

博士論文

**Robotic Manipulation Teaching System Based  
on Object-pair Mutual Function Knowledge  
with a Few Visual Demonstrations**

( 少数の視覚演示と対物体相互機能知識に基づ  
くロボットマニピュレーション教示システム )

令和2年12月04日提出  
指導教員 岡田 慧 教授

東京大学大学院 情報理工学系研究科  
知能機械情報学専攻  
48177511 李 梓佳



---

# Abstract

Robot manipulation teaching system is a crucial intermedium when general users want to teach robots new manipulation knowledge without the help of experts. However, given robots a limited amount of demonstrations and require them to generalize the learned knowledge to new instances is a difficult problem. Towards reducing user effort in robot teaching, this thesis contributes a robot manipulation teaching system that enables robots to extract meaningful knowledge from a few visual demonstrations and leverage the learned knowledge in handling complex variations.

The thesis first defines an Object-pair Mutual Function Knowledge model that enables robots to generalize the elements of the knowledge model to new instances. The main difference between the proposed knowledge model and the previous knowledge models is that the proposed knowledge model adopts the object function part to determine whether a new instance can be used to achieve a known function, while the previous knowledge models have difficulty in handling new instances because they require prior knowledge for the instances.

Next, the thesis presents an on-site teaching approach that can extract knowledge from human teaching. The whole process has three main steps including handheld object extraction, object function part extraction, and object motion trajectory extraction. Each step is transparent to the demonstrator because the demonstrator can see the extraction outcomes. Unlike previous methods that only record a series of robot end-effector positions or human hand positions for trajectory learning. The proposed method applied a hand keypoint detector to figure out the interaction between human hand and object, and thus enables the robot to extract more visual variables for learning. The virtual teaching tool is supplementary to the on-site teaching approach because in some situations using only hand gestures for teaching is difficult. The virtual teaching tool also allows users to teach robots through 3D CAD models.

This thesis also proposed a data augmentation framework and an Object-pair Mutual Function Knowledge Acquisition Network. The data augmentation framework is able to

enrich the demonstration samples with different position changes. Since the demonstration samples given by users are limited, the learning algorithm is difficult to notice how position changes affect the output results. By given more augmented demonstrations with different relative object positions, the network has learned to choose proper pose trajectories to handle different situations. The relationship between the number of augmented samples and network training time is also estimated.

In the task reproduction phases, this thesis proposed a framework that has three main components: the semantic instance segmentation component receives the user command as inputs and detect target objects in the environment through the RGB images. The point cloud processor receives the image masks given by the semantic instance segmentation component and converts the pixel within the masks to point clouds, according to the user input function, this component attaches the activation signals to the corresponding objects. The trajectory and grasping pose generation component takes the point cloud as inputs and outputs motion trajectory and grasping pose for the desired function to guide robots to perform tasks. Comparing different trajectory generation methods, this thesis shows that the object function knowledge-based method outperforms the instance-based method in handling new instances for known manipulation tasks.

Finally, this thesis shows how to integrate the robot teaching system to a real robot platform. Through the experiment, this thesis shows how the proposed system enables the robot to gain new knowledge to carry out daily tasks. The experiment also shows the robot has successfully handled the position variation, the intra-category shape variation, and the initial state variation after learning from a few human teaching.

By enabling robots to generalize the learned manipulation knowledge from a few human demonstrations, this thesis contributes towards an effective robot manipulation teaching system that not only allows transparent teaching but also is able to handle different kinds of variations through a limited amount of demonstrations.



## 目次



<b>第 1 章</b>	<b>Introduction</b>	<b>11</b>
1.1	Research Goals . . . . .	15
1.2	Challenges . . . . .	19
1.3	Overview of the Robot Teaching Process . . . . .	20
1.4	Major Contributions . . . . .	21
1.5	Thesis Outline . . . . .	22
<b>第 2 章</b>	<b>Motivation and Related Work</b>	<b>25</b>
2.1	Object Manipulation Knowledge . . . . .	27
2.1.1	Object Shape Knowledge . . . . .	27
2.1.2	Object Function Knowledge . . . . .	28
2.1.3	Trajectory Knowledge . . . . .	28
2.1.4	Policy . . . . .	29
2.2	Demonstratoin Approaches . . . . .	29
2.2.1	Kinesthetic Teaching . . . . .	30
2.2.2	Teleoperation . . . . .	30
2.2.3	Robot Learning-by-Watching . . . . .	31
2.3	Manipulation Learning Ability and Generalizability in Robot Learning from Human Teaching . . . . .	32
2.3.1	Evaluating Manipulation Learning Ability in Robot Manipulation Teaching . . . . .	33
2.3.2	Evaluating Generalizability in Robot Manipulation Teaching . . .	36
2.3.3	Function Part Detection Based Trajectory Imitation . . . . .	38
2.4	Summary . . . . .	39
<b>第 3 章</b>	<b>Object-pair Mutual Function Knowledge Description and Representa- tion</b>	<b>41</b>
3.1	Manipulation Knowledge Model . . . . .	43
3.2	Object-pair Mutual Function Knowledge Model . . . . .	44
3.3	Representing the Object-pair Mutual Function Knowledge Model . . . .	45
3.3.1	Representation of the Object-pair ( $O_{ac}$ , $O_{as}$ ) . . . . .	45

3.3.2	Representation of the Function F . . . . .	45
3.3.3	Representation of the Function Part P . . . . .	46
3.3.4	Representation of the Trajectory T . . . . .	47
3.4	Discussion . . . . .	49
3.4.1	The Faulty Generalization Problem and Its Solution . . . . .	50
3.5	Summary . . . . .	50
<b>第 4 章</b>	<b>Acquiring Manipulation Knowledge through Transparent Teaching</b>	<b>53</b>
4.1	Transparent Teaching . . . . .	56
4.2	An On-site Teaching Approach . . . . .	56
4.2.1	Overview of the Teaching Process . . . . .	58
4.2.2	Human Hand Keypoint Estimation . . . . .	58
4.2.3	Handheld Object Extraction . . . . .	59
4.2.4	Extracting Object Function Part through Human-Object Interaction	61
4.2.5	Trajectory Demonstration and Key Object State Extraction . . . .	61
4.3	Experiments . . . . .	63
4.3.1	Acquiring Manipulation Knowledge from On-site Teaching . . .	63
4.4	Virtual Teaching . . . . .	68
4.4.1	Synthetic Object Point Cloud Generation . . . . .	68
4.4.2	Graphical User Interface Annotation Tool for Virtual Teaching . .	70
4.5	Discussion . . . . .	73
4.6	Summary . . . . .	74
<b>第 5 章</b>	<b>Learning Manipulation Knowledge through a Few Visual Demonstra-</b>	
	<b>tions</b>	<b>75</b>
5.1	Manipulation Learning through Imitation . . . . .	77
5.2	Demonstration Data Processing and Separation Augmentation . . . . .	78
5.2.1	6D Pose Trajectory Separation and Assignment . . . . .	79
5.2.2	Object-pair Recombination . . . . .	80
5.3	Manipulation Knowledge Learning on Augmented Visual Demonstration	
	Samples . . . . .	80

5.4	Object-pair Mutual Function Knowledge Acquisition Network . . . . .	80
5.4.1	Function Detection Branch . . . . .	82
5.4.2	Offset Pose Prediction Branch . . . . .	83
5.4.3	Fusion . . . . .	84
5.4.4	Grasping Pose Detection . . . . .	85
5.4.5	Network Training . . . . .	86
5.5	Experiments . . . . .	86
5.5.1	Demonstration Data . . . . .	88
5.5.2	Test Objects . . . . .	88
5.5.3	Evaluating the Number of Augmented Demonstrations Needed for Handling Position Variation . . . . .	88
5.5.4	Test the Trained Network on New Instances with Different positions	91
5.5.5	Network Training Time Evaluation . . . . .	91
5.6	Summary . . . . .	93
<b>第 6 章</b>	<b>A Framework for Manipulation Task Reproduction</b>	<b>95</b>
6.1	Instance Matching Based Task Reproduction . . . . .	98
6.2	Object Function Knowledge-Based Task Reproduction . . . . .	99
6.2.1	Overview of the Framework . . . . .	99
6.2.2	Semantic Instance Segmentation . . . . .	99
6.2.3	Point Cloud Processing Pipeline . . . . .	101
6.2.4	Trajectory and Grasping Pose Generation . . . . .	103
6.3	Experiments . . . . .	104
6.4	Comparison of the Instance Matching Based Method and the Function Part Detection Based Method in Handling New Instances . . . . .	106
6.5	Discussion . . . . .	109
6.6	Summary . . . . .	109
<b>第 7 章</b>	<b>Home-Assistant Robot Applications Using the Manipulation Teaching System with a Few Visual Demonstrations</b>	<b>111</b>
7.1	Robot Platform . . . . .	113

7.1.1	Hardware . . . . .	114
7.1.2	Middleware . . . . .	116
7.2	Integration of the Manipulation Teaching System on the Robot System . .	118
7.3	Teaching a Robot New Manipulation Knowledge in Daily Life . . . . .	120
7.3.1	Task Description . . . . .	120
7.3.2	Objects . . . . .	122
7.3.3	Demonstrations . . . . .	122
7.3.4	Learning Manipulation Knowledge from Virtual Teaching . . . .	123
7.3.5	Learning to Detect Object in Unexpected State with Negative Sample . . . . .	124
7.3.6	Learning Manipulation Knowledge from a Few On-site Demon- strations . . . . .	126
7.4	Teaching a Robot to Tidy a Room . . . . .	130
7.4.1	Objects . . . . .	130
7.4.2	Demonstration Samples . . . . .	132
7.4.3	Learning to Tidy a Room with a Few Demonstrations . . . . .	132
7.5	Discussion . . . . .	132
7.6	Summary . . . . .	136
<b>第 8 章</b>	<b>Conclusion</b>	<b>137</b>
8.1	Conclusion . . . . .	139
8.2	Future work . . . . .	144
	<b>参考文献</b>	<b>147</b>
	<b>Appendix A</b>	<b>173</b>
	<b>Appendix B</b>	<b>177</b>

# 第1章

## Introduction





Robotic research in home-assistant has increasingly attracted public attention during recent years due to the increasing proportion of seniors in many countries. On the one hand, people have great expectations of entrusting household chores to robots. On the other hand, currently, most commercial robot platforms are lacking intelligent sensing abilities to carry out complex tasks in the human living environment.

