) from the "components zone" to the "assembly zone", finally assemble the two objects, as shown in Figure 3.13b.

To validate if the robot was able to assemble by itself, the RedBlue block was placed under the wrist camera within the "components zone". The robot inferred a good grasping pose and grasped the RedBlue block successfully. After the robot placed the RedBlue block in the "assembly zone", the Red block was placed at a random location. The robot found a suitable grasping pose and assembled the two blocks successfully at last (see Figure 3.13c).

It should be noted that the assembly movement was controlled by a force threshold. When the robot was executing the assembly movement, the force increased gradually until it reached the threshold. The threshold was manually adjusted according to experience data. In this experiment, the threshold was set at $14N$. In the teaching process, the human demonstrator taught the robot the grasping point and orientation, as well as the assembly sequence, which shortened the learning progress of robotic assembly.

3.5 Summary

In this chapter, a new method for learning grasping pose used in an assembly task was proposed. The learning of robotic assembly task is divided into two phases: teaching and


(a) Human demonstrates how to pick a Lego block from an effective position and orientation by kinesthetic guiding.

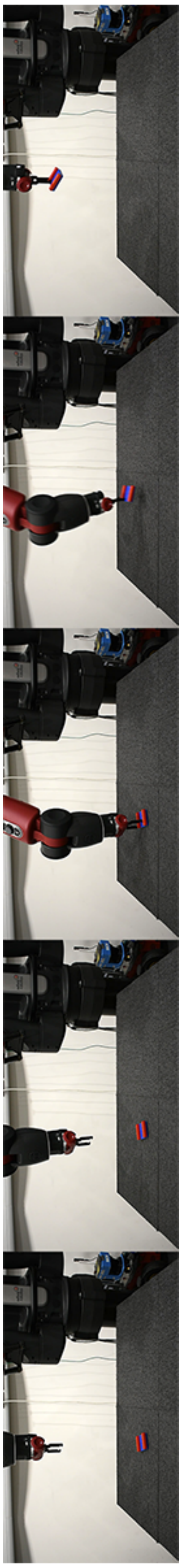
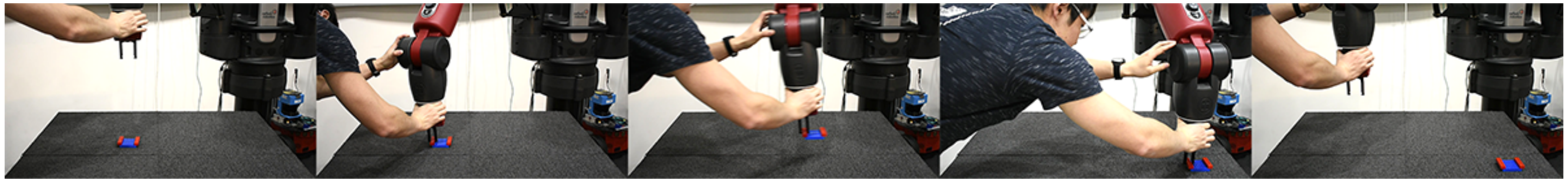
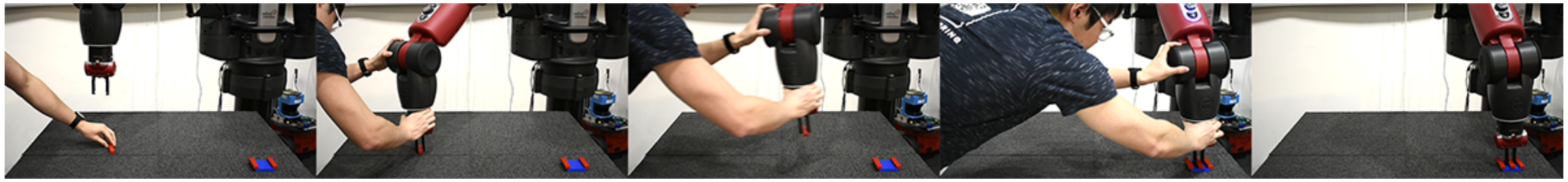

(b) Robot reproduces picking skill with the learned object in arbitrary positions.

Fig. 3.12 Robot learns the skill of picking.


(a) The human demonstrator teaches the robot to pick the first leg block to the assembly location by kinesthetic guiding.


(b) The human demonstrator teaches the robot to pick the second leg block to the assembly location and assemble it into the first leg block.

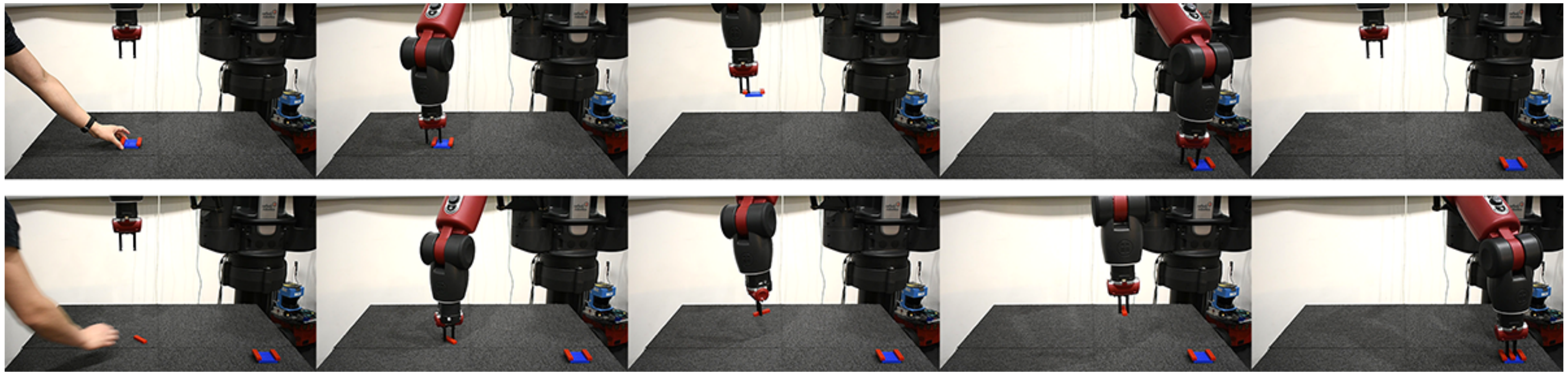

(c) The robot reproduces the assembly task autonomously.

Fig. 3.13 Robot learns the skill of assembly.

reproduction. During the teaching phase, a wrist camera was used to scan an object on the workbench and extract its SIFT feature. The human demonstrator showed the robot how to grasp an object from an effective position and orientation. During the reproduction phase, the robot used the learned knowledge to reproduce the grasping manipulation autonomously. The robustness of the robotic assembly system was evaluated through a series of grasping trials. The dual-arm Baxter robot was used to perform the Peg-in-Hole task by using the proposed approach. Experimental results showed that the robot was able to accomplish the assembly task by learning from human demonstration without traditional dedicated programming.

It should be noted that due to the Baxter's relatively low accuracy, the successful assembly rate could be further improved if using a higher precision robot or bigger size objects. In Chapter 5, the assembly objects are changed to appropriate size that fits Baxter's limit of precision.

Moreover, kinesthetic guiding was used for robot learning. Force control was implemented for controlling assembly movement. The key target was to simplify the teaching process of the assembly task. Experiments from the Lego Blocks assembly task show that the proposed method can be used in teaching robots to do assembly tasks through simple demonstrations. However, further experiments are needed to study the robustness of the system over different assembly tasks, such as slide-in-the-groove, bolt screwing, and finally chair assembly. During the assembly phase, the force control strategy needs to be optimized to ensure a smooth motion and correct the assembly positions. The single-arm manipulation could be extended to dual-arm manipulation so that the additional arm and wrist camera can enable the efficient transfer of assembly skills to robots.

Chapter 4

Human Movement Mapping

Although the kinesthetic guiding method presented in the previous chapter is easy for the robot to 'sense' the demonstrated example, it is not intuitive and unable to be demonstrated in a remote position. This chapter aims to solve the problem and present a mapping algorithm that maps the human motion from human joint space to robot motor space. Section 4.1 introduces the procedure of the human skeleton joints tracked by Kinect V2 sensor. Section 4.3 presents the principle of the space mapping algorithm to transfer the human joint space to the robot motor space. In Section 4.4, Holt's exponential equations are used to reduce the data noise problem. Section 4.5 presents initial results on joint smoothing and joint mapping to show the feasibility and performance of the proposed mapping algorithm. Finally, a summary is given in Section 4.7.

4.1 Introduction

Human gestures are a natural and effective way of communication and allow complex information to be expressed. Using gestures for interactive learning with robots can be of great benefit, particularly in scenarios where traditional input devices are impractical [112]. For the purpose of tracking human full-body pose in real-time, camera-based motion capture systems is usually applied. The camera systems typically require additional markers or suits worn by the person to be tracked. To imitate human motion, a low-level controller takes the generated trajectories as input and generates motor commands for the robot [67].

Colour markers are a simple and effective method for tracking human motions. Acosta-Calderon and Hu [3] implemented a robotic manipulator imitating system by using colour markers that attached to the human demonstrator's arm. The human's shoulder and wrist position data was used to solve the human-robot corresponding problem at the sacrifice of losing dexterity of demonstrated hand motion. Because only two arm joints of the demonstrator were used for robot imitating. The elbow joint of human hand was ignored, which could be useful for planning an obstacle-avoiding trajectory.

To improve the dexterity of human motion tracking, more colour markers were used in recent research [60]. The human shoulder, elbow, and wrist are marked by wearing yellow bandages, and for each one of them, the 3D coordinates (x, y, z) are tracked and derived by the Kinect camera. The human hand gestures could be represented by a 9D representation consisted of the three arm joints. The robot then learns the latent relationship between the human 9D space and the robot arm joint space through hand-by-hand kinesthetic teaching. However, the marker-based visual tracking and learning from kinesthetic teaching could be improved by introducing a marker-less tracking and learning from remote visual teaching method.

In this chapter, a camera tracking system which derives the human joints directly is used. Besides, a new skeleton-joint mapping algorithm is proposed to transfer the human skeleton space to robot joint space. The main contribution of this chapter is:

- A teaching method is proposed for the robot to observe human demonstrations by a visual sensor so that the assembly skill demonstrated by a human can be effectively mapped from task space to configuration space for robot learning. Compared with existing works that use low-accuracy computer-vision-based human tracking method or marker-based tracking with additional devices, the proposed method uses a Kinect-based skeleton-joint mapping with improved accuracy of human motion tracking and without additional markers attached to human demonstrator.

4.2 Human Skeleton Tracking Mechanism

Figure 4.2 shows the human skeleton joints tracked by the Kinect sensor. There are 25 joints numbered from 0 to 24 in sequence: SpineBase, SpineMid, Neck, Head, Shoulder-Left, ElbowLeft, WristLeft, HandLeft, ShoulderRight, ElbowRight, WristRight, HandRight, HipLeft, KneeLeft, AnkleLeft, FootLeft, HipRight, KneeRight, AnkleRight, FootRight, SpineShoulder, HandTipLeft, ThumbLeft, HandTipRight, ThumbRight. Each joint has 11 properties: colour coordinates (x, y); depth coordinates (x, y); camera coordinates (x, y, z); and orientation coordinates (x, y, z, w).

- Colour Coordinates (X, Y). The colour coordinates (x, y) are the coordinates from the colour camera with a resolution of 1920×1080. A (x, y) represents a 2D point on the colour image. The 2D position is a row/column location of a pixel on the image, where $x = 0, y = 0$ is the pixel at the top left of the colour image, and $x = 1919, y = 1079(width - 1, height - 1)$ corresponds to the bottom right pixel.

- Depth Coordinates (X, Y). The depth coordinates are the coordinates from the depth camera with a resolution of 512×424. The depth position describes a 2D location on the depth image. A pixel is a row/column location where x is the row and y is the column. So $x = 0, y = 0$ is the pixel at the top left of the colour image, and $x = 511, y = 423(width - 1, height - 1)$ corresponds to the bottom right corner of the image.

- Camera Coordinates (X, Y, Z). The Kinect uses infrared (IR) sensor to locate 3D points of the joints in space. The camera coordinates are used to describe points referred to

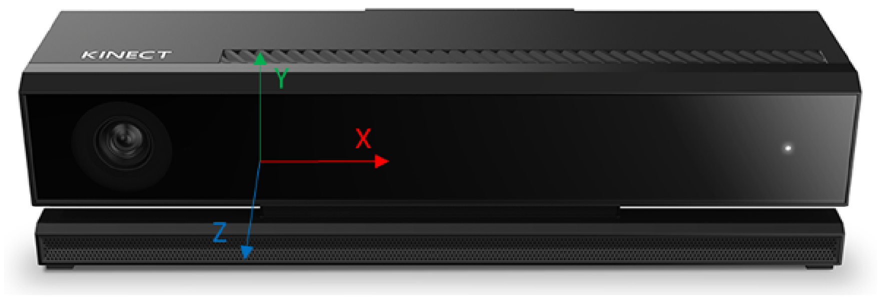


Fig. 4.1 The definition of the camera coordinates of the Kinect sensor. Source: msdn.microsoft.com

the 3D coordinate system used by the Kinect. The origin of the coordinate is located at the centre of the IR sensor on Kinect, see Figure 4.1. X grows to the sensor's left (from the sensor's point-of-view). Y grows up (note that this direction is based on the sensor's tilt). Z grows out in the direction the sensor is facing.

- Orientation Coordinates (X, Y, Z, W). The orientation coordinates are used to deliver the orientation of some joints, like the head. Quaternions are a 4D way to store the 3D orientation and provide a great way to store and animate rotations.

The joint filtering implementation in a typical application receives joint positions from Skeleton Tracking (ST) system as input in each frame and returns the filtered joint positions as output. The filter treats each joint's x, y, and z coordinates independently from other joints or other dimensions. That is, a filter is independently applied to the x, y, and z coordinate of each joint separately—and potentially each with different filtering parameters. Note that, though it is typical to filter the Cartesian position data returned by ST directly, one can apply the same filtering techniques to any data calculated from joint positions.

Figure 4.3 shows an example of skeleton tracking by Kinect in real-time. The green body is the tracked human that is framed at the left corner. The tracked joints are marked on the green body and listed on the top left.

A useful technique to reduce latency is to tweak the joint filter in order to predict future joint positions. That is to say that the filtered output could be a smoothed estimate of joint

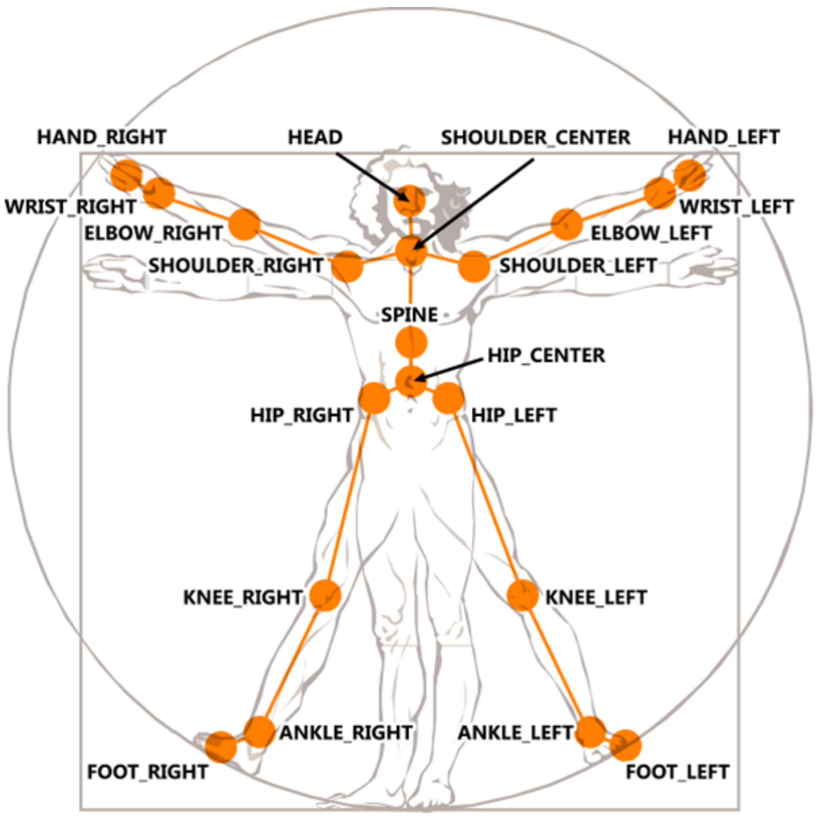
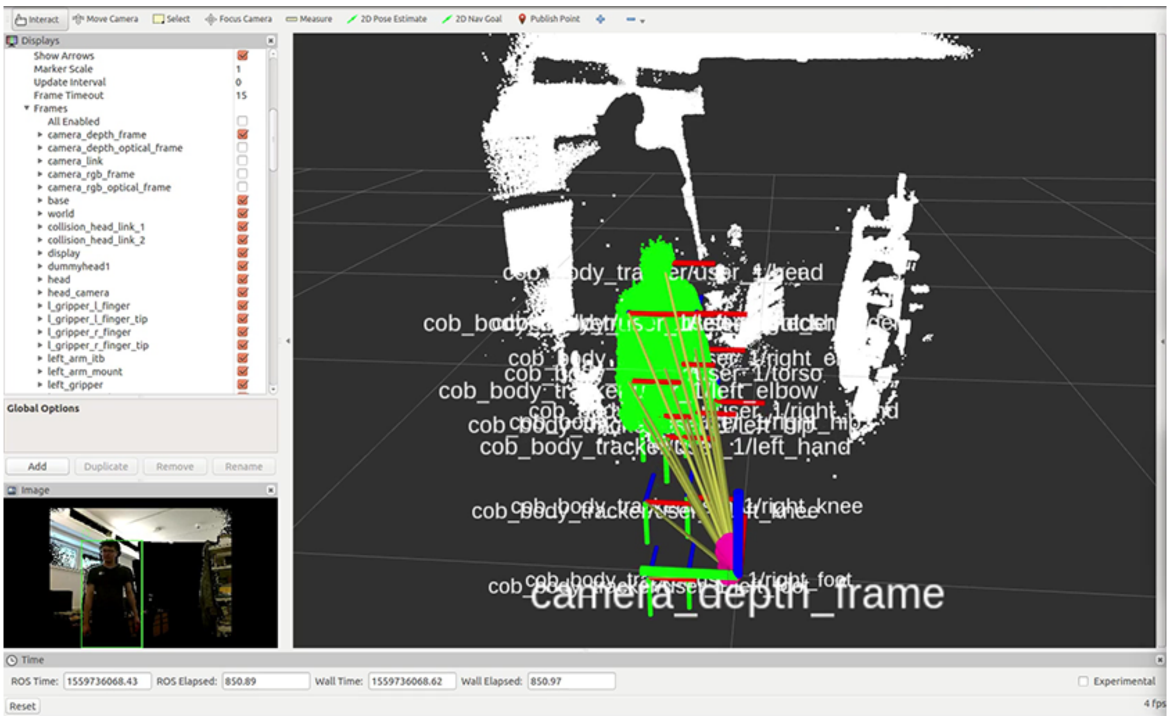


Fig. 4.2 Human skeleton joints tracked by Kinect V2 sensor. There are 25 joints numbered from 0 to 24 in sequence: SpineBase, SpineMid, Neck, Head, ShoulderLeft, ElbowLeft, WristLeft, HandLeft, ShoulderRight, ElbowRight, WristRight, HandRight, HipLeft, KneeLeft, AnkleLeft, FootLeft, HipRight, KneeRight, AnkleRight, FootRight, SpineShoulder, HandTipLeft, ThumbLeft, HandTipRight, ThumbRight. Source: msdn.microsoft.com



Show Arrows	✓
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
All Enabled	☐
camera_depth_frame	✓
camera_depth_optical_frame	☐
camera_link	☐
camera_rgb_frame	☐
camera_rgb_optical_frame	☐
base	✓
world	✓
collision_head_link_1	✓
collision_head_link_2	✓
display	✓
dummyhead1	✓
head	✓
head_camera	✓
l_gripper_l_finger	✓
l_gripper_l_finger_tip	✓
l_gripper_r_finger	✓
l_gripper_r_finger_tip	✓
left_arm_itb	✓
left_arm_mount	✓
left_gripper	✓

ROS Time: 1559736068.43 ROS Elapsed: 850.89 Wall Time: 1559736068.62 Wall Elapsed: 850.97 ☐ Experimental

Fig. 4.3 GUI for Kinect skeleton tracking. The left corner is the human demonstrator who is being tracked. The green human frame in the middle is the tracked body from Kinect point cloud view.

positions in subsequent frames. If forecasting is used, the joint filtering will reduce the overall latency. However, since the predicted outputs are estimated from previous data, the forecasting data may not always be accurate, especially when a movement is suddenly started or stopped. Forecasting may propagate and magnify the noise in previous data to future data, and hence, may increase the noise.

One should understand how latency and smoothness affect the user experience, and identify which one is more important to create a pleasant experience. Then, it is crucial to carefully choose a filter and fine-tune its parameters to match the specific needs of the application. In most Kinect applications, data output from the ST system is used for a variety of purposes. Joints have different characteristics from one another in terms of how fast they can move. For example, in some applications, a person's hands can move much faster than the spine joint, and therefore different filtering techniques should be used for hands than the spine and other joints. No single filtering solution could fit the needs of all joints.

4.3 Human-Robot mapping mechanism

In the example teaching phase, the human skeleton positions are captured by the Kinect sensor. However, the human skeleton data can not be used to control the robot motors immediately as their working spaces are different. To solve the mapping problem between the human joint space (i.e., demonstrating space) and the robot motor space (i.e., configuration space), a human-robot mapping mechanism is introduced into the example teaching phase. The mapping mechanism means to abstract the keyframes of the human movements to define a LfD demonstration example.

Figure 4.4 shows the human skeleton joints that can be tracked by a Kinect sensor. These human skeleton joints are head h_e, torso t, right shoulder s, right elbow e, right hand h_a, left shoulder $s^{'}$, left elbow $e^{'}$, and left hand $h^{'}{}_a$. It should be noticed that the arrow lines are the vectors between two joints, which are used to calculate the angles of the limbs.

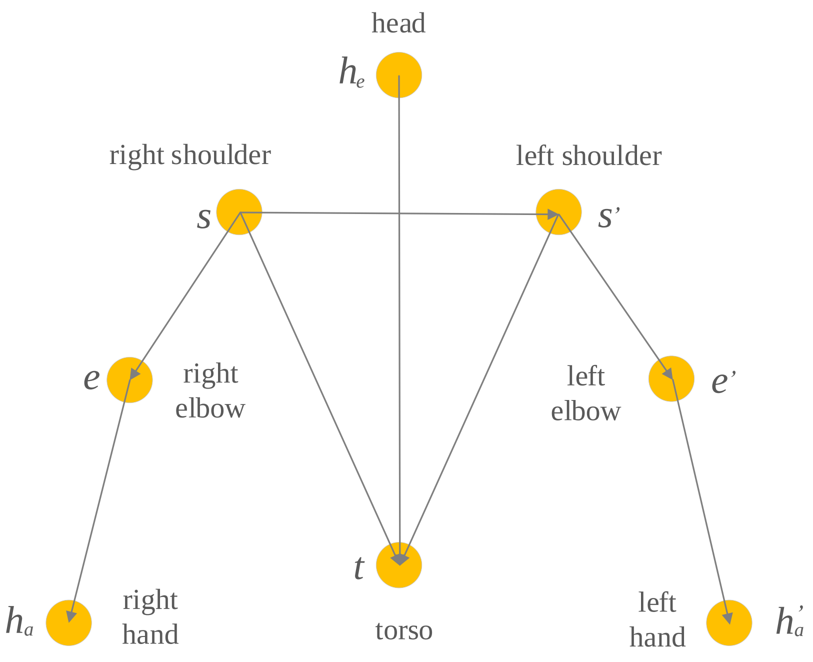


Fig. 4.4 The human skeleton joints tracked by the Kinect sensor are head h_e, torso t, right shoulder s, right elbow e, right hand h_a, left shoulder $s^{'}$, left elbow $e^{'}$, and left hand $h^{'}{}_a$. The arrow lines are the vectors between two joints and used to calculate the angles of the limbs.

Here we firstly consider the mapping of the right human arm to the right arm of a Baxter robot, which is shown in Figure 4.5. The arm joints are named in the following manner: $S0$-Shoulder Roll, $S1$ - Shoulder Pitch, $E0$ - Elbow Roll, $E1$ - Elbow Pitch, $W0$ - Wrist Roll, $W1$ - Wrist Pitch, $W2$ - Wrist Roll. In the proposed method, only the shoulder and elbow joints are used for mapping. The two wrist joints $W1$ and $W2$ are set to keep the default perpendicular position, which simplifies the picking and assembly tasks for the robot.

The vector from head h_e to torso t is $\boldsymbol{V}_{h_e t}$. The vector from the shoulder joint s to the elbow joint e is $\boldsymbol{V}_{se}$. For the plane given by the vectors $\boldsymbol{V}_{se}$ and $\boldsymbol{V}_{h_e t}$, the normal vector is

$$\boldsymbol{n}_1 = \boldsymbol{V}_{se} \times \boldsymbol{V}_{h_e t}. \tag{4.1}$$

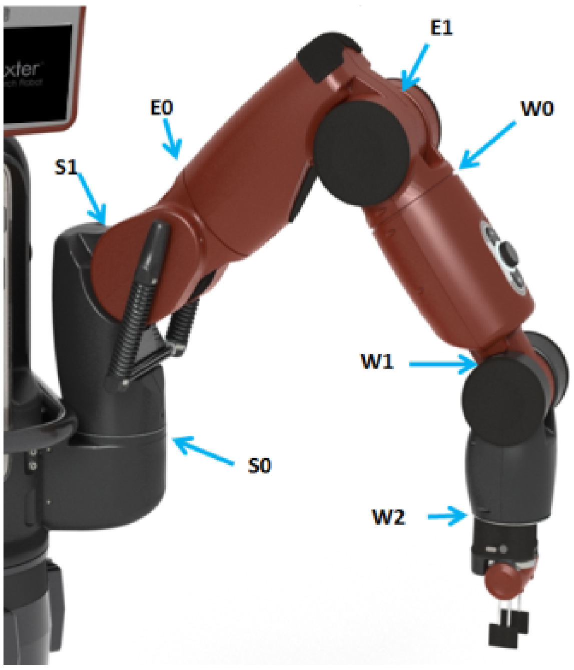


Fig. 4.5 The arm joints of Baxter. They are named in the following manner: $S0$- Shoulder Roll, $S1$ - Shoulder Pitch, $E0$ - Elbow Roll, $E1$ - Elbow Pitch, $W0$ - Wrist Roll, $W1$ - Wrist Pitch, $W2$ - Wrist Roll. In the proposed method, only the shoulder and elbow joints are used for mapping. The two wrist joints $W1$ and $W2$ are set to keep the default perpendicular position, which simplifies the picking and assembly tasks for the robot. Source: http://sdk.rethinkrobotics.com/wiki/File:Baxter_arm.png

Let the $\boldsymbol{V}_{ss'}$ represent the vector from the right shoulder s to the left shoulder s', then the angle θ_0 between $\boldsymbol{n_1}$ and $\boldsymbol{V}_{ss'}$ is

$$cos\theta_0 = \frac{\boldsymbol{n_1} \cdot \boldsymbol{V}_{ss'}}{||\boldsymbol{n_1}||\,||\boldsymbol{V}_{ss'}||}. \tag{4.2}$$

The mapping motor value for robot joint S_0 is

$$S_0 = w_{s0}(\frac{\pi}{4} - \theta_0) = w_{s0}(\frac{\pi}{4} - \arccos\frac{\boldsymbol{n_1} \cdot \boldsymbol{V}_{ss'}}{||\boldsymbol{n_1}||\,||\boldsymbol{V}_{ss'}||}), \tag{4.3}$$

where w_{s0} is the scale factor that used to adjust the sensitivity of the mapping for robot joint S_0. For the angle θ_1 between the vectors $V_{h_{et}}$ and $\boldsymbol{V}_{se}$,

$$cos\theta_1 = \frac{\boldsymbol{V}_{h_{et}} \cdot \boldsymbol{V}_{se}}{||\boldsymbol{V}_{h_{et}}||\,||\boldsymbol{V}_{se}||}. \tag{4.4}$$

Then the mapping motor value for robot joint S_1 is

$$S_1 = w_{s1}(\frac{\pi}{2} - \theta_1) = w_{s1}(\frac{\pi}{2} - \arccos\frac{\boldsymbol{V}_{h_{et}} \cdot \boldsymbol{V}_{se}}{||\boldsymbol{V}_{h_{et}}||\,||\boldsymbol{V}_{se}||}), \tag{4.5}$$

where w_{s1} is the scale factor that used to adjust the sensitivity of the mapping for robot joint S_1.

Let $\boldsymbol{V}_{eh_a}$ present the vector from the right elbow e to the right hand h_a, for the plane given by the vectors $\boldsymbol{V}_{se}$ and $\boldsymbol{V}_{eh_a}$, the normal vector is

$$\boldsymbol{n_2} = \boldsymbol{V}_{se} \times \boldsymbol{V}_{eh_a}. \tag{4.6}$$

Then the angle θ_2 between $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ is

$$cos\theta_2 = \frac{\boldsymbol{n_1} \cdot \boldsymbol{n_2}}{||\boldsymbol{n_1}||\,||\boldsymbol{n_2}||}. \tag{4.7}$$

Here we get the mapping motor for the robot joint E_0,

$$E_0 = w_{e0}\theta_2 = w_{e0} \arccos \frac{\boldsymbol{n_1} \cdot \boldsymbol{n_2}}{||\boldsymbol{n_1}||\,||\boldsymbol{n_2}||}, \tag{4.8}$$

where w_{e0} is the scale factor that used to adjust the sensitivity of the mapping for robot joint E_0.

Let θ_3 represents the angle between the vectors $\boldsymbol{V}_{se}$ and $\boldsymbol{V}_{eh_a}$, then

$$cos\theta_3 = \frac{\boldsymbol{V}_{se} \cdot \boldsymbol{V}_{eh_a}}{||\boldsymbol{V}_{se}||\,||\boldsymbol{V}_{eh_a}||}, \tag{4.9}$$

Here we get the mapping motor for the robot joint E_1,

$$E_1 = w_{e1}\theta_3 = w_{e1} \arccos \frac{\boldsymbol{V}_{se} \cdot \boldsymbol{V}_{eh_a}}{||\boldsymbol{V}_{se}||\,||\boldsymbol{V}_{eh_a}||}, \tag{4.10}$$

where w_{e1} is the scale factor that used to adjust the sensitivity of the mapping for robot joint E_1.

Now, all the joint values are ready for mapping from the human demonstrator to the robot. The scale factors w_{s0}, w_{s1}, w_{e0}, and w_{e1}, used to adjust the sensitivity of the mapping for robot joint S_0, S_1, E_0 and E_1, are set at different values. In the ideal situation, these values are all equal to 1. In the practical experiment, these scale parameters are set around 1 to control the sensitivity of the mapping. It should be noted that the rest robot joints, W_0, W_1, and W_2, stay at the initial positions. In the robotic assembly tasks (see Section 5.3.4), the joint W_1 is set to an initial position that keeps the robot hand at the perpendicular pose to help the learning of assembly tasks.

4.4 Holt's Two Parameter Exponential Smoothing

Holt's model is known as linear exponential smoothing, which is a popular smoothing model for forecasting data with trend [45]. It uses two parameters, one is used for the overall smoothing, and the other is used for the trend smoothing equation. The method is also called double exponential smoothing or trend-enhanced exponential smoothing. Holt's model consists of three separate equations that work together to compute a final prediction. The first equation conducts a basic smoothing to directly control the last smoothed value for the last period's trend. The trend itself is presented as the difference between the last two smoothed values and updated over time through the second equation. The third equation is used to produce the final forecast.

Local optimization and filtering have been widely exploited in model-based 3D human motion capture. As a promising alternative solution for tracking and initialization, global stochastic optimization has recently been proposed. As we know, single smoothing does not excel in catching the data when there is a trend. This situation can be strengthened by introducing a second equation with a second constant, γ, which must be chosen in conjunction with smoothing factor α. Assume that the raw data sequence of observations is expressed as y_t, starting at time $t = 0$. S_t is used to represent the smoothed value for time t, and b_t is our best estimate of the trend at time t. The first two equations associated with Double Exponential Smoothing are as follows [45]:

$$S_t = \alpha y_t + (1-\alpha)(S_{t-1} + b_{t-1}), 0 \leq \alpha \leq 1 \tag{4.11}$$

$$b_t = \gamma(S_t - S_{t-1}) + (1-\gamma)b_{t-1}, 0 \leq \gamma \leq 1 \tag{4.12}$$

where α is the data smoothing factor and γ is the trend smoothing factor.

Finally, the forecast of the algorithm $\hat{y}_{t+k}$, for any $k > 1$ at time $t+k$ based on the raw data up to time t is given by:

$$\hat{y}_{t+k} = S_t + kb_t \tag{4.13}$$

Note that the current value of the iterations is used to compute its smoothed value replacement in double exponential smoothing. There are a variety of strategies to set initial values for S_t and b_t in double smoothing. Generally, S_1 is set to y_1. For the initialization of b_1, we set it as bellow:

$$b_1 = y_2 - y_1 \tag{4.14}$$

The first smoothing equation (4.11) adjusts S_t directly for the trend of the previous period, b_{t-1}, by adding it to the last smoothed value, S_{t-1}. The equation helps to eliminate the lag and updates S_t to the appropriate base of the current value. The second smoothing equation (4.12) then updates the trend, which is represented as the difference between the last two values.

The forecast function is no longer flat but trending. The k-step-ahead forecast is equivalent to the last estimated level plus k times the last estimated trend value. The values for α and γ can be obtained via non-linear optimization techniques, such as the Levenberg–Marquardt algorithm. Experiments on the human joints smoothing are presented in the next section.

4.5 Experiment Results

4.5.1 Experiment setup

The demonstrator's motions are recorded by a depth visual sensor Kinect, which provides the colour information and the position in a three-dimensional Cartesian coordinate system. The Kinect is connected to a Ubuntu workstation through USB cable. The Baxter robot used in the experiment is running in a simulator Gazebo. All the computation and information are implemented through the ROS platform. From the software level, the Kinect sensor works as ROS nodes that perform the computation of visual related messages, such as extracting human skeleton positions. Gazebo is a robotic simulator which allows the user to create 3D scenarios on the computer with robots, obstacles and many other objects. Gazebo supports illumination, gravity, inertia, etc, with a physical engine. Compared with the real robot, the simulator can be used for evaluating and testing the robot in difficult or dangerous environments without worrying about harming the robot. Besides, usually, it is faster to run a simulator with the same environment than operating with the real robot system and the real experiment setup. Figure 4.6 gives an overview of the experiment platform.

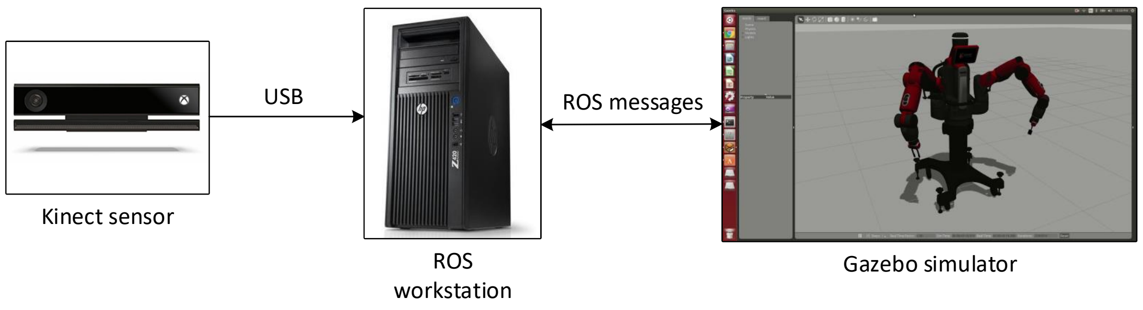


Fig. 4.6 The experiment hardware setup, including a Kinect sensor, a ROS workstation and a Gazebo simulator installed in the ROS.

4.5.2 Software implementation

This section introduces how the experimental environment is implemented with ROS. There are many nodes running in the ROS package, like handling the interface, processing the various messages, transforming between multiple coordinate frames, and so on. Here only major nodes that consist of the proposed method are presented. The core nodes used for constructing the human skeleton-joint map is presented in Figure 4.7, namely *cob_body_tracker, kinect2_launch_joint_sub, kinect2_launch_joint_pub*, and *baxter_arm_control*. More specifically,

- *cob_body_tracker* node tracks human skeleton positions and smooths the joints with Holt's smoothing algorithm before publishing joint data to *human skeleton position topic*.
- *kinect2_launch_joint_pub* node subscribes the human skeleton positions from the *cob_body_tracker* node through the *human skeleton position topic* and handles the computing of human skeleton-joint mapping.
- *kinect2_launch_joint_sub* node subscribes the mapped robotic motor values from the *kinect2_launch_joint_pub* node through *robot joints topic* and then publishes motor values to *baxter_arm_control* node through *robot joints command topic* for the robot to perform the mapped motions.

Firstly, the code B.1 is used to retrieve the human skeleton joints' 3D positions in the Cartesian coordinate system, i.e., the human task space. The human skeleton joints used for building the *human skeleton-joint map* are head, torso, right shoulder, right elbow, right hand, left shoulder, left elbow, and left hand.

Then the tracked joints' positions are smoothed by Holt's two-parameter exponential algorithm, see code B.2.

Next, the smoothed joints' positions are published to given topics, see code B.3.

In the node of *kinect2_launch_joint_pub*, the positions of human skeleton joints are retrieved through ROS *tf* tool and mapped into robotic arm joints values. The *tf* tool maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time, see code B.4.

After the mapping process, the corresponding robotic arm joints' values are returned. Next, these motor values are further published to given topics, see code B.5.

In the node of *kinect2_launch_joint_sub*, the motor values are subscribed and sent to the robot for performing motions, see code B.6.

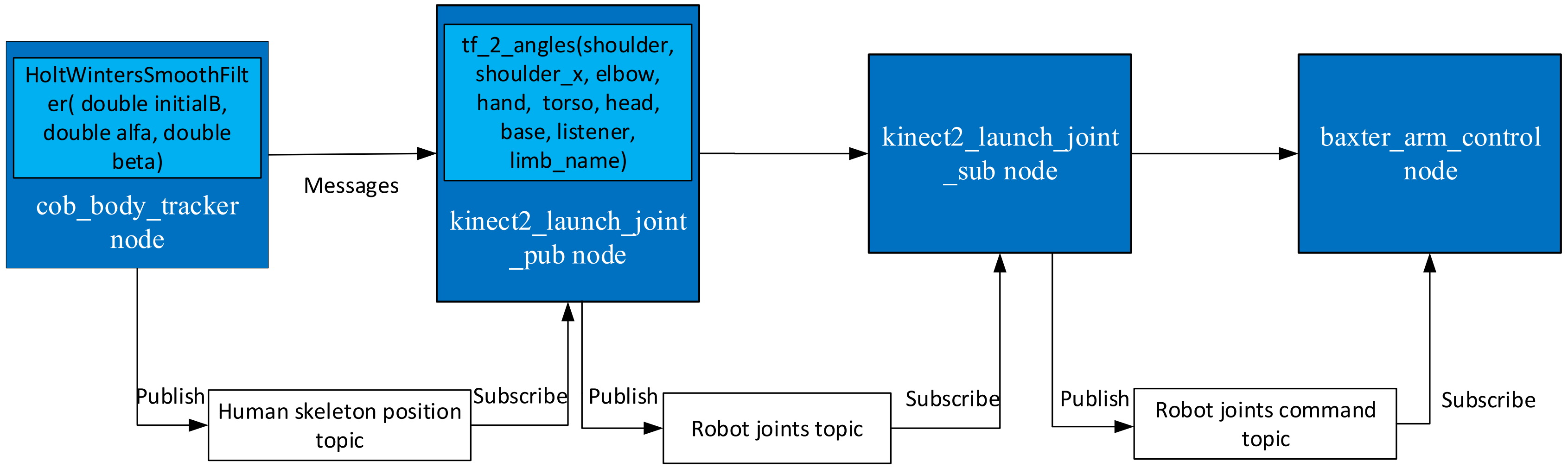

HoltWintersSmoothFilter(double initialB, double alfa, double beta)
cob_body_tracker node
Messages
tf_2_angles(shoulder, shoulder_x, elbow, hand, torso, head, base, listener, limb_name)
kinect2_launch_joint _pub node
kinect2_launch_joint _sub node
baxter_arm_control node
Publish
Human skeleton position topic
Subscribe
Publish
Robot joints topic
Subscribe
Publish
Robot joints command topic
Subscribe

Fig. 4.7 The implementation of the human skeleton-joint mapping with ROS and the core nodes used for computing and passing messages. *cob_body_tracker node* tracks human skeleton positions and publishes the smoothed joints data to *human skeleton position topic*. *kinect2_launch_joint_pub* node subscribe the human skeleton positions from the *cob_body_tracker* node through the *human skeleton position topic* and handles the computing of human skeleton-joint mapping. *kinect2_launch_joint_sub* node subscribes the mapped robotic motor values from the *kinect2_launch_joint_pub* node and then publishes motor values to *baxter_arm_control* node for the robot to perform the mapped motions.

Last, the robot receives the motor position commands and perform the commands, see B.7.

4.5.3 Joints smoothing experiments

The joint smoothing experiments show how the exponential smoothing improves the smoothness of human skeleton position data recorded from the Kinect sensor. Figure 4.8 gives an overview of the joints smoothing experiments and the human-robot skeleton mapping experiments in Section 4.5.4.

The human demonstrator stands in front of the Kinect sensor, moving the upper limbs. The Kinect sensor detects the human skeleton and extracts joint positions, which are further smoothed with the double exponential smoothing algorithm. The skeleton-joint mapping and robot imitating steps will be presented in Section 4.5.4.

The smoothing algorithm is tested with real human joints data. The human demonstrates a rolling gesture with his right hand, 81 frames are recorded. In each frame, the data consists of right shoulder, left shoulder, right elbow, right hand, torso and head.

The criteria of optimising the two parameters α and γ of double exponential algorithm consist of keeping the smoothness of the trajectory and containing the MSE (mean squared error) of the smoothed trajectory. The MSE is define as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{4.15}$$

where n is the total number of frames, y_i is the original position of human skeleton joints, $\hat{y}_i$ is the smoothed value of human skeleton joints. The MSE and the smoothness is a tradeoff problem, which needs to be balanced between low MSE and smoothing trajectory. This is a multi-objective optimisation problem. We first manually narrow down the range of α to $\{0.10, 0.30\}$ and γ to $\{0.60, 0.85\}$ based on the smoothing applied on different human skeleton joints, as presented in Figure 4.9, 4.10, and Figure A.1 – A.10. Then we optimise

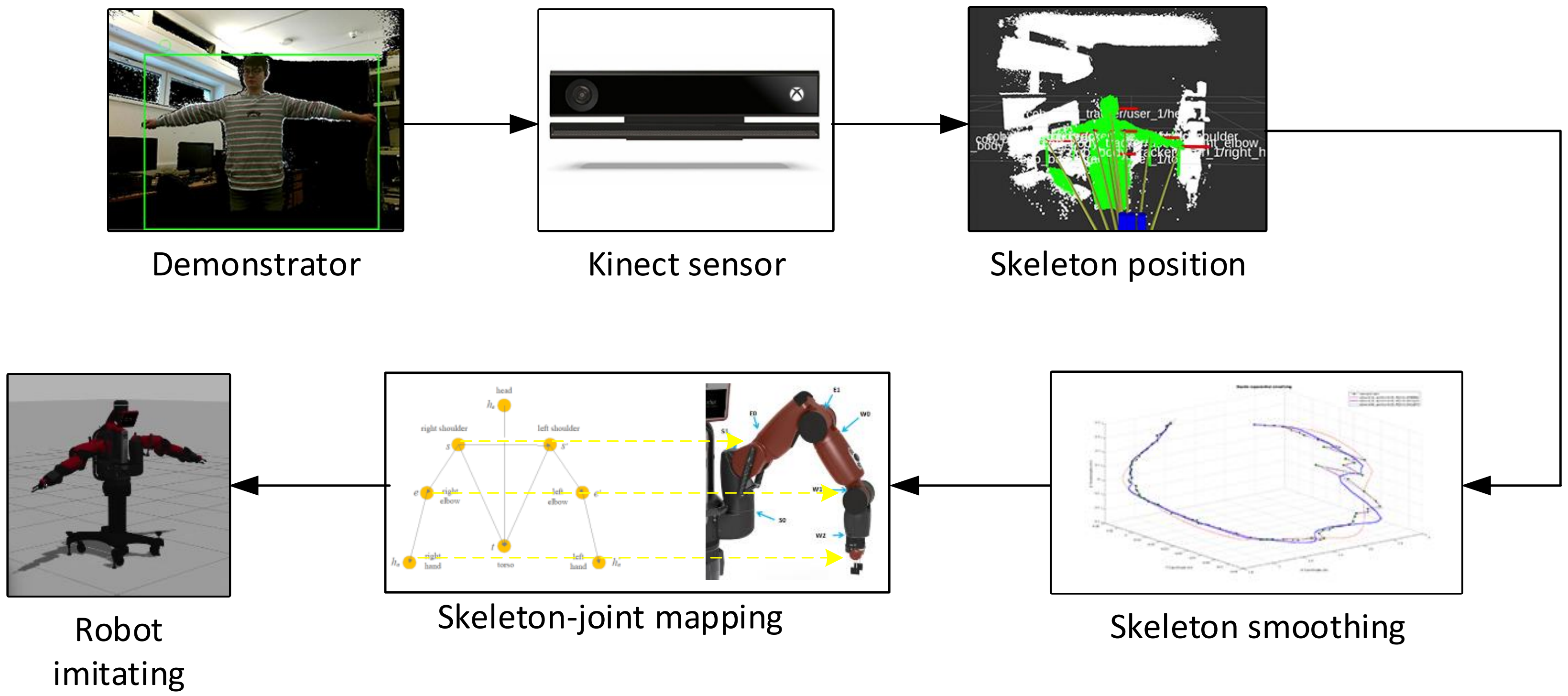


Fig. 4.8 The flowchart of human-robot skeleton mapping. The human demonstrator demonstrates in front of the Kinect. The human's skeleton positions are then tracked by the robot. Next, the human skeleton positions are transferred into robot joint values through the *skeleton-joint mapping* after being smoothed. Then the robot imitates human motions from the transferred joint values.

the MSE within the above range, i.e, $\alpha \in \{0.10, 0.30\}$ and $\gamma \in \{0.60, 0.85\}$, as shown in Figure 4.11 and Figure A.11 – A.15. A detailed distribution of the MSE with some selected parameter samples is presented in Table 4.1.

- Figure 4.9 shows that the human hand joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$. When increase the α, the smoothing performance decreases while the precision increases.
- Figure 4.10 shows that the human hand joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$. When increases the γ, the smoothing performance slightly decreases while the precision increases.
- Other human skeleton joints, like elbow, right shoulder, left shoulder, torso, and head, have same trend as the hand joint. These figures are placed in Appendix A, see Figure A.1–A.10. To balance the smoothness and precision, we manually select $\alpha \in \{0.10, 0.30\}$ and $\gamma \in \{0.60, 0.85\}$ as the preferred parameters of double exponential algorithm for human skeleton joint smoothing.

To compute the smoothing factors α and γ of the equations 4.11 and 4.12, a trial-and-error method was used: simply running the algorithm over and over again to obtain a MSE distribution figure, such as Figure 4.11, it is the MSE distribution of human hand joint after being smoothed based on different smoothing factors. The top picture is the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.082e-04, where\ \alpha = 0.61, \gamma = 0.69$. The picture at the bottom is the local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.5256e-04, where\ \alpha = 0.30, \gamma = 0.85$. More figures are placed in the Appendix A, see Figure A.11 to A.15. From these figures we can see that the optimum parameter set is $\alpha = 0.30, \gamma = 0.85$. It should be noted that this pair of parameters is not the

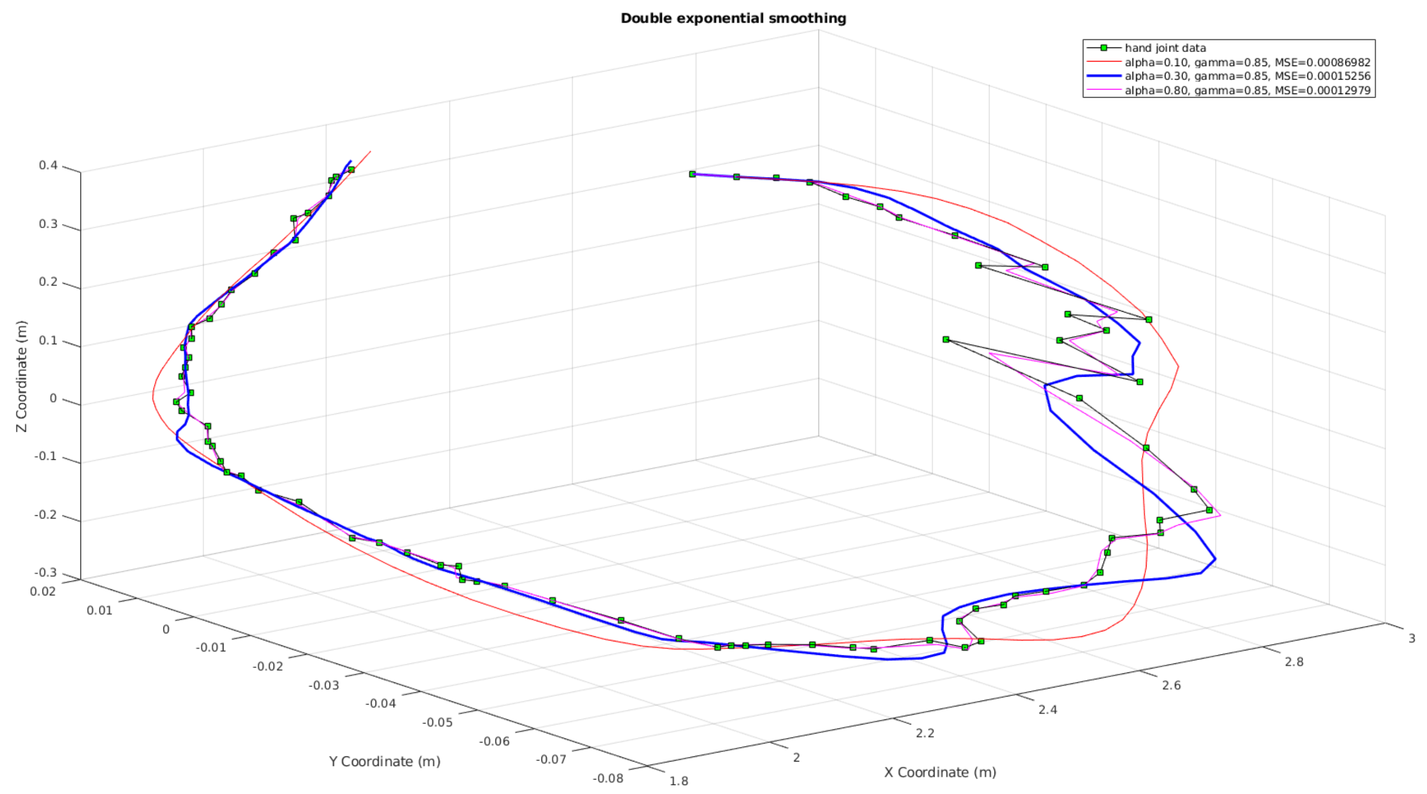


Fig. 4.9 Double exponential smoothing applied on human hand joint data with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.

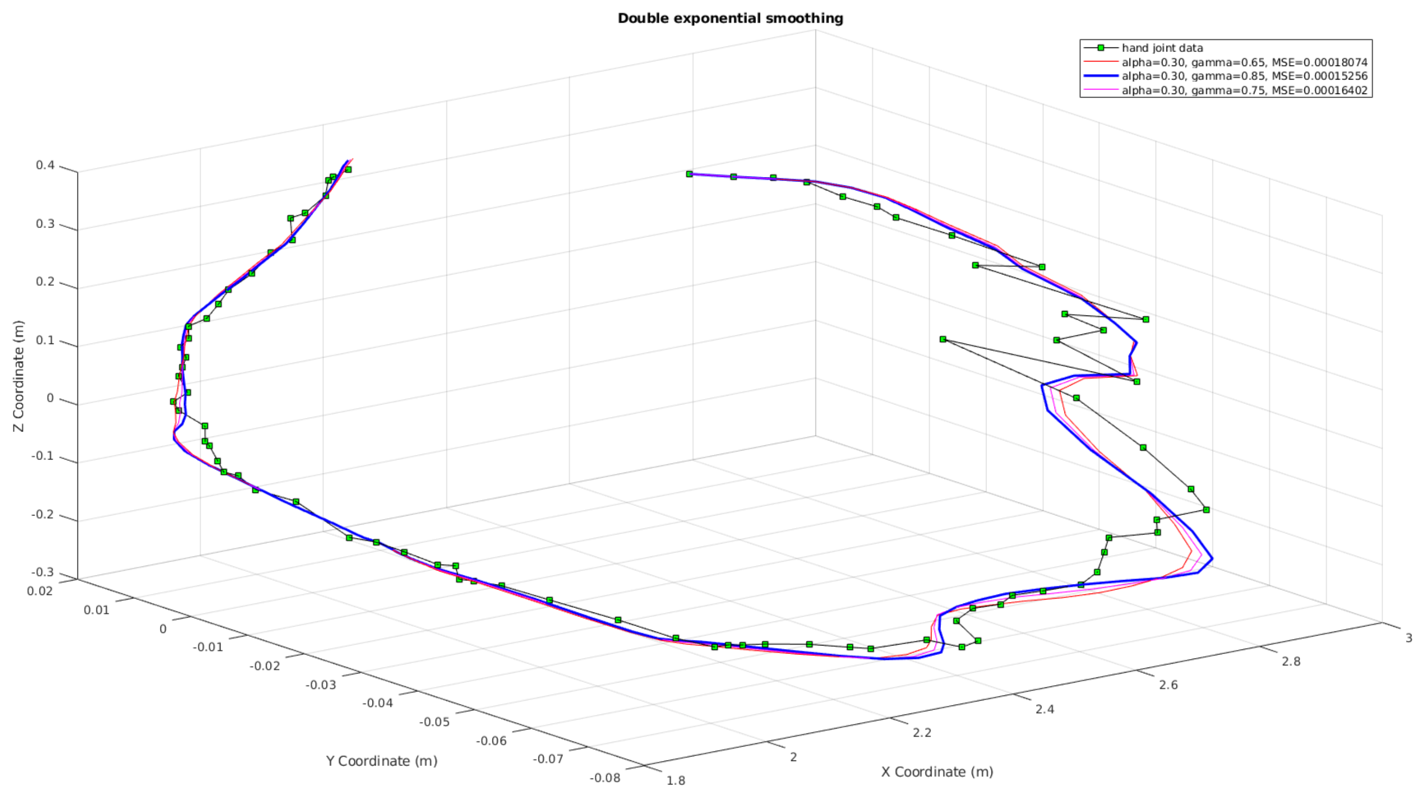


Fig. 4.10 Double exponential smoothing applied on human hand joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

Table 4.1 Double exponential smoothing applied on human joint data with different α and γ

	α	γ	MSE		α	γ	MSE
Hand joint	0.10	0.85	8.6982×10^{-4}	Elbow joint	0.10	0.85	3.5977×10^{-4}
	0.30	0.85	1.5256×10^{-4}		0.30	0.85	1.0231×10^{-4}
	0.80	0.85	1.2979×10^{-4}		0.80	0.85	9.3858×10^{-5}
	0.30	0.65	1.8074×10^{-4}		0.30	0.65	1.0971×10^{-4}
	0.30	0.75	1.6402×10^{-4}		0.30	0.75	1.0557×10^{-4}
	0.30	**0.85**	1.5256×10^{-4}		**0.30**	**0.85**	1.0231×10^{-4}
Right shoulder joint	0.10	0.85	5.2168×10^{-5}	Left shoulder joint	0.10	0.85	4.8778×10^{-5}
	0.30	0.85	8.8919×10^{-6}		0.30	0.85	1.0029×10^{-5}
	0.80	0.85	5.2011×10^{-6}		0.80	0.85	5.7503×10^{-6}
	0.30	0.65	1.1243×10^{-5}		0.30	0.65	1.0928×10^{-5}
	0.30	0.75	9.9690×10^{-6}		0.30	0.75	1.0418×10^{-5}
	0.30	**0.85**	8.8919×10^{-6}		**0.30**	**0.85**	1.0029×10^{-5}
Torso joint	0.10	0.85	1.0975×10^{-5}	Head joint	0.10	0.85	1.2621×10^{-5}
	0.30	0.85	4.2925×10^{-6}		0.30	0.85	5.0714×10^{-6}
	0.80	0.85	2.7519×10^{-6}		0.80	0.85	5.0699×10^{-6}
	0.30	0.65	4.2920×10^{-6}		0.30	0.65	5.1497×10^{-6}
	0.30	0.75	4.2947×10^{-6}		0.30	0.75	5.1195×10^{-6}
	0.30	**0.85**	4.2925×10^{-6}		**0.30**	**0.85**	5.0714×10^{-6}

optimum for torso trajectory as the torso joint hardly move during human demonstrations. In addition, the MSE is almost same as the one when $\alpha = 0.30, \gamma = 0.665$.

Table 4.1 summarises the distribution of the MSE with some selected parameter samples. From the table, considering the smoothing optimisation in Figure 4.9, 4.10, and Figure A.1–A.10, it is clear that $\alpha = 0.30$ and $\gamma = 0.85$ ensure a low MSE and avoid over-smoothing.

4.5.4 Joint mapping experiments

In this section, we illustrate the effectiveness of the proposed method through human gesture mapping experiments. The flowchart of the experiment scenario is presented in Figure 4.8. The joint mapping experiments are extended from the previous joint smoothing experiments. After the skeleton smoothing, the proposed skeleton-joint mapping is applied to the human

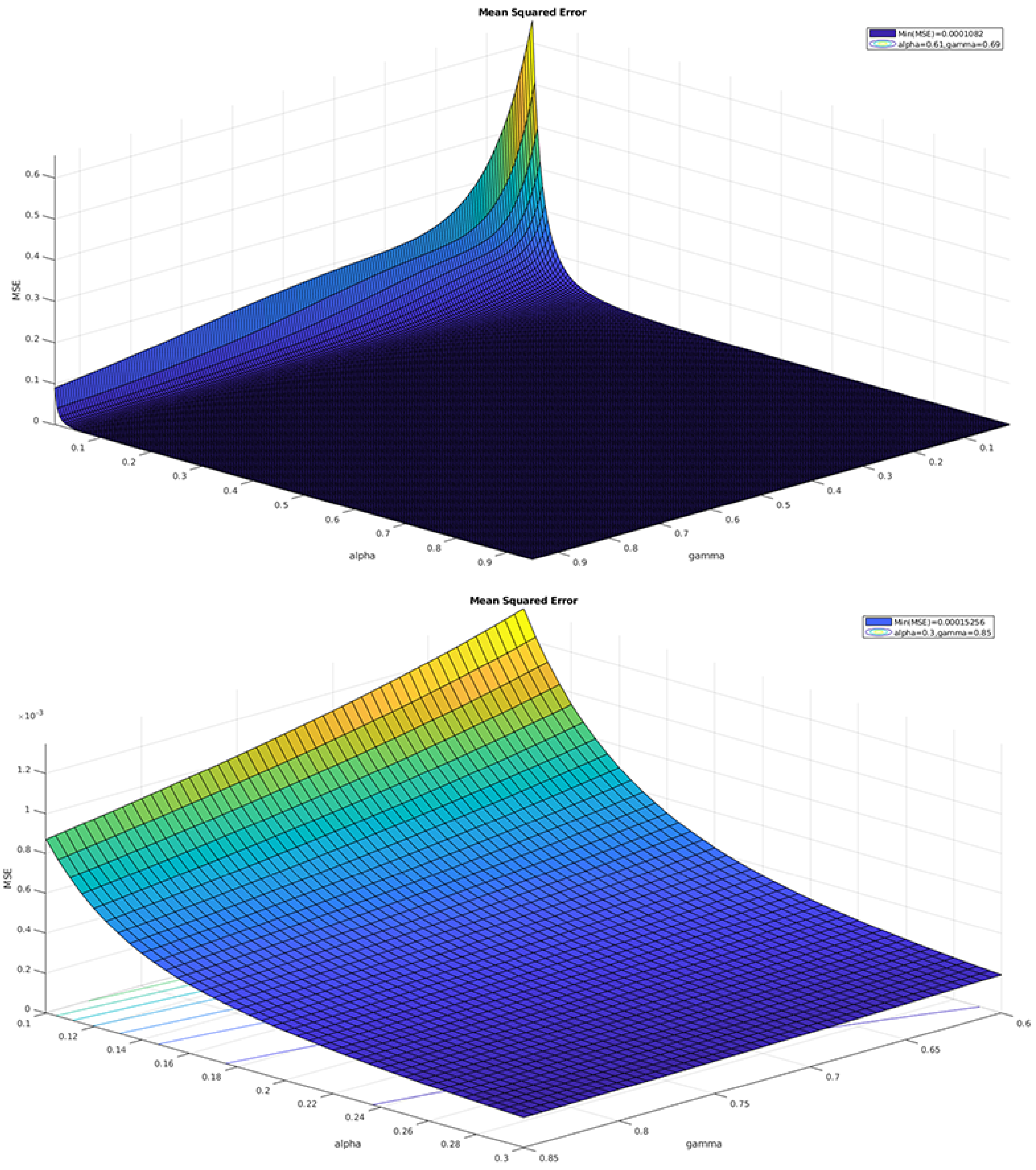


Fig. 4.11 The MSE distribution of human hand joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.082e - 04, where\ \alpha = 0.61, \gamma = 0.69$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.5256e - 04, where\ \alpha = 0.30, \gamma = 0.85$.

motions. Then we get a converted motion for the robot that fits the robot's motor configuration space. Lastly, the robot imitates the learned motions in the Gazebo simulator.

Figure 4.12 shows skeleton mapping from human to robot with a Gazebo simulator. The mapping algorithm is tested with three symbolic gestures: lateral raising (Figure 4.12a), forward reaching (Figure 4.12b), and forward bending (Figure 4.12c).

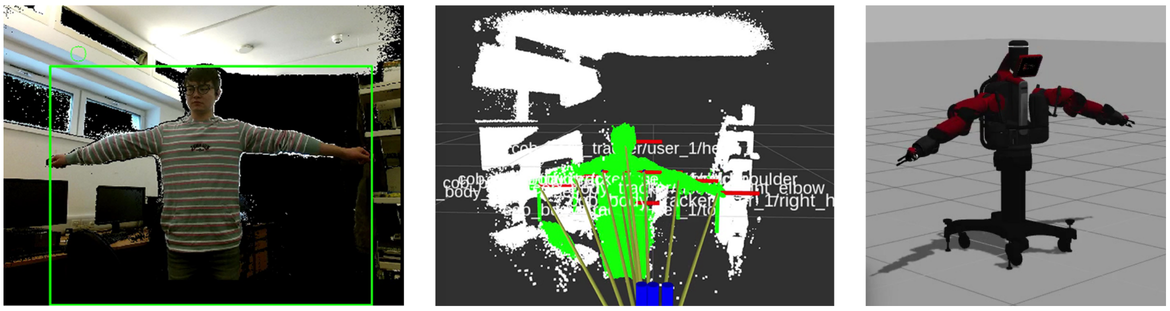
More specifically, the detailed description of three columns in Figure 4.12 is given below.

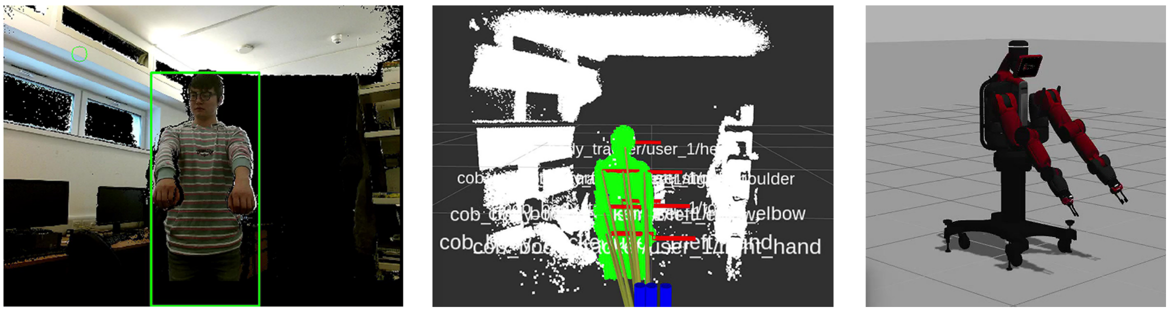
- The left column in Figure 4.12 is what the robot sees through the Kinect camera, from where the human demonstrator is detected and separated from the background environments.
- The middle column shows the human joints tracking information, which is highlighted with green colour. The tracked arm and torso joints are marked with unique names. The blue bar at the bottom means the coordinate origin of the Kinect depth frame, on whom all of the joint's positions are based. These human joints information is transferred into robot motor values using the joint mapping algorithm.
- The right column shows the robot imitating the demonstrator's gestures through the proposed mapping approach.

As can be seen from Figure 4.12b, the robot learns to reach forward, but its arms are a little bit lower than the ones of the human demonstrator. The reason is that the robot reaches the limitation, as the robot's manipulation space is not as big as human's.

4.6 Discussion

The effectiveness of the skeleton-joint mapping is evaluated in quality rather than in quantity by robot imitating three symbolic gestures: lateral raising, forward reaching, and forward bending. These three gestures cover the wide range of the robotic arms' workspace. The


(a) Skeleton mapping pose: lateral raising


(b) Skeleton mapping pose: forward reaching

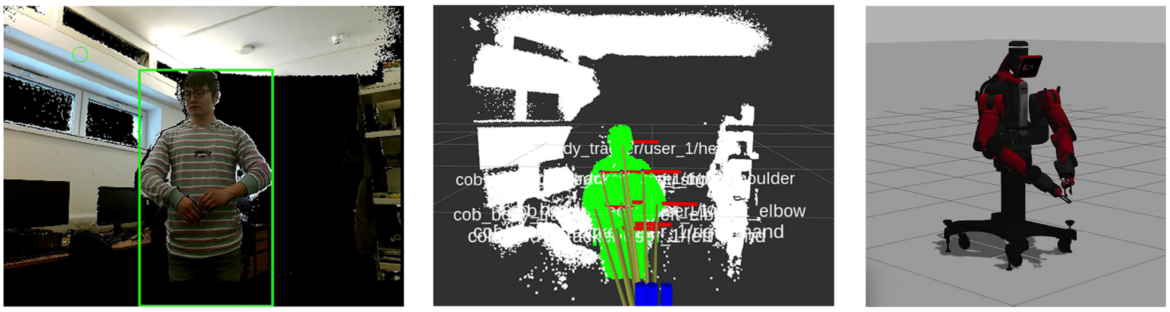

(c) Skeleton mapping pose: forward bending

Fig. 4.12 Skeleton mapping from human to robot with a Gazebo simulator. Left column: human demonstrator shows various of poses. Middle column: skeleton joints information observed by Kinect camera. Right column: Baxter imitates demonstrator's gestures through the mapping algorithm.

experiment results indicate that skeleton-joint mapping is working well and able to map human motions into robot motions. A further evaluation of the mapping algorithm will be carried out through a robotic assembly task in Chapter 5. In the robotic assembly task, the human demonstrator's hand motions are more complicated, which requires a precise mapping between the human skeleton space and the robot motor space. The emphasis of the evaluation is placed on the robotic assembly task, on which this thesis focuses, rather than simply targeting the precision or smoothness of robot imitating. In fact, overfitting either of the precision or the smoothness will result in bad performance of robot imitating.

The generality of the human skeleton-joint mapping is deducible from the fundamental theory of building the skeleton-joint mapping mechanism, i.e., Equation (4.1) to (4.10). We can see from the equations that the angles between skeleton pieces determines the motor values of robotic arms. The size of human skeleton does not influence the output of the mapping, which means that the skeleton-joint mapping works with different human demonstrators who have various lengths of limbs. The parameters of Holt double exponential smoothing algorithm are optimised based on a set of various human skeleton joints whose trajectories span from a small space (the torso joint) to a large space (the wrist joint). The various span of joints ensures that the parameters have a good generality and adaptability, which also supports the view that the size of human demonstrators does not matter for robot imitating.

The applicability of the mapping is secured as the trajectory samples used for learning the two parameters α and γ of double exponential algorithm are random movements demonstrated by human teacher. The parameters have a good balance between the smoothness and the precision for all the movements with different skeleton joints. On one hand, the double exponential smoothing algorithm ensures the position data of the tracked human joints is effective for robot learning. On the other hand, the mapping equations secures that the human motions can be mapped into robot motions, which is validated through three symbolic human demonstrations. Furthermore, the applicability of the mapping algorithm is evaluated with a robotic assembly task in the next chapter.

4.7 Summary

In this chapter, the joint data smoothing and mapping algorithm are investigated. Holt's two-parameter exponential algorithm is applied in the human joint's smoothing process to reduce the latency of the demonstrated motion. For the demonstration strategy, a mapping algorithm is proposed for the demonstrator to teach the robot skills. The mapping algorithm transfers the human joints pose information into robotic motor angles which prompt the robot to imitate the demonstrator. The experiments of three typical gestures validate the effectiveness of the proposed method. Furthermore, the skeleton-joint mapping will be validated in teaching robots performing assembly tasks in the next chapter.

Chapter 5

Robotic Assembly

5.1 Introduction

In this chapter, we investigate how Kinetic-based teaching can be used in a Learning from Demonstration framework, and how assembly task can benefit from LfD. These experiments are concerned with the implementation of LfD algorithm for robotic assembly tasks, notably concerning the mapping of human motion from human joint space to robot motor space, the generalisation of the assembly skills using Gaussian Mixture Model, and regeneration of the learned skill model with Gaussian Mixture Regression for reproducing the assembly tasks. The chapter is organised as follows:

- Section 5.2 presents the scenario of the robotic assembly experiments from three phases, *What robot sees*, *What robot learns*, and *What robot does*. Each of the phases explores how LfD is implemented via flowcharts.
- Section 5.3 presents the GMM produced in the robotic assembly demonstrations and GMR reproduced using the learned GMM. Experiments are conducted to show the robot learning assembly skill from human demonstrations.
- Section 5.4 presents the conclusion and discussion of the chapter.

5.2 Robotic Assembly Experiment Scenario

In the LfD for robotic assembly experiment, the user needs to teach assembly actions by skeleton-joint mapping based visual demonstration in a task of manipulating with the MEGA blocks. Figure 5.1 shows MEGA blocks used in this thesis research.

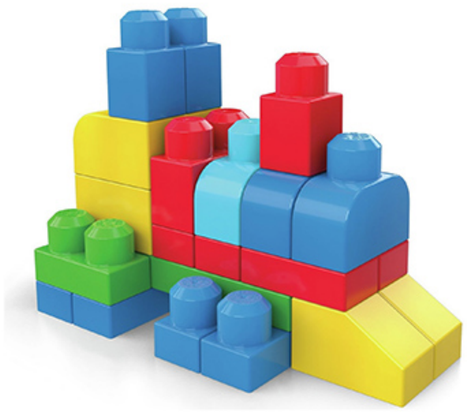


Fig. 5.1 MEGA blocks used for robotic assembly experiments.