Towards solving this problem, many home-assistant and human support robot platforms like HRP [1], AERO [2], ASIMO [3], and HSR [4] have been designed to assist people in daily housework. These robot platforms are equipped with visual, audio, and force sensors that enable robots to collect information from the surrounding, while the integrated software systems allow robots to make appropriate decisions according to prior knowledge and dynamically changing.

Relevant studies on home-assistant robots [5, 6] have shown the blueprints of how home-assistant robots support human lives by carrying out household chores such as tidying and cleaning rooms [7, 8], preparing meals [9, 10, 11, 12], laundry folding [13], and ironing [14].

Even though previous studies achieved significant improvements in handling complex domestic tasks, the development of new manipulation tasks via traditional programming methods requires expertise in coding, which is a significant time investment. In addition, human environments have many uncertainties, it is impossible for task developers to foresee all the situations. Therefore, a robot teaching system is essential because it allows users to customize their robot applications without the help of experts. Recent studies have shown that robots can acquire many skills from human demonstrations, including simple actions [15, 16, 17, 18, 19, 20, 21, 22], rigid object manipulation [23, 24], pizza dough rolling [25], and compliant food manipulation [26]. These studies show great potential in enabling robots to learn to handle housework through human teaching.

Generally, a robot teaching system should allow demonstrators with less experience and knowledge in robotics to teach robots manipulation skills. Therefore, a robot teaching system has to solve some basic problems: (1) what knowledge should be used for teaching and learning; (2) how to extract this knowledge from human demonstrations; (3) how the robot learns the knowledge; and (4) how the robot reproduces the manipulation tasks.

In the early stage of robot teaching, the human teacher maneuvers the robot's arm to move it through desired motions or set a series of waypoints through a remote device [27], at the same time, the system will record some key parameters to replay the trajectories. This demonstration method is called kinesthetic teaching. Similarly, teleoperation allows demonstrators to use external devices such as a joystick or a virtual reality (VR) headset to control the robot to perform tasks. The most natural demonstration approach is robot learning-by-watching. In this approach, the users perform tasks using their own body and robots obtain information from their visual sensors.

The robot learning-by-watching problem is complicated [28, 29, 30]. Unlike the other two approaches that human demonstrators can give the robot desired poses by directly driving robots. Learning from observation requires a robot to extract meaningful information from visual sensors and convert the information to manipulation knowledge, then the robot must learn the knowledge and apply it to solve problems. The basic paradigm for robot learning-by-watching is requiring a robot to imitate human behavior to complete a task, this paradigm is called imitation learning.

When reproducing the learned manipulation tasks, previous methods [31] require the target objects' CAD models for 6D pose estimation, but it is difficult for general users to create CAD models for every object in the environment. On the other hand, deep learning technology has a powerful generalization ability that can solve the problem of generalizing the learned ability to new instances. In 2012, AlexNet [32] won the ImageNet [33] competition with a top-5 error rate of 15.3%, compared to the second place top-5 error rate of 26.2%. In 2015, the ResNet [34] reached a top-5 error of 3.6%, which is believed to surpass human-level performance. In 2016, DeepMind's AlphaGo [35, 36] defeated the legendary Go player Lee Sedol using the deep reinforcement learning technique. In 2019, OpenAI Five [37] beat the world champions in an esports game, DOTA2. These landmark events showed that the growth of computing power allows machines to implement complex algorithms and outperform humans in specific tasks. However, these improvements are built upon a huge amount of data, which requires a lot of time and human effort for collecting the demonstration data.

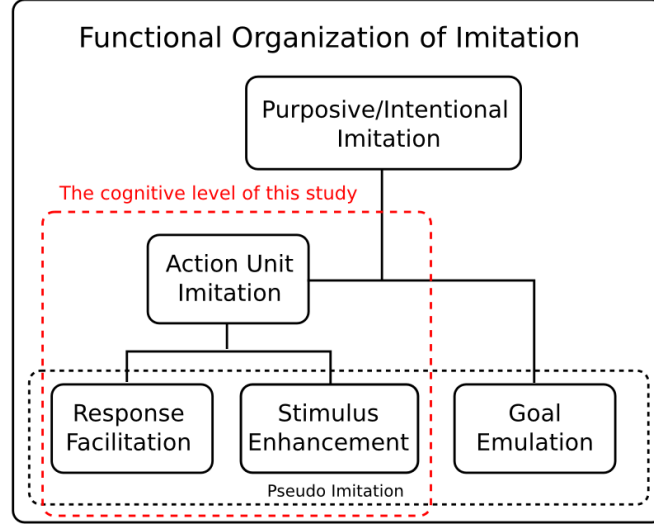


Figure 1.1: Functional organization of imitation presented in [38, 39]

## 1.1 Research Goals

This thesis targets the problem of reducing user effort in robot teaching by enabling a robot to learn new manipulation knowledge with a few demonstrations and apply the knowledge to new instances. More than simply imitating human behavior of using specified instances that appeared in the demonstrations, the goal of the thesis is to enable robots to leverage the learned knowledge in manipulating new instances while adapting to variations.

To clarify the cognitive range, we introduce the functional organization of imitation presented in [38, 39] (Figure 1.1). This model has three pseudo-imitation functions: the response facilitation function, the stimulus enhancement function, and the goal emulation function. Where response facilitation means an imitator receives a stimulus of a behavior pattern by observing the demonstration behavior, stimulus enhancement means the existence of a particular object that evokes the imitator a behavior pattern, the goal emulation means achieving the same goal as demonstration without understanding how the demonstrator achieved the goal. Action unit imitation combines response facilitation and stimulus enhancement, it mainly concerns about reusing the action on a kind of objects.

Finally, purposive imitation combines the action unit imitation and goal emulation, it requires an agent to understand the causal relations between the behavior and the goal, so it is also called true imitation.

This thesis mainly focuses on implementing the action unit imitation through human teaching and robot learning (Figure 1.2). On the teaching side, the user teaches the robot knowledge about the object and shows the robot how to use the object. On the learning side, the robot reproduces the learned knowledge to solve problems.

Determining what variables to be used in a knowledge model for teaching and learning is not a simple task. If a knowledge model is too simple that only contains point trajectories [40, 41, 42, 43, 44], the robot may have difficulty in handling variations such as position and instance changes. On the other hand, a complicated knowledge model poses challenges in learning and teaching. To solve this problem, this thesis proposes a knowledge model that contains information of the objects, objects' function parts and trajectories for reproducing the skills, the variable in the knowledge model is easy for human teaching and robot learning. Furthermore, the function part property allows the robot to generalize the learned trajectories to new instances.

During the teaching process, a system that shows demonstrators what has been recorded by the robot is called “a transparent teaching system”. The traditional industrial robot teaching system is a typical transparent teaching system because users can see the recorded waypoints through the user interface, while the teaching systems presented in [45, 46, 12] are not transparent because the teaching outputs are symbolic instructions and users can not know about the concrete robot execution. In single-step manipulation tasks, a transparent teaching system is more preferred because users can know about how the robot is going to perform a task. The difficulty of building a transparent teaching system is how to extract essential information from visual demonstrations, existing transparent teaching systems [17, 18] only extract trajectories for manipulating specific objects, which is insufficient for generalization on new instances. This thesis presents a teaching framework that is able to extract the target object and its function part, as well as the object's key states for achieving a task, given these variables, a robot can easily apply the learned skills on new instances.