Fig. 5.2 shows the overall structure of the proposed framework, in which human adjusts teaching performance in real-time based on the observing of the robot's movements to ensure the robot learns a good example. *Object tracking* works as a trigger/segmentation of demo examples. The framework consists of three main parts: ***What robot sees***, ***What robot learns***, and ***What robot does***. The proposed method is based on the utilisation of the LfD paradigm, enabling robots to perform new tasks autonomously.

5.2.1 What robot sees

In this phase, the demonstrator demonstrates the necessary skills to the robot, see Figure 5.3. A Kinect sensor is placed in front of the demonstrator for tracking the human skeleton joints (arms, head, and torso), and the object being manipulated is also tracked by using the RGBD colour filter. The human's movement is mapped into robot arm joint values through

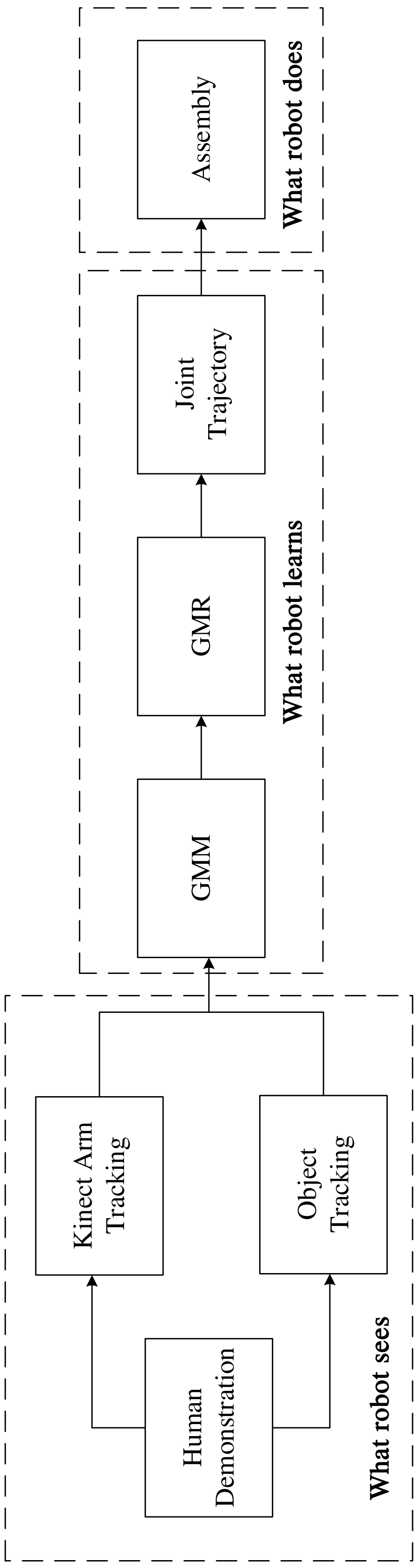


Fig. 5.2 The proposed LfD framework that consists of three main blocks: **What robot sees**, **What robot learns**, and **What robot does**.

skeleton-joint mapping. The joint values are raw data of robot hand trajectories, which will be used in the next stage, **What robot learns**, for learning GMM and GMR models.

Human demonstrates assembly skill

The robot observes human demonstrating assembly skills through the Kinect sensor. The human joints are mapped into robot motors from task space to configuration space. The manipulated object is tracked when the human performs assembly examples. In Figure 5.3, the demonstrator picks a MEGA block and place it on the desk, then picks another block and assembles these two blocks ($\mathrm{A} \rightarrow \mathrm{B}$). The assembly task is performed by just one hand, i.e., the right arm. The human keeps watching over the robot's motions mapped from the human's arm motions, ensuring the skills are mapped successfully by adjusting the demonstrator's hand motions.

Kinect sensor

The Kinect sensor works as the visual input, providing the function of tracking the manipulated object and human skeleton positions. The functions are implemented through two individual ROS nodes, one for object tracking, one for human skeleton tracking.

Skeleton smoothing and skeleton position

The original human skeleton position data subscribed from the Kinect sensor node contains noise, which will produce motor shaking if used directly. As presented in Section 4.4, Holt's two-parameter exponential smoothing is efficient for filtering out the skeleton noise from the Kinect tracking node. We use a trial-and-error method to run the smoothing algorithm again and again to get an MSE distribution. From the distribution, we get the best smoothing factor parameters, as experiments presented in Section 4.5.3. In this experiment, we use the same smoothing model to get the skeleton position as the environments are almost the same.

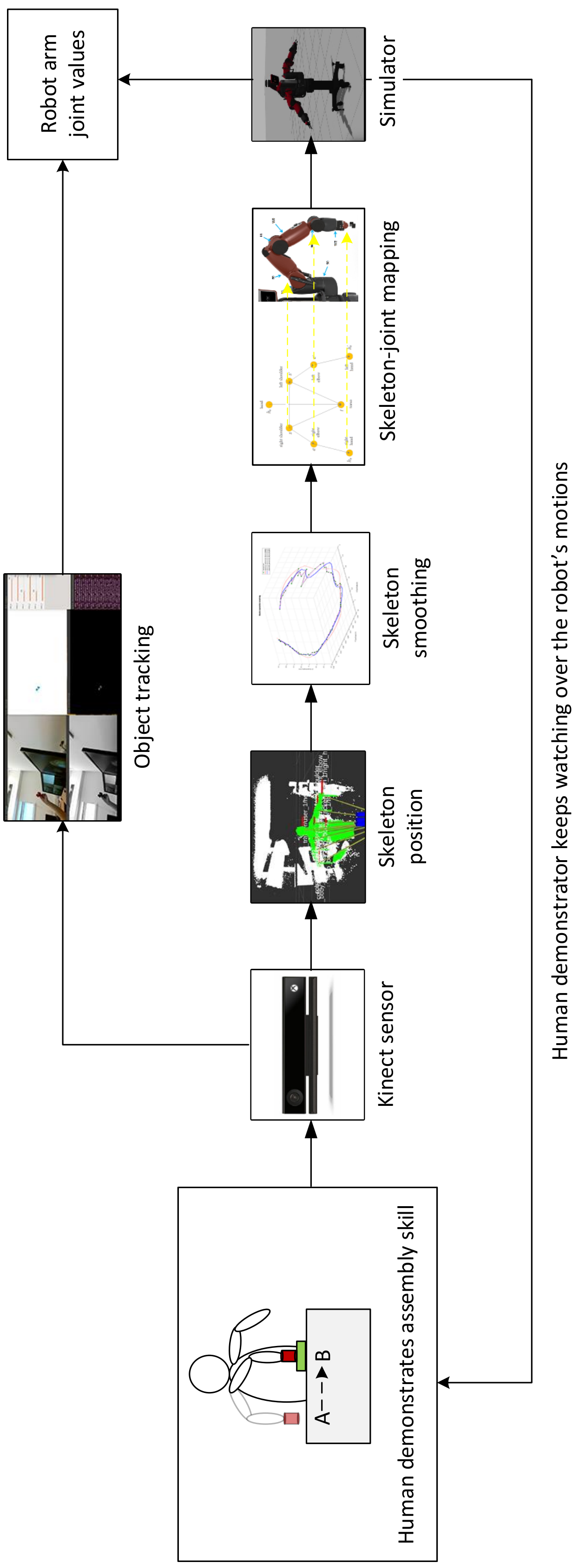


Fig. 5.3 The flowchart of the human teaching assembly skill to the robot. Human adjusts demo performance in real-time based on the observing of the robot's movements to ensure the robot learns a good example. *Object tracking* works as a trigger/segmentation of demo examples.

Skeleton mapping

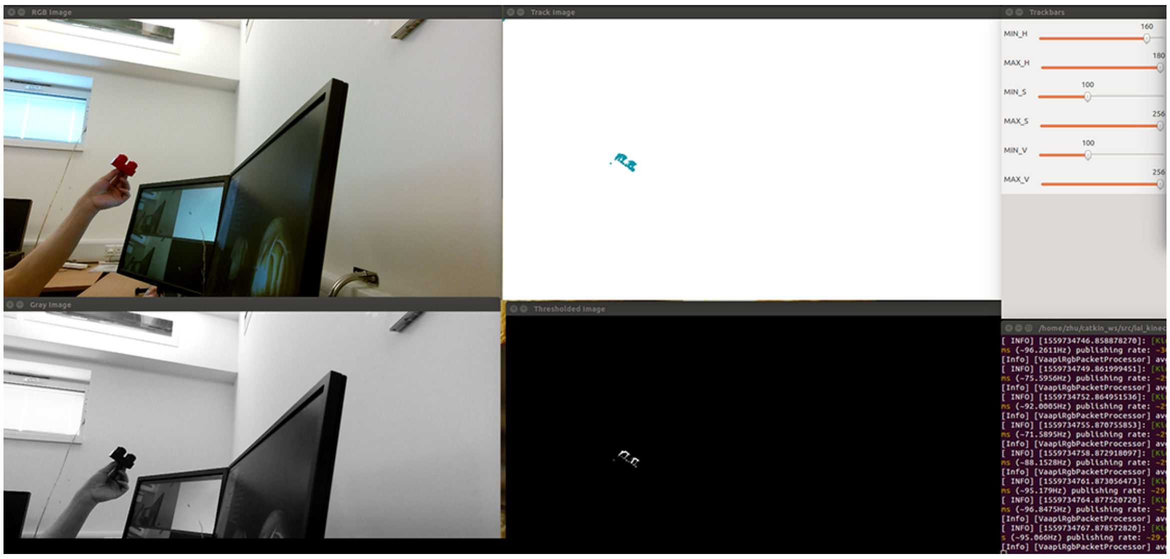
Skeleton mapping is an essential process for robot learning from human demonstration. A teacher demonstrates some skills in front of the robot, i.e. within the field of view of Kinect. The motion of the teacher's hands is captured through joint tracking, from where we get the human joints' real-time positions. These positions are based on the Kinect coordination in the task space, which is different from the robot's motor coordination in the configuration space. Human joints' positions can not be directly used to drive the robot's motors or end effector in the way that the teacher demonstrates. Thereby, a mapping mechanism between the human joints and robot motors are fundamental for the teaching process. In this proposed research, we use the angles formed by the adjacent joints of human limbs. Then map the angles into the robot's motor values, see section 4.3 for details on how vector maps between two coordinates.

Simulator and robot arm joint values

In the simulator, the simulated Baxter robot performs the trajectory mapped from the human skeleton trajectory. The simulated robot provides real-time feedback for the human demonstrator, so the later knows whether the demonstrated skill has been understood by the robot or not. As the trajectory has been run on the simulated platform, it would be reliable when it comes to the real robot in the next reproducing stage. The output of the simulator is validated joint trajectories mapped from the human demonstration.

Object tracking

Object tracking involves the process of tracking an object as it moves around frames in a video, maintaining the assignment of the unique ID. There are many ways to implement a simple object tracking algorithm using the OpenCV library. In [139], SIFT feature is used for object detection, classification and pose estimation.



MIN_H 160
MAX_H 180
MIN_S 100
MAX_S 256
MIN_V 100
MAX_V 256

```
[ INFO] [1559734746.858878270]: [Kl
ms (~96.2611Hz) publishing rate: ~3
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734749.861999451]: [Kl
ms (~75.5956Hz) publishing rate: ~2
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734752.864951536]: [Kl
ms (~92.0005Hz) publishing rate: ~2
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734755.870755853]: [Kl
ms (~71.5895Hz) publishing rate: ~2
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734758.872918097]: [Kl
ms (~88.1528Hz) publishing rate: ~2
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734761.873056473]: [Kl
ms (~95.179Hz) publishing rate: ~29
[ INFO] [1559734764.877520720]: [Kl
ms (~96.8475Hz) publishing rate: ~2
[Info] [VaapiRgbPacketProcessor] av
[ INFO] [1559734767.878572820]: [Kl
s (~95.066Hz) publishing rate: ~29.
[Info] [VaapiRgbPacketProcessor] av
```

Fig. 5.4 The manipulated object is tracked using HSV range. For a red object, H(160 - 180), S(100 - 256), V(100 - 256).

Here, the object tracking is simplified by using the colour filter and Kinect camera. For a red MEGA block, we use an HSV range to extract the object out from the environment, like H(160 - 180), S(100 - 256), V(100 - 256), as can be seen in Fig. 5.4. Firstly, the contours of the object are detected, so the boundaries of the shape are obtained. Once we have the boundary coordinates, we can compute the "centroid", or more simply, the centre coordinates of the bounding shape. Using the centre coordinates, we can simply get the location of the object combined with the depth information provided by the Kinect camera.

During the teaching phase, the *Object tracking* function works together with the *Skeleton mapping* feature. The object detecting feature starts working when an object of interest is placed in the field of view. Once the demonstrator picks the object, the object's position changes. Then the position's change is detected by the tracking function. From then on, the *Skeleton mapping* feature starts working, i.e. the robot starts learning from human demonstrations.

In the phase of reproducing assembly skills, the *Object tracking* also plays a vital function. The assembly task is to assemble two objects, differentiated by colour, as an entirety. The positions of the two manipulated objects are tracked simultaneously when the robot is reproducing the assembly task. Before assembly, the two objects' relative distance is tracked as feedback for promoting the accuracy of assembly. It should be noted that the feedback only works after the planned trajectory has been executed. The visual feedback is complementary to the LfD algorithm, which regenerates the assembly trajectory using GMR.

5.2.2 What robot learns

From the module of *What robot sees*, the robot 'sees' the demonstrated skills in the format of the robot's motor values in the configuration space through *Skeleton mapping*. The mapped motor values are primitive data which is similar to a recorded trajectory of the observed motion. In order to transfer a skill to the robot, a large body of work adopted the perspective that the robot should have the essential ability to encode the demonstrations in an efficient way for skill generalisation and reproduction. In the proposed research, we first use GMM to generalise the observed demonstration by encoding the temporal and spatial values, then retrieve smooth trajectories by GMR from the previously encoded joint representation. Figure 5.5 shows the flowchart of the robot learning model.

Data optimisation

Before the robot motor data being sent to train the GMM model, data optimisation needs to be done first. As presented in Figure 5.5, the robot joint values are first processed by a spline function, then the GMM is initialised, following by a parameter optimisation algorithm EM.

The spline function resamples the robot joint data, as they are recorded at a high frequency. The resampling helps reduce the redundancy and computation complexity but keeps the original trajectory shape. Only optimising the input data is not enough; the k-Means algorithm is also used for initialising the parameters of GMM. Next, the EM algorithm is used to update the parameters to minimise the covariance of GMM.

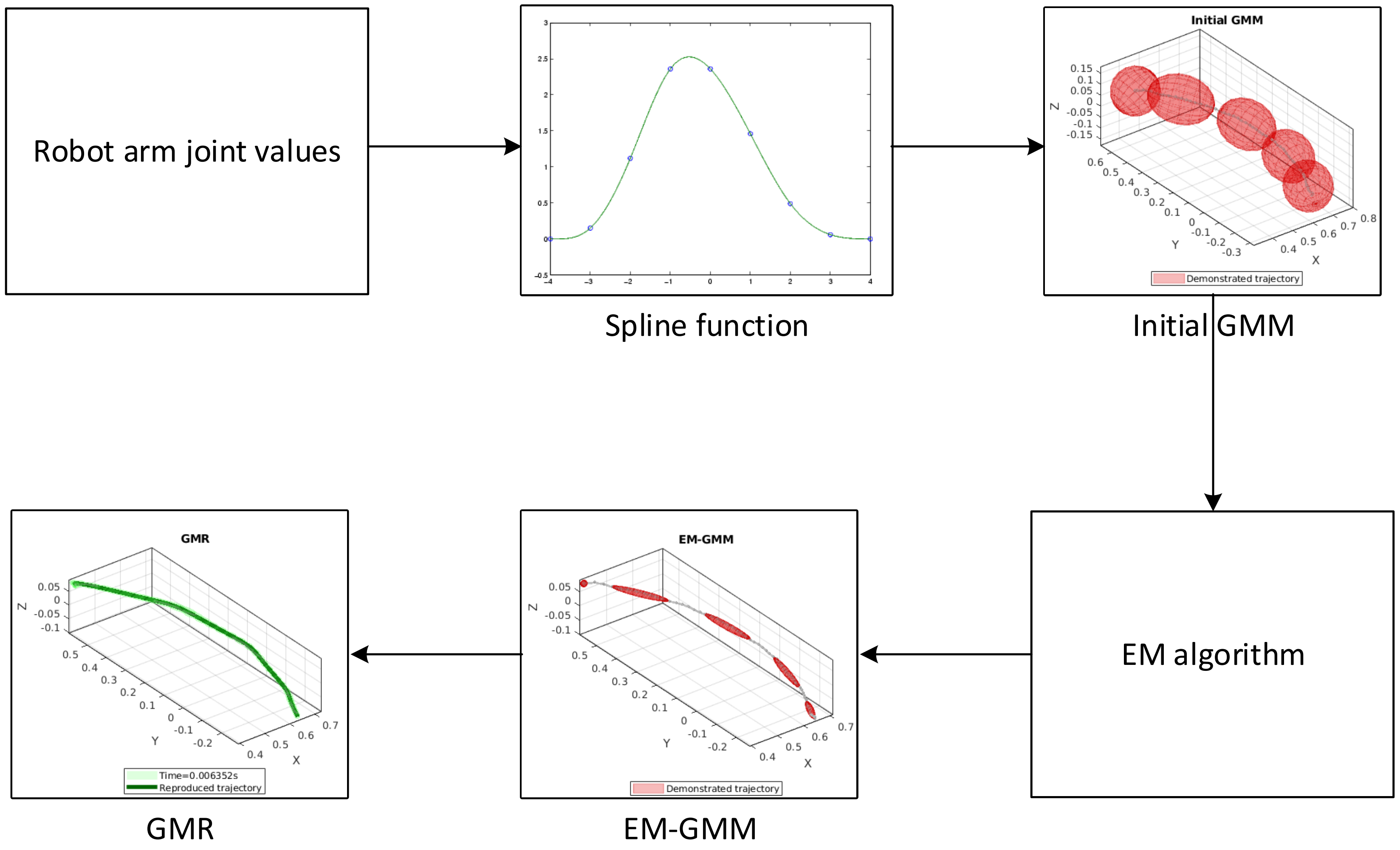


Fig. 5.5 The flowchart of robot learning model. The robot arm joint values are sampled and encoded with GMM. Then GMM is optimised by EM algorithm to have a good performance. Next, the GMR is learned by partially sharing the parameters of GMM.

Gaussian mixture models

In our framework, GMMs are used to encode the temporal spatial components of continuous trajectories. Trajectory-GMM provides here a way to handle partial demonstrations without further modification of the model. Learning of the parameters is achieved by log-likelihood maximisation subject to the constraint. The data in the different frames arose from the same source, resulting in an expectation-maximisation (EM) algorithm to iteratively update the model parameters until convergence [37].

For a dataset of N datapoints $\{\boldsymbol{\xi}_j\}_{j=1}^N$ with $\boldsymbol{\xi}_j \in \mathbb{R}^D$, which can be either joint angles, hand paths, or hand–object distance vectors, a Gaussian mixture model with a weighted sum of M components $\{\boldsymbol{C}_i\}_{i=1}^M$ is defined by a probability density function [32]:

$$p(\boldsymbol{\xi}_j|\lambda) = \sum_{i=1}^{M} \omega_i g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{5.1}$$

where $\boldsymbol{\xi}_j$ is a D-dimensional data vector, $\omega_i, i = 1, \ldots, M$, are the mixture component weights of component $\boldsymbol{C}_i$, with the constraint that $\Sigma_{i=1}^M \omega_i = 1$, and $g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, ..., M, j = 1, ..., N$ are the component Gaussian densities. For each component $\boldsymbol{C}_i$, the Gaussian density is defined as [32]:

$$g(\boldsymbol{\xi}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{\xi}_j - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\xi}_j - \boldsymbol{\mu}_i)\}, \tag{5.2}$$

where $\boldsymbol{\mu}_i$ are the component means, $\boldsymbol{\Sigma}_i$ are the covariance matrices. $\lambda = \{\omega_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^M$ are the set of parameters to be estimated in the density function; these parameters define a Gaussian mixture model. λ is usually estimated by maximum likelihood estimation using the standard expectation maximisation (EM) algorithm.

Gaussian mixture regression

We come to the use of regression to retrieve smooth trajectories from a set of observations, and particularly to the use of Gaussian Mixture Regression (GMR) [42], by previously encoding the temporal and spatial values of a set of trajectories in a joint representation using GMM, as presented in Section 5.2.2.

GMM is used to generalise the trajectory, GMR is used to regenerate the trajectory for the robot. To reconstruct a general form for the signals we apply GMR. In a generic problem, regression is to estimate the conditional expectation of Y given X on the basis of a set of observational samples {X, Y}, where $X \in R^p$ is the predictor variable, and $Y \in R^q$ is the response variable. Consecutive temporal values ξ_t are used as query points and the corresponding spatial values $\widehat{\xi_s}$ are estimated through regression. For each GMM, the temporal and spatial components (input and output parameters) are separated, i.e. the mean and covariance matrix of the Gaussian component k are defined by [22]:

$$\mu_k = \{\mu_{t,k}, \mu_{s,k}\},\ \Sigma_k = \begin{pmatrix} \Sigma_{t,k} & \Sigma_{ts,k} \\ \Sigma_{st,k} & \Sigma_{s,k} \end{pmatrix} \tag{5.3}$$

For each Gaussian component k, the conditional expectation of $\xi_{s,k}$ given ξ_t, and the estimated conditional covariance of $\xi_{s,k}$ given ξ_t are [22]:

$$\widehat{\xi}_{s,k} = \mu_{s,k} + \Sigma_{st,k}(\Sigma_{t,k})^{-1}(\xi_t - \mu_{t,k}), \tag{5.4}$$

$$\widehat{\Sigma}_{s,k} = \Sigma_{s,k} - \Sigma_{st,k}(\Sigma_{t,k})^{-1}\Sigma_{ts,k}. \tag{5.5}$$

$\widehat{\xi}_{s,k}$ and $\widehat{\Sigma}_{s,k}$ are mixed according to the probability that the Gaussian component $k \in \{1,\ldots,K\}$ has of being responsible for ξ_t [22]:

$$\beta_k = \frac{p(\xi_t|k)}{\sum_{i=1}^{K} p(\xi_t|i)}. \tag{5.6}$$

Using (5.4) (5.5) and (5.6). for a mixture of K components, the condition expectation of ξ_s given ξ_t, and the conditional covariance of ξ_s given ξ_t are:

$$\widehat{\xi}_s = \sum_{k=1}^{K} \beta_k \widehat{\xi}_{s,k} \tag{5.7}$$

$$\widehat{\Sigma}_s = \sum_{k=1}^{K} \beta_k^2 \widehat{\Sigma}_{s,k} \tag{5.8}$$

Thus, by evaluating $\{\widehat{\xi}_s, \widehat{\Sigma}_s\}$ at different time steps ξ_t, a generalised form of the motions $\widehat{\xi} = \{\xi_t, \widehat{\xi}_s\}$ and associated covariance matrix are produced. The temporal interval between two-time steps can be directly related to the controller requirements of the robot. Note that it is not equivalent to taking the mean and variance of the data at each time step, which would produce jerky trajectories and increase dramatically the number of parameters (the mean and variance values would be kept in memory for each time step). With a probabilistic model, only the means and covariance matrices of the Gaussian are kept in memory.

The advantage of the GMR is that it provides an efficient solution to produce a smooth motion which can be used for robot motion planning. The associated covariance matrices describing the variations and correlations across different variables encapsulated by GMM are directly used by GMR for robot motion planning, which makes the real-time control efficient by imposing considerably low-computational costs [22]. Also, the output trajectory is free of vibrations to perform any task that requires precise control, like the assembly task.

Joint trajectory

GMR models generate joint trajectories for the robot to perform the assembly skills. The joint trajectories are defined in the joint space with 7 Degrees of Freedom (DoF): S_0, S_1, E_0, E_1, W_0, W_1, W_2. Each of the joint represents a motor of the robot arm (see Figure 4.5).

5.2.3 What robot does

In the final stage, the robot tries to repeat the learned skills from human demonstrations. As shown in Figure 5.6 the robot executes the assembly tasks using the learned skills. The assembly experiments are presented in Section 5.3.4. It should be noted that not all of the skills that human used for demonstrating are learned by the robot. For example, the force used for assembling is predefined in the proposed research, as it is not included in our research field.

Firstly, the robot recognises and locates the location (x, y, z) of workpiece on the table using the Kinect sensor. From the IK solver, the robot plans a trajectory to pick the object. Then, the robot uses the learned GMR to generate motion planning with a time sequence started at the current moment as input data. Next, the robot moves the picked workpiece towards another workpiece. Then, the robotic hand starts moving down little by little to perform the assembly task until the detected assembly force reaches the threshold. Finally, the assembly task finishes with the two workpieces assembled. The task is designed to show that the robot can be taught to assemble an object with an existed object placed on the table which is different from the task demonstrated in Chapter 3.

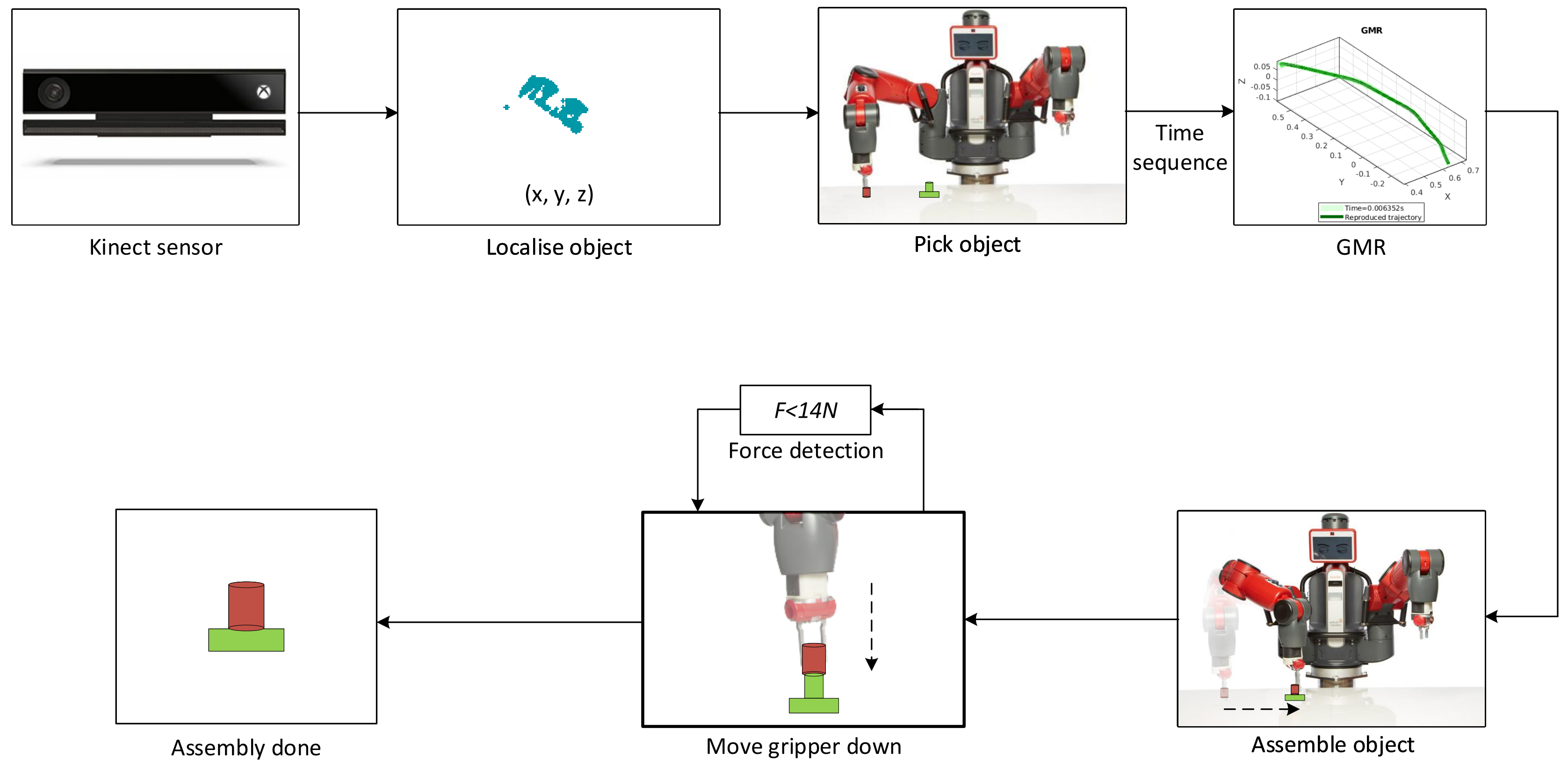


Fig. 5.6 The flowchart of robot reproducing assembly skill. To further validate the skeleton-joint mapping in teaching robots, a robotic assembly task was experimented with Baxter robot. In the assembly task, the robot first finds the object and pick it up, then use the learned model GMR to reproduce the assembly skill. During the last step of assembly, the force detection is used to control the assembly process.

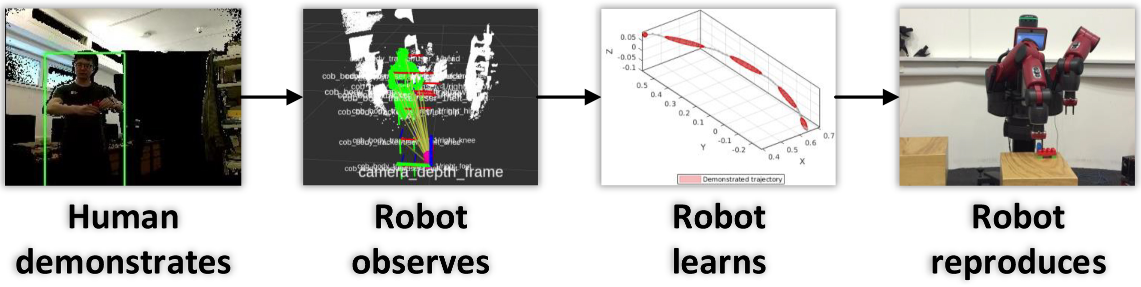


Fig. 5.7 The overall experiment sequence for the robotic assembly task.

5.3 Experimental Results

5.3.1 Experiment setup

The demonstrator's motions are tracked by 3D visual sensors in front of the human demonstrator. The Kinect camera is fixed on the head of the Baxter robot. A desk is placed between the human demonstrator and the robot. The overall configuration of the test system is shown in Figure 5.7. Firstly, the human teacher demonstrates in the visual field of the Kinect. Then the robot observes the teacher's motion which later is converted to joints angles through the mapping algorithm. Next, the robot learns the demonstrated trajectories with GMM and GMR. Lastly, the robot reproduces the assembly skills by completing an assembly task with MEGA blocks.

During the teaching phase, the human stands within the field of the Kinect sensor, about 2 meters away from the camera. Then, the demonstrator uses his right hand to pick a workpiece A. From now on, the robot starts tracking the demonstrator's hand motion. Next, the demonstrator moves the workpiece A towards the workpiece B that stays at another place of the desk and assembles the former with the latter. During the teaching phase, the demonstrator keeps watching the robot's motions from the mapping and avoiding the motions that are out of the robot's physical ability.

The joint angle trajectories of the head joint, torso joint, right shoulder joint, right elbow joint, and right hand joint are reconstructed by taking the Kinect sensor as reference. The left hand joints could be tracked and used to teach the robot assembly skills. However, we just use the right hand to demonstrate so that the research topic is limited to single-arm manipulation instead of dual-arm manipulation, and the later is out of our research field.

Each of these motions is demonstrated five times consecutively. In each example, the demonstrator starts from the same position. While observing the demonstrations, the robot tries to extract the demonstrator's skill underlying the task by using the skeleton-joint mapping mechanism. After the demonstration ends, the robot models the trajectory with the learned GMM that satisfies best the constraints extracted during the demonstration and reproduce a motion that follows this trajectory. Only one example is selected for the robot to generate the learned skill.

5.3.2 Software implementation

This section introduces how the experimental environment is implemented with ROS. The codes mean to explain the core framework of the proposed work and how ROS nodes work for the robotic system. Besides, in order to keep the concision of the paragraph, the indirect and secondary information is omitted.

The implementation of human teaching module is similar as the implementation of human skeleton-joint mapping presented in Section 4.5.2. In this framework, the object tracking node is introduced to work with the visual mapping method. The nodes that used for handling the interface, processing the various messages, transforming between multiple coordinate frames and so on, are omitted for a clear description. The main nodes are presented in Figure 5.8, namely *find_object_3d, cob_body_tracker, kinect2_launch_joint_sub, kinect2_launch_joint_pub, baxter_arm_control, model_generalisation*, and *robot_assembly*. More specifically,

- *find_object_3d* node detects the object to be assembled, the object's position is tracked and published to *objects topic*.
- *cob_body_tracker* node tracks human skeleton positions and smooths the joints with Holt's smoothing algorithm before publishing joint data to *human skeleton position topic*.
- *kinect2_launch_joint_pub* node subscribes the human skeleton positions from the *cob_body_tracker* node through the *human skeleton position topic* and handles the computing of human skeleton-joint mapping.
- *kinect2_launch_joint_sub* node subscribes the mapped robotic motor values from the *kinect2_launch_joint_pub* node through *robot joints topic* and then publishes motor values to *baxter_arm_control* node through *robot joints command topic* for the robot to perform the mapped motions.
- *model_generalisation* node uses the robot trajectory data as training sample to build the GMM and GMR that plans the motions for the robotic assembly task. Then the regenerated joint values are published to the *Robot joint value topics*.
- *robot_assembly* node manages the execution of robotic assembly task by subscribing to the joint values published from the *model_generalisation* node. Besides, the object position is also subscribed as a feedback for assembly.

Firstly, the *find_object_3d* node starts tracking the object that to be assembled by the human demonstrator. The object's position is then published to the topic *objects*, see code C.1.