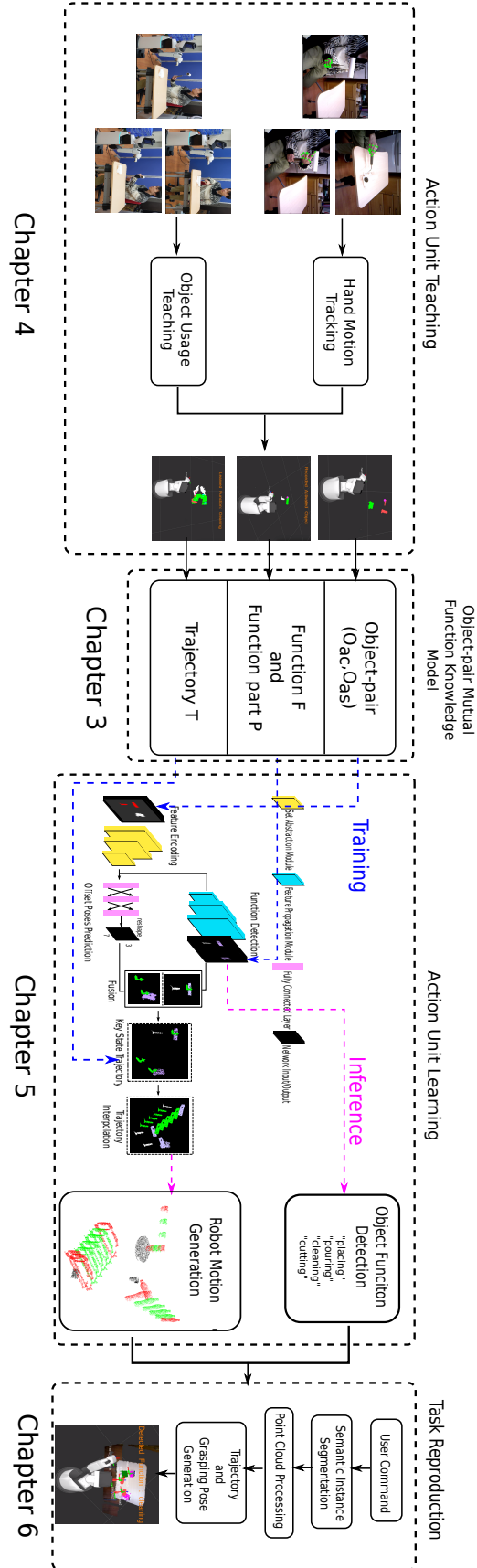


Figure 1.2: An overview of the proposed system and thesis structure.

The ability to learn new manipulation knowledge with a few demonstrations is also desired. The superior performance of deep learning technology enables robots to learn different kinds of variations, but this improvement relies upon a large amount of data. For some computer vision tasks such as object detection and segmentation from 2D images [47, 48, 49, 50, 34], it is not difficult to obtain a huge amount of data from the internet or public datasets [33, 51]. However, when teaching a robot on-site, human demonstrators usually need to perform a task in front of the robot, some existing deep learning-based approaches [52, 53] may require demonstrators to perform tasks many times to collect a sufficient amount of data for training the deep neural networks. It increased the user burden in robot teaching. To address this problem, this thesis presents an efficient knowledge acquisition framework to maximize the use of existing demonstration samples for learning. Having the demonstration data, a learning algorithm should be able to learn the differences in the samples, for example, given different input objects, it has to determine what the objects can be used for, it also needs to generate a proper trajectory for guiding a robot to carry out the task. This thesis presents an Object-pair Mutual Function Acquisition Network that is able to automatically select optimal manipulation strategies to handle different manipulation tasks.

The last stage of a teaching system is how robots reproduce the manipulation tasks. In the previous manipulation process, robots require cad models of the manipulation objects for 6D pose estimation [54, 55, 56, 57, 58], based on the objects' poses, the robot can search for optimal strategies for manipulation. However, this instance-based method limited the use range of knowledge. To solve this problem, this thesis presents an object function knowledge-base method and the corresponding framework for manipulation task reproduction.

To summarize, this thesis describes a system that enables robots to learn new manipulation skills from a few visual demonstrations. To reduce the user effort in robot teaching, human demonstrators only need to provide a few demonstration samples to teach a robot new skills. The demonstration process is transparent, therefore the demonstrators can see the demonstration outcomes. Utilizing the object-pair Mutual Function Knowledge model and the learning algorithm, the system enables robots to leverage the learned manipulation

knowledge to manipulate new instances while handling different kinds of variations.

## 1.2 Challenges

The difficulties in building a robust and user-friendly robot teaching system are described as follows:

- Firstly, the system has no prior manipulation knowledge of a new task. It is difficult because a user must be able to teach the system a new task from zero. Most of the existing teaching systems usually require some preconditions, such as the manipulated object is already known or the manipulated object's position is fixed.
- Secondly, the target object may have different shapes or sizes from the demonstration objects. In other words, a robot should not only know how to manipulate the objects that appeared in the demonstrations but also be able to apply the learned knowledge to other objects that have the same functions. choosing what to learn is a difficult problem because too many variables pose a challenge to the teaching and learning process, while too few variables limit the knowledge's use range.
- Thirdly, transparent teaching requires a system to extract essential variables from visual demonstration and showing explainable feedbacks to the demonstrator. Unlike previous transparent teaching systems that only extract the demonstrator's hand trajectory in 3D space, to achieve generalization, the system requires information of the object function part and object's 6D trajectory when performing a task, extracting these variables is a difficult task.
- Finally, the system must learn to handle variations through limited demonstrations provided by users. Existing manipulation learning algorithms usually require a lot of data for training but the demonstration samples are not easy to get. Instead of asking the demonstrator to perform one task many times, how to maximize the use of the demonstration sample is a challenging problem.

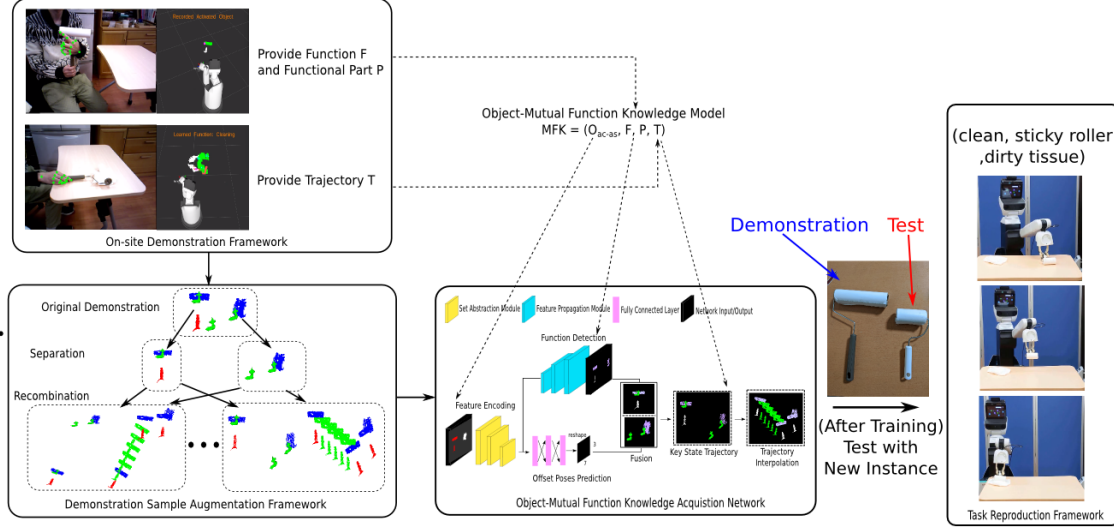


Figure 1.3: An overview of the Robot Manipulation Teaching Process.

### 1.3 Overview of the Robot Teaching Process

An overview of the robot teaching and learning process is shown in Figure 1.3. The process has four main components: a knowledge model, a robot teaching approach, a data augmentation framework, and a deep neural network for manipulation knowledge learning and task reproduction.

The knowledge model presented in this thesis is called the Object-pair Mutual Function Knowledge model. This model contains four elements to describe an object and its function, which part of the object has this function, and how to perform this function. Compared to the existing object-based models, this knowledge model enables robots to generalize the learned knowledge to new instances.

The on-site demonstration framework allows general users to teach robots new manipulation knowledge in a transparent way. The teaching process has several stages to obtain the essential information, and thus the demonstrator teaches the robot in an interactive way. In the first stage, the demonstrator shows the target objects and the corresponding function parts to the robot, as the same time, the robot extracts the variables and returns the point clouds of the target objects and function labels. In the second stage, the demon-



strator shows the robot how to use the object by presenting a motion trajectory, the robot extracts the manipulated object's key states by estimating its 6D poses.

The demonstration samples are sent to the augmentation framework to generate more samples with different relative object positions for learning. To maximize the use of these demonstration samples, the augmentation framework first separates the two objects in a demonstration, then assigns key states to each of the objects and calculates the relative poses, after that, the framework assigns random position transforms and randomly selects object-pairs to make up new demonstrations.

The deep neural network is responsible for learning features from the demonstration samples and providing configures for manipulation task reproduction. Given the demonstration inputs, the object-pair is a global feature while the object's function part is a local feature, the deep neural network is good at learning this kind of hierarchical features. On the other hand, the proposed network structure allows the robot to simultaneously infer the function and manipulation strategy given a specific object-pair point cloud input.

After training the network, users can reproduce the task by providing a user command such as (clean, sticky roller, dirty tissue), then the robot can not only use the demonstration sticky roller, but also another sticky roller to perform the task.

## 1.4 Major Contributions

To reduce the user effort in robot teaching, we presented robot manipulation teaching systems to enable robots to learn manipulation knowledge from a few visual demonstrations. The learned knowledge can be adapted to new instances so that users do not have to provide extra demonstrations for the new instances. The main contributions can be summarized as follows:

- 1) An Object-pair Mutual Function Knowledge model. Compared to the existing instance-based methods, the proposed knowledge model enables robots to understand the relationship between object function part and manipulation action, so that robots are able to apply the action to new instances with similar function parts.