The Kinect sensor also tracks human skeleton information with the *cob_body_tracker* node that tracks human skeleton positions and publishes them to *human skeleton position*

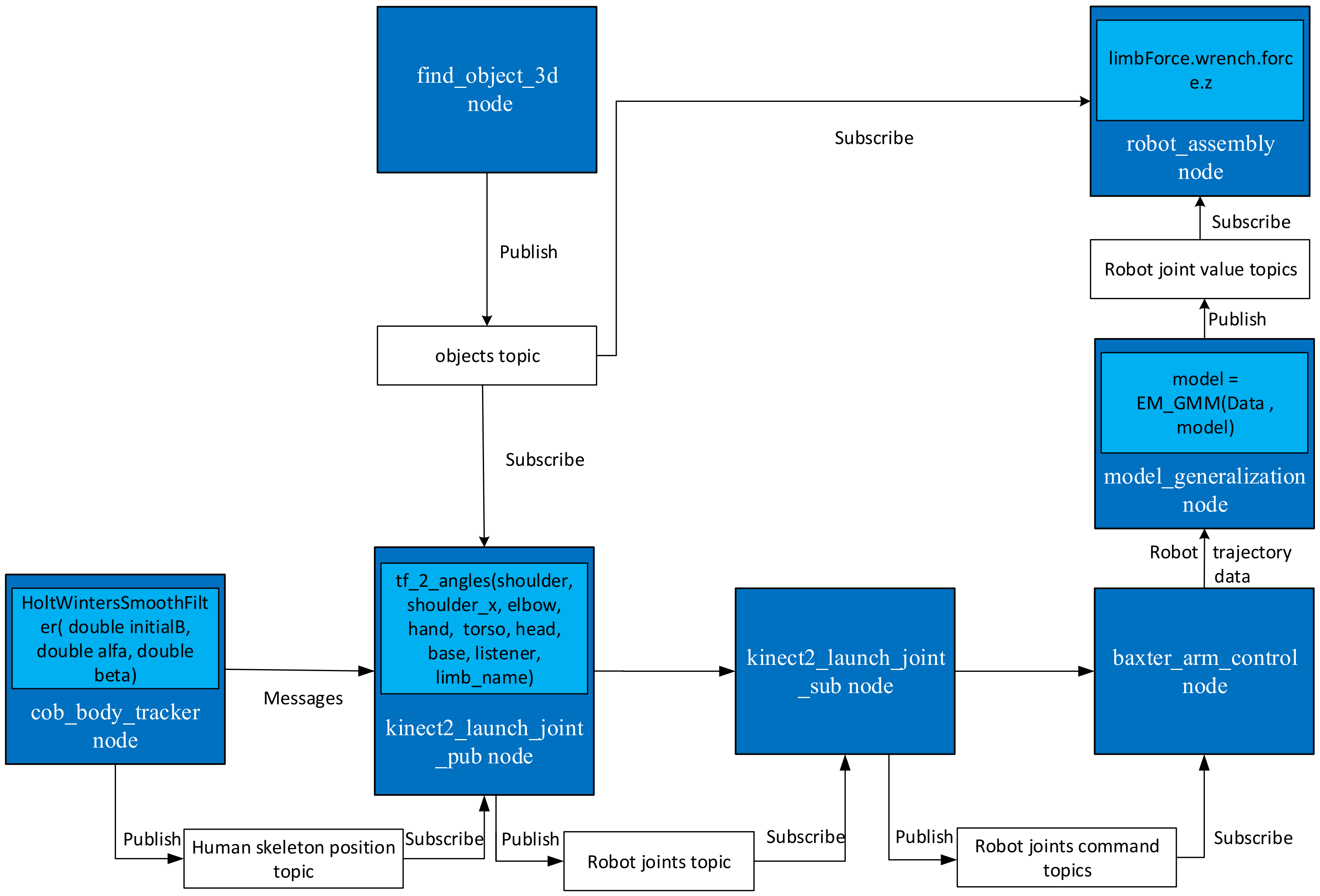

find_object_3d node
Publish
objects topic
Subscribe
Subscribe
limbForce.wrench.force.z
robot_assembly node
Subscribe
Robot joint value topics
Publish
model = EM_GMM(Data , model)
model_generalization node
Robot trajectory data
HoltWintersSmoothFilter(double initialB, double alfa, double beta)
cob_body_tracker node
Messages
tf_2_angles(shoulder, shoulder_x, elbow, hand, torso, head, base, listener, limb_name)
kinect2_launch_joint_pub node
kinect2_launch_joint_sub node
baxter_arm_control node
Publish
Human skeleton position topic
Subscribe
Publish
Robot joints topic
Subscribe
Publish
Robot joints command topics
Subscribe

Fig. 5.8 The implementation of the robotic assembly framework with ROS and the core nodes used for computing and passing messages.

topic. But before publishing the human joint's positions, Holt's smoothing is used to filter the noise of position data.

The *cob_body_tracker* node, *kinect2_launch_joint_sub* node, and *baxter_arm_control* node works same as presented in Setion 4.5.2. However, *kinect2_launch_joint_pub* node does more computational work, as it processes the object tracking information to control the start of robot learning, see code C.2.

During the demonstrating phase, the robot follows the human's motions. After the human demonstrator finishing the demonstration, the data of the robot's motions is saved for training the GMMs. The motion planner GMR is built and used for generating the trajectories for the robot. The trajectory data is then published to the robot joint topics, see code C.3.

In the node of *robot_assembly*, the robot subscribes to the joint values published from *model_generalization* node. Besides, the robot keeps watching the force sensor value in the crane direction (z axis direction) by subscribing to the */robot/limb/<side>/endpoint_state* topic. The force feedback is used as a control signal to finish the assembly task, see code C.4.

5.3.3 Model generalisation

In the proposed framework, GMMs are used to encode the temporal and/or spatial components of continuous trajectories. The 2D GMM/GMR algorithm is extended to the 3D case in which the robot task space is changed from 2 dimensions to 3 dimensions [21]. Here, the trajectory-GMM provides a way to handle partial demonstrations without further modification of the model. The parameter learning is achieved by the log-likelihood maximisation subject to the constraint of the different frames of data aroused from the same source. This is resulting in an Expectation-Maximisation (EM) algorithm to update the model parameters repeatedly until convergence. Figure 5.9 gives an intuitive comparison.

GMR overwhelms other models, like HMM, in terms of keeping the covariance information included in the Gaussian distributions to produce data, which could be smooth trajectories that can be run on the robot. In the case of a robot joint command, a smooth trajectory is critical to assure the robot avoid from vibration and overusing of torque. Therefore, Gaussian Mixture Regression (GMR) is introduced to retrieve smooth generalised trajectories with associated covariance matrices representing the variations and correlations across the different variables.

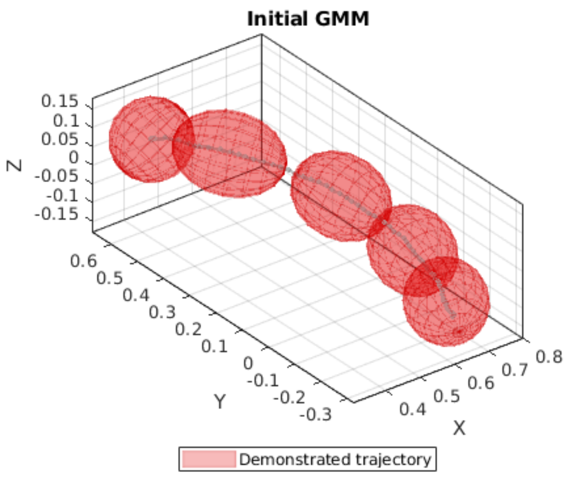
The ellipses in Figure 5.9(a) are much bigger than the ellipses in Figure 5.9(b). The reason is that the 'Initial GMM' has a larger covariance, which is initialised with k-Means clustering using k random cluster centre picked from the input data. The parameters of k-Means algorithm are listed in Table 5.1, where *cumdistThreshold* is the threshold of cumulative distance from each clustering centroid to data points, *maxIter* is the maximum iteration time, and *nbStates* is the number of clustering centre.

Table 5.1 k-Means clustering algorithm parameters for initialising GMM.

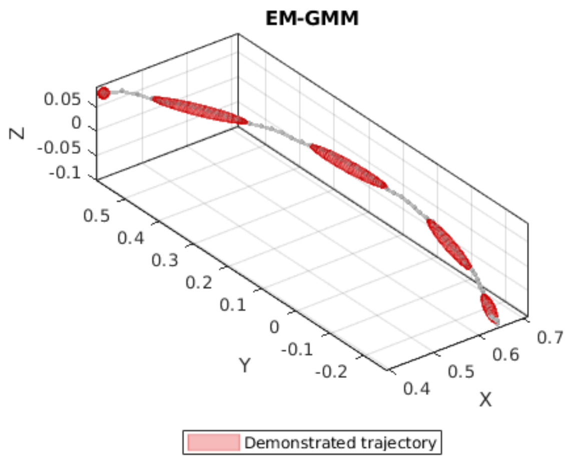
Parameters	Values
cumdistThreshold	1.00E-10
maxIter	100
nbStates	5

When the cumulative distance does not change, i.e. reaches the *cumdistThreshold* at 1.00E-10, or reaches the maximum iteration time *maxIter* at 100, the k-Means becomes convergence. The number of clustering centre is set at 5, which is enough for generalising the demonstrated trajectory. For optimising the number of clustering centre, we explored with a couple of experiments, see Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14.

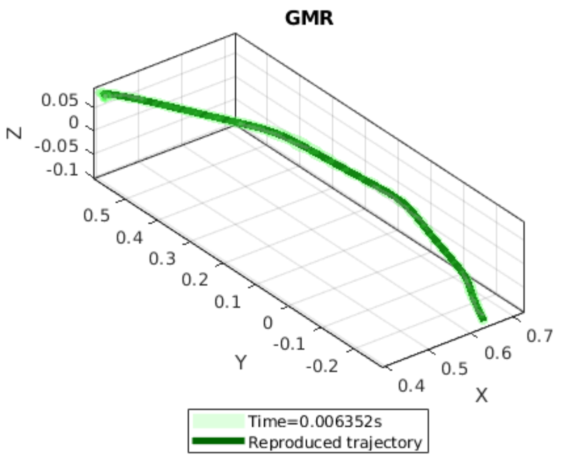
To minimise the covariance of GMM, EM algorithm is used to update the parameters $\lambda = \{\omega_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{M}$ from equations (5.1) and (5.2). The algorithm keeps updating λ until the log-likelihood converges. The parameters used for EM are presented in Table 5.2, where *nbMinSteps* is the minimum number of iterations allowed, *nbMaxSteps* is the maximum number of iterations allowed, and *maxDiffLL* is the convergence threshold for EM. The parameter values are set based on the empirical values, as EM usually converge after 15 to 70



(a) The initialised GMM with K-means algorithm.



(b) Initialization and optimisation the GMM model with EM algorithm.



(c) Reproduce the demonstrated trajectory with GMR model.

Fig. 5.9 Optimise the GMM with EM algorithm.

times of iteration, which locates in the range of *nbMinSteps* and *nbMaxSteps*. The *maxDiffLL* is small enough compared to the input data, thereby there is no need to promote the accuracy which brings unignorable computing and time delay.

Table 5.2 EM algorithm parameters for optimising GMM

Parameters	Values
nbminSteps	5
nbMaxSteps	100
maxDiffLL	1E-4

In Figure 5.10, we compared the GMR with EM optimisation and without EM optimisation. The figure at the top left shows that the demonstrated trajectory is encoded with initial GMM without optimise. The figure at the top right is the demonstrated trajectory with GMR reproduced from the top left GMM. The figure at the bottom left shows the demonstrated trajectory with EM-optimised GMM. The figure at the bottom right is the demonstrated trajectory with GMR reproduced from the GMM at the bottom left. From the right column, we can see that the 'Initial GMR' has large covariances (i.e. large light green ellipses). In contrast, the 'EM-GMR' has trivial covariances (i.e. tiny light green ellipses, just the same size as the deep green line). Through the comparison of these figures, it is clear that EM optimisation is vital for the reproducing of GMR model.

Figures 5.11, 5.12, and 5.13 show the same demonstrated trajectory modeled with different GMMs and regenerated with corresponding GMRs for the sub-task of assembly. The cluster (red eclipse) in the left column represents the Gaussian model, each cluster corresponding to one Gaussian model. Initially, the Gaussian model number starts from 1, i.e., $nbStates = 1$, which also means the state number of Gaussian Mixture Model is 1. Due to the numerous trajectory joint data, it is obvious that one Gaussian model is not precise enough to describe the data distribution. The size and shape of the cluster represent the covariance of the Gaussian model.

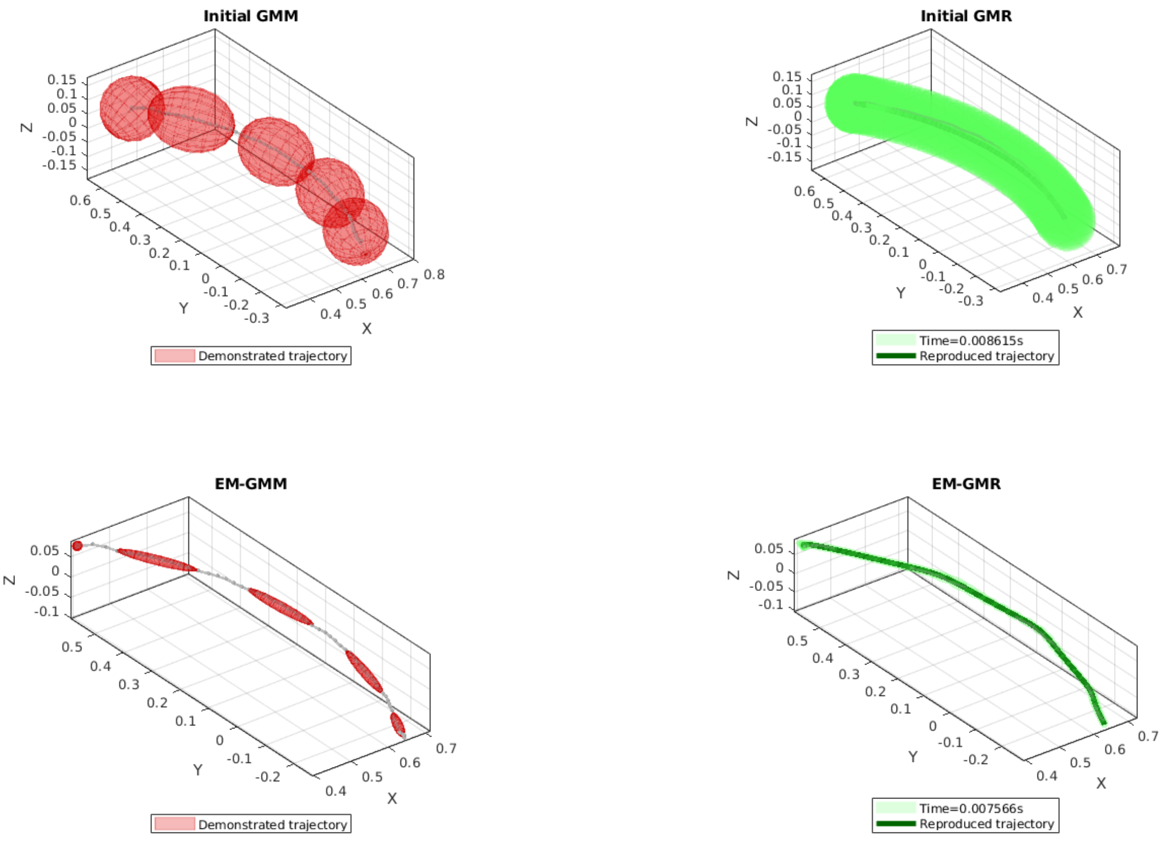


Fig. 5.10 Top left: Encode the demonstrated trajectory with initial GMM without optimising; Top right: Reproducing the demonstrated trajectory with GMR based on the top left GMM; Bottom left: Encode the demonstrated trajectory with EM-optimised GMM; Bottom right: Reproducing the demonstrated trajectory with GMR based on GMM at the bottom left.

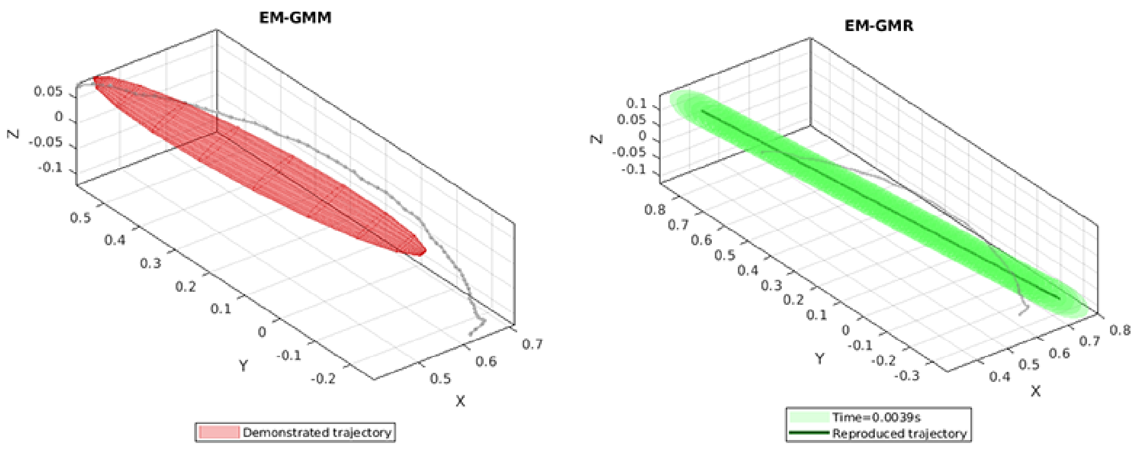
When $nbStates = 1$, the eclipse is the largest one compared to the other Gaussian models, which means the Gaussian model has a large covariance. The covariance matrix of a Gaussian distribution determines the directions and lengths of the axes of its density contours, all of which are ellipsoids. From the right side of Figure 5.11a, we can find that the Gaussian Mixture Regression has poor performance as the regenerated trajectory does not fit the original example (the grey line), especially the start position and end position are too far away, which will lead to failure in an assembly task.

Indeed, in Figure 5.11b, we can see that the generated trajectory(the thin green line) is almost a straight line, which is entirely independent of the original demonstrated example(the thin grey line). Furthermore, the vibration represented by the green eclipses(there are about 100 clusters represented by the green eclipse overlapping each other, which looks like a green pillar) is conspicuous that the learned GMR is not successful. With the increase of *nbState* number, the size of the GMM cluster decreasing, which means that the GMM model is more and more precise. At the same time, the vibration of the GMR also sees a continuous decreasing. Therefore, more clusters mean more precise both in modelling the demonstrated example and generating the learned skill.

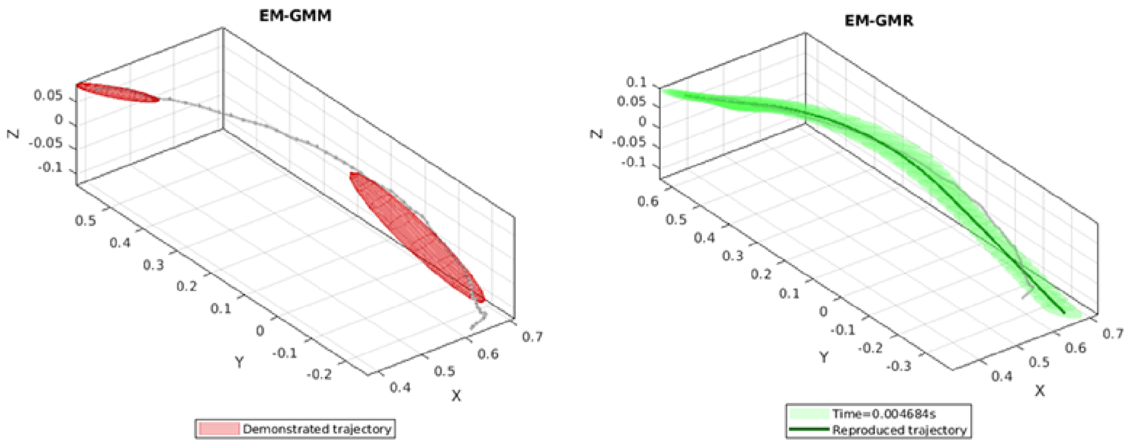
If we focus on the start position and end position of the trajectory generated by the GMR, we can find that with the increase of *nbState* number, the example trajectory and the learned trajectory are getting closer until overlapping together. This kind of precision is helpful in the assembly task, as the robot is easy to assemble the workpieces at the same place just demonstrated by the human teacher. However, the precision brings the sacrifice of smoothness of the generated motion. In the trajectory planning of robotics, smoothness is always the priority. Thence, we should take a compromise at the precision to buy some smoothness.

As we know, in a regression method, we must balance between having an accurate estimation of the response and having a smooth response. It is also known as the bias-variance tradeoff. For a high-performance regression model, we need to optimise the model parameters for two goals: (1) have a low bias between the estimated output and the sample data, i.e., high accuracy of regression output with respect to the observed data; (2) have a low variance on the estimate, i.e., free of vibrations to the regression output. In the GMR model, the bias-variance tradeoff is determined by the number of Gaussian components consisted of the GMM.

With the increase of *nbStates*, the learned model GMR needs more time to make the trajectory planning. The consumed time starts from 0.0039s at $nbStates = 1$ to 0.023s at $nbStates = 20$, showing a consistent growing trend, see Figure 5.15. Though more clusters

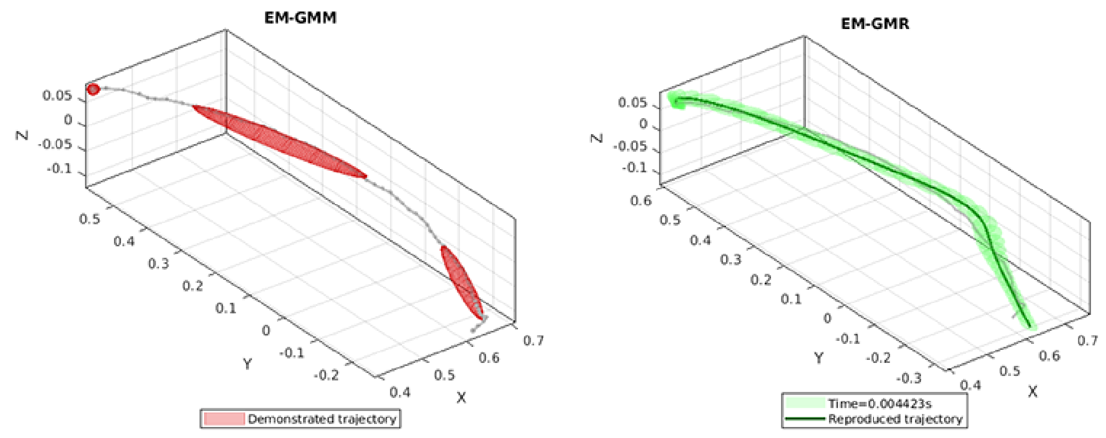


(a) When *nbStates* = 1

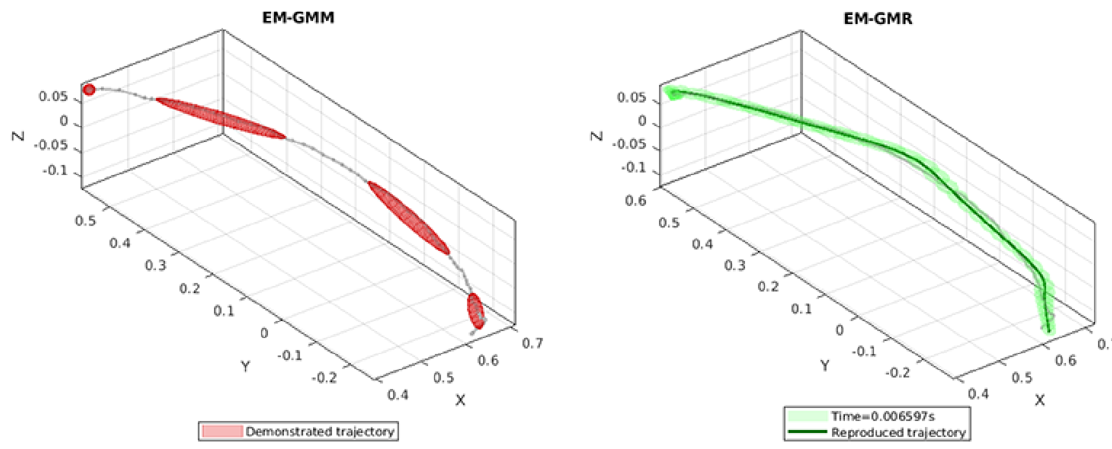


(b) When *nbStates* = 2

Fig. 5.11 Different models using two *nbStates* values (*nbStates* $\in \{1,2\}$). Left column: the demonstrated trajectories and the GMMs used to generalise these demonstrations. Right colum: the demonstrated trajectories, the reproduced trajectories and their covariances (See the light green ellipses). The time showed in the legend is the time used by EM-GMR to reproduce the trajectory.

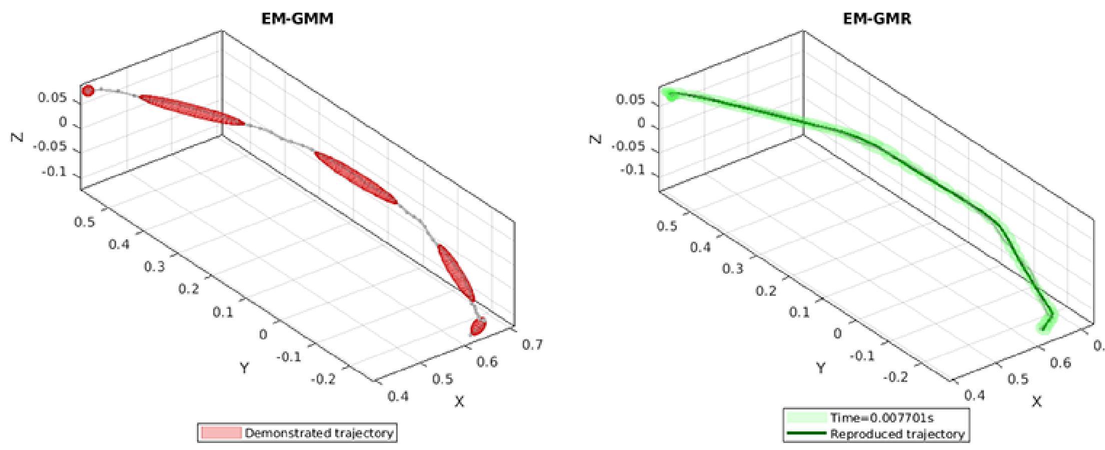


(a) When *nbStates* = 3

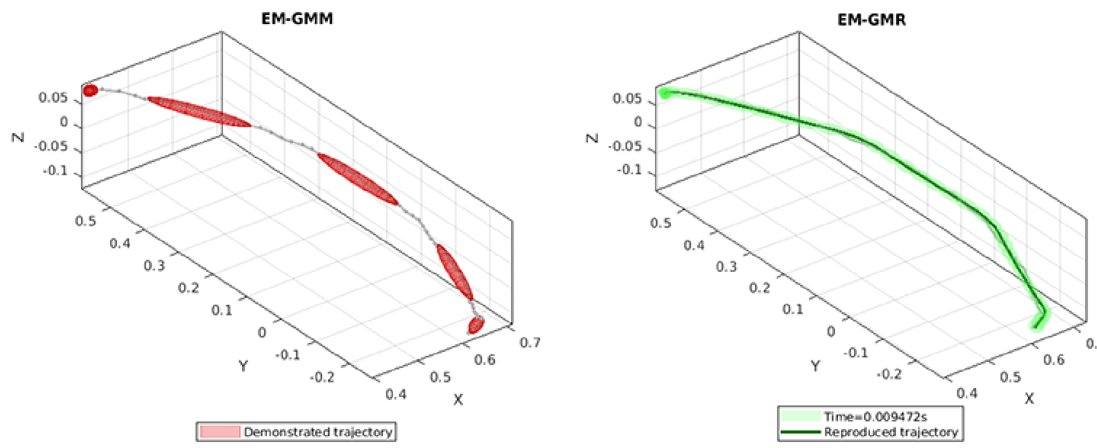


(b) When *nbStates* = 4

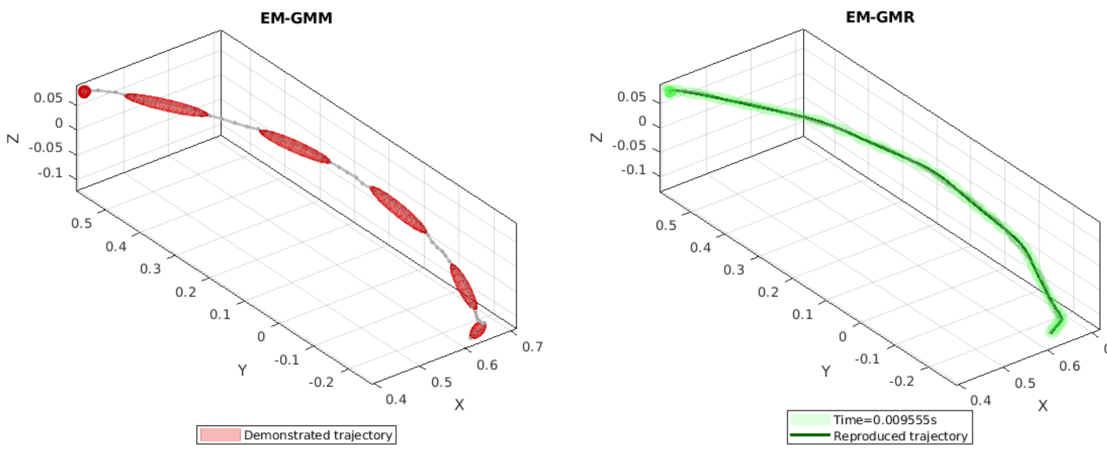
Fig. 5.12 Different models using two *nbStates* values (*nbStates* $\in \{3,4\}$). Left column: the demonstrated trajectories and the GMMs used to generalise these demonstrations. Right colum: the demonstrated trajectories, the reproduced trajectories and their covariances (See the light green ellipses). The time showed in the legend is the time used by EM-GMR to reproduce the trajectory.



(a) When *nbStates* = 5

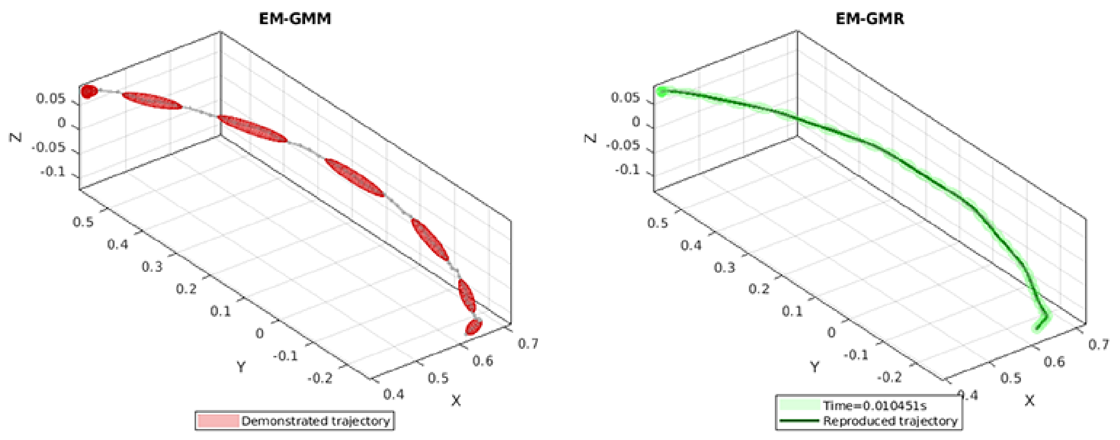


(b) When *nbStates* = 6

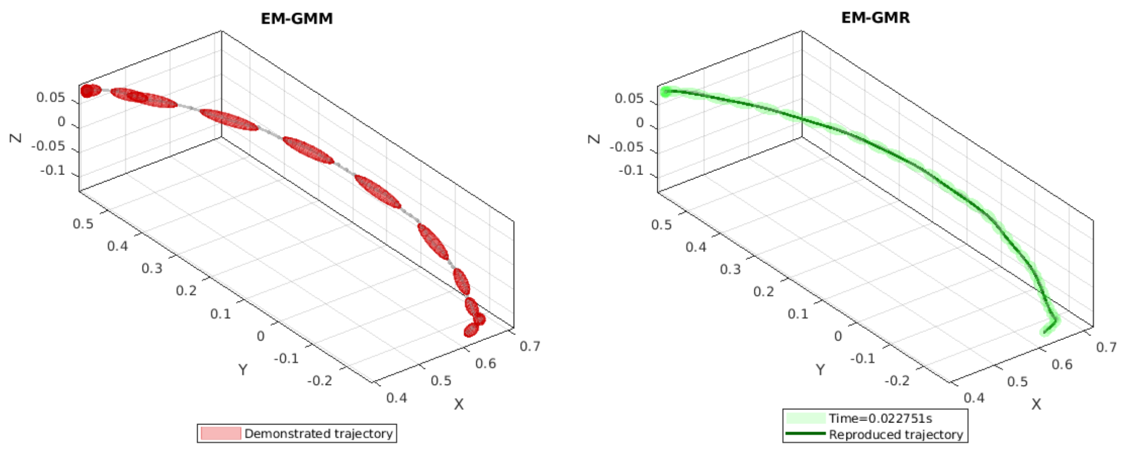


(c) When *nbStates* = 7

Fig. 5.13 Different models using three *nbStates* values ($nbStates \in \{5,6,7\}$). Left column: the demonstrated trajectories and the GMMs used to generalise these demonstrations. Right colum: the demonstrated trajectories, the reproduced trajectories and their covariances (See the light green ellipses). The time showed in the legend is the time used by EM-GMR to reproduce the trajectory.



(a) When *nbStates* = 10



(b) When *nbStates* = 20

Fig. 5.14 Different models using two *nbStates* values ($nbStates \in \{10,20\}$). Left column: the demonstrated trajectories and the GMMs used to generalise these demonstrations. Right colum: the demonstrated trajectories, the reproduced trajectories and their covariances (See the light green ellipses). The time showed in the legend is the time used by EM-GMR to reproduce the trajectory.

mean more time needed for trajectory planning, the time could be neglected as it is on the scale of 0.01 seconds. As a comparison, a robot task usually takes several seconds, sometimes even minutes. Therefore, the time is not considered as an optimisation goal in our GMR based planner system.

To find out the best *nbStates* listed in Table 5.1, we tested a series of *nbStates* values. More specifically, the *nbStates* values are set at (1, 2) in Figure 5.11, the *nbStates* values are set at (3, 4) in Figure 5.12, and (5, 6, 7) in Figure 5.13. Note that the last figure in Figure 5.14 is set at 10 and 20. Within these figures, the right columns show that the reproduced trajectories are far away from the demonstrated trajectories at the beginning (see the right picture at the top of Figure 5.11), but when *nbStates* increases to 5 (see the right picture at the top of Figure 5.13), the regenerated trajectory almost coincides with the demonstrated trajectory, especially the end position. This trend illustrates that when $nbStates \in [1,5]$, with the increase in *nbStates*, the accuracy of the estimation increases. When $nbStates \in [5,20]$, with the increase in *nbStates*, the smoothness of the estimation decrease. We see that with 20 Gaussian components, the model is more accurate but the generalised trajectory is less smooth than when using 5 Gaussian components.

For the assembly task, where the reproduced trajectory ends decides the accuracy of the assembly task. When the *nbStates* reaches 6 or more, the reproduced trajectory can not be more precise than the demonstrated, while the time used by EM-GMR for reproducing trajectories keeps increasing. Besides, the covariances start converging when *nbStates* reach at 5. Therefore, $nbStates = 5$ is used in this thesis research as listed in Table 5.1.

When the demonstrated skill becomes more complicated, the trajectory of the robot hand becomes more extended, which requires a more accurate model. In Figure 5.16, we extended the *nbStates* to 10 so that the new GMM and GMR can generalise and regenerate the demonstrated skill precisely. The trajectory shows the robot hand moves from one position

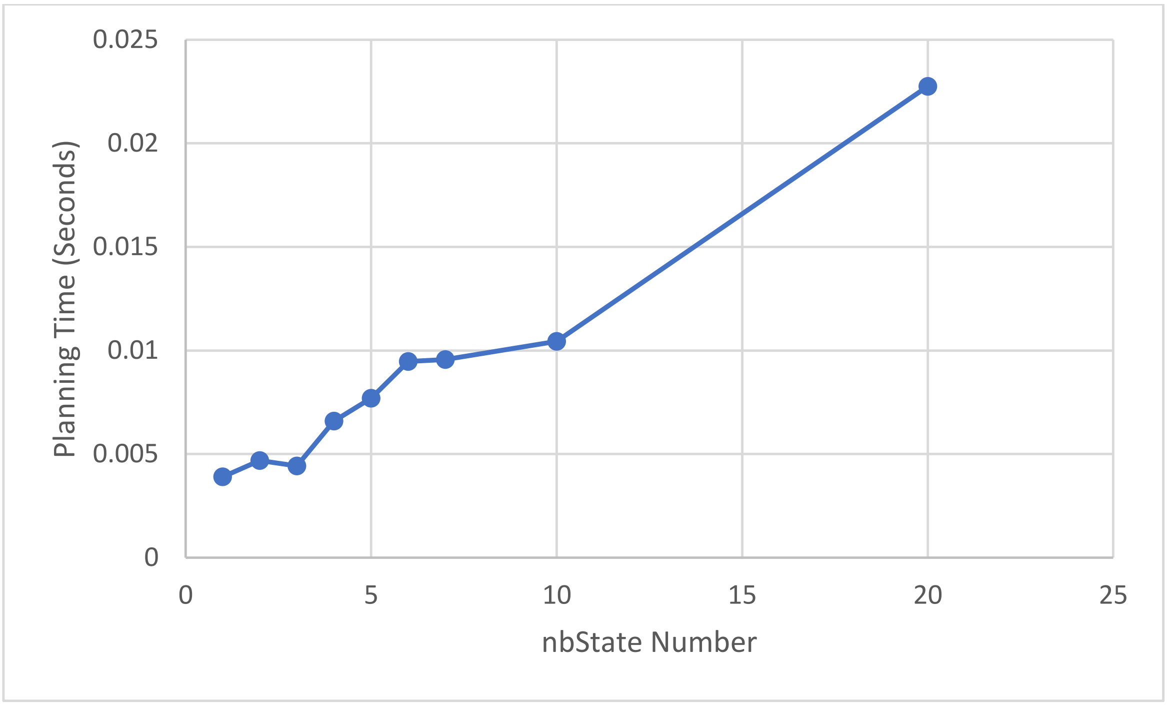


Fig. 5.15 Time used by GMR to make trajectory planning under different nbStates.