- 2) An on-site demonstration framework. This framework enables general users to teach robots manipulation knowledge without the help of experts. Compared to the existing transparent teaching approaches, the proposed teaching approach can extract not only the point cloud of the target object but also the object's function part and its 6D pose trajectory when performing a task.
- 3) A virtual demonstration tool that allows users to teach robots manipulation knowledge using synthetic data.
- 4) An Object-pair Mutual Function Knowledge Acquisition Network and the relevant frameworks for manipulation knowledge learning and task reproduction. Given an object-pair point cloud and the activation signal provided by the previous components, the network infers the corresponding function, function parts, and key object states to guide a robot to perform manipulation tasks.
- 5) A robot manipulation teaching system. Integrating different components, this thesis proposed a complete system that can gain knowledge from a few human demonstrations and the system is robust to different kinds of variations.
- 6) A comprehensive evaluation of the algorithms on real robotic applications.

## 1.5 Thesis Outline

This chapter first introduced the overall objectives of the thesis and challenges that need to be solved, then described the overview of the proposed framework and the main contributions of the thesis. The remaining chapters are organized as follows:

- **Chapter 2** describes core concepts in this thesis and reviews the state-of-the-art methods that are relevant to robot manipulation teaching, the advantages and limitations of these methods are also discussed.
- **Chapter 3** introduces the definition and representation of the Object-pair Mutual Function Knowledge model. This chapter also discusses the advantages of using this knowledge model to solve the robot teaching problem.

- **Chapter 4** presents an on-site robot teaching approach and a virtual teaching tool. This chapter also shows the details of how to extract essential variables from on-site visual demonstrations.
- **Chapter 5** presents a framework for demonstration sample augmentation and an Object-pair Mutual Function Knowledge Acquisition Network for manipulation knowledge learning. This chapter also evaluates the performance of the proposed network.
- **Chapter 6** presents a framework for manipulation task reproduction. This chapter also evaluates the performance of using different methods for trajectory generation.
- **Chapter 7** integrates all of the components to build a system that enables robots to learn new knowledge from a few visual demonstrations and reproduce the manipulation tasks under different kinds of variations. This chapter also presents details of implementing the system on a real robot platform.
- **Chapter 8** summarizes our work and discusses the advantages and limitations of the work, as well as some suggestions for future research and possible extensions of the proposed framework.



## **第2章**

# **Motivation and Related Work**



This chapter reviews the state-of-the-art approaches that are related to the proposed robot teaching system, including robot teaching methods, teaching outcomes, and manipulation knowledge learning and reproducing methods. Section 2.1 provides details of various manipulation knowledge used in robot teaching. Section 2.2 shows different ways of how humans show manipulation knowledge to robots. Section 2.3 reviews the demonstration outcomes after teaching. Section 2.4 and 2.5 discuss the learning algorithm and how to achieve few-sample learning. Section 2.6 reviews the existing methods in manipulation tasks reproduction. Section 2.7 discusses the advantages of the existing approaches and their limitations.

## 2.1 Object Manipulation Knowledge

In manipulation tasks, many variables can be used to represent object properties and manipulation behavior. Choosing what variables for teaching and learning is challenging because abstract knowledge and implicit variables are difficult to show to the robot through vision.

### 2.1.1 Object Shape Knowledge

The object shape knowledge contains an object's 3D geometry information, combining with the 6D pose estimation approaches such as the Iterative Closest Point (ICP) algorithm [59] or the LineMOD [60] algorithm, a robot is able to estimate the target objects' states during the task reproducing phase. However, the estimation results rely on the similarity of the template data and the target data, and thus the template data usually can not be used to estimate other instances. Although knowing objects' 6D poses allow robots to perform some complex manipulation tasks such as laundry folding [13], cutting [61], and tidying [62], it requires users to create templates for every new instance, which increases the user burden in robot teaching.

### 2.1.2 Object Function Knowledge

Most of the existing object function learning methods allow the robot to learn object functions off-site. In this paradigm, human teachers need to prepare various kinds of objects and attach function labels to the object parts. Detecting object function (also called object affordance in a loose way) from visual is considered as a segmentation problem, different parts of an object may have different function [63, 64, 65, 66]. For example, a hammer's head has the function "pounding" and its handle has the function "grasping". The function detection results indicate which parts are manipulatable, thus bring benefits for solving robotic manipulation problems. Nguyen et al. [67] presented a deep learning approach to identify object function at the pixel level. Building upon the state of the art architecture, the SegNet [68], their network is able to detect different kinds of functions with relatively high accuracy. Finally, they showed how to use the detect functions to guide a real robot to perform the grasp tasks. On this basis, Nguyen et al. [64], Do et al. [65], and Chaudhary et al. [66] improved the deep learning network structure to reach a higher detection accuracy. On the other hand, Authors in [64] believed that detecting functions directly from image pixels will weaken the relationship between function and object, they proposed a framework to detect objects from an image at the first stage, then apply another network to predict functions on that object. Due to the lack of motion information, these methods can only handle simple manipulation cues like where to grasp.

### 2.1.3 Trajectory Knowledge

Trajectories are commonly used to tell a robot how to perform tasks during on-site demonstrations. In traditional trajectory optimization approaches [69, 70, 71], human teachers show robots how to do a task with a series of waypoints, the algorithm then generates efficient trajectories for robots to move smoothly between any two waypoints. Learning trajectory can be achieved in either the operation space or the joint space. In the former case, some positions of the generated trajectory may be unreachable, while the latter one depends on the kinematic chain of the robot and can not be directly applied to other robot platforms.



In the visual learning paradigm, trajectories are represented by a sequence of keyframes that express crucial states which are for completing a given task. The learning algorithms are responsible for learning variables from the given keyframes and mapping the variables to robot actions [72].

#### 2.1.4 Policy

The Policy is introduced to describe the relationship between state and action. In particular, a policy can be represented as  $\pi: X \rightarrow A$ , where  $X$  is the input domain, in the robot teaching problem, this input can be raw observations or RGB images [73, 74, 75, 76, 77],  $A$  is the action space. An extra reward function is also needed to evaluate whether a policy is good or bad. Given an input, the agent should choose a proper action within the action space to maximize the reward. learning policy allows the robot to make decisions depending on the current state/environment, thus this approach performs better than other approaches in dynamic environments. However, this flexible decision-making strategy also brings unpredictable robot behavior, owing to the system can only tell people the robot's next action but not the goal of a task. This unpredictable robot behavior may bring dangers in complex environments.

### 2.2 Demonstratoin Approaches

There are two main approaches that a robot can acquire knowledge from a human teacher. The first approach is the human teacher labels data off-site and the robot learns from the annotated data. The second approach is the human teacher demonstration on-site and the robot learns from the demonstration.

Compared to off-site data annotation, general users are usually more inclined to the on-site demonstration, because the latter is more intuitive and natural. Currently, the most popular on-site demonstration approaches are kinesthetic teaching, teleoperation, and robot learning-by-watching.



Figure 2.1: An example of the kinesthetic teaching.

### 2.2.1 Kinesthetic Teaching

Kinesthetic teaching is the simplest and most intuitive approach (Figure 2.1). During a demonstration, the robot is set on a special mode, this model allows the teacher to maneuver the robot joints to desired positions with minimal effort [78, 79, 80, 81, 82]. Meanwhile, robot states such as joint angles or torques are recorded by on-board sensors. Most of the industrial robots are equipped with the kinesthetic teaching function because this approach does not require the human teacher to have experience in robotics, but experienced teacher usually provides a higher quality of demonstrations.

Kinesthetic teaching does have some limitations. Firstly, this method is not suitable for high degree-of-freedom (DOF) platforms such as humanoid robots or robots with dexterous hands because the demonstrations are difficult to perform. Secondly, this method is not suitable for complex manipulation tasks owing to the limited information obtained from the demonstrations.

### 2.2.2 Teleoperation

Teleoperation is another popular demonstration approach (Figure 2.2). In this approach, a human teacher uses an external remote device such as a joystick or a teach pendant to



Figure 2.2: An example of the teleoperation.

control the robot. With the development of the user interface, Virtual Reality (VR) device [52, 83, 84, 85, 86] has been introduced to teleoperate the robot. Compared to kinesthetic teaching, teleoperation is safer because it does not require the teacher to stand nearby. Also, the images captured by the robot head-mounted cameras have higher quality due to the elimination of visual artifacts.

Since the robot motion is computed by some predetermined algorithms such as Jacobian transpose, the robot does not always move as expected. the teacher usually needs more time to adapt to the unnatural feedback.

### 2.2.3 Robot Learning-by-Watching

This approach does not require a human teacher to maneuver the robot or operate the external devices, the teacher can perform tasks using his own body (Figure 2.3). During the demonstration, the teacher only needs to ensure the relevant hits such as the human hand and the manipulated objects appear within the robot’s field of view. This approach is also widely used in imitation learning [87, 88, 89, 90, 91, 92], in which the robot imitates a human teacher’s behavior.

The learning-by-watching method is challenging because the robot and the demonstrator are observing a task in different spaces, therefore the robot needs to map observations in the demonstrator’s space to its space.



Figure 2.3: An example of the robot learning-by-watching.

## 2.3 Manipulation Learning Ability and Generalizability in Robot Learning from Human Teaching

The previous section has mentioned the manipulation knowledge and demonstration approaches for robot teaching. However, there are two crucial problems in robot learning from human teaching:

- How well a system learns from human teaching?
- How well a system reproduces a manipulation task?