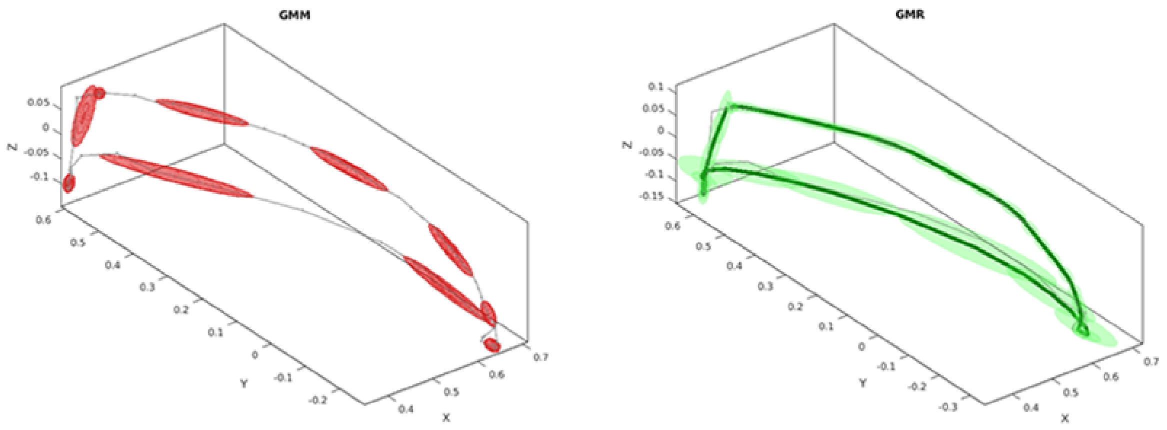


Fig. 5.16 Left: Gaussian distributions learned by expectation–maximisation with 10 states. Right: GMM with time-based Gaussian mixture regression (GMR) used for reproduction. Grey line: demonstrated task trajectory; Green line: reproduced task trajectory.

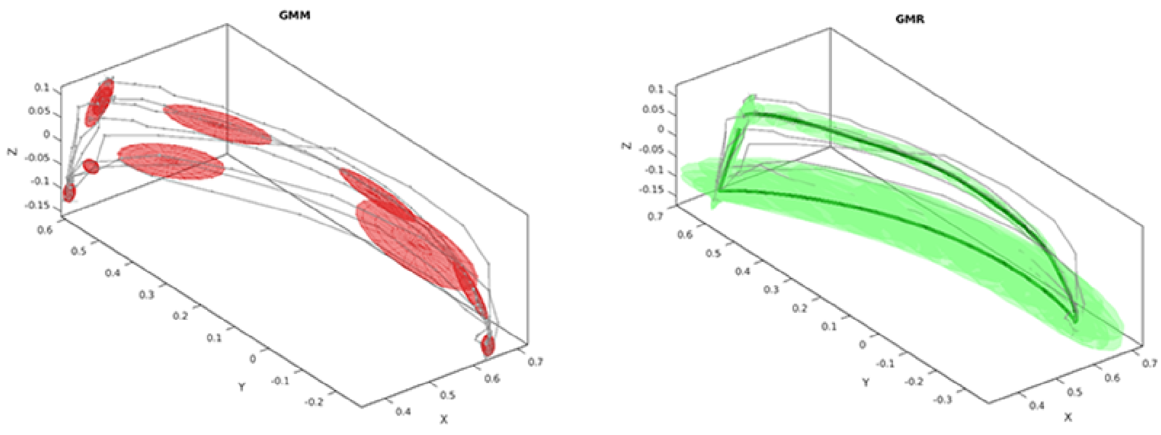
to another position and moves back. It is an example of the robot picking an object from the desk and assembles it with another object located in a different location, then retreating.

In Figure 5.17, we demonstrated pick-place-retreat multiple times, the robot observes all the demos and model them into one single GMM, see the left column. Then, the robot repeats the learned skill with a generalised GMR, see the right column. We can see from the figure that the GMM and GMR have the potential to learn the features from multiple demonstrations and reproduce the learned trajectories. As we can see from the figure, especially evident in the GMR figure, the new learned model locates in the middle of the examples, showing an ability to generalise the example features. This kind of feature learning ability is desirable in the situation that the robot needs to learn logical principles. However, if the demonstrated skill needs high precision in position control, then generalisation of multiple models is not the best solution. Therefore, learning from a single demonstration is desirable for robotic assembly.

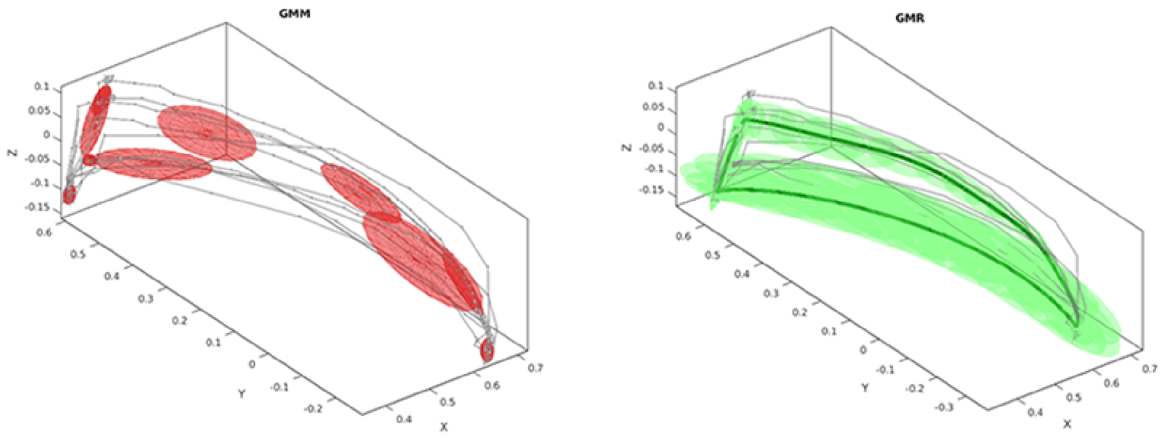
5.3.4 Robotic assembly tasks

In the Peg-in-Hole experiment, the robot must pick up one piece of the MEGA block shown in Figure 5.1, from the "components zone", and then assemble it with another piece of the MEGA block which is placed in the "assembly zone". We demonstrate the assembly task in front of the Kinect One camera, as shown in Figure 5.18. The human demonstrator stands within the visual field of the Kinect sensor, about 2 meters away from the camera. Then, he uses his right hand to pick the workpiece A. From now on, the robot starts tracking the demonstrator's hand motion using the Kinect motion tracking feature. At the same time, the human-robot skeleton mapping function starts working, so the human motion is converted into robot joint trajectories at the same time.

The robot joint data is recorded for the modelling of the skill in the model learning stage. Next, the demonstrator moves the workpiece A towards the workpiece B, which stays at



(a) 4 demonstrations are generalised by GMM model and regenerated by GMR



(b) 5 demonstrations are generalised by GMM model and regenerated by GMR

Fig. 5.17 GMM/GMR have been successful in representing sets of demonstrated examples that are time indexed, using time as the Gaussian conditioning variable to perform the regression.

another place of the desk, and assembles the former with the latter. During the teaching phase, the demonstrator keeps watching the robot's motions from the mapping and avoiding the motions being out of the robot's physical ability. The real-time correcting assures the effectiveness and efficiency of the teaching process.

By introducing the real-time correcting feature during the demonstrating, we can reduce the demonstrating times to as less as only one time, which is a tremendous privilege compared to general demonstrating methods which require multiple examples. Figure 5.19 shows that the Baxter robot imitates the human demonstrator's movements to do the assembly accordingly. The experiment runs on a Ubuntu 14.04 workstation with Core i5-6600 CPU @ 3.30Hz. It is a consumer-level platform which is easy to set up.

The motion of an assembly task is demonstrated by the human demonstrator. At the same time, the poses of the MEGA block is tracked by the Kinect camera with a sampling rate of 100 Hz and 30 Hz separately. It should be noted that the motion frequency is recorded from the motor encoder of the robot, which is mapped from the human motion (See Section 4.3). After the teacher starts performing the demonstration, the Kinect camera keeps tracking the human motion when the object, i.e. the MEGA block, is detected being picked by the teacher. The demonstrator moves the object after picking up and assemble it with another MEGA block which is placed on the table. The motion tracking system keeps tracking the teacher's hand motion until the two pieces of assembly objects are assembled. The assembly process stops when the force sensor embedded in the gripper reaches the threshold. In this experiment, we assume the threshold as prior knowledge and set it at 14N. The robot uses the torque sensor to detect table and keeps pressing down until the torque threshold is met.

For the feedback of the final assembly, the robot uses a separate trajectory planner which is different from the GMR. After the robot finished the trajectory planned by the GMR, the gripper reaches the position that is over the assembly workpiece. Then, the robot moves the gripper slowly down, so the robot gripper can sense the torque at each movement step until

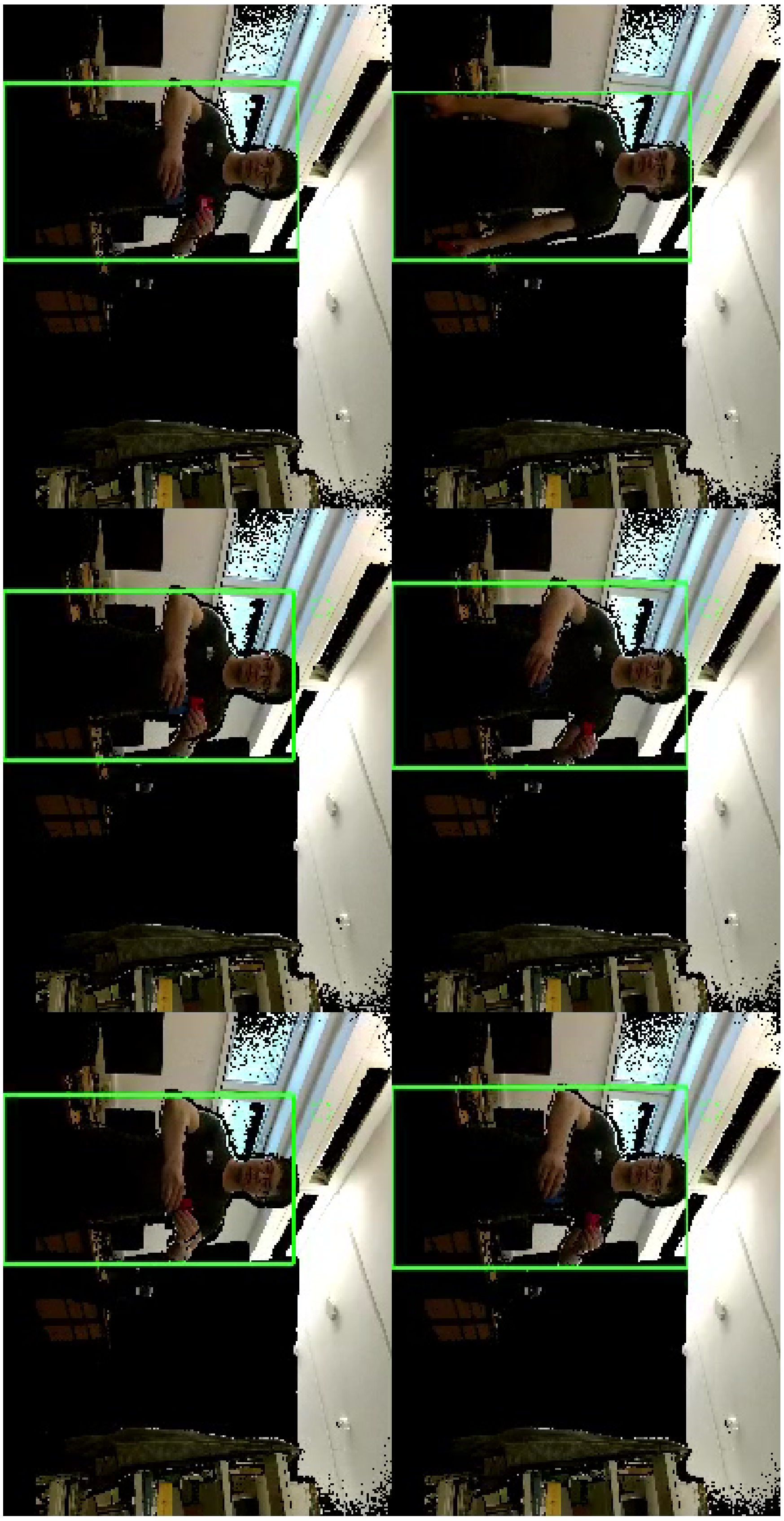


Fig. 5.18 Human demonstrates the two blocks assembly in front of Kinect 3D sensor.

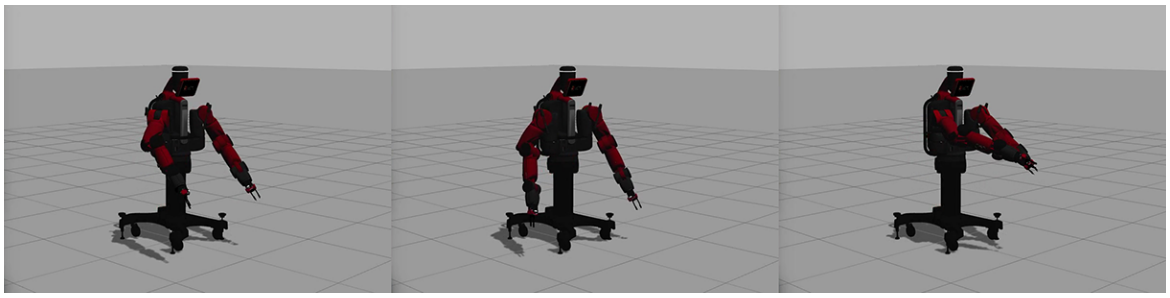


Fig. 5.19 The robot imitates the assembly movements using the skills learned from human demonstration in Figure 5.18.

the torque reaches the threshold which assures the two workpieces are assembled. The robot gripper keeps in crane pose to make sure the two objects to be assembled are coincident. Besides, the crane pose and moves along the perpendicular direction also assure the torque sensor of the gripper feels the complement of the insertion movement. Indeed, as the robot's arms permit to position the hands in the space with different joint angle configurations, an inverse kinematics problem arises when hands paths are reproduced. So some of the joints are set at fixed values to avoid multiple joint angle solutions.

In the data processing phase, the raw motion data is resampled at the length of 100 key points using spline interpolation. The resampling process simplifies the computing complexity for GMM and GMR, as well as keeps the essential trajectory information. A total of five demonstrated trajectories were recorded to illustrate the capability of the proposed learning system to learn assembly skill. Figure 5.20 shows the experiment with Baxter using LfD scenario to complete an assembly task, in which six pictures describes a sequence of picking and placing actions from the top left to the bottom right.

5.3.5 Discussion

In this Chapter, only MEGA blocks are considered for robotic assembly task, which does not mean that the proposed skeleton-joint mapping can not be transferred to other robot learning fields. In fact, the mapping mechanism is designed to follow human teacher's arm motions

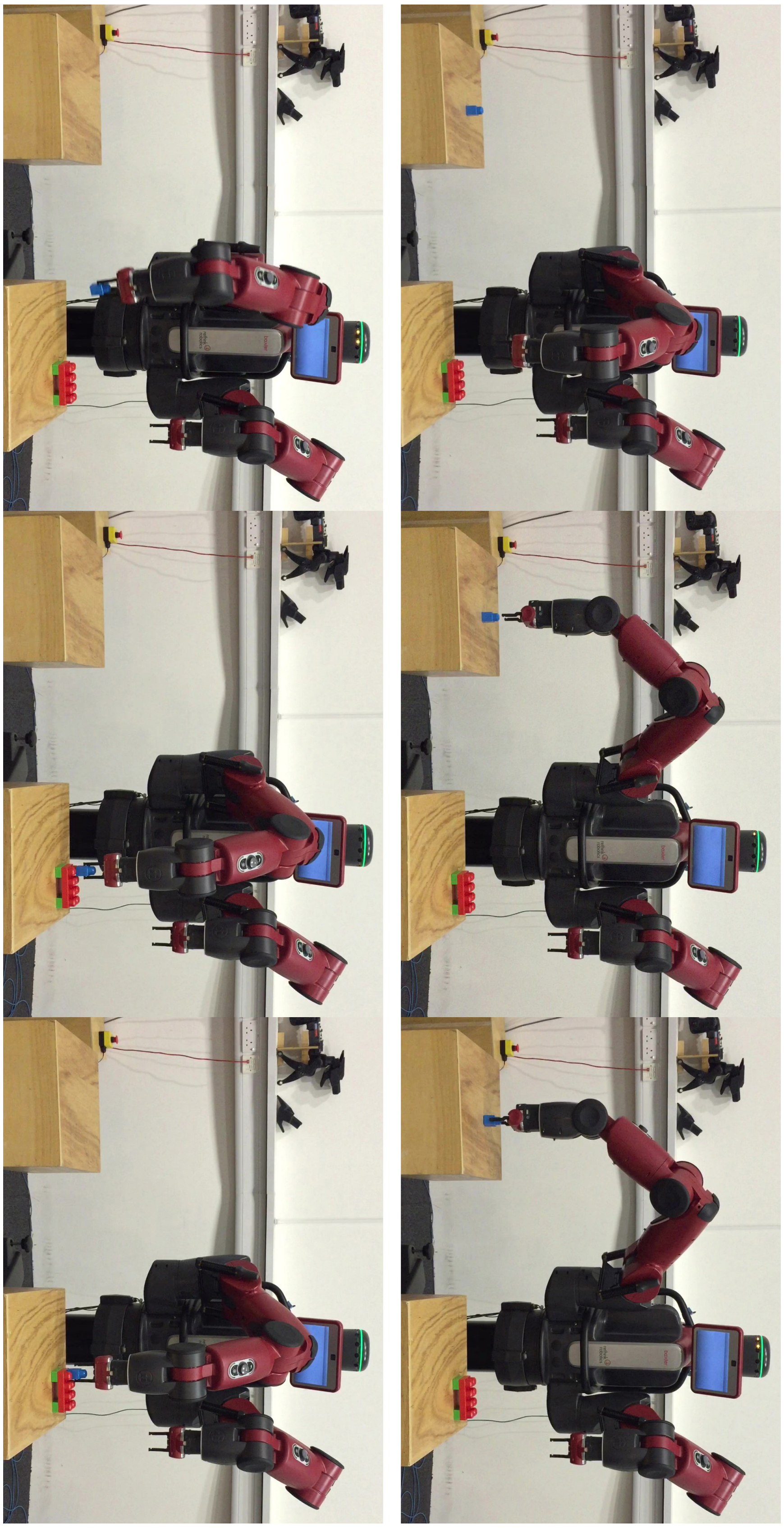


Fig. 5.20 Robot demonstrates the learned skill of picking an object and assembling it with another object.

which can be applied in many tasks like folding clothes, piling objects, pushing/pulling, and so on. However, the robotic gripper used for learning specific tasks, like assembly slippery or deformable objects, need to be physically capable. For example, a soft robotic hand with multiple fingers is good at manipulating deformable objects. While a robotic hand with two-rigid fingers, like Baxter, can only handle rigid and rough objects.

Currently, there are many ways for demonstrating the examples to the robot, like kinesthetic demonstration, motion-sensor demonstration, and teleoperated demonstration [138]. In general, the kinesthetic demonstration can achieve high accuracy in example teaching, as the demonstrated skill is transferred to the robot hand-by-hand without any other middle sensors between the teacher hand and the robot hand. In contrast, the motion-sensor demonstration relies on marker-based tracking devices or motion sensors integrated with the gloves to capture the demonstrator's hand motion, and the teleoperated demonstration requires a specially designed controller to deliver human's skill to the robot [138].

The latter two forms of teaching methods require additional devices or sensors, which introduce system errors in practical application. Besides, additional devices or sensors require a specially programmed system to handle the data stream, which makes the research question very tricky and time-consuming. Therefore, we selected the kinesthetic demonstration as a comparison reference to the proposed human-robot skeleton mapping based 3D visual demonstrating, see Figure 5.21. From the figure, we can see that both demonstration methods have an excellent reproducing ability. The difference between the two methods is ignorable.

In the comparative experiments, we demonstrated the assembly task with kinesthetic demonstration and visual demonstration, respectively. In the demonstrated task, the human teacher shows the robot to pick one workpiece and move towards another workpiece, then assemble the two workpieces and move the hand back to the original place where the first workpiece is placed. To simplify the comparison, the first workpiece is always placed at the same place that the robot can pick directly by just moving down towards the workpiece,

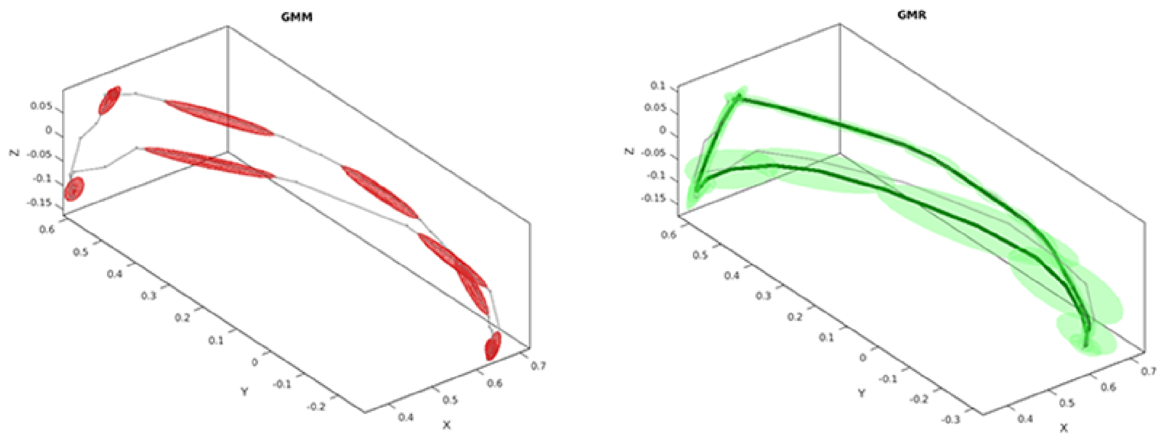
Table 5.3 A feature comparison between the existed demonstrating methods and the proposed method in robotic assembly.

	Kinesthetic demonstration	Motion-sensor demonstration	Teleoperated demonstration	Visual mapping demonstration
Non-expert teaching	✓	✓	✓	✓
Supervised teaching	✓		✓	✓
Remote teaching		✓	✓	✓
Single demonstration	?	?	?	✓
Hands-free sensor	✓			✓

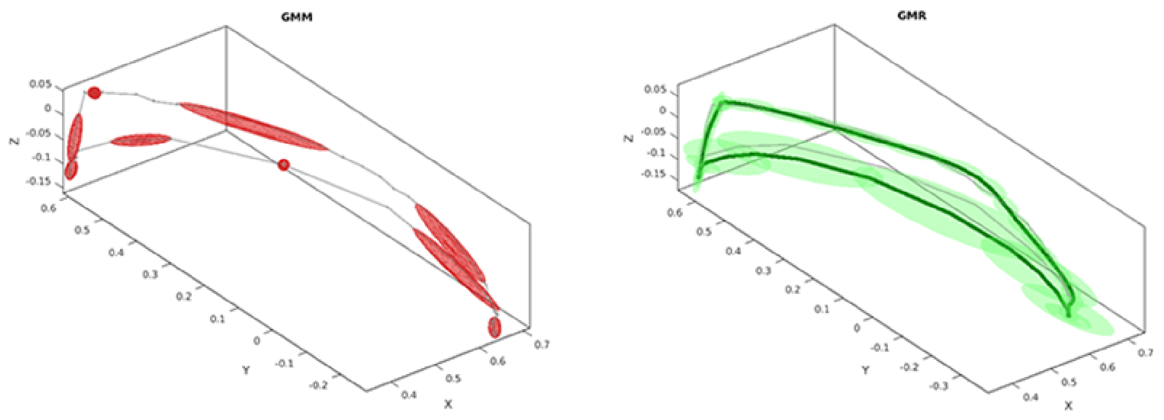
without using the vision sensor to search and locate the object. The other parts remain the same as the previous experiments.

Table 5.3 shows that the proposed visual mapping demonstration has advantages over others, comparing with other demonstration methods in many desirable features of LfD, like non-expert teaching, supervised learning, remote teaching, single demonstration, and hands-free teaching.

1. Non-expert teaching. In the learning from demonstration system, the robot could learn new skills from non-expert teaching. The teacher just needs to demonstrate the skills in front of the robot using a natural way of how the normal human performs. All the existed demonstration methods and the proposed visual mapping demonstration support the robot to learn from non-expert teaching, which is the essential advantage of LfD compared with other robot learning methods[11].



(a) The existed kinesthetic teaching



(b) The proposed human-robot skeleton mapping based 3D visual teaching

Fig. 5.21 Comparison of the two demonstrating method, (a) the existing kinesthetic teaching VS (b) the proposed human-robot skeleton mapping based 3D visual teaching.

2. Supervised teaching. When the human teaches the robot new skills, the robot is supervised by the human in real-time as the teacher monitors the robot's performance during the teaching process. For example, in kinesthetic demonstration, the human guides the robot hand to perform the task, which means the robot's learning is under supervising of the teacher [26, 22, 136, 78, 2]. In teleoperated demonstration, the robot is not guided through hand-by-hand; instead, the human operator uses a control box or delivers hints to control a robot to execute tasks, but they are still under supervising of the operator [109, 65, 28, 16]. While in motion-sensor demonstration, the human performs the skill first and the robot just keeps watching before reproducing the skill, in which there is no bidirectional communication between the teacher and the learner [3, 67, 66]. In contrast, the proposed visual mapping demonstration is similar to the teleoperated demonstration. The human teacher monitors the robot's real-time following performance. When the robot does something that is not as desired, the human teacher can stop the current teaching and demonstrates again. The repeated teaching is slightly different from the failed one which means to avoid the physical limitations of the robot. For example, the demonstrator teaches the same assembly skill but changes the motions and trajectory when performs the assembly task. So the robot can easily learn from the new samples without motor limitations that stop the robot following the demonstrator. This kind of feedback helps the human teacher improve the teaching quality. In LfD, this kind of supervised teaching is efficient communication between the teacher and the learner[105].

3. Remote teaching. Teaching the robot hand-by-hand is convenient and easy, but demonstrating the task from a remote place related to the robot and the working space of the robot is safer. All the listed demonstration methods could be performed remotely except for the kinesthetic demonstration, which needs hand-by-hand guiding. Motion-sensor demonstration uses trackers to capture human motions. Teleoperated demonstration

uses a controller to deliver human motions. The proposed visual mapping demonstration uses the Kinect 3D sensor to capture human motions and map into robot joint values, carrying out from a remote way.

4. Single demonstration. All the demonstrative examples of LfD system required for robot learning is not numerous, but not all of them use only one demonstration for teaching. In the kinesthetic demonstration, the human operator usually demonstrates the example skill several times. For example, Calinon and Billard [22] showed the robot 5 times for each demonstrated skill, Ye and Alterovitz [136] provided 7 examples for the robot to learn. In the motion-sensor demonstration, the demonstrated examples needed for teaching could be more, like Skoglund et al. [113] used 26 examples in a pick-and-place demonstration, Kulić et al. [67] uses 751 motion segments from a data set that consists of 16 minutes of continuous whole-body motion data of a single human subject. In the teleoperated demonstration, Tanwani and Calinon [123] used 8 demonstrations for an opening-valve task. In the proposed visual mapping demonstration, the teaching example could be as less as just one, which simplifies the teaching process. It should be noted that the number of examples is subject to the task to be learned and the models used to generalize and reproduce the learned skills. Therefore, the example number of other demonstration methods could be as less as just a single demonstration.

5. Hands-free sensor. The motion-sensor and teleoperated demonstration need extra marker sensors or controllers held in hand [109, 65, 28, 16, 3, 67, 66]. The kinesthetic and the proposed visual mapping demonstration do not need any sensor or devices attached to the human body, which frees the human hand for demonstrating any hand-related manipulating tasks, like grasping.

Through the above comparison, we come to the conclusion that the proposed visual mapping demonstration has comprehensive advantages over kinesthetic, motion-sensor, and teleoperated demonstration.

5.4 Summary

This chapter proposes a novel learning from demonstration method for robotic assembly task based on visual teaching. A new teaching method is proposed to visually observe human demonstrations so that the assembly skill demonstrated by a human can be effectively mapped from task space to configuration space for robot manipulation. A new approach is proposed for simultaneous human movement tracking and object position tracking, which is suitable for assembly task demonstration and robot learning. The effectiveness of the proposed method was verified through a manipulation task conducted in a Peg-in-Hole assembly task with the Baxter robot. An overall comparison between the proposed visual mapping demonstration and other existed demonstration is presented, which shows the proposed method has comprehensive advantages over others.

The contribution of this chapter is that a real-time feedback feature is introduced for the robot to learn assembly skill from observing human demonstrating the manipulating skills. Compared with the existing works that requires multiple or enormous demonstrations for robot learning assembly tasks, the proposed LfD paradigm improves the demonstrating quality by using the real-time simulating robot to provide feedback. Furthermore, the proposed method has comprehensive advantages over kinesthetic, motion-sensor, and teleoperated demonstration in robot learning assembly tasks.

Chapter 6

Conclusion and Future Work

6.1 Research so far

In this thesis, a simple visual skeleton mapping method was exploited in an LfD system to teach the robot assembly skills. The visual skeleton-joint mapping was approximated by a set of formulations that builds a map between the human skeleton joint space and the robot motor space. Before mapping, the human joint data was filtered and smoothed by Holt's two-parameter exponential smoothing algorithm to reduce the robot's motor babbling. Learning from demonstration (LfD) has been used to help the robot implement manipulation tasks autonomously, particularly to learn manipulation behaviors from observing the motion executed by the human demonstrator.

Firstly, this thesis reviewed recent research and development in the field of LfD for robotic assembly. The main focus of the review was placed on how to demonstrate the example behaviors to the robot in assembly operations, and how to extract the manipulation features for robot learning and generating imitative behaviors. Various metrics used to evaluate the performance of robot imitation learning were analysed. Specifically, the application of LfD in the robotic assembly is a focal point in this thesis.

Secondly, this thesis presented a novel approach for a robot to conduct assembly tasks, namely robot learning from human demonstrations. The learning of robotic assembly task was divided into two phases: teaching and reproduction. During the teaching phase, a wrist camera was used to scan the object on the workbench and extract its SIFT feature. The human demonstrator taught the robot to grasp the object from the effective positions and orientations. During the reproducing phase, the robot used the learned knowledge to reproduce the grasping manipulation autonomously, even though the environment changed. The robustness of the robotic assembly system was evaluated through a series of grasping trials. A dual-arm Baxter robot was used to perform the Peg-in-Hole task by using the proposed approach. Experimental results were given to show that the robot was able to accomplish the assembly task by learning from human demonstrations without traditional dedicated programming.

Thirdly, the joint data smoothing and mapping algorithm were investigated in this thesis. Holt's two-parameter exponential algorithm was applied in the human joint's smoothing process to reduce the vibration of the demonstrated motion. For the demonstration strategy, a mapping algorithm was proposed for the demonstrator teaching the robot skills. The mapping algorithm transferred the human joints pose information into robot motor angles that prompt the robot to imitate the demonstrator. The experiments of three typical gestures were conducted to validate the effectiveness of the proposed method.

Lastly, this thesis proposed a novel learning from the demonstration method for robotic assembly tasks based on visual teaching and real-time feedback from a human teacher. A new teaching method was proposed for the robot to visually observe human demonstrations so that the assembly skill demonstrated by the human can be effectively mapped from task space to configuration space for robot learning. A new approach was proposed for human movement tracking and object position tracking to deal with assembly task demonstration and robot learning effectively. The effectiveness of the proposed method was verified through a manipulation task conducted in a Peg-in-Hole assembly task with the Baxter robot.

The three main experimental parts, Chapters 3, 4, and 5, constitute the three main contributions of this thesis. Chapter 3 used a widely-used demonstration method, kinesthetic teaching, for robot learning effective grasping poses. This chapter mainly solved one step of the robotic assembly: how to learn effective grasping poses for an unstructured object for robotic assembly. Chapter 3 is independent of Chapter 4 and 5 while the latter two chapters are together. Chapter 4 presented a skeleton-joint mapping and applied in LfD as a new demonstrating method. Chapter 5 further evaluated the mapping with a robotic assembly task with Baxter robot.

6.2 List of Publications

The following journal and conference papers have been published during my PhD study:

- **Z. Zhu** and H. Hu (2018). Robot learning from demonstration in robotic assembly: A survey. Journal of Robotics, 7(2):17.
- **Z. Zhu**, H. Hu and D. Gu, "Robot Performing Peg-in-Hole Operations by Learning from Human Demonstration," The Proceedings of the 10th Computer Science and Electronic Engineering (CEEC), Colchester, United Kingdom, 2018, pp. 30-35.
- F. Chao, **Z. Zhu**, C. Lin, H. Hu, L. Yang, C. Shang, and C. Zhou (2018). Enhanced robotic hand-eye coordination inspired from human-like behavioral patterns. IEEE Transactions on Cognitive and Developmental Systems, 10(2):384–396.
- R. Wu, C. Zhou, F. Chao, **Z. Zhu**, C.M. Lin, and L. Yang (2017). A developmental learning approach of mobile manipulator via playing. Journal of Frontiers in Neurorobotics, Vol. 11, Article 53.

The following paper has been submitted for potential publication:

- **Z. Zhu** and H. Hu (2020), "A Novel Skeleton-Joint Mapping Framework for Robot Learning Assembly Skills from Observing Human Demonstrations, submitted to Journal of Robotics and Autonomous Systems for review.

6.3 Discussion

In this work, modelling and learning trajectories were deployed rather than learning symbols which is a high-level representation of the skill that decomposes the skill in a sequence of action-perception units [12]. As the assembly task needs precise control in terms of the end actuator's position and force feedback during an insertion task. Although high-level symbol learning and reasoning has been used in the LfD field [1] [95], more research is required to be addressed to fill the gap between trajectory learning and symbol learning, especially in the area of robotic assembly.