Based on these two questions, we consider two main factors that affect the performance of a robot manipulation teaching system. The first factor is the manipulation learning ability, which focuses on the problem of how a robot learns a manipulation task. For example, industrial robots typically perform only one function, in these robot systems, the robot actions are predefined and unlearnable, and thus their manipulation learning ability is low. On the contrary, a high manipulation learning ability system requires a robot to learn the intention behind the manipulation behavior. Take imitation learning as an example, in the action unit imitation, a robot only need to mimic the demonstrator's

actions without concerning whether the goal of a task is achieved, while the purposive imitation requires a robot to learn the actions and goals of different tasks. Commonly, a system with strong manipulation learning ability allows robots to learn many complex manipulation tasks, and thus this factor is important to robot manipulation teaching.

On the other hand, generalizability describes how well a system reproduces a manipulation task. In this thesis, we use two metrics to evaluate the generalizability. The first metric is whether a system can be applied to different robot platforms to achieve the same task without adjustments after teaching. A system that relies heavily on the robot configuration is usually fragile in task reproduction because a trained model may become useless in other robot platforms, which means users have to do the demonstration again. The second metric is whether a system can use never-before-seen objects to achieve the same task. Since a system that can generalize the learned skills to new instances can effectively reduce the user effort in robot teaching.

### 2.3.1 Evaluating Manipulation Learning Ability in Robot Manipulation Teaching

Currently, there are two main paradigms that can be used to guide a robot to accomplish manipulation tasks. The first paradigm is trajectory imitation. During a demonstration, the system extracts a trajectory  $T = p_1, p_2, p_3, \dots, p_n$  that contains a series of positions or 6D poses of the human hand. The trajectory will then be converted to control signals, such as joint state/angle [93, 94, 78], end-effector position [95, 96, 40, 97, 43, 44, 41, 98, 42, 99], end-effector pose [100, 101, 102], and end-effector force [103, 104, 105, 106]. According to different tasks and objects, the manipulation trajectories are usually different, therefore the system must be able to generate suitable trajectories for the corresponding tasks. The advantage of this paradigm is the system outcomes are explainable because users know that the robot will move along a specific trajectory. The disadvantage is that the robot has difficulty in handling dynamic tasks, say, the cup's position is suddenly changed when the robot is executing a pouring task.

The second paradigm is reinforcement learning. In this paradigm, an agent learns skills

by interacting with the outer environment then uses the feedback to update its policy to decide its next move. This process is modeled as a Markov Decision Process (MDP), which consists of a 4-tuple  $(S, A, T, R)$ , where  $S$  is a set of state,  $A$  is a set of actions,  $T(S', A, S) = P(S'|S, A)$  is the probability that performing action  $A$  in state  $S$  will lead to state  $S'$ ,  $R$  is the reward received for performing action  $A$  in state  $S$ . In particular, the agent maintains a policy that describes the estimated reward of choosing different actions in a state, and the goal of reinforcement learning is to learn an optimal policy, which maximizes the long-term cumulative rewards. Combining with the deep learning technique, many significant works have been proposed to show how to apply the reinforcement learning method to some robot manipulation tasks such as robotic grasping [107, 108], pushing an object [109], assembly [110], hanging a coat hanger on a clothes rack [111], opening a door [112], and manipulating deformable objects [113, 114, 115]. In contrast to the trajectory imitation paradigm, the reinforcement learning paradigm is good at handling dynamic tasks, but the robot's motion is not predictable, when the agent makes a wrong decision, the results can be catastrophic.

### Instance Matching Based Trajectory Imitation

In this method, a robot imitates a manipulation task by observing the demonstrator's hand movement, meanwhile, it records the demonstrator's hand positions at a short interval to compose a motion trajectory. This trajectory is linked to a specified instance. The next time when the robots want to manipulate the instance, it can simply apply the trajectory. Welschhold et al. [18] considered a manipulation action includes two 6-DOF trajectories, one for the hand of the teacher and one for the target object. They presented a hyper-graph to learn the correspondence of these two trajectories and showed how to use their methods in some robot manipulation tasks such as opening a swivel door, opening a drawer, etc. The instance matching based trajectory imitation method usually does not require many demonstrations for task reproduction, however, the trajectory can not be used in other instances.

## Behavioral Cloning

Behavioral cloning belongs to the reinforcement learning paradigm. This method uses supervise learning to learn the state-action pairs provided by demonstrators [116, 117], and thus the reward can be ignored. For example, Pomerleau et al.[118] use the sensor input of a land vehicle and the steering angle as the state-action pair, then learn the correspondence of the action to the state through a large number of demonstrations to achieve autonomous navigation. Similarly, authors in [119, 120, 121, 122] applied the behavioral cloning method for robot navigation or motion control. In the robot manipulation field, estimating reward from observing human's demonstrations is difficult, Finn et al.[123] applied the behavioral cloning method by sampling state-action pairs from the demonstrations then provided 1,300 human demonstrations to train a meta-model. Given the meta-trained model, their method can quickly adapt to new instances in the placing task via one-shot learning. Rahmatizadeh et al.[124] modified the deep neural to allow the model to detect multiple tasks and trained without meta-learning. According to their results, the robot has successfully learned the pick and place, pushing tasks through around 500 human demonstration. The shortcoming of behavioral cloning is obvious, when the agent visited a state that is never trained on, it can not find a proper action to match the state and this can lead to task failure. To solve this problem, users must provide a lot of demonstrations to improve the system performance for a single task, which affects the user experience in robot teaching.

## Inverse Reinforcement Learning

Inverse reinforcement learning is proposed to solve the problem of learning the reward function from the demonstrations provided by experts, and thus it allows a robot to learn the goal information from a task. In [125], the reward function is presented by a linear combination of feature functions. On the basis of the linear inverse reinforcement learning method, Abbeel et al. [126] proposed the feature expectations vector to estimate the reward function from demonstrations. When it comes to robot control, the problem be-

comes complex. Finn et al.[127] has presented a deep spatial autoencoder method to learn feature points for state representation, after the carefully designed network training steps and feature pruning algorithm, their method has successfully learned the representation that corresponds to image-space coordinates of objects in different tasks. This visual feature extraction method is then be used in [128] to learn the cost functions for their guided cost learning algorithm, results showed that their method has achieved success in teaching a robot to place a dish into a rack and pour almonds into a cup. Although the inverse reinforcement learning method shows promising results in the car [129] and robot control tasks [128], the criteria of designing feature functions for reward estimation is ambiguous.

### **The Position of This Study**

Compared to the reinforcement learning paradigm, we tend to adopt the trajectory imitation paradigm due to two main reasons: Firstly, estimating goal information in some tasks is difficult. Currently, we can hardly find a reward function that can be used for all manipulation tasks. For a general user-oriented robot manipulation teaching system, asking a user to provide reward functions is difficult. Secondly, the state in reinforcement learning usually contains many uncontrollable variables such as background changes, robot inner parameter changes, etc. These uncontrollable variables making the robot's behavior unpredictable, and this can lead to catastrophic failures. In contrast, from the outcomes of the trajectory imitation method, users can know about the robot's motion trajectory before execution, and thus the robot's behavior is predictable. In this study, we consider teaching simplicity and safety are more important than goal achievement, and thus the study only learns the action without concerning about the goal.

### **2.3.2 Evaluating Generalizability in Robot Manipulation Teaching**

There are usually many different choices for us to complete a task in our daily lives. For example, we can ask different robots to perform a task and the robot can decide to use which object to perform the task given many candidates. In this study, generalizability



is presented to describe the problem of how well a robot performs a task after teaching. Generally, teaching a robot a lot of manipulation tasks usually takes much time, we do not want to do the demonstration again when the robot platform is changed. On the other hand, when performing a task, a robot may need to decide to use which object when there are many candidates. The instance matching based method focuses more on the properties of a specific instance, while the task representation based method focuses more on whether the object in the current state can be used for a task. For example, given two cups in a pouring task, one of the cups is recorded in the database but this cup is placed upside-down, another cup is never-before-seen but in a normal state. An instance matching based method tends to rotate the known cup then perform the pouring action, while the task representation based method tends to directly perform the pouring action on the never-before-seen cup.

### **Object 6D Pose Estimation**

Object 6D pose estimation is a typical instance matching based method. In this method, users need to provide templates for the target objects such as 3D geometrical models or object images in different viewpoints. Since the target objects are already known, the robot can easily estimate the objects' pose using the Iterative Closest Point (ICP) [59] algorithm for point cloud alignment or the Line-MOD [60] algorithm for visual template matching. Knowing an object's 6D pose can simplify the manipulation process by manipulating the object along specified trajectories with correct poses [130, 131, 8, 61, 60]. However, object instance matching based methods have difficulty in handling new instances with different shapes, sizes, or appearances, the slight changes in these properties may cause task failures due to alignment error.

### **Robotic Grasp Detection**

In robotic grasp detection, a system only concerns about which part of the object can be grasped by the robot, and thus it is a classical task representation based method. In

this method, representations are designed to fulfill the grasping purpose. For example, the point representation presented by Saxena et al. [132] is used to detect grasping points of the target objects in RGB Images. To fit the shape of the parallel robot gripper for robotic grasping detection, Jiang et al. [133] proposed a 2-D rectangle representation that consists of 5 parameters ( $rG, cG, bG, mG, \theta G$ ), where  $rG$  and  $cG$  represent the upper-left corner of the rectangle,  $bG$  and  $mG$  are the dimensions of the rectangle and  $\theta G$  is the angle. This representation is then be introduced in other works [134, 135, 136, 137, 138] for robotic grasping pose detection. Although the robot grasp representation allows robots to grasp different objects, it can not be used for other manipulation tasks.

### The Position of This Study

In this study, we consider the task representation based method is more suitable for a robot manipulation teaching system. This is because in most manipulation tasks, users usually do not care about using which object to perform a task, they care more about whether the robot can perform the task successfully. Besides, instance matching based method requires users to provide templates for every instance, which is a difficult job for general users. On the other hand, existing task representations only focus on solving a specific manipulation task such as grasping. In this study, we need to design representations that can be used for multiple manipulation tasks.

### 2.3.3 Function Part Detection Based Trajectory Imitation

In this study, we proposed a function part detection based trajectory imitation method. The position of the proposed method is shown in Figure 2.4. Although inverse reinforcement learning allows a system to learn about the goal information of a manipulation task, this goal information is implicit and it is difficult to know whether the robot has learned a true goal. Furthermore, in the reinforcement learning paradigm, the robot makes a decision based on the current state, which means the change of background may lead to strange behaviors, sometimes it will be dangerous. We consider safety and predictable be-

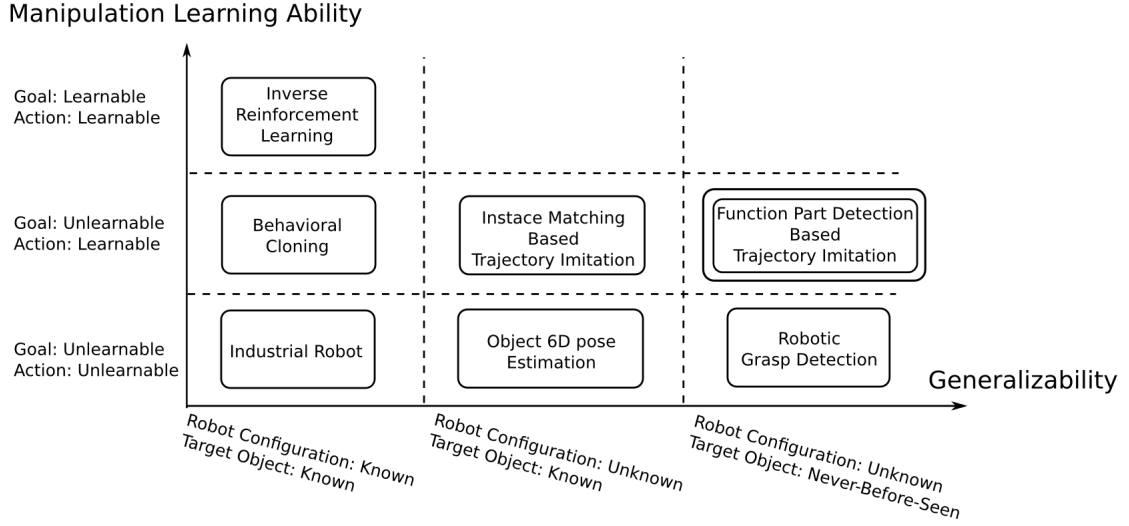


Figure 2.4: The position of the proposed function part detection based trajectory imitation method in manipulation learning ability and generalizability.

havior are more important than goal estimation in a general user-oriented system. Therefore, this study adopts the trajectory imitation method, which means the manipulation learning ability is in the middle level.

On the other hand, a system that can generalize the learned manipulation knowledge to new instances definitely reduces the user effort in robot teaching, and thus the proposed system should have high generalizability. To achieve this goal, we proposed a combinative representation, which links the object state trajectory to the object function part. In this representation, the object state trajectory allows the robot to perform different kinds of manipulation tasks by reproducing the object states, while the function part allows the robot to apply the same manipulation strategy to instances with similar function parts so that users do not need to provide demonstrations for every new instance.

## 2.4 Summary

This chapter introduced the related work of the robot teaching system from various aspects. In robot teaching, the most widely used robot teaching approaches are kinesthetic

teaching and teleoperation because they are easy to implement. However, these two approaches require the user to maneuver the robot joints during the teaching process. Robot learning-by-watching is an ideal teaching method but extracting manipulation knowledge from demonstrations is a difficult problem. Manipulation knowledge determines the variables for teaching and learning, existing robot teaching methods mainly focus on the object shapes and motion trajectories, but the object shape knowledge is difficult to generalize to other instances when reproducing a task. Next, we summarize the existing robot learning from human teaching methods from two aspects: manipulation learning ability and generalizability. The former focuses more on how to learn the useful information for demonstrations. The latter focuses on applying the learned knowledge to more situations as possible during the task reproduction phase. After analyzing the existing methods, this thesis proposed a function part detection based trajectory imitation method that has middle manipulation learning ability but high generalizability.

## 第3章

# **Object-pair Mutual Function Knowledge Description and Representation**



In visual demonstration, many variables can be used to represent manipulation tasks. The knowledge model decides what variables should be extracted from human demonstrations and learned by robots, thus it is a crucial component in bridging the behavior of teaching and learning (Figure 3.1). If the knowledge model is too simple, robots can not acquire enough information for understanding the usage of objects, while a complicated knowledge model pose challenges in human teaching and robot learning. This chapter introduces an Object-pair Mutual Function Knowledge model for robot manipulation teaching. This chapter will discuss what are the essential variables when building a robust knowledge model and how to represent these variables.

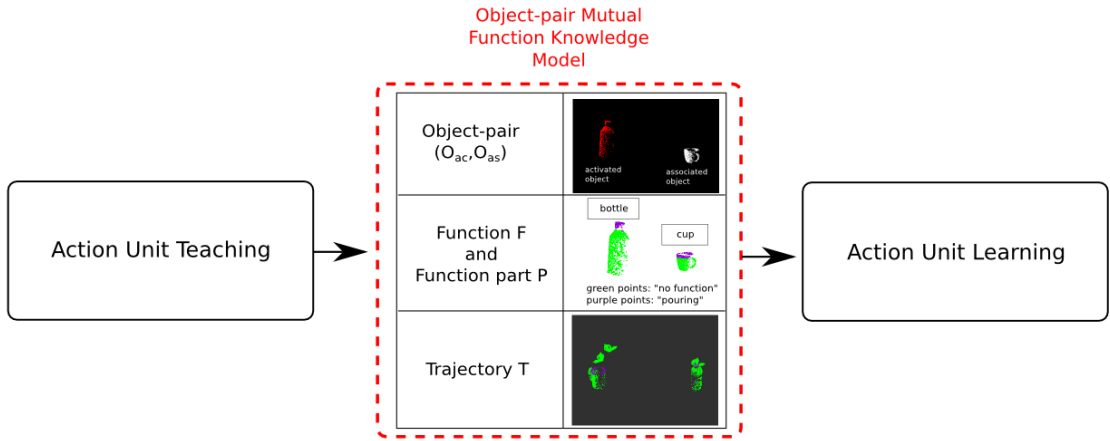


Figure 3.1: The position of the knowledge model in the system.

### 3.1 Manipulation Knowledge Model

In the neurocognitive system, many knowledge models [139, 140, 141] are presented to explain the relationship between vision and action. When manipulating an object, a person needs to first recognize the object and its properties, then execute manipulation actions under visual guidance.

When it comes to robotics, a typical knowledge model including the shape of objects [142], the 6D pose of objects [56, 55], and trajectories for manipulating the objects [143,

18]. This knowledge can reproduce the manipulation when the tested instances are the same as the demonstration objects. However, limited by the object shapes, when using other instances that have different shapes or sizes in the same category, this knowledge is hard to guide a robot to accomplish tasks. To solve this problem, we can link the trajectory to the object function part, but not a specified instance. Besides, a knowledge model should be able to answer the following problems: (1) what is the function of the object; (2) which part of the object is related to this function and (3) how to use the object to complete the function.

### 3.2 Object-pair Mutual Function Knowledge Model

To solve the above problem, this thesis proposes an Object-pair Mutual Function Knowledge model MFK, which is defined as follows:

$$MFK = (O_{ac-as}, F, P, T) \quad (3.1)$$

where  $O_{ac-as}$  is an object-pair  $(O_{ac}, O_{as})$ .  $O_{ac}$  is the activated object and  $O_{as}$  is the associated object.  $F$  is a set of functions,  $F = \{\text{"no function"}, \text{"pouring"}, \text{"placing"}, \text{"cleaning"}, \text{"cutting"} \dots\}$ . An agent can use the object-pair  $O_{ac-as}$  to achieve one of the function  $F_{O_{ac-as}}$  within the function set  $F$ .  $P$  represents function parts of the objects that are related to the function, according to the data format,  $P$  can be a pixel set (image) or a point set (point cloud).  $T$  is a motion trajectory that represents an action for a specific function using the object-pair  $(O_{ac}, O_{as})$ . The difference between the proposed knowledge model and the existing knowledge model is that the proposed model focuses more on explaining how to use one object to act on another object to achieve a specific function, therefore we adopt an object-pair as input instead of a single object. Other elements also play important roles in representing a function, and we will discuss all the terms one by one, including their necessity and how to represent the symbols in robotic applications.