Although the skeleton-joint mapping works for both robotic arms and was validated in Section 4.5.3, this thesis only used one arm of the Baxter robot to learn and perform the assembly skill in the robotic assembly tasks. This simplification helps the robotic arm to obtain robust force feedback during the final insertion action, as two-arm interactions and force modeling are avoided. Besides, the robot arm is assumed to keep the perpendicular pose, which assures the workpiece to stay stationary on the desk during the assembly.

Besides, the joints mapped with human skeleton are limited to 4 joints, s_0, s_1, e_0, e_1 due to Baxter's hardware limitation. For a more dexterous robotic arm with more motor joints, the *human skeleton-joint mapping* is still useful. As presented in Figure 4.2, there are more human skeleton joints could be used for mapping, like *HandTipLeft, ThumbLeft, HandTipRight, ThumbRight, WristLeft,* and *WristRight*. Using a similar mapping strategy, these human joints could be utilised in teaching robots to learn more complicated skills.

The proposed visual mapping demonstration provides feedback during teaching, as the human teacher monitor the robot's real-time following performance. However, this kind of supervising is limited as it works more like real-time feedback to the teacher, so the teacher knows that he/she is performing right and the skill has been saw by the robot. If the teacher makes a mistake, like moving towards the wrong direction, the robot can not correct himself as the robot still does not know what is correct. The robot is still learning like a student learns a new lesson and cannot correct the teacher. To improve the teaching quality, the teacher needs to be sophisticated enough to demonstrate the example skills for the robot. For example, the demonstrator need to consider about the robot's physical limitation.

In this thesis, the force threshold used for controlling the assembly task is assumed as prior knowledge. From the point of efficient learning for the robot, this is a good strategy to simplify the learning process, as well as to specify and focus the research topic on motion learning. However, to build a highly autonomous learning robotic system, the force used in the assembly task should be learned through interactive learning between the robot and the environment. The autonomous learning feature is especially important when the robot performs tasks in unstructured environments. In the case of novel environments, reinforcement learning (RL) could be introduced to help the robot adapt to the new situation and find a proper parameter or model.

The proposed research method has many advantages compared with existing work, while it also has its limitations. Firstly, the visual skeleton-joint mapping is not as precise as kinesthetic teaching. The latter demonstration method uses hand-by-hand teaching, and the robotic arm is under the direct control of the human demonstrator. So there is no lag in the teaching. Besides, the demonstrator can teach the robot in a more handy way. Secondly, the robotic assembly tasks learned in this thesis do not cover all the assembly sub-tasks. This thesis mainly focuses on the pick-and-place and the insertion task, while robotic assembly consists of many other tasks like slide-in-the-groove, bolt screwing, pipe connection, and a combination of these tasks like chair assembly. Besides, some of these tasks need special robotic hand like a hand with suction mechanism.

6.4 Future work

A unique framework for LfD learning with observing, representing, and reproducing steps was created for robotic assembly tasks. Rather than the traditional kinesthetic teaching, the robot observes the demonstrations through a visual sensor. The observed skills were modelled with GMM and reproduced with GMR. By optimizing with the EM algorithm, the models converge faster and have smaller MSE errors. Besides, the object tracking was integrated with the observing and reproducing stages to effectively start an assembly task. The effectiveness and usefulness of the approach were demonstrated with practical examples of the assembly task conducted by a Baxter robot. The experimental results showed how a non-expert could easily teach the robot skills by using this approach.

During the skeleton-joint mapping, some of the tracked joints (see Figure 4.2 and Figure 4.3) were used in the transforming formulations. More specifically, only 8 skeleton joints (see Figure 4.4) were used to track the arm motion of the human, which is adequate for the robot to learn simple skills, e.g., picking and assembling. However, more joints should be utilised if the demonstrated skills need more dexterity to be achieved. For example, the wrist joint could be mapped into the motor W_1, and the fingertip of the human hand can be mapped into the motor W_2, see Figure 4.5. The fingertip, wrist joint, and thumb joint can be used to demonstrate the bolt-screwing task, which is critical for advanced robotic assembly like chair assembly. Improve the dexterity of the skeleton-joint mapping is of the interest of future research.

Appendix A

Supplementary Figures of Chapter 4

A.1 Double Exponential Smoothing Applied on Human Joints

- Figure A.1 shows that the human elbow joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.
- Figure A.2 shows that the human elbow joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.
- Figure A.3 shows that the human right shoulder joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.
- Figure A.4 shows that the human right shoulder joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.
- Figure A.5 shows that the human left shoulder joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.
- Figure A.6 shows that the human left shoulder joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

- Figure A.7 shows that the human torso joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.
- Figure A.8 shows that the human torso joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.
- Figure A.9 shows that the human head joint trajectory is smoothed by double exponential algorithm with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.
- Figure A.10 shows that the human head joint trajectory is smoothed by double exponential algorithm with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

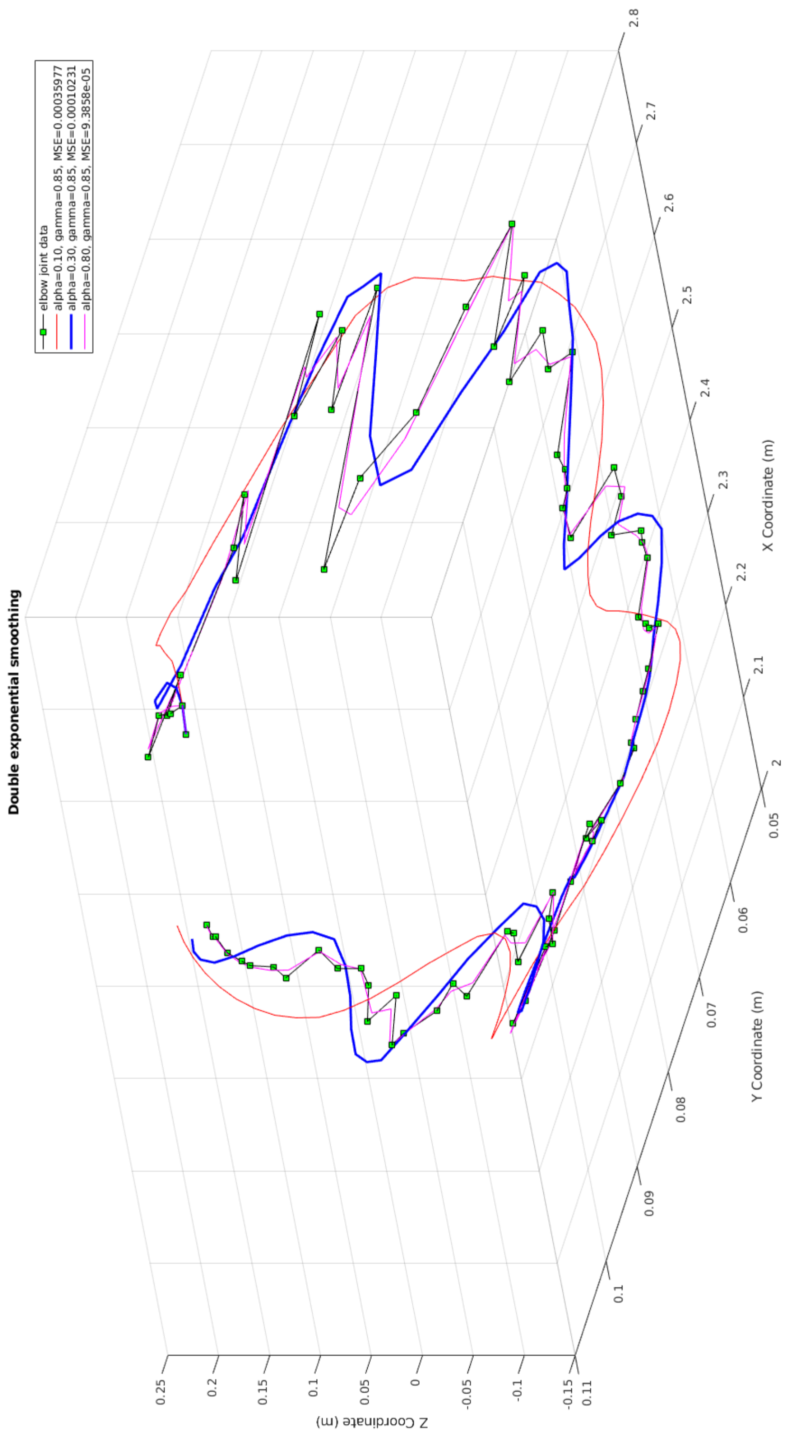


Fig. A.1 Double exponential smoothing applied on human elbow joint data with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.

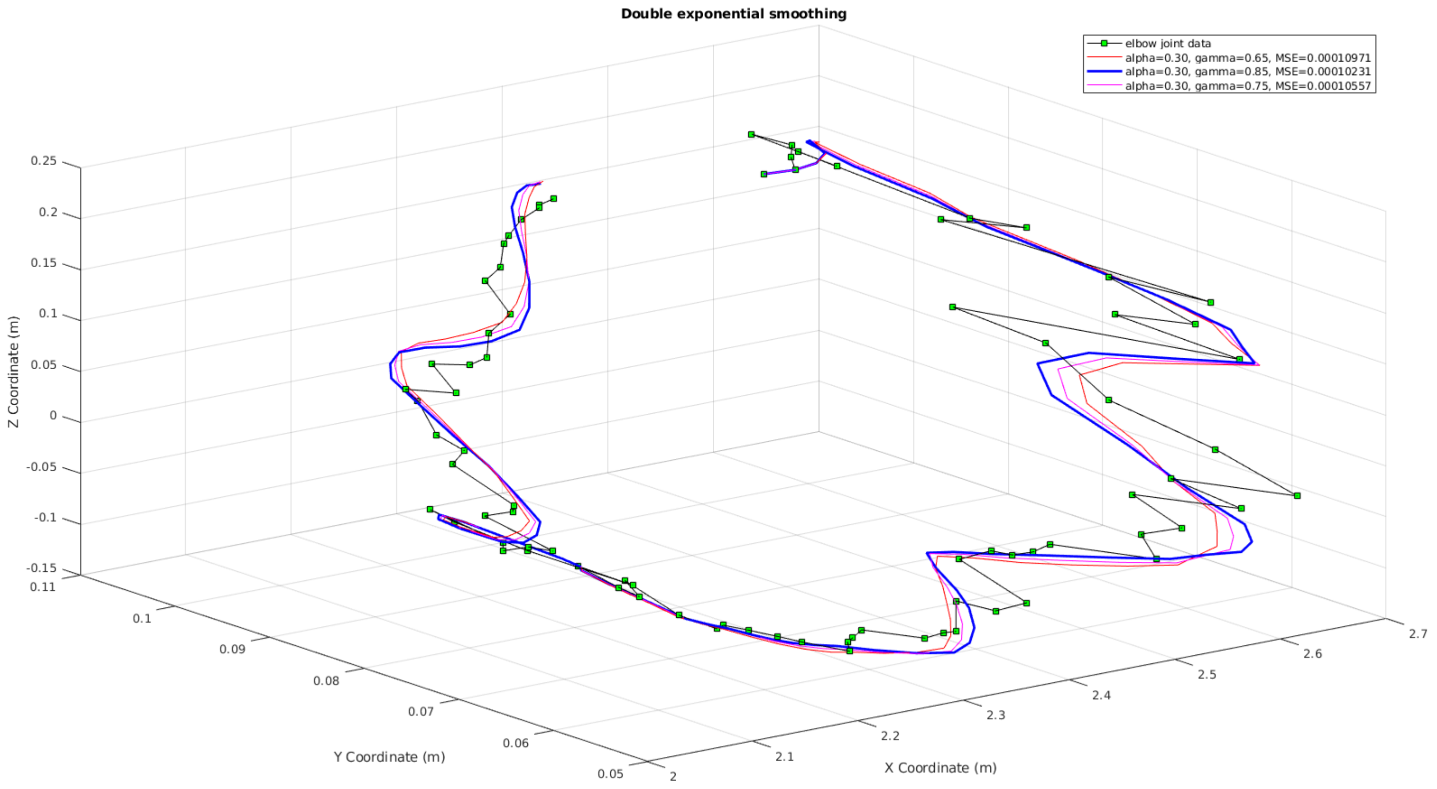


Fig. A.2 Double exponential smoothing applied on human elbow joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

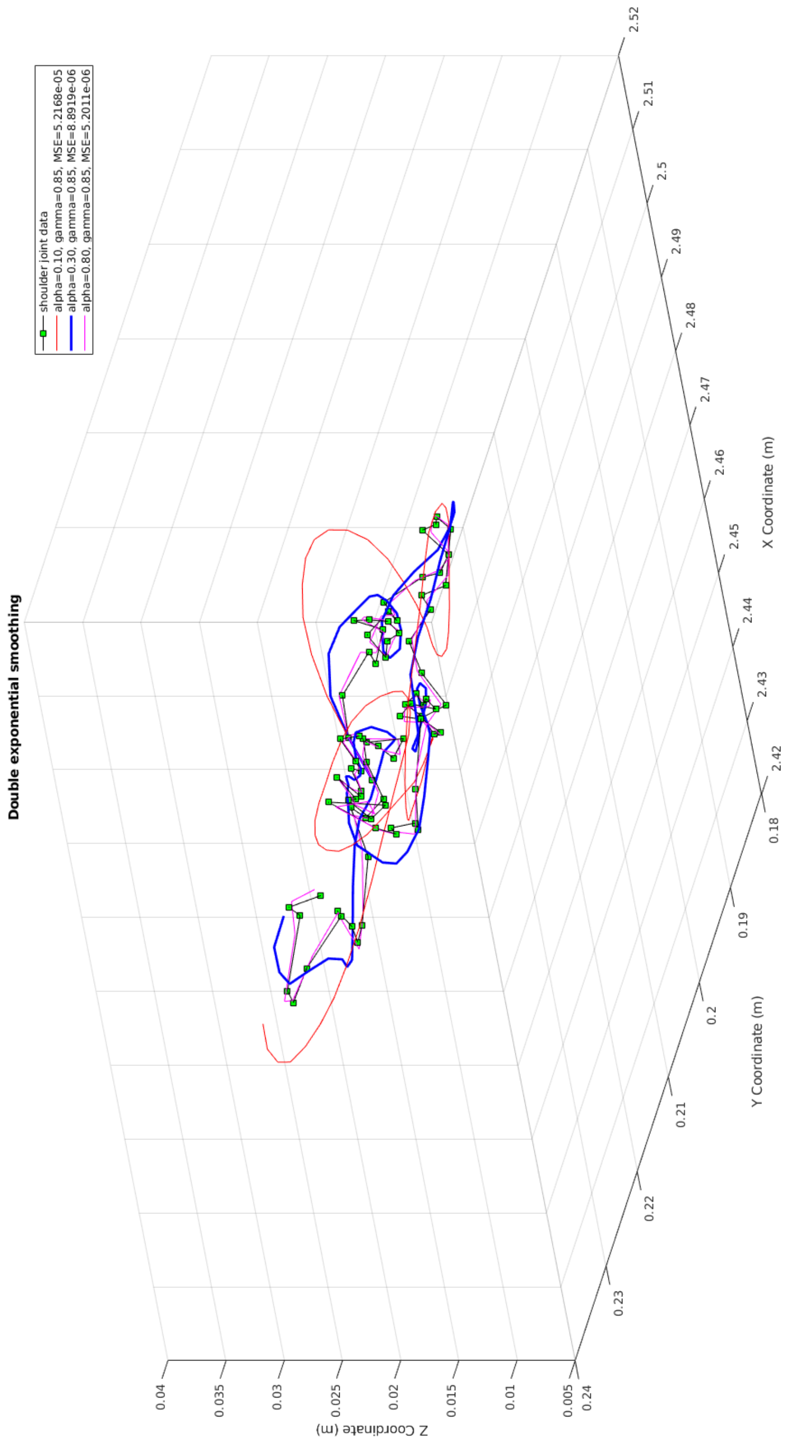


Fig. A.3 Double exponential smoothing applied on human right shoulder joint data with different α, $\alpha \in \{0.10, 0.30, 0.80\}$, and fixed γ, $\gamma = 0.85$.

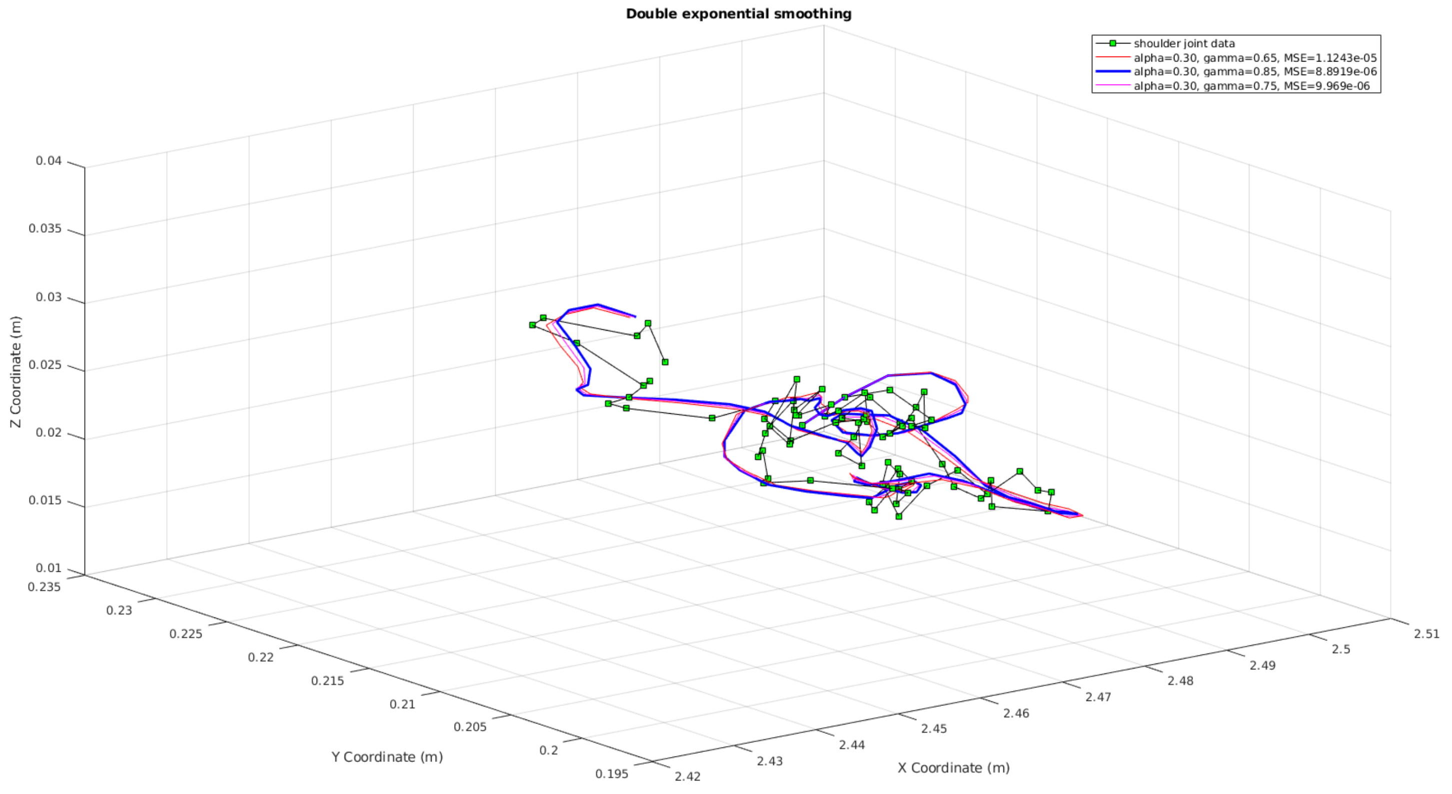


Fig. A.4 Double exponential smoothing applied on human right shoulder joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

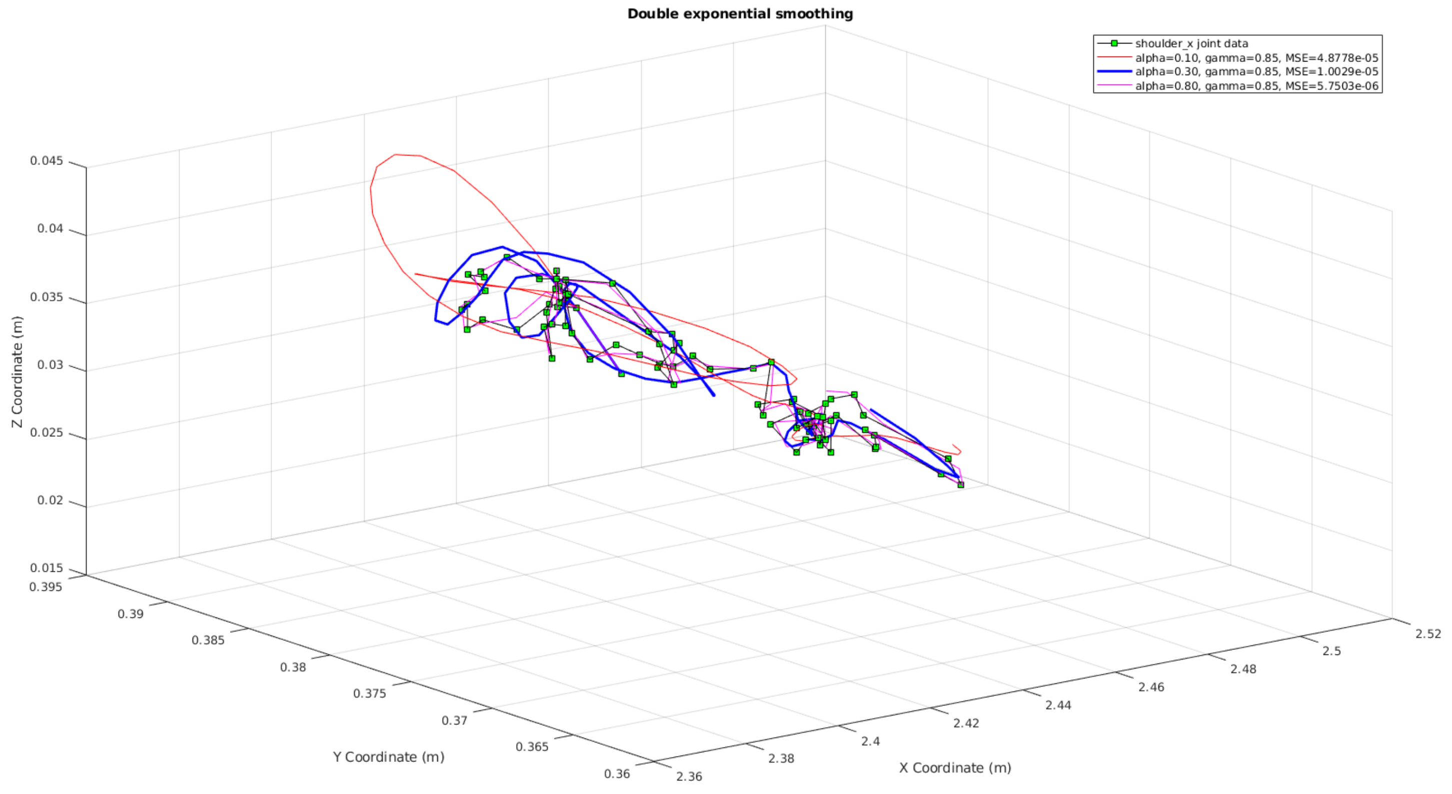


Fig. A.5 Double exponential smoothing applied on human left shoulder joint data with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.

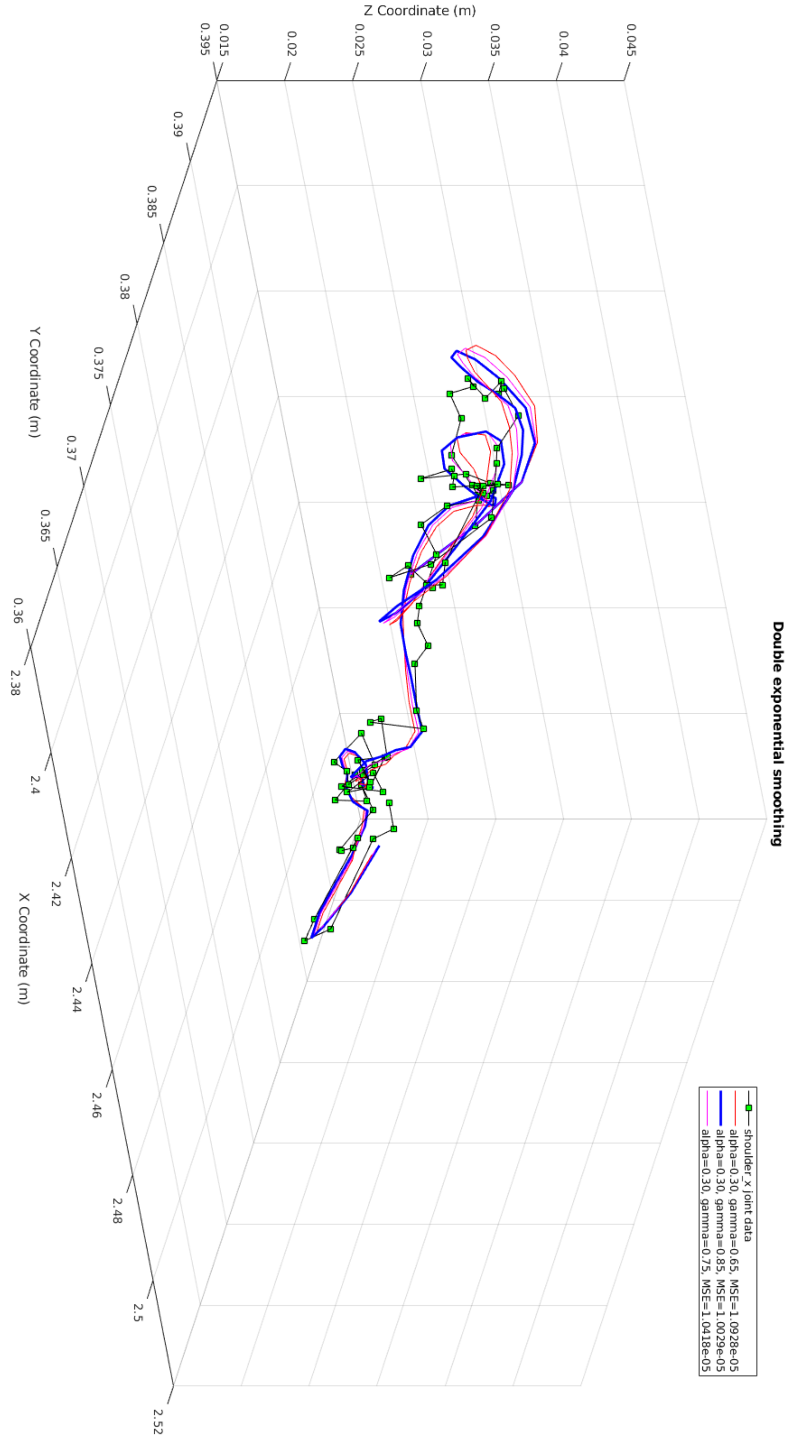


Fig. A.6 Double exponential smoothing applied on human left shoulder joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

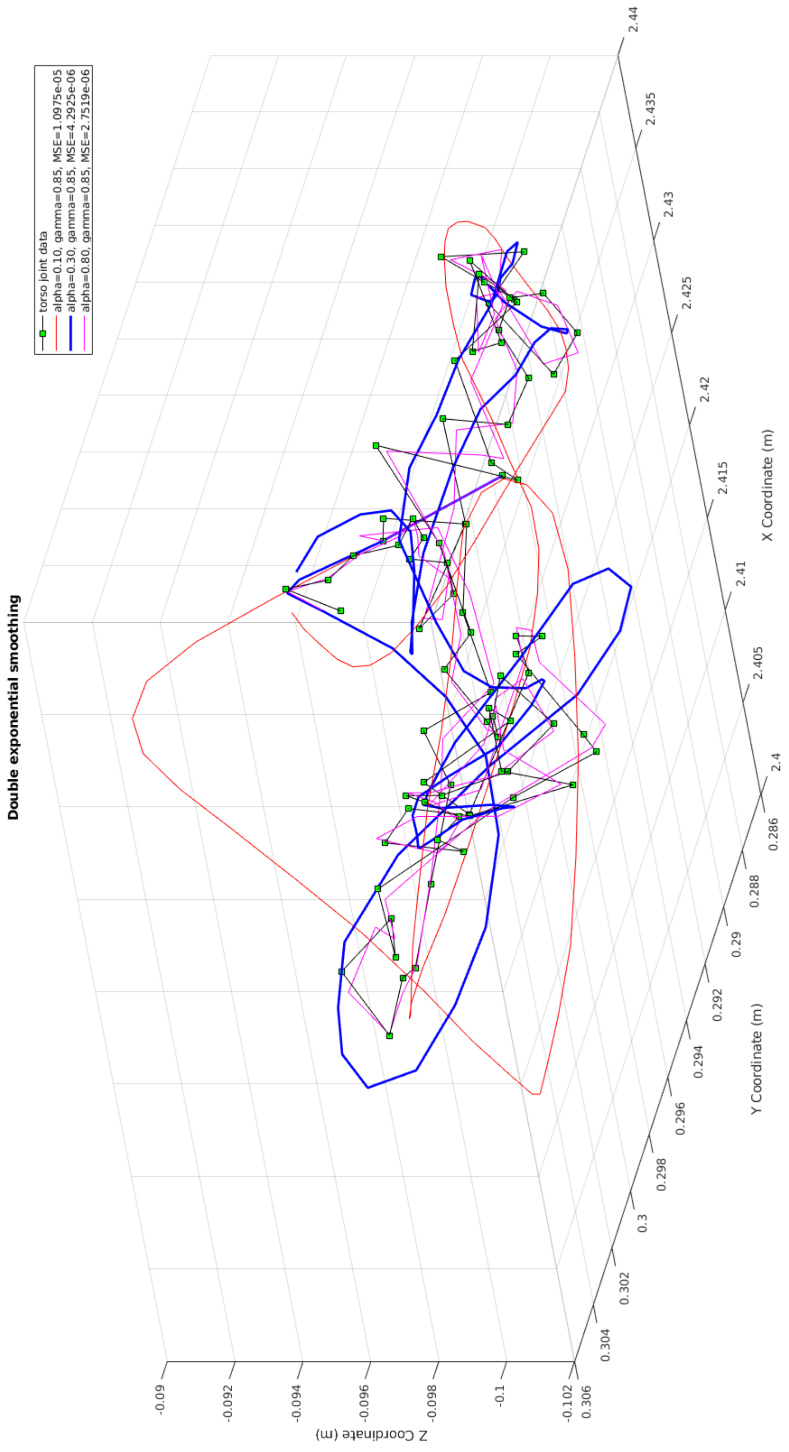


Fig. A.7 Double exponential smoothing applied on human torso joint data with different $\alpha, \alpha \in \{0.10, 0.30, 0.80\}$, and fixed $\gamma, \gamma = 0.85$.

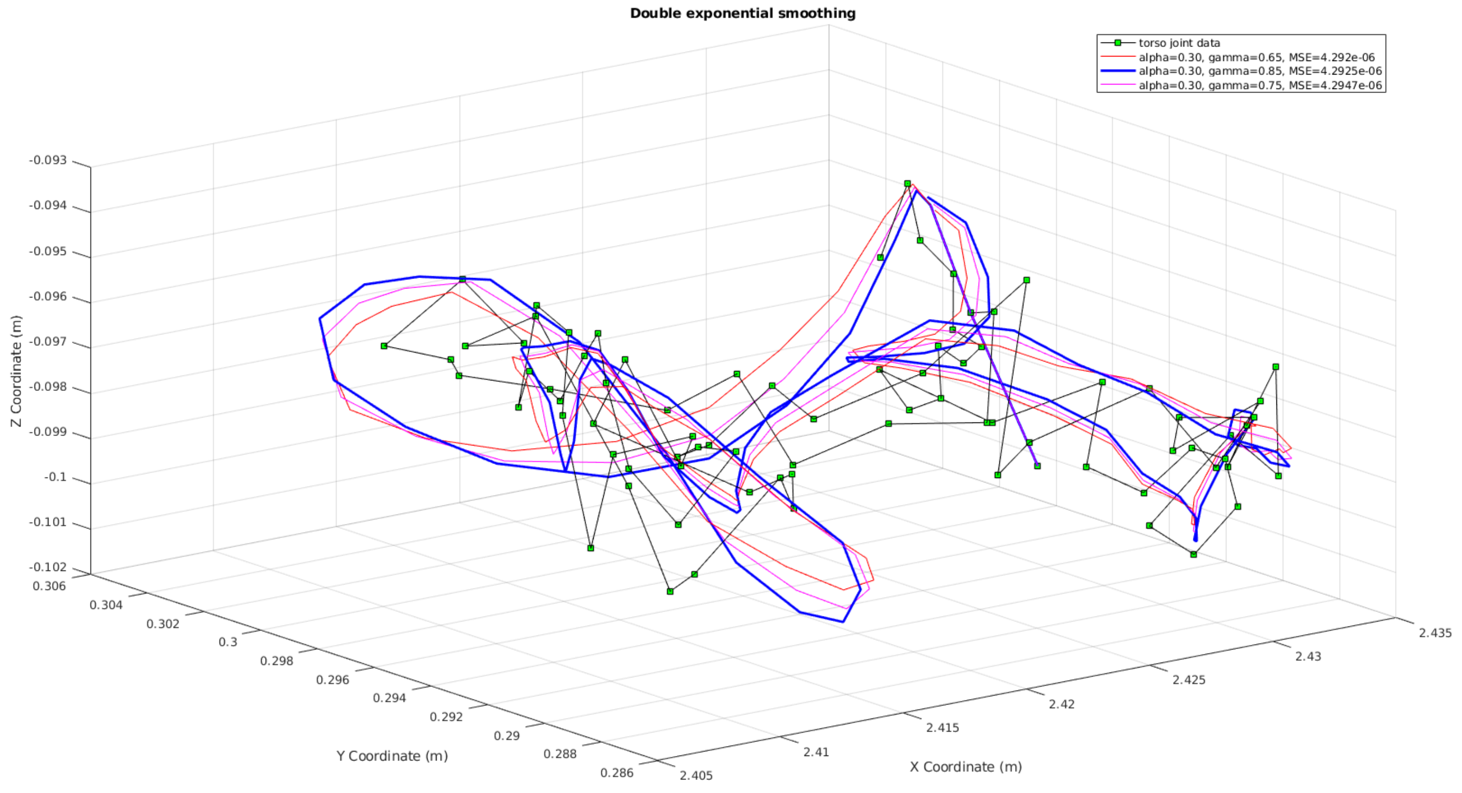


Fig. A.8 Double exponential smoothing applied on human torso joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

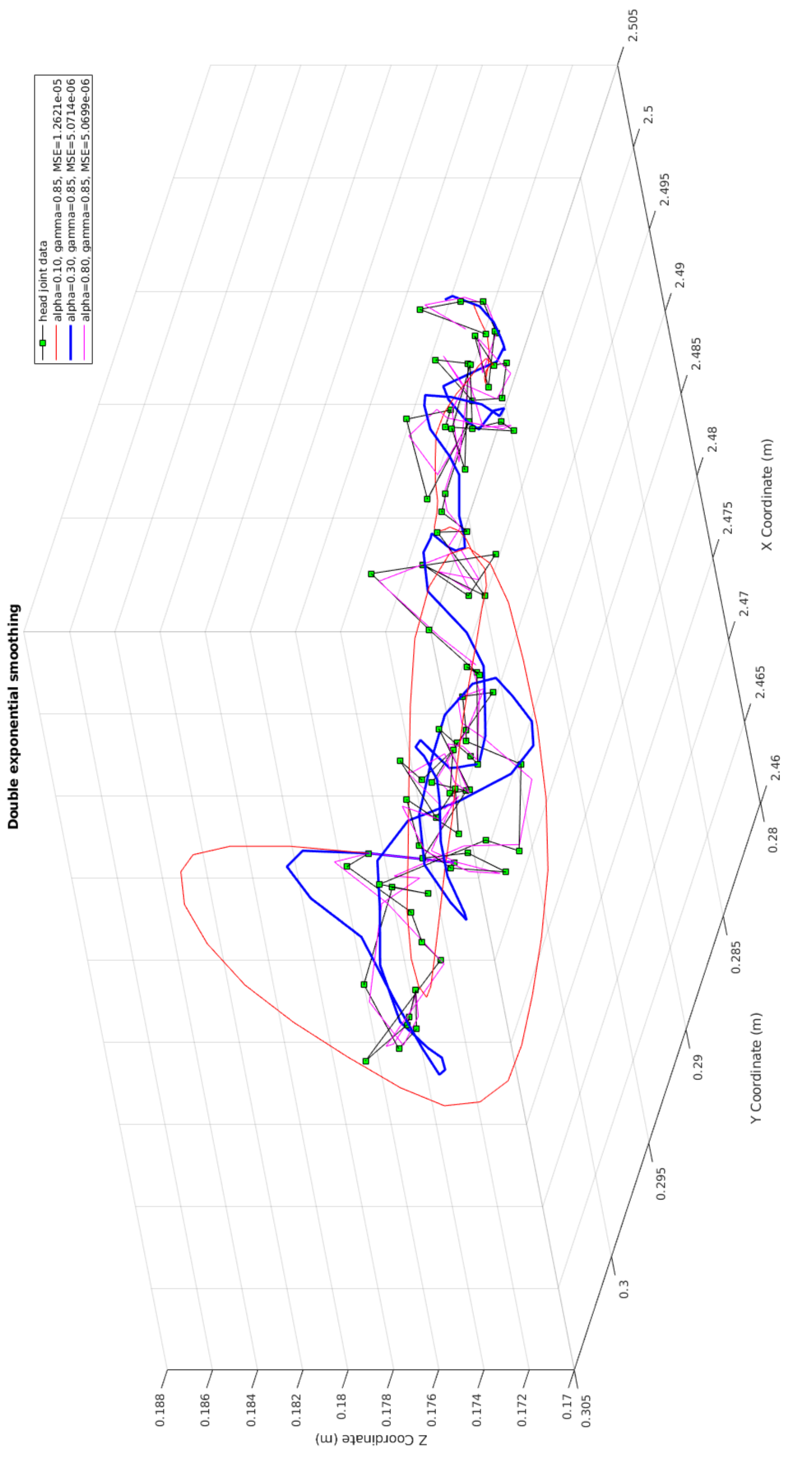


Fig. A.9 Double exponential smoothing applied on human head joint data with different α, $\alpha \in \{0.10, 0.30, 0.80\}$, and fixed γ, $\gamma = 0.85$.

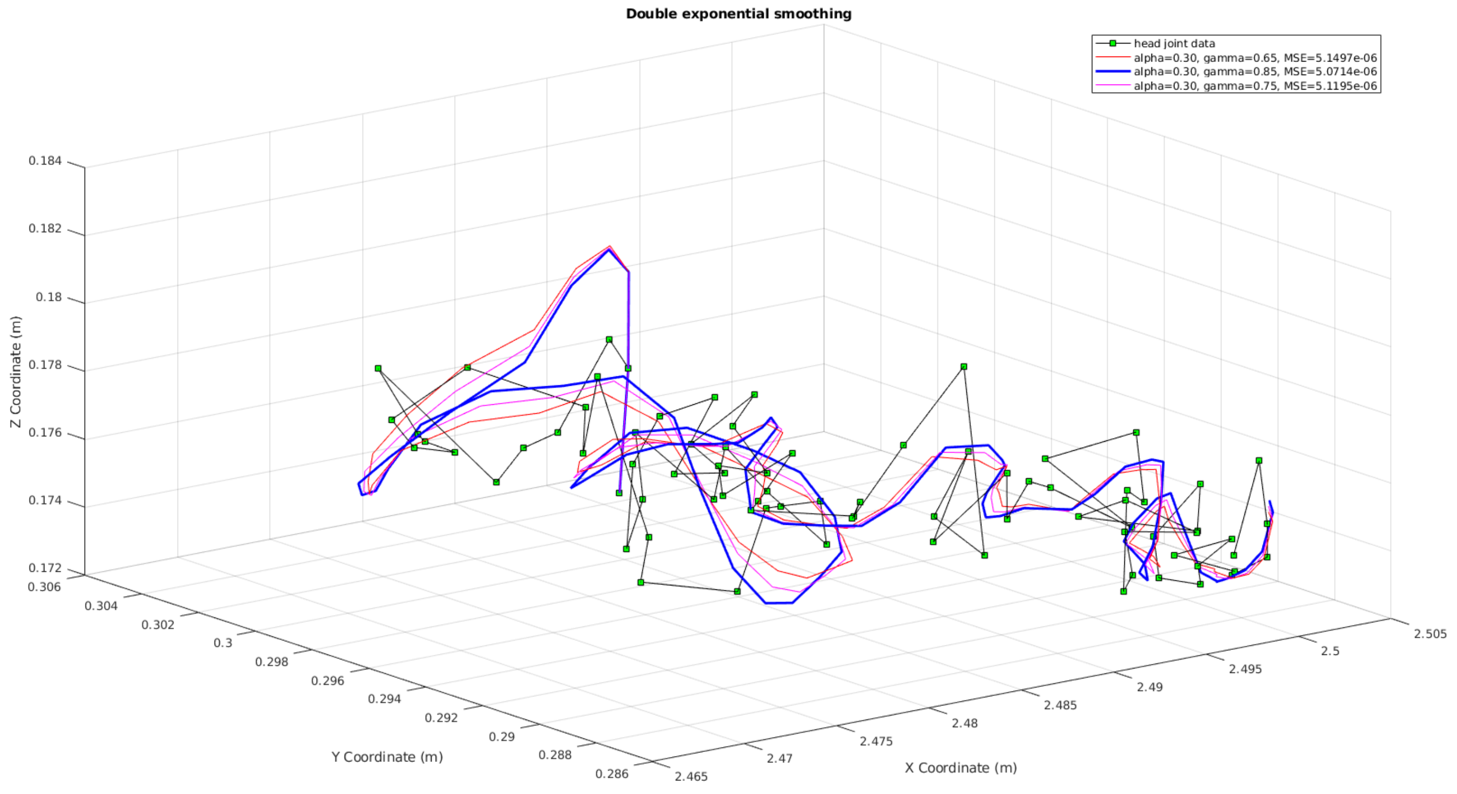


Fig. A.10 Double exponential smoothing applied on human head joint data with different $\gamma, \gamma \in \{0.65, 0.75, 0.85\}$, and fixed $\alpha, \alpha = 0.30$.

A.2 The MSE Distribution of Smoothed Human Joint Trajectories

Figure A.11 is the MSE distribution of human elbow joint after being smoothed based on different smoothing factors. The picture at the top shows the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 7.3166e-04, where\ \alpha = 0.62, \gamma = 0.57$. The picture at the bottom is the local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.0231e-04, where\ \alpha = 0.30, \gamma = 0.85$.

Figure A.12 is the MSE distribution of human right shoulder joint after being smoothed based on different smoothing factors. The picture at the top shows the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 4.6028e-06, where\ \alpha = 0.95, \gamma = 0.385$. The picture at the bottom is the local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 8.8919e-06, where\ \alpha = 0.30, \gamma = 0.85$.

Figure A.13 is the MSE distribution of human left shoulder joint after being smoothed based on different smoothing factors. The picture at the top shows the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 5.111e-06, where\ \alpha = 0.95, \gamma = 0.35$. The picture at the bottom is the local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.0029e-05, where\ \alpha = 0.30, \gamma = 0.85$.

Figure A.14 is the MSE distribution of human torso joint after being smoothed based on different smoothing factors. The picture at the top shows the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 2.1853e-06, where\ \alpha = 0.95, \gamma = 0.20$. The picture at the bottom is the local distribution, $\alpha \in$

$\{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 4.2919e-06, where\ \alpha = 0.30, \gamma = 0.665$.

Figure A.15 is the MSE distribution of human head joint after being smoothed based on different smoothing factors. The picture at the top shows the global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 3.5854e-06, where\ \alpha = 0.765, \gamma = 0.185$. The picture at the bottom is the local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 5.0714e-06, where\ \alpha = 0.30, \gamma = 0.85$.

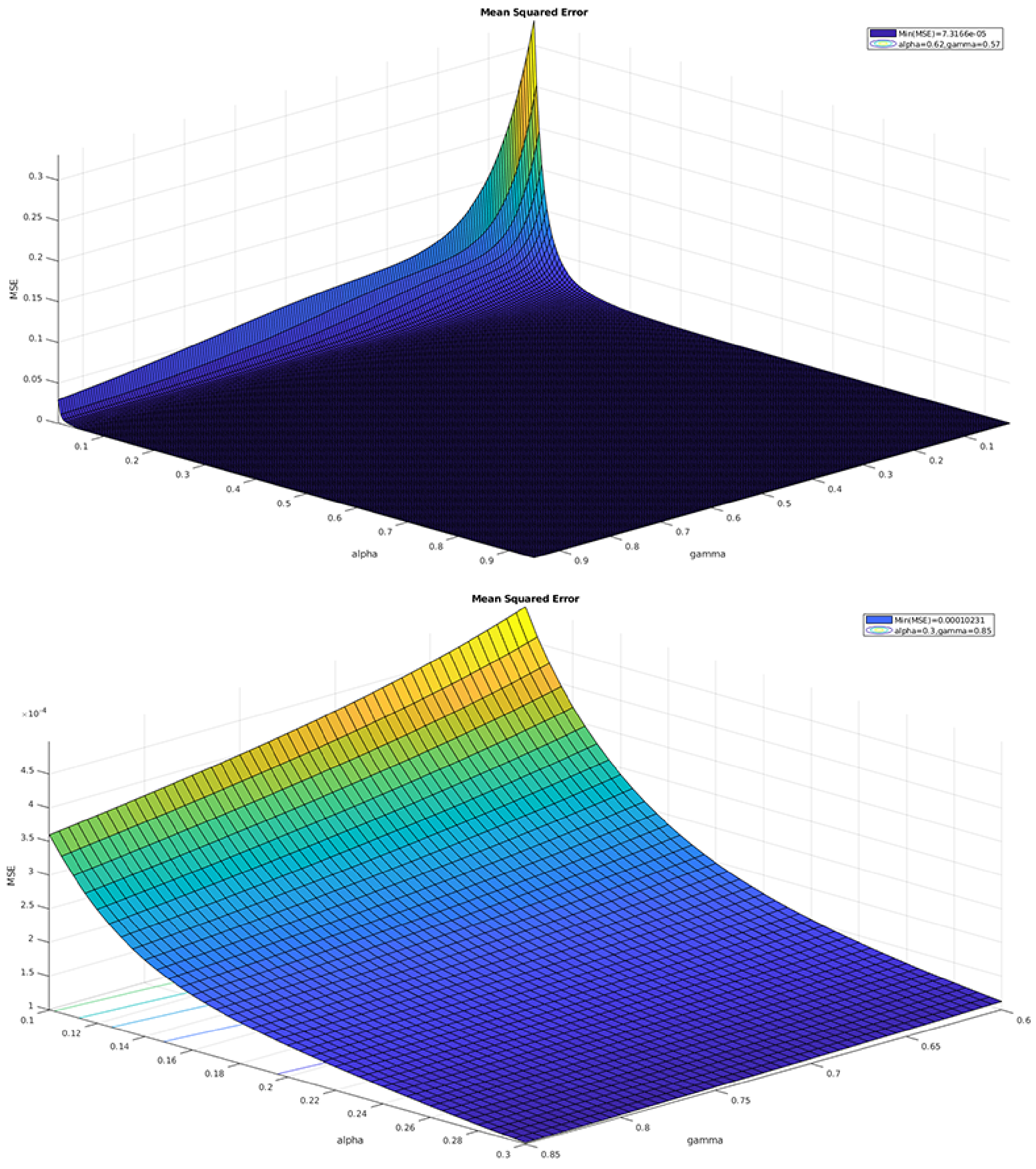


Fig. A.11 The MSE distribution of human elbow joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 7.3166e-04, where\ \alpha = 0.62, \gamma = 0.57$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.0231e-04, where\ \alpha = 0.30, \gamma = 0.85$.

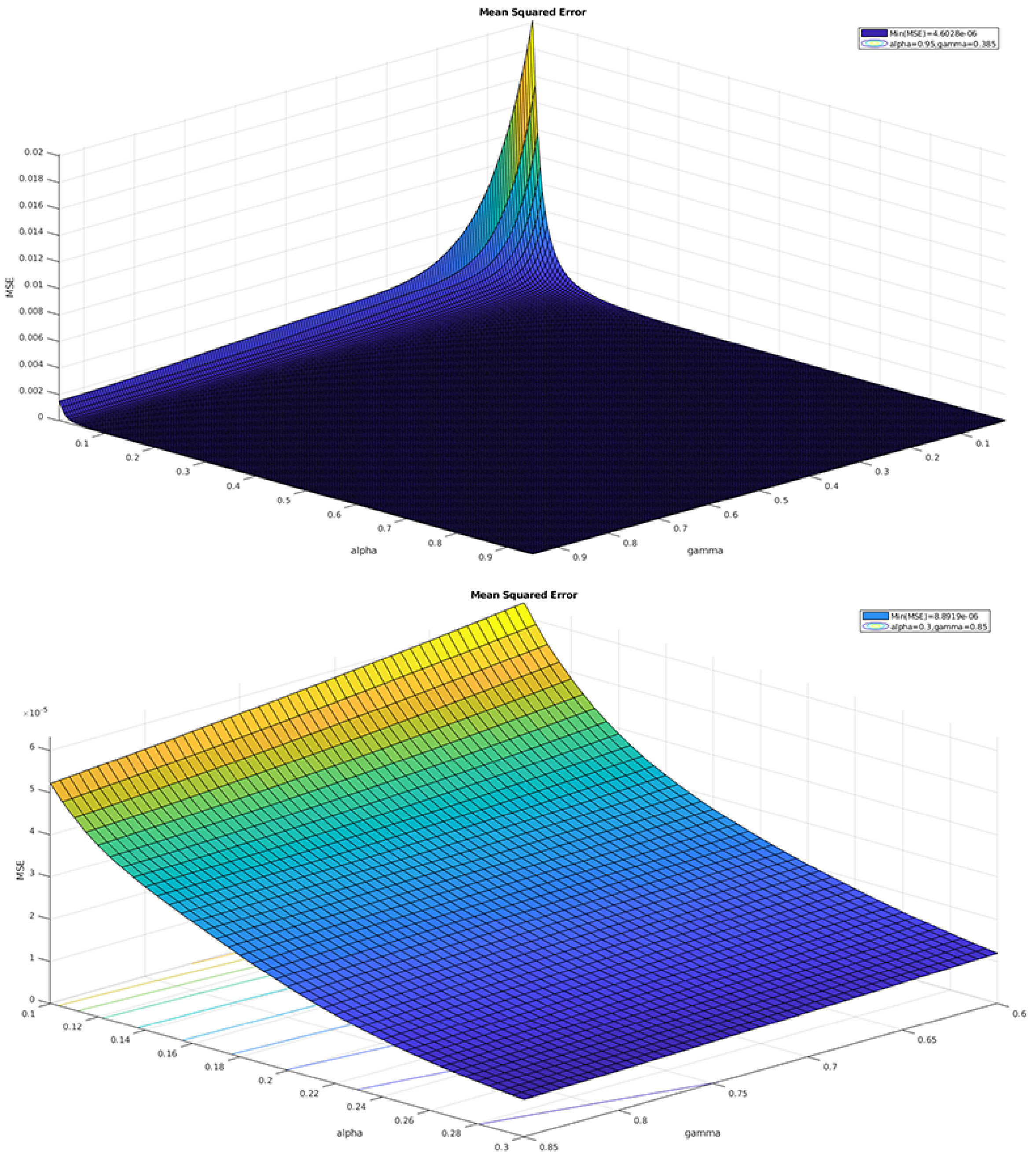


Fig. A.12 The MSE distribution of human right shoulder joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha,\gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 4.6028e-06, where\ \alpha = 0.95, \gamma = 0.385$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 8.8919e-06, where\ \alpha = 0.30, \gamma = 0.85$.

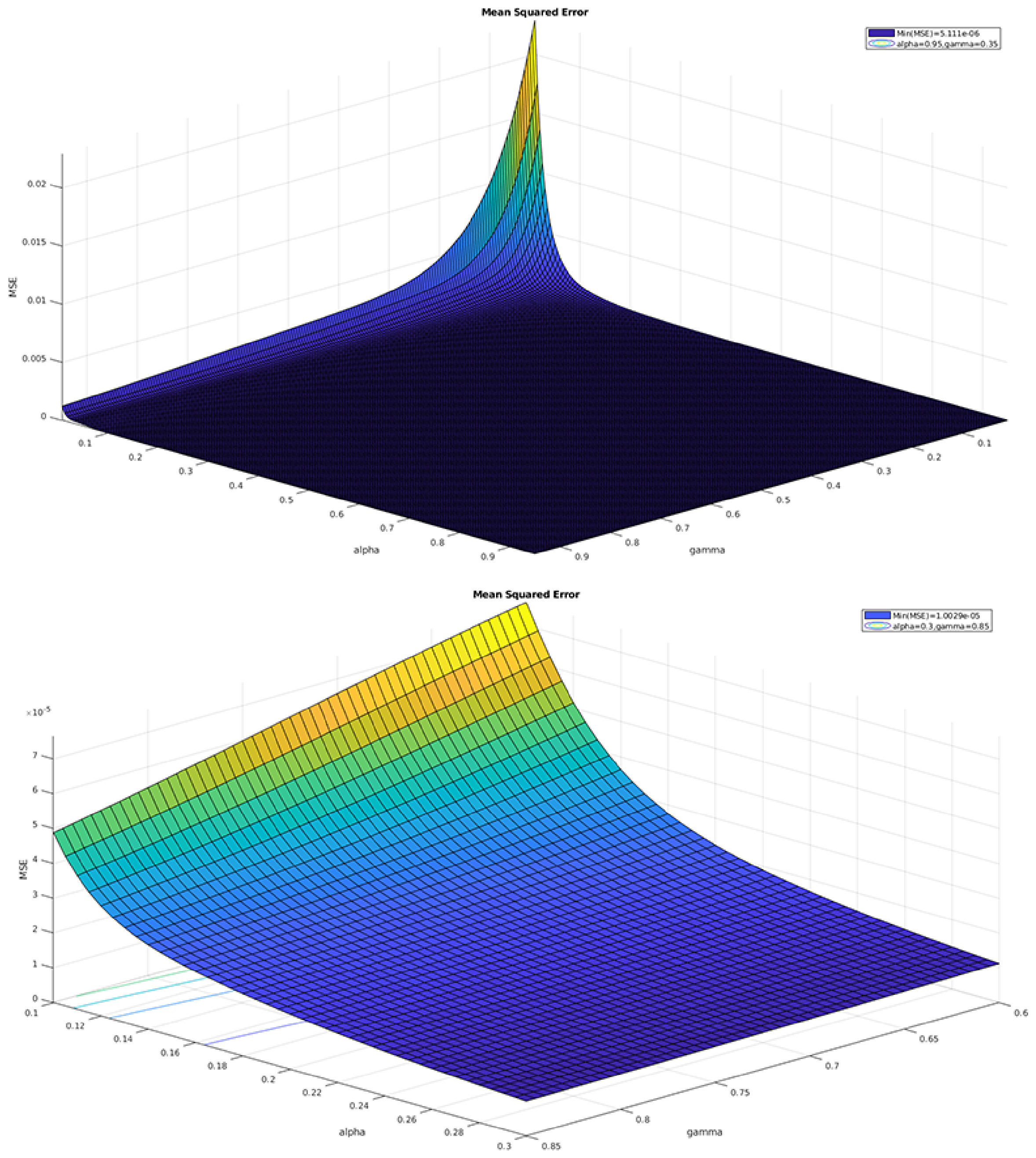
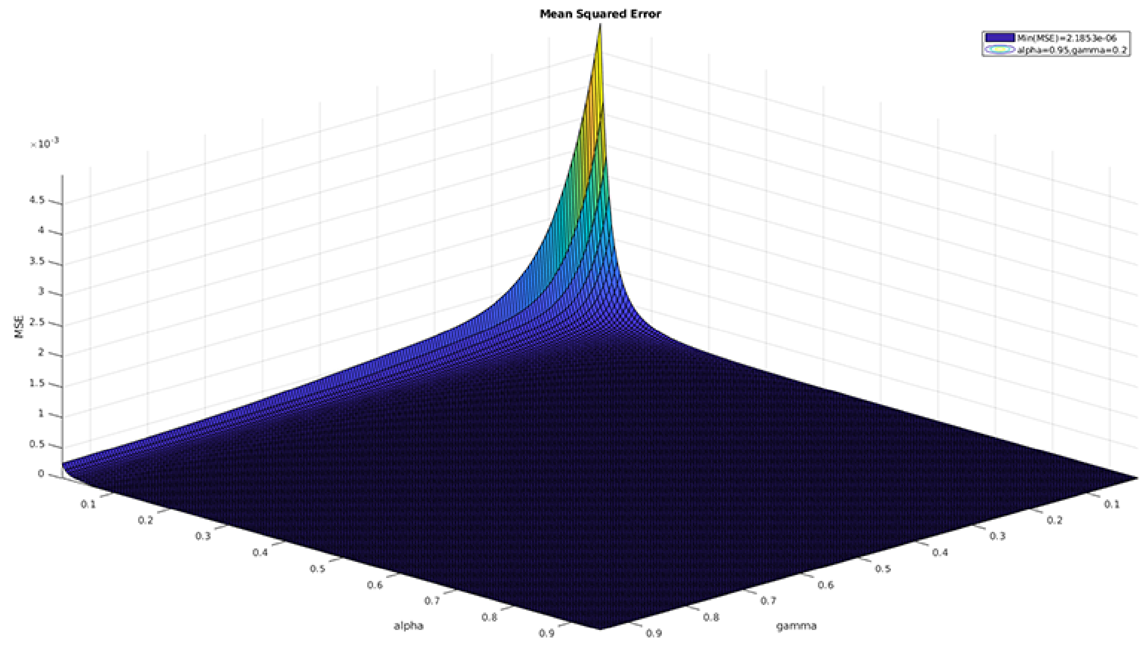


Fig. A.13 The MSE distribution of human left shoulder joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 5.111e-06, where\ \alpha = 0.95, \gamma = 0.35$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 1.0029e-05, where\ \alpha = 0.30, \gamma = 0.85$.



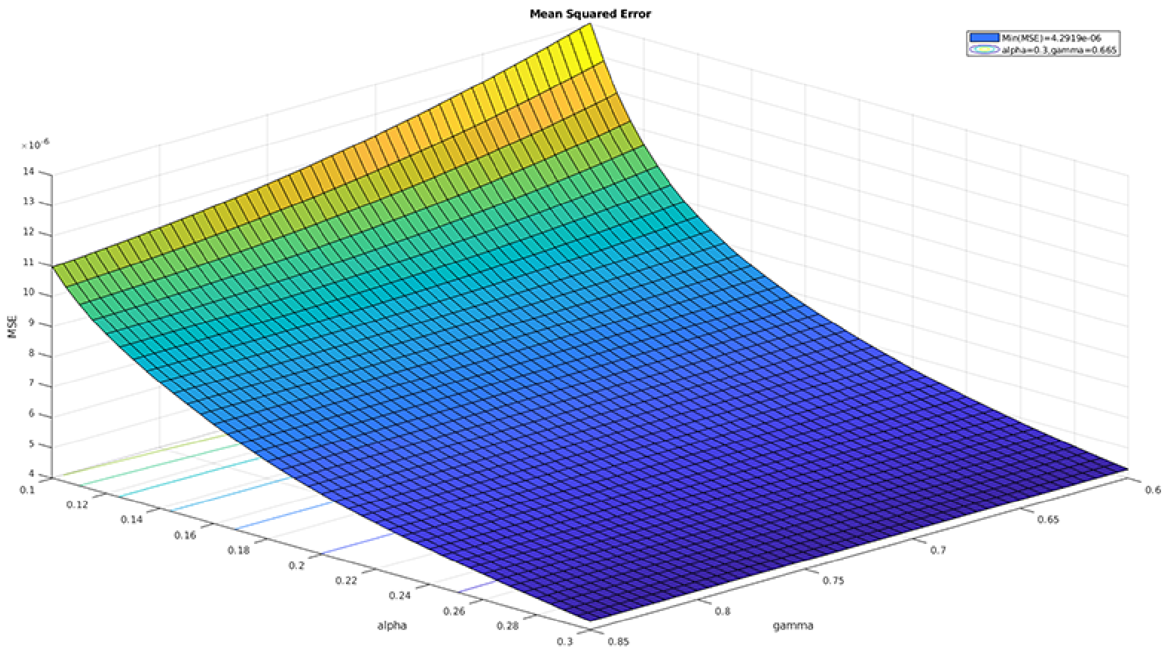
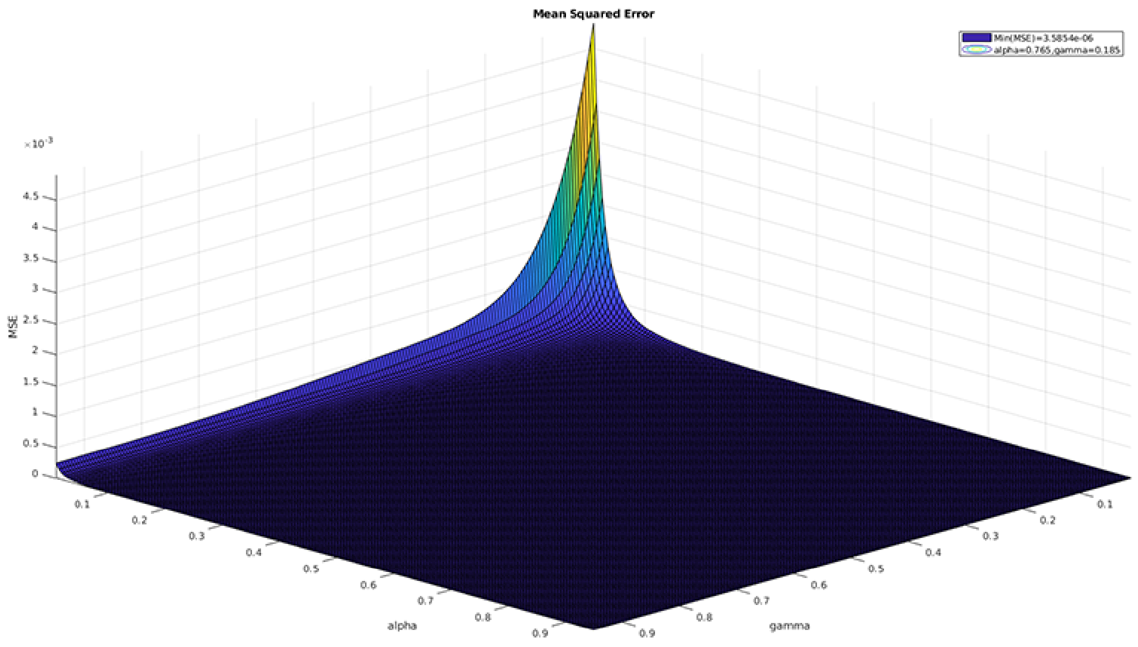


Fig. A.14 The MSE distribution of human torso joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 2.1853e-06, where\ \alpha = 0.95, \gamma = 0.20$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 4.2919e-06, where\ \alpha = 0.30, \gamma = 0.665$.



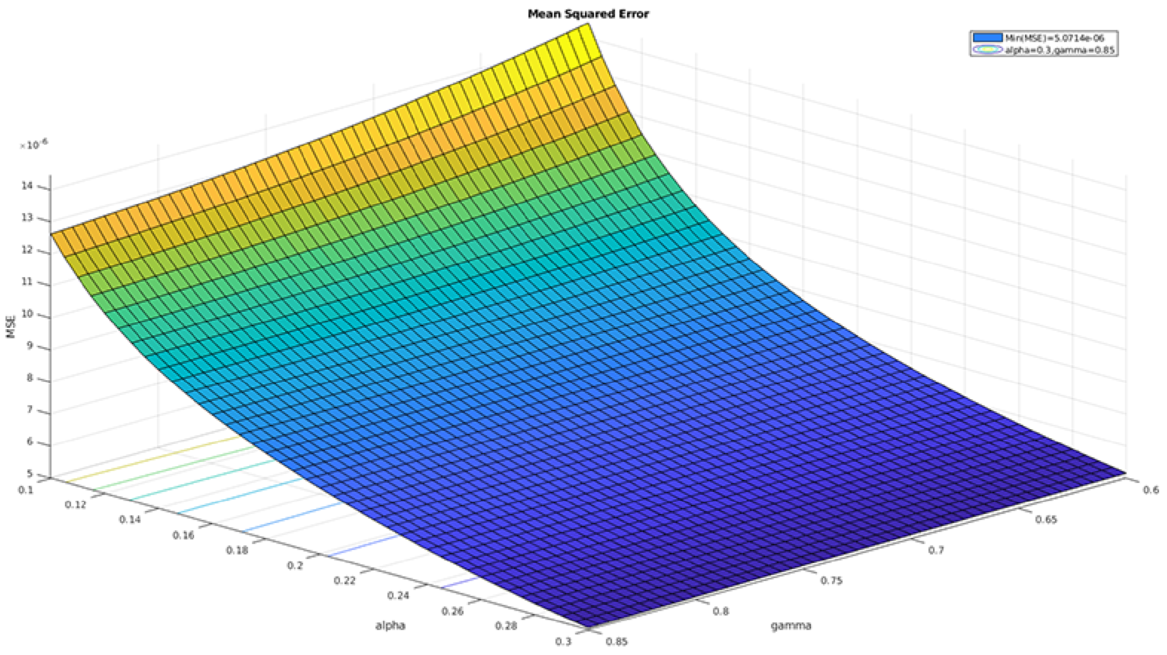


Fig. A.15 The MSE distribution of human head joint after being smoothed based on different smoothing factors. Top: Global distribution, $\alpha, \gamma \in \{0.01, 0.95\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 3.5854e - 06, where\ \alpha = 0.765, \gamma = 0.185$. Bottom: Local distribution, $\alpha \in \{0.10, 0.30\}$, $\gamma \in \{0.60, 0.85\}$, iteration step $\Delta\alpha = 0.005, \Delta\gamma = 0.005$. The minimum $MSE = 5.0714e - 06, where\ \alpha = 0.30, \gamma = 0.85$.