### 3.3 Representing the Object-pair Mutual Function Knowledge Model

The visual information captured by the robot's camera may contain many features and patterns, these features and patterns can not be directly used to drive a robot, thus we need an intermediate which is called representation to connect the visual information and robot action.

#### 3.3.1 Representation of the Object-pair ( $O_{ac}, O_{as}$ )

Generally, a target object is needed when performing a function. For example, we can not use a knife to cut itself, we can only use it to cut other things. Therefore, we use the object-pair ( $O_{ac}, O_{as}$ ) in the knowledge model.

In this study, the object-pair is represented by a 4-channel point cloud, where the first three channels are x, y, z coordinates of the points and the last channel is the activation signal, for points from the activated object, this value is set to one, otherwise set to zero.

#### 3.3.2 Representation of the Function F

According to which object is activated, the function can be different. For example, given a bottle and a cup, if the bottle is activated, the expected output function is “pouring”, due to physical limitation, if the cup is activated, the output function is “no function” (Figure 3.2). Similarly, given a plate and an apple, we can place the apple on the plate but we can not place the plate on the apple. Therefore, if the plate is activated, the output function ought to be “placing”, if the apple is activated, the output function ought to be “no function”.

The mutual function set in this study is  $F = \{\text{“no function”}, \text{“pouring”}, \text{“placing”}, \text{“cleaning”}, \text{“cutting”} \dots\}$ , users can also extend the function set by adding functions.

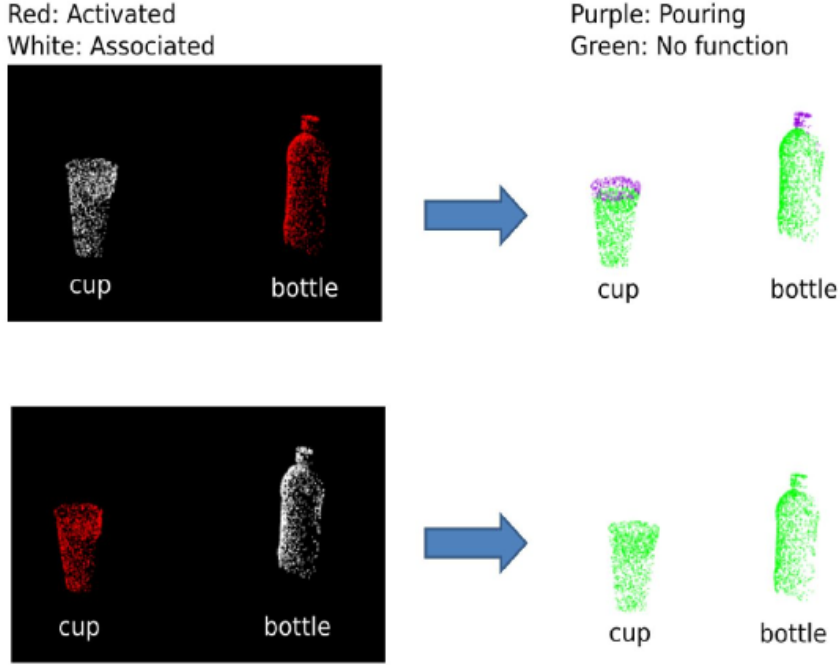


Figure 3.2: Illustration of the relationship between the activated object  $O_{ac}$  and the corresponding function  $F$ . If the bottle is activated, the corresponding function is “pouring”, if the cup is activated, the corresponding function is “no function”.

### 3.3.3 Representation of the Function Part P

The function part  $P$  is a set of points, it describes which parts of the objects are related to the function. An example is shown in Figure 3.3, given an object pair (bottle, cup), the expected function is “pouring” and the function parts  $P_{ac-as}$  are points of the opening part of the bottle and the cup.

Similarly, for the object-pair (hacksaw, wood), the expected function is “cutting” and the function parts  $P_{ac-as}$  are the points of the blade of the hacksaw and the points of the whole wood, which means we can cut somewhere of the wood using the blade. Therefore, the function part of the object-pair (hacksaw, wood) is  $P_{ac-as} = \{\text{all points with the “cutting” label}\}$ , each point has four channels, including the  $x$ ,  $y$ ,  $z$  coordinates of the point and a function label  $F_i$  that is defined by the function set  $F$ .

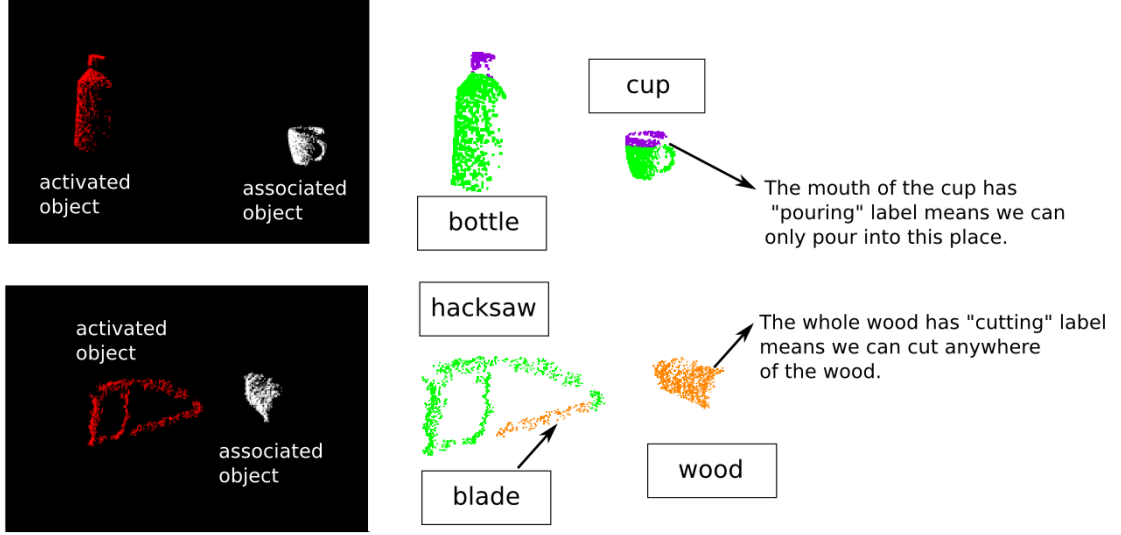


Figure 3.3: Illustration of the function parts of different object-pairs.

### 3.3.4 Representation of the Trajectory T

Instead of linking the motion trajectory  $T$  to an object, we link  $T$  to the function part, the merit of linking motion to the function part is that when a robot finds a never-before-seen object, it can observe whether the object has known function part  $P$ , if the number of points with the function part  $|P|$  is larger than a specific threshold, the robot can apply the learned motion to complete the corresponding function.

This study represents a motion trajectory  $T$  with a sequence of the states  $T = \{S_s, S_{m1}, S_{m2}, \dots, S_{mn}, S_g\}$  of the object being grasped, where  $S_s$  is the start state,  $S_g$  is the goal state and the rest are middle states. Every object state is represented with a 6D pose. To simplify then trajectory, this study only uses three key states  $T = \{S_s, S_m, S_g\}$ , the middle state  $S_m$  is the one that has the largest position variance with respect to the centroids of start state  $S_s$  and the goal state  $S_g$ , an example is shown in Figure 3.4. Concretely, we use the centroid of the start state and end state to create a vector, then compute the distances between the centroids of the middle states and the vector, finally, the middle state with the largest distance will be selected as the key middle state. An equation is shown below:

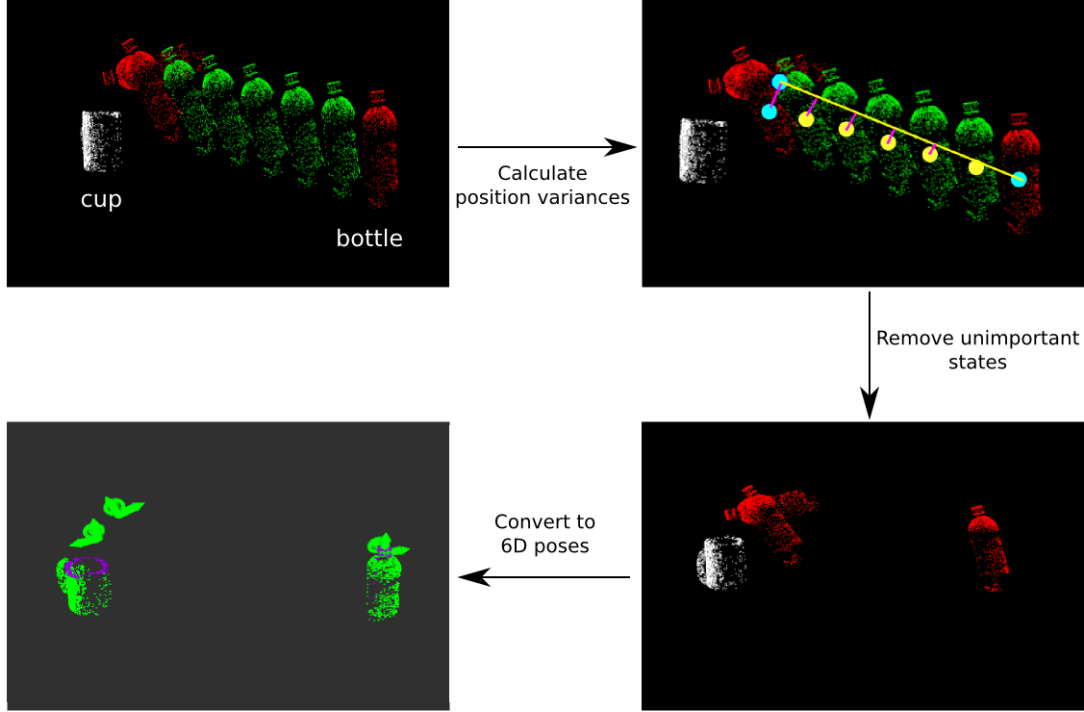


Figure 3.4: Illustration of how to extract key poses from a motion trajectory.