Appendix B

Supplementary Code of Chapter 4

B.1 *cob_body_tracker* node

```
/*
 * Publishes user message [@cob_perception_msgs::Person] on the given
    topic of [@people_pub] publisher.
 */
void BodyTracker::publishTrackedUserMsg()
{
  //publish tracked users
  std::vector<cob_perception_msgs::Person> detected_people;
  for(std::list<nite::UserData>::iterator iter_ = tracked_users_->
   begin(); iter_ != tracked_users_->end(); ++iter_)
  {
      cob_perception_msgs::Person person;
      cob_perception_msgs::Skeleton skeleton;
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).
   getSkeleton().getJoint(nite::JOINT_HEAD)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).
   getSkeleton().getJoint(nite::JOINT_NECK)));
```

```
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_SHOULDER)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_SHOULDER)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_ELBOW)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_ELBOW)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_HAND)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_HAND)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_TORSO)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_HIP)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_HIP)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_KNEE)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_KNEE)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_LEFT_FOOT)));
      skeleton.joints.push_back(convertNiteJointToMsgs((*iter_).getSkeleton().getJoint(nite::JOINT_RIGHT_FOOT)));
    }
}
```

Listing B.1 BodyTracker

```
HoltWintersSmoothFilter::HoltWintersSmoothFilter( double initialB, double alfa, double beta) {
    alfa_ = alfa;
    beta_ = beta;
    reset(initialB);
}
void HoltWintersSmoothFilter::insert(const double& v )
{
   double s_prev = s_;
   if(!gotFirst_) {
       s_ = v;
       gotFirst_ = true;
   }
   else {
       // update the estimate
       s_ = alfa_ * v + (1 - alfa_) * (s_ + b_);
       // update the estimate of the derivative
       b_ = beta_ * (s_ - s_prev) + (1 - beta_) * b_;
   }
}
// Returns the current smoothed value
double HoltWintersSmoothFilter::getFiltered() {
   return s_;
}
void HoltWintersSmoothFilter::reset(double initialB) {
    gotFirst_ = false;
    b_ = initialB;
}
void HoltWintersSmoothFilter::setAlfa(double alfa) {
    HoltWintersSmoothFilter::alfa_ = alfa;
}
void HoltWintersSmoothFilter::setBeta(double beta) {
    HoltWintersSmoothFilter::beta_ = beta;
```

```
}
```

Listing B.2 HoltWintersSmoothFilter

```
  cob_perception_msgs::People array;
  array.header.stamp = ros::Time::now();
  array.header.frame_id = depth_optical_frame_;
  array.people = detected_people;
  people_pub_.publish(array);
```

Listing B.3 people_pub_.publish

B.2 *kinect2_launch_joint_pub* node

```
def tf_2_angles(shoulder, shoulder_x, elbow, hand,  torso, head, base
   , listener, limb_name):
    //retrieve positions of human joints, then continue transforming
    try:
        (b_2_s, trash) = listener.lookupTransform(base, shoulder,
   rospy.Time(0))
        (b_2_s_x, trash) = listener.lookupTransform(base, shoulder_x,
    rospy.Time(0))
        (b_2_e, trash) = listener.lookupTransform(base, elbow, rospy.
   Time(0))
        (b_2_hand, trash) = listener.lookupTransform(base, hand,
   rospy.Time(0))
        (b_2_t, trash) = listener.lookupTransform(base, torso, rospy.
   Time(0))
        (b_2_head, trash) = listener.lookupTransform(base, head,
   rospy.Time(0))
    except (tf.LookupException, tf.ConnectivityException, tf.
   ExtrapolationException):
        print "tf exception"
        return {}
    b_2_s = list(b_2_s)
    b_2_s_x = list(b_2_s_x)
    b_2_e = list(b_2_e)
    b_2_hand = list(b_2_hand)
    b_2_t = list(b_2_t)
    b_2_head = list(b_2_head)
    s_2_s_x = map(operator.sub, b_2_s_x, b_2_s)
    s_2_t = map(operator.sub, b_2_t, b_2_s)
    s_2_e = map(operator.sub, b_2_e, b_2_s)
    e_2_hand = map(operator.sub, b_2_hand, b_2_e)
    head_2_t = map(operator.sub, b_2_t, b_2_head)
    my_v2 = head_2_t
    my_v3 = vector_cross_product(s_2_e, my_v2)
    my_theta_1 = angle_query(my_v3, s_2_s_x)
    if limb_name == 'right':
        radius_s0 = w_s0*(math.pi/4-my_theta_1)
    else:
        radius_s0 = w_s0*( math.pi*3/4-my_theta_1)

    if mirror:
        if limb_name == 'left':
            radius_s0 = w_s0*( -radius_s0+math.pi/2)
        else:
            radius_s0 = w_s0*( -radius_s0-math.pi/2)

    my_theta_2 = angle_query(my_v2, s_2_e)
    radius_s1 = w_s1*(math.pi/2-my_theta_2)
    my_v4 = vector_cross_product(s_2_e, e_2_hand)
    my_theta_3 = angle_query(my_v3,my_v4)

    radius_e0 = w_e0*my_theta_3
    if limb_name == 'left':
        radius_e0 = w_e0*(-radius_e0)
    my_theta_4 = angle_query(s_2_e,e_2_hand)
    radius_e1 = w_e1*my_theta_4
    return {"s0":radius_s0, "s1":radius_s1, "e0":radius_e0, "e1":
radius_e1}
```

Listing B.4 tf_2_angles

```
def talker():
    global pub_list
    rospy.init_node("kinect2_launch_joint_pub", anonymous=True)
    pub_left_s0 = rospy.Publisher('left_s0', Float64, queue_size=10)
    pub_left_s1 = rospy.Publisher('left_s1', Float64, queue_size=10)
    pub_left_e0 = rospy.Publisher('left_e0', Float64, queue_size=10)
    pub_left_e1 = rospy.Publisher('left_e1', Float64, queue_size=10)
    pub_right_s0 = rospy.Publisher('right_s0', Float64, queue_size
=10)
    pub_right_s1 = rospy.Publisher('right_s1', Float64, queue_size
=10)
    pub_right_e0 = rospy.Publisher('right_e0', Float64, queue_size
=10)
    pub_right_e1 = rospy.Publisher('right_e1', Float64, queue_size
=10)
    pub_list = {'left_s0':pub_left_s0,
                'left_s1':pub_left_s1,
                'left_e0':pub_left_e0,
                'left_e1':pub_left_e1,
                'right_s0':pub_right_s0,
                'right_s1':pub_right_s1,
                'right_e0':pub_right_e0,
                'right_e1':pub_right_e1}
```

```

def track_one_arm(shoulder, shoulder_x, elbow, hand, torso, head,
    base, listener, limb_name):
    pub_list[limb_name+'_s0'].publish(radius_s0)
    pub_list[limb_name+'_s1'].publish(radius_s1)
    pub_list[limb_name+'_e0'].publish(radius_e0)
    pub_list[limb_name+'_e1'].publish(radius_e1)
```

Listing B.5 talker

B.3 *kinect2_launch_joint_sub* node

```
    rospy.Subscriber("left_s0", Float64, call_back_gen('left', 0))
    rospy.Subscriber("left_s1", Float64, call_back_gen('left', 1))
    rospy.Subscriber("left_e0", Float64, call_back_gen('left', 2))
    rospy.Subscriber("left_e1", Float64, call_back_gen('left', 3))
    rospy.Subscriber("right_s0", Float64, call_back_gen('right', 0))
    rospy.Subscriber("right_s1", Float64, call_back_gen('right', 1))
    rospy.Subscriber("right_e0", Float64, call_back_gen('right', 2))
    rospy.Subscriber("right_e1", Float64, call_back_gen('right', 3))

    angles = left_limb.joint_angles()
    angles['left_s0'] = command['left'][0]
    angles['left_s1'] = command['left'][1]
    angles['left_e0'] = command['left'][2]
    angles['left_e1'] = command['left'][3]
    angles['left_w0'] = 0
    angles['left_w1'] = 0
    angles['left_w2'] = 0
    left_limb.set_joint_positions(angles)

    angles = right_limb.joint_angles()
    angles['right_s0'] = command['right'][0]
    angles['right_s1'] = command['right'][1]
    angles['right_e0'] = command['right'][2]
    angles['right_e1'] = command['right'][3]
    angles['right_w0'] = 0
    angles['right_w1'] = 0
    angles['right_w2'] = 0
    right_limb.set_joint_positions(angles)
```

Listing B.6 set_joint_positions

B.4 *baxter_arm_control* node

```
  // Get controller topic name that it will subscribe to
  if( nh_.getParam("topic", topic_name) ) {
    // Get a node handle that is relative to the base path
    ros::NodeHandle nh_base("~");
    // Create command subscriber custom to baxter
    position_command_sub_ = nh_base.subscribe<baxter_core_msgs::
   JointCommand>
      (topic_name, 1, &BaxterPositionController::commandCB, this);
  }
  else // default "command" topic
  {
    // Create command subscriber custom to baxter
    position_command_sub_ = nh_.subscribe<baxter_core_msgs::
   JointCommand>
      ("command", 1, &BaxterPositionController::commandCB, this);
  }

 void BaxterPositionController::starting(const ros::Time& time)
{
  baxter_core_msgs::JointCommand initial_command;

  // Fill in the initial command
  for(int i=0; i<n_joints_; i++)
  {
    initial_command.names.push_back( position_controllers_[i]->
   getJointName());
    initial_command.command.push_back(position_controllers_[i]->
   getPosition());
  }
  position_command_buffer_.initRT(initial_command);
  new_command_ = true;
}

void BaxterPositionController::commandCB(const baxter_core_msgs::
   JointCommandConstPtr& msg)
{
  // the writeFromNonRT can be used in RT, if you have the guarantee
   that
  //  * no non-rt thread is calling the same function (we're not
   subscribing to ros callbacks)
  //  * there is only one single rt thread
  position_command_buffer_.writeFromNonRT(*msg);
  new_command_ = true;
}
```

Listing B.7 BaxterPositionController

Appendix C

Supplementary Code of Chapter 5

C.1 *find_object_3d* node

```
ros::init(argc, argv, "find_object_3d");
FindObjectROS::FindObjectROS(QObject * parent) :
  FindObject(true, parent),
  objFramePrefix_("object"),
  usePnP_(true)
{
  ros::NodeHandle pnh("~"); // public
  pnh.param("object_prefix", objFramePrefix_, objFramePrefix_);
  pnh.param("pnp", usePnP_, usePnP_);
  ROS_INFO("object_prefix = %s", objFramePrefix_.c_str());
  ROS_INFO("pnp = %s", usePnP_?"true":"false");
  ros::NodeHandle nh; // public
  pub_ = nh.advertise<std_msgs::Float32MultiArray>("objects", 1);
  pubStamped_ = nh.advertise<find_object_2d::ObjectsStamped>("
    objectsStamped", 1);
  pubInfo_ = nh.advertise<find_object_2d::DetectionInfo>("info", 1);
  this->connect(this, SIGNAL(objectsFound(const find_object::
    DetectionInfo &, const QString &, double, const cv::Mat &, float))
    , this, SLOT(publish(const find_object::DetectionInfo &, const
    QString &, double, const cv::Mat &, float)));
}
void FindObjectROS::publish(const find_object::DetectionInfo & info,
    const QString & frameId, double stamp, const cv::Mat & depth,
    float depthConstant)
{ // send tf before the message
  if(info.objDetected_.size() && !depth.empty() && depthConstant !=
    0.0f)
  { // Find center of the object
      QPointF center = iter->map(QPointF(objectWidth/2, objectHeight
    /2));
      cv::Vec3f center3D = this->getDepth(depth,
          center.x()+0.5f, center.y()+0.5f,
          float(depth.cols/2)-0.5f, float(depth.rows/2)-0.5f,
          1.0f/depthConstant, 1.0f/depthConstant);
      if(!usePnP_)
      {
          QPointF xAxis = iter->map(QPointF(3*objectWidth/4,
    objectHeight/2));
          axisEndX = this->getDepth(depth,
              xAxis.x()+0.5f, xAxis.y()+0.5f,
              float(depth.cols/2)-0.5f, float(depth.rows/2)-0.5f,
              1.0f/depthConstant, 1.0f/depthConstant);
          QPointF yAxis = iter->map(QPointF(objectWidth/2, 3*
    objectHeight/4));
          axisEndY = this->getDepth(depth,
              yAxis.x()+0.5f, yAxis.y()+0.5f,
              float(depth.cols/2)-0.5f, float(depth.rows/2)-0.5f,
              1.0f/depthConstant, 1.0f/depthConstant);
      }
      transform.setOrigin(tf::Vector3(
        tvec.at<double>(0)*(center3D.val[2]/tvec.at<double>(2)),
        tvec.at<double>(1)*(center3D.val[2]/tvec.at<double>(2)),
        tvec.at<double>(2)*(center3D.val[2]/tvec.at<double>(2))));
  if(pub_.getNumSubscribers() || pubStamped_.getNumSubscribers() ||
   pubInfo_.getNumSubscribers())
  {
      if(pub_.getNumSubscribers()){
          pub_.publish(msg);
      }
      if(pubStamped_.getNumSubscribers()){// use same header as the
   input image (for synchronization and frame reference)
          msgStamped.header.frame_id = frameId.toStdString();
          msgStamped.header.stamp.fromSec(stamp);
          pubStamped_.publish(msgStamped);
      }
      if(pubInfo_.getNumSubscribers()){// use same header as the
   input image (for synchronization and frame reference)
          infoMsg.header.frame_id = frameId.toStdString();
          infoMsg.header.stamp.fromSec(stamp);
          pubInfo_.publish(infoMsg);
      }
  }
}
```

Listing C.1 FindObjectROS

C.2 *kinect2_launch_joint_pub* node

```
def retrieve_hand_position(hand, listener, limb_name):
  try:
      (hand_p, trash) = listener.lookupTransform(base, hand, rospy.
   Time(0))
  except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
      print "tf exception"
      return {}
    return {limb_name:hand_p}

def calculate_distance_object_hand(x,y):
  squared_dist = np.sum((x-y)**2, axis=0)
  dist = np.sqrt(squared_dist)
  return dist

def track_one_arm(shoulder, shoulder_x, elbow, hand, torso, head, base, listener, limb_name):
    pub_list[limb_name+'_s0'].publish(radius_s0)
    pub_list[limb_name+'_s1'].publish(radius_s1)
    pub_list[limb_name+'_e0'].publish(radius_e0)
    pub_list[limb_name+'_e1'].publish(radius_e1)

def callback(objectCenter3D):
  distance = calculate_distance_object_hand(objectCenter3D.data, handPos)
  if distance < MinDistThreshold:
      disCount++
  if disCount > MaxDistCount:
      track_one_arm(shoulder, shoulder_x, elbow, hand, torso, head, base, listener, limb_name)

rospy.Subscriber("objects", std_msgs::Float32MultiArray, callback)
```

Listing C.2 calculate_distance_object_hand

C.3 *model_generalisation* node

```
rosinit('master_host');
modelNode = ros.Node('model_generalization');
for i = 1:nbSamples
    filename = strcat('RobotLearning',num2str(i),'Pos');
    Pos_t(i).data = load(filename);
    [mpos, npos] = size(Pos_t(i).data);
    Pos = Pos_t(i).data(:,:);
    demos{i}.pos = Pos';
    s(i).Data = [[1:nbData]*model.dt; spline(1:size(demos{i}.pos,2),
   demos{i}.pos, linspace(1,size(demos{i}.pos,2),nbData))]; %
   resampling
  Data = [Data s(i).Data];
end
model = init_GMM_kmeans(Data, model);
model = EM_GMM(Data, model);
[DataOut, SigmaOut] = GMR(model, [1:nbData]*model.dt, 1, 2:model.
   nbVar);
jointPosRaw = inverseKinematicsSolver(DataOut);

robot_s0_pub = ros.Publisher(modelNode,'right_s0','std_msgs/
   Float64multiarray') ;
robot_s1_pub = ros.Publisher(modelNode,'right_s1','std_msgs/
   Float64multiarray') ;
robot_e0_pub = ros.Publisher(modelNode,'right_e0','std_msgs/
   Float64multiarray') ;
robot_e1_pub = ros.Publisher(modelNode,'right_e1','std_msgs/
   Float64multiarray') ;
jointPos_s0 = rosmessage(robot_s0_pub);
jointPos_s1 = rosmessage(robot_s1_pub);
jointPos_e0 = rosmessage(robot_e0_pub);
```

```
jointPos_e1 = rosmessage(robot_e1_pub);
jointPos_s0.Data = jointPosRaw[:,0];
jointPos_s1.Data = jointPosRaw[:,1];
jointPos_e0.Data = jointPosRaw[:,2];
jointPos_e1.Data = jointPosRaw[:,3];
send(robot_s0_pub,jointPos_s0);
send(robot_s1_pub,jointPos_s1);
send(robot_e0_pub,jointPos_e0);
send(robot_e1_pub,jointPos_e1);
```

Listing C.3 modelNode

C.4 *robot_assembly* node

```
def pick(pose):
    gripper_open()
    _approach(pose)
    _servo_to_pose(pose)
    gripper_close()
    _retract()
def call_back_gen(limbForce):
  if limbForce.wrench.force.z > forceThreshold
      _control_arm.move_to_neutral()
  else
      right_limb.set_joint_positions(angles)
def callback_obj(objectCenter3D):
  distance = calculate_distance_object_hand(objectCenter3D.data,
   handPos)
  if distance < MinDistThreshold:
      disCount += 1
  if disCount > MaxDistCount:
       end_effector_pose = ik_request(objectCenter3D.data)
```

```
        _guarded_move_to_joint_position(end_effector_pose)
        pick(end_effector_pose)
def talker():
    rospy.init_node("robot_assembly", anonymous=True)
    rospy.Subscriber("right_s0", Float64, call_back_gen('right', 0))
    rospy.Subscriber("right_s1", Float64, call_back_gen('right', 1))
    rospy.Subscriber("right_e0", Float64, call_back_gen('right', 2))
    rospy.Subscriber("right_e1", Float64, call_back_gen('right', 3))
    rospy.Subscriber("objects", std_msgs::Float32MultiArray,
   callback_obj)
    angles = right_limb.joint_angles()
    forceSub = rospy.Subscriber("/robot/limb/right/endpoint_state",
   Float64, call_back_gen(limbForce))
```

Listing C.4 assembly

References

[1] Abbas, T. and MacDonald, B. A. (2011). Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In *2011 IEEE International Conference on Robotics and Automation*, pages 3816–3821.

[2] Abu-Dakka, F. J., Nemec, B., Kramberger, A., Buch, A. G., Krüger, N., and Ude, A. (2014). Solving peg-in-hole tasks by human demonstration and exception strategies. *Industrial Robot: An International Journal*, 41(6):575–584.

[3] Acosta-Calderon, C. A. and Hu, H. (2005). Robot imitation: Body schema and body percept. *Applied Bionics and Biomechanics*, 2(3-4):131–148.

[4] Albu-Schäffer, A., Ott, C., and Hirzinger, G. (2007). A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *The International Journal of Robotics Research*, 26(1):23–39.

[5] Aleotti, J., Caselli, S., and Reggiani, M. (2003). Toward programming of assembly tasks by demonstration in virtual environments. In *Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003. The 12th IEEE International Workshop on*, pages 309–314. IEEE.

[6] Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (2006). Evaluation of robot imitation attempts: comparison of the system's and the human's perspectives. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 134–141. ACM.

[7] Argall, B. D., Browning, B., and Veloso, M. (2008). Learning robot motion control with demonstration and advice-operators. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 399–404. IEEE.

[8] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

[9] Asfour, T., Azad, P., Gyarfas, F., and Dillmann, R. (2008). Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202.

[10] Bahubalendruni, M. R., Biswal, B. B., Kumar, M., and Nayak, R. (2015). Influence of assembly predicate consideration on optimal assembly sequence generation. *Assembly Automation*, 35(4):309–316.

[11] Bakker, P. and Kuniyoshi, Y. (1996). Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11.

[12] Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). *Robot Programming by Demonstration*, pages 1371–1394. Springer.

[13] Billard, A., Epars, Y., Calinon, S., Schaal, S., and Cheng, G. (2004). Discovering optimal imitation strategies. *Robotics and autonomous systems*, 47(2-3):69–77.

[14] Billard, A. and Matarić, M. J. (2001). Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2):145–160.

[15] Billard, A. G., Calinon, S., and Guenter, F. (2006). Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384.

[16] Bohren, J., Papazov, C., Burschka, D., Krieger, K., Parusel, S., Haddadin, S., Shepherdson, W. L., Hager, G. D., and Whitcomb, L. L. (2013). A pilot study in vision-based augmented telemanipulation for remote assembly over high-latency networks. In *2013 IEEE International Conference on Robotics and Automation*, pages 3631–3638.

[17] Breazeal, C. and Aryananda, L. (2002). Recognition of affective communicative intent in robot-directed speech. *Autonomous robots*, 12(1):83–104.

[18] Breazeal, C., Berlin, M., Brooks, A., Gray, J., and Thomaz, A. L. (2006). Using perspective taking to learn from ambiguous demonstrations. *Robotics and Autonomous Systems*, 54:385–393.

[19] Breazeal, C., Gray, J., and Berlin, M. (2009). An embodied cognition approach to mindreading skills for socially intelligent robots. *The International Journal of Robotics Research*, 28(5):656–680.

[20] Breazeal, C. and Scassellati, B. (2002). Robots that imitate humans. *Trends in cognitive sciences*, 6(11):481–487.

[21] Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29.

[22] Calinon, S. and Billard, A. (2007). Incremental learning of gestures by imitation in a humanoid robot. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 255–262. ACM.

[23] Calinon, S., D'halluin, F., Sauser, E., Caldwell, D., and Billard, A. (2010). A probabilistic approach based on dynamical systems to learn and reproduce gestures by imitation. *IEEE Robotics and Automation Magazine*, 17(2):44–54.

[24] Calinon, S., Evrard, P., Gribovskaya, E., Billard, A., and Kheddar, A. (2009). Learning collaborative manipulation tasks by demonstration using a haptic interface. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6. IEEE.

[25] Calinon, S., Guenter, F., and Billard, A. (2005). Goal-directed imitation in a humanoid robot. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 299–304. IEEE.

[26] Calinon, S., Guenter, F., and Billard, A. (2006). On learning the statistical representation of a task and generalizing it to various contexts. In *Robotics and Automation, ICRA 2006. Proceedings 2006 IEEE International Conference on*, volume 37, pages 2978–2983. IEEE.

[27] Chella, A., Dindo, H., and Infantino, I. (2006). A cognitive framework for imitation learning. *Robotics and Autonomous Systems*, 54(5):403–408.

[28] Chen, J. and Zelinsky, A. (2003). Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research*, 22(5):299–319.

[29] Chernova, S. and Veloso, M. (2007). Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 233. ACM.

[30] Choi, C., Taguchi, Y., Tuzel, O., Liu, M.-Y., and Ramalingam, S. (2012). Voting-based pose estimation for robotic assembly using a 3d sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1724–1731. IEEE.

[31] Chong, N. Y., Kotoku, T., Ohba, K., Komoriya, K., and Tanie, K. (2001). Exploring interactive simulator in collaborative multi-site teleoperation. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 243–248. IEEE.

[32] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1995). Active learning with statistical models. In *Advances in neural information processing systems*, pages 705–712.

[33] Dantam, N., Essa, I., and Stilman, M. (2012). Linguistic transfer of human assembly tasks to robots. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 237–242. IEEE.

[34] Dattalo, A. (2020). Ros introduction.

[35] Demiris, J. and Hayes, G. (1996). *Imitative learning mechanisms in robots and humans*. University of Edinburgh, Department of Artificial Intelligence.

[36] Demiris, Y. (2007). Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–158.

[37] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[38] Dixon, K. R. and Khosla, P. K. (2004). Learning by observation with mobile robots: A computational approach. In *Robotics and Automation. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 102–107. IEEE.

[39] Dogar, M., Spielberg, A., Baker, S., and Rus, D. (2015). Multi-robot grasp planning for sequential assembly operations. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 193–200. IEEE.

[40] Dong, S. and Williams, B. (2011). Motion learning in variable environments using probabilistic flow tubes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1976–1981. IEEE.

[41] Edmonds, M., Gao, F., Xie, X., Liu, H., Qi, S., Zhu, Y., Rothrock, B., and Zhu, S.-C. (2017). Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles. In *International Conference on Intelligent Robots and Systems (IROS), IEEE.*

[42] Ghahramani, Z. and Jordan, M. I. (1994). Supervised learning from incomplete data via an em approach. In *Advances in neural information processing systems*, pages 120–127.

[43] González-Fierro, M., Balaguer, C., Swann, N., and Nanayakkara, T. (2014). Full-body postural control of a humanoid robot with both imitation learning and skill innovation. *International Journal of Humanoid Robotics*, 11(02):1450012.

[44] Hersch, M., Guenter, F., Calinon, S., and Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics*, 24(6):1463–1467.

[45] Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10.

[46] Hovland, G. E., Sikka, P., and McCarragher, B. J. (1996). Skill acquisition from human demonstration using a hidden markov model. In *Robotics and Automation. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2706–2711. Ieee.

[47] Hu, S., Ko, J., Weyand, L., ElMaraghy, H., Lien, T., Koren, Y., Bley, H., Chryssolouris, G., Nasr, N., and Shpitalni, M. (2011). Assembly system design and operations for product variety. *CIRP Annals*, 60(2):715 – 733.

[48] Humans to Robots Laboratory, B. U. (2017). Ein.

[49] Hyon, S.-H., Hale, J. G., and Cheng, G. (2007). Full-body compliant human–humanoid interaction: Balancing in the presence of unknown external forces. *IEEE Transactions on Robotics*, 23(5):884–898.

[50] Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1398–1403. IEEE.

[51] Inamura, T., Kojo, N., Sonoda, T., Sakamoto, K., and Okada, K. (2005). Intent imitation using wearable motion capturing system with on-line teaching of task attention. In *5th IEEE-RAS International Conference on Humanoid Robots.*, pages 469–474.

[52] Inamura, T., Toshima, I., and Nakamura, Y. (2003). Acquiring motion elements for bidirectional computation of motion recognition and generation. *Experimental robotics viii*, pages 372–381.

[53] Jain, R. K., Majumder, S., and Dutta, A. (2013). Scara based peg-in-hole assembly using compliant ipmc micro gripper. *Robotics and Autonomous Systems*, 61(3):297–311.

[54] Jansen, B. and Belpaeme, T. (2006). A computational model of intention reading in imitation. *Robotics and Autonomous Systems*, 54(5):394–402.

[55] Jasim, I. F. and Plapper, P. W. (2014). Contact-state modeling of robotic assembly tasks using gaussian mixture models. *Procedia CIRP*, 23:229–234.

[56] Kim, Y.-L., Song, H.-C., and Song, J.-B. (2014). Hole detection algorithm for chamferless square peg-in-hole based on shape recognition using f/t sensor. *International journal of precision engineering and manufacturing*, 15(3):425–432.

[57] Knepper, R. A., Layton, T., Romanishin, J., and Rus, D. (2013). Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 855–862. IEEE.

[58] Knight, W. (2013). Baxter: The blue-collar robot. https://www.technologyreview.com/s/513746/baxter-the-blue-collar-robot/. Accessed: 05-03-2020.

[59] Kormushev, P., Calinon, S., and Caldwell, D. G. (2011). Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25(5):581–603.

[60] Koskinopoulou, M. (2019). *Learning from Demonstration to Accomplish Robotic Manipulation Tasks*. PhD thesis, University of Crete.

[61] Kramberger, A., Gams, A., Nemec, B., Chrysostomou, D., Madsen, O., and Ude, A. (2017). Generalization of orientation trajectories and force-torque profiles for robotic assembly. *Robotics and Autonomous Systems*, 98:333–346.

[62] Kramberger, A., Piltaver, R., Nemec, B., Gams, M., and Ude, A. (2016). Learning of assembly constraints by demonstration and active exploration. *Industrial Robot: An International Journal*, 43(5):524–534.

[63] Krüger, J., Lien, T. K., and Verl, A. (2009). Cooperation of human and machines in assembly lines. *CIRP Annals-Manufacturing Technology*, 58(2):628–646.

[64] Krüger, N., Ude, A., Petersen, H. G., Nemec, B., Ellekilde, L.-P., Savarimuthu, T. R., Rytz, J. A., Fischer, K., Buch, A. G., and Kraft, D. (2014). Technologies for the fast set-up of automated assembly processes. *KI-Künstliche Intelligenz*, 28(4):305–313.

[65] Kuklinski, K., Fischer, K., Marhenke, I., Kirstein, F., aus der Wieschen, M. V., Solvason, D., Kruger, N., and Savarimuthu, T. R. (2014). Teleoperation for learning by demonstration: Data glove versus object manipulation for intuitive robot control. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on*, pages 346–351. IEEE.

[66] Kulic, D. and Nakamura, Y. (2008). Scaffolding on-line segmentation of full body human motion patterns. In *Intelligent Robots and Systems, IROS 2008. IEEE/RSJ International Conference on*, pages 2860–2866. IEEE.

[67] Kulić, D., Ott, C., Lee, D., Ishikawa, J., and Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345.

[68] Kulić, D., Takano, W., and Nakamura, Y. (2008). Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *The International Journal of Robotics Research*, 27(7):761–784.

[69] Kuniyoshi, Y., Yorozu, Y., Inaba, M., and Inoue, H. (2003). From visuo-motor self learning to early imitation-a neural architecture for humanoid learning. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3132–3139. IEEE.

[70] Lambrecht, J., Kleinsorge, M., Rosenstrauch, M., and Krüger, J. (2013). Spatial programming for industrial robots through task demonstration. *International Journal of Advanced Robotic Systems*, 10(5):254.

[71] Laursen, J. S., Ellekilde, L.-P., and Schultz, U. P. (2018). Modelling reversible execution of robotic assembly. *Robotica*, pages 1–30.

[72] Laursen, J. S., Schultz, U. P., and Ellekilde, L.-P. (2015). Automatic error recovery in robot assembly operations using reverse execution. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1785–1792. IEEE.

[73] Lee, C. and Yangsheng, X. (1996). Online, interactive learning of gestures for human/robot interfaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 2982–2987 vol.4.

[74] Lee, D. and Nakamura, Y. (2006). Stochastic model of imitating a new observed motion based on the acquired motion primitives. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4994–5000.

[75] Lee, D. and Nakamura, Y. (2007). Mimesis scheme using a monocular vision system on a humanoid robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2162–2168.

[76] Lee, K., Su, Y., Kim, T.-K., and Demiris, Y. (2013). A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems*, 61(12):1323–1334.

[77] Lee, M. A., Zhu, Y., Srinivasan, K., Shah, P., Savarese, S., Fei-Fei, L., Garg, A., and Bohg, J. (2019). Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*.

[78] Li, W. and Fritz, M. (2015). Teaching robots the use of human tools from demonstration with non-dexterous end-effectors. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 547–553. IEEE.

[79] Liang, Y.-S., Pellier, D., Fiorino, H., and Pesty, S. (2017). Evaluation of a robot programming framework for non-experts using symbolic planning representations. In *26th IEEE International Symposium on Robot and Human Interactive Communication*.

[80] Likar, N., Nemec, B., Žlajpah, L., Ando, S., and Ude, A. (2015). Adaptation of bimanual assembly tasks using iterative learning framework. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 771–776. IEEE.

[81] Liu, Q., Li, R., Hu, H., and Gu, D. (2016). Extracting semantic information from visual data: A survey. *Robotics*, 5(1):8.

[82] Lopes, M. and Santos-Victor, J. (2007). A developmental roadmap for learning by imitation in robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):308–321.

[83] Makiishi, T. and Noborio, H. (1999). Sensor-based path-planning of multiple mobile robots to overcome large transmission delays in teleoperation. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, volume 4, pages 656–661. IEEE.

[84] Manschitz, S., Gienger, M., Kober, J., and Peters, J. (2020). Learning sequential force interaction skills. *Robotics*, 9(2):45.

[85] Mataric, M. J. (2000). Getting humanoids to move and imitate. *IEEE Intelligent Systems and their Applications*, 15(4):18–24.

[86] Mistry, M., Mohajerian, P., and Schaal, S. (2005). An exoskeleton robot for human arm movement study. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 4071–4076. IEEE.

[87] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126.

[88] Mollard, Y., Munzer, T., Baisero, A., Toussaint, M., and Lopes, M. (2015). Robot programming from demonstration, feedback and transfer. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1825–1831. IEEE.

[89] Muhlig, M., Gienger, M., Hellbach, S., Steil, J. J., and Goerick, C. (2009). Task-level imitation learning using variance-based movement optimization. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1177–1184. IEEE.

[90] Navarro-Gonzalez, J., Lopez-Juarez, I., Rios-Cabrera, R., and Ordaz-Hernández, K. (2015). On-line knowledge acquisition and enhancement in robotic assembly tasks. *Robotics and Computer-Integrated Manufacturing*, 33:78–89.

[91] Nemec, B., Abu-Dakka, F. J., Ridge, B., Ude, A., Jorgensen, J. A., Savarimuthu, T. R., Jouffroy, J., Petersen, H. G., and Kruger, N. (2013). Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–7. IEEE.

[92] Nicolescu, M. and Mataric, M. J. (2005). Task learning through imitation and human-robot interaction. *Models and mechanisms of imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions*, pages 407–424.

[93] Niekum, S., Chitta, S., Barto, A., Marthi, B., and Osentoski, S. (2013). Incremental semantically grounded learning from demonstration. In *Robotics: Science and Systems*, volume 9.

[94] Nottensteiner, K., Sagardia, M., Stemmer, A., and Borst, C. (2016). Narrow passage sampling in the observation of robotic assembly tasks. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 130–137. IEEE.

[95] Pardowitz, M., Knoop, S., Dillmann, R., and Zollner, R. D. (2007). Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):322–332.

[96] Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. (2009). Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE.

[97] Peternel, L., Petric, T., and Babic, J. (2015). Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1497–1502. IEEE.

[98] Peternel, L., Petrič, T., and Babič, J. (2018). Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation. *Autonomous Robots*, 42(1):1–17.

[99] Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

[100] Peters, R. A., Campbell, C. L., Bluethmann, W. J., and Huber, E. (2003). Robonaut task learning through teleoperation. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2806–2811. IEEE.

[101] Pollard, N. S., Hodgins, J. K., Riley, M. J., and Atkeson, C. G. (2002). Adapting human motion for the control of a humanoid robot. In *Robotics and Automation. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1390–1397. IEEE.

[102] Pook, P. K. and Ballard, D. H. (1993). Recognizing teleoperated manipulations. In *Proceedings IEEE International Conference on Robotics and Automation*, volume Vol.2, pages 578–585.

[103] Riley, M., Ude, A., and Atkeson, C. G. (2000). Methods for motion generation and interaction with a humanoid robot: Case studies of dancing and catching. In *2000 Workshop Interactive Robot. Entertainment*, pages 35–42.

[104] Rozo, L., Calinon, S., Caldwell, D., Jiménez, P., and Torras, C. (2013a). Learning collaborative impedance-based robot behaviors. *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 1(1):1.

[105] Rozo, L., Jiménez, P., and Torras, C. (2013b). A robot learning from demonstration framework to perform force-based manipulation tasks. *Intelligent Service Robotics*, 6:33–51.

[106] Ruchanurucks, M., Nakaoka, S., Kudoh, S., and Ikeuchi, K. (2006). Humanoid robot motion generation with sequential physical constraints. In *Robotics and Automation. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2649–2654. IEEE.

[107] Rybski, P. E. and Voyles, R. M. (1999). Interactive task training of a mobile robot through human gesture recognition. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 1, pages 664–669 vol.1.

[108] Sarić, A., Xiao, J., and Shi, J. (2014). Reducing uncertainty in robotic surface assembly tasks based on contact information. In *Advanced Robotics and its Social Impacts (ARSO), 2014 IEEE Workshop on*, pages 94–100. IEEE.

[109] Savarimuthu, T. R., Buch, A. G., Schlette, C., Wantia, N., Rossmann, J., Martinez, D., Alenya, G., Torras, C., Ude, A., and Nemec, B. (2017). Teaching a robot the semantics of assembly tasks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–23.

[110] Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 358(1431):537–547.

[111] Schmitt, R. and Cai, Y. (2014). Recognition of dynamic environments for robotic assembly on moving workpieces. *The International Journal of Advanced Manufacturing Technology*, 71(5-8):1359–1369.

[112] Schwarz, L. A., Mkhitaryan, A., Mateus, D., and Navab, N. (2012). Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, 30(3):217–226.

[113] Skoglund, A., Iliev, B., and Palm, R. (2010). Programming-by-demonstration of reaching motions—a next-state-planner approach. *Robotics and Autonomous Systems*, 58(5):607–621.

[114] Stenmark, M. and Topp, E. A. (2016). From demonstrations to skills for high-level programming of industrial robots. In *AAAI Fall Symposium Series*.

[115] Stolt, A., Linderoth, M., Robertsson, A., and Johansson, R. (2012). Force controlled robotic assembly without a force sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1538–1543. IEEE.

[116] Suárez-Ruiz, F., Zhou, X., and Pham, Q.-C. (2018). Can robots assemble an ikea chair? *Science Robotics*, 3(17).

[117] Suomalainen, M. and Kyrki, V. (2016). Learning compliant assembly motions from demonstration. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 871–876. IEEE.

[118] Suomalainen, M. and Kyrki, V. (2017). A geometric approach for learning compliant motions from demonstration. *arXiv preprint arXiv:1709.02589*.

[119] Suárez-Ruiz, F. and Pham, Q.-C. (2016). A framework for fine robotic assembly. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 421–426. IEEE.

[120] Sweeney, J. D. and Grupen, R. (2007). A model of shared grasp affordances from demonstration. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 27–35. IEEE.

[121] Takamatsu, J., Ogawara, K., Kimura, H., and Ikeuchi, K. (2007). Recognizing assembly tasks through human demonstration. *The International Journal of Robotics Research*, 26(7):641–659.

[122] Tang, T., Lin, H.-C., Zhao, Y., Fan, Y., Chen, W., and Tomizuka, M. (2016). Teach industrial robots peg-hole-insertion by human demonstration. In *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*, pages 488–494. IEEE.

[123] Tanwani, A. K. and Calinon, S. (2016). Learning robot manipulation tasks with task-parameterized semitied hidden semi-markov model. *IEEE Robotics and Automation Letters*, 1(1):235–242.

[124] Thorndike, E. L. and Jelliffe (1911). *Animal Intelligence. Experimental Studies*, volume 39. Macmillan.

[125] Trafton, J. G., Cassimatis, N. L., Bugajska, M. D., Brock, D. P., Mintz, F. E., and Schultz, A. C. (2005). Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(4):460–470.

[126] Tso, S. K. and Liu, K. P. (1996). Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In *Industrial Technology. (ICIT '96), Proceedings of The IEEE International Conference on*, pages 294–298.

[127] Tzafestas, C. S. (2001). Teleplanning by human demonstration for vr-based teleoperation of a mobile robotic assistant. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 462–467. IEEE.

[128] Ude, A., Atkeson, C. G., and Riley, M. (2004). Programming full-body movements for humanoid robots by observation. *Robotics and autonomous systems*, 47(2):93–108.

[129] Ude, A., Gams, A., Asfour, T., and Morimoto, J. (2010). Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815.

[130] Voyles, R. M. and Khosla, P. K. (2001). A multi-agent system for programming robots by human demonstration. *Integrated Computer-Aided Engineering*, 8(1):59–67.

[131] Wahrburg, A., Zeiss, S., Matthias, B., and Ding, H. (2014). Contact force estimation for robotic assembly using motor torques. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1252–1257. IEEE.

[132] Wan, W. and Harada, K. (2016). Integrated assembly and motion planning using regrasp graphs. *Robotics and biomimetics*, 3(1):18.

[133] Yang, J., Xu, Y., and Chen, C. S. (1994). Hidden markov model approach to skill learning and its application to telerobotics. *IEEE Transactions on Robotics and Automation*, 10(5):621–631.

[134] Yang, J., Xu, Y., and Chen, C. S. (1997). Human action learning via hidden markov model. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(1):34–44.

[135] Yang, Y., Lin, L., Song, Y., Nemec, B., Ude, A., Buch, A. G., Krüger, N., and Savarimuthu, T. R. (2014). Fast programming of peg-in-hole actions by human demonstration. In *Mechatronics and Control (ICMC), 2014 International Conference on*, pages 990–995. IEEE.

[136] Ye, G. and Alterovitz, R. (2017). Demonstration-guided motion planning. *Springer Tracts in Advanced Robotics*, pages 291–307.

[137] Zeng, A., Yu, K.-T., Song, S., Suo, D., Walker, E., Rodriguez, A., and Xiao, J. (2017). Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1386–1383. IEEE.

[138] Zhu, Z. and Hu, H. (2018). Robot learning from demonstration in robotic assembly: A survey. *Robotics*, 7(2):17.

[139] Zhu, Z., Hu, H., and Gu, D. (2018). Robot performing peg-in-hole operations by learning from human demonstration. In *2018 10th Computer Science and Electronic Engineering (CEEC)*, pages 30–35. IEEE.