$$\mathbf{c}_1 = (x_s, y_s, z_s) \quad (3.2)$$

$$\mathbf{c}_2 = (x_g, y_g, z_g) \quad (3.3)$$

$$\mathbf{c}_{mi} = (x_{mi}, y_{mi}, z_{mi}) \quad (3.4)$$

$$d_i = \frac{|(\mathbf{c}_{mi} - \mathbf{c}_1) \times (\mathbf{c}_{mi} - \mathbf{c}_2)|}{|\mathbf{c}_2 - \mathbf{c}_1|} \quad (3.5)$$

Where  $\mathbf{c}_1$  is the centroid of the start state,  $\mathbf{c}_2$  is the centroid of the end state,  $\mathbf{c}_{mi}$  is the centroid of the  $i$ th middle state, and  $d_i$  is the distance of the  $i$ th middle state.

According to the function, the being grasped object can be the activated object  $O_{ac}$  or the associated object  $O_{as}$ , for example, in the function “pouring”, the being grasped object is a bottle ( $O_{ac}$ ), however, in the function “placing”, the activated object ( $O_{ac}$ ) is a plate and the being grasped object is an apple ( $O_{as}$ ).

### 3.4 Discussion

Different from previous methods that the manipulation behavior is triggered by the object shape, in the proposed model, the manipulation behavior is triggered by the function part. This characteristic brings many benefits. Take the pouring function as an example (Figure 3.5), since all the cups and bowls have an opening part, the manipulation knowledge of pouring into a cup also works for pouring into a bowl.

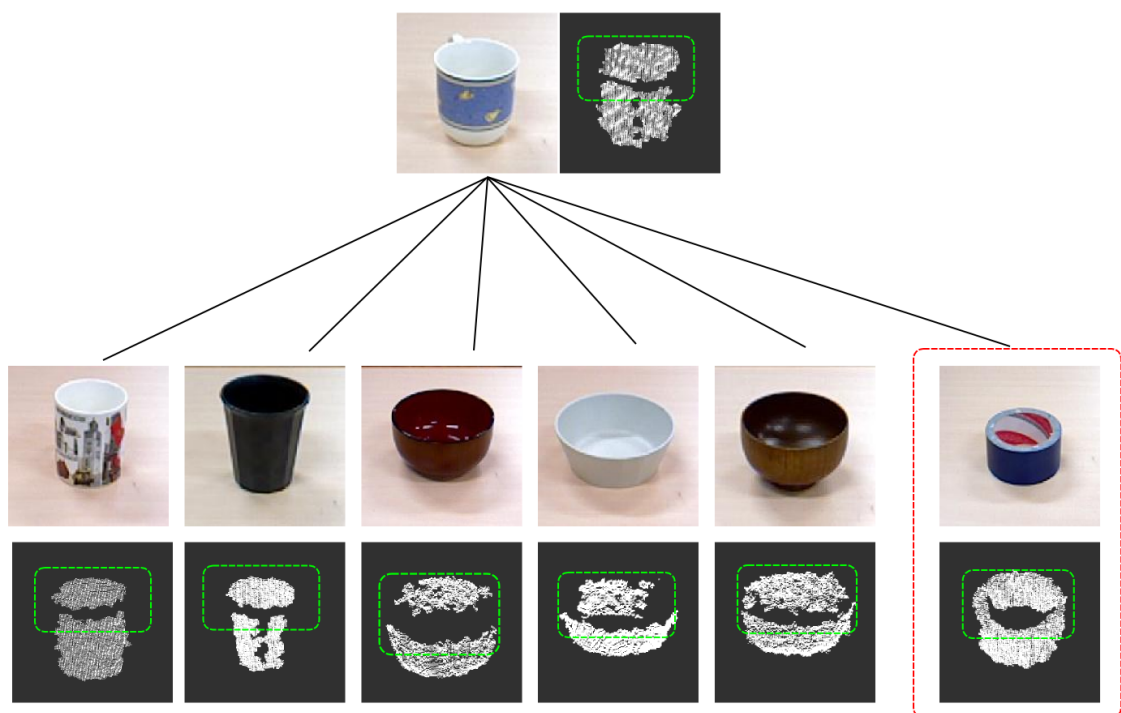


Figure 3.5: Illustration of the advantages and disadvantages of the proposed knowledge model. Given a cup for demonstration, since other cups and bowls have a similar opening part (colored in green), therefore the pouring skill can be generalized to these instances. However, a tape that lying on the table (colored in red) also has a similar opening part, which causes faulty generalization.

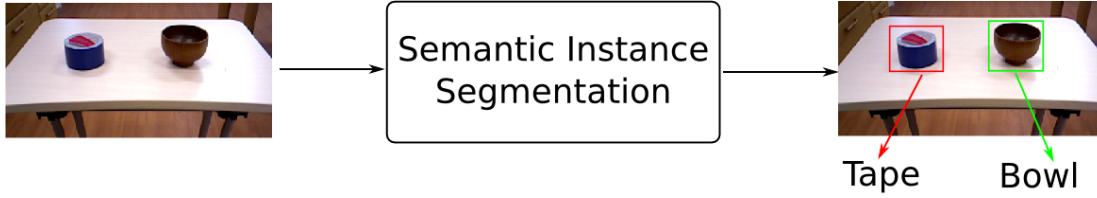


Figure 3.6: Handling faulty generalization with the semantic instance segmentation component.

### 3.4.1 The Faulty Generalization Problem and Its Solution

However, instances in other categories that have a known function part may trigger a wrong manipulation behavior, this problem is considered as a faulty generalization. For instance, when a tape is lying on the table (Figure 3.5, colored in red), it also has a similar point cloud geometry to the cup and also has an opening part, these similarities will lead to wrong manipulation decision.

Fortunately, although it is difficult to distinguish a tape from a bowl using only point cloud geometry due to partial observation, it is still possible to distinguish them from RGB image because they have different visual features (Figure 3.6). In the reproducing stage, we can apply a semantic instance segmentation component to find out the desired objects, more details will be discussed in chapter 6.

## 3.5 Summary

This chapter introduced the Object-pair Mutual Function Knowledge model, which consists of the object pair ( $O_{ac}, O_{as}$ ), the function set  $F$ , the function parts  $P$ , and the trajectory  $T$ , which is summarized in Figure 3.7. In addition, the object pair is represented in a point cloud format, the function set has some concrete functions such as “pouring”, “cutting”, the trajectory is represented by three 6D poses. Compared to the previous object shape-based knowledge models, the proposed knowledge model also pays attention to the object’s function part, this property allows the proposed method to generalize the

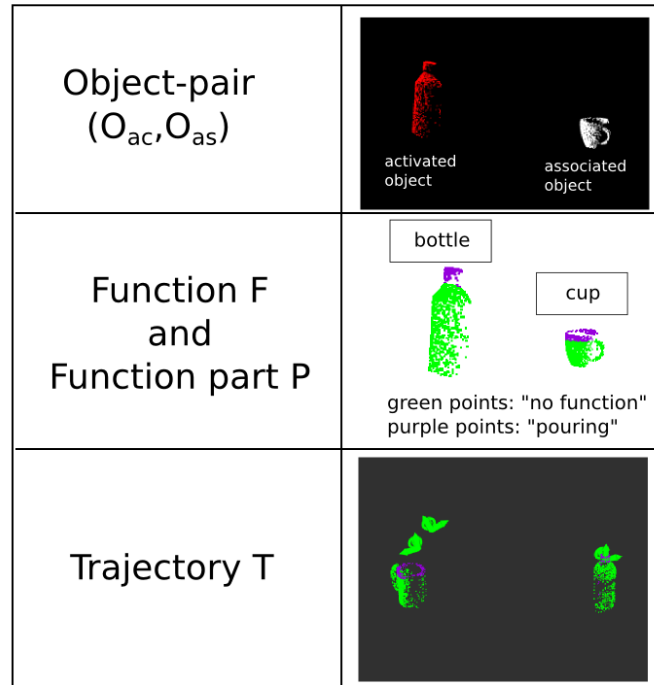


Figure 3.7: Illustration of the elements in the Object-pair Mutual Function Knowledge model.

knowledge to new instances easily. However, it also presents a problem of how to acquire the corresponding variables from human teaching. In response to this problem, the framework presented in Chapter 6 has introduced a semantic instance segmentation components. In the following chapter, an approach that can extract desired variables for the knowledge model from visual demonstrations will be presented.





## **第4章**

# **Acquiring Manipulation Knowledge through Transparent Teaching**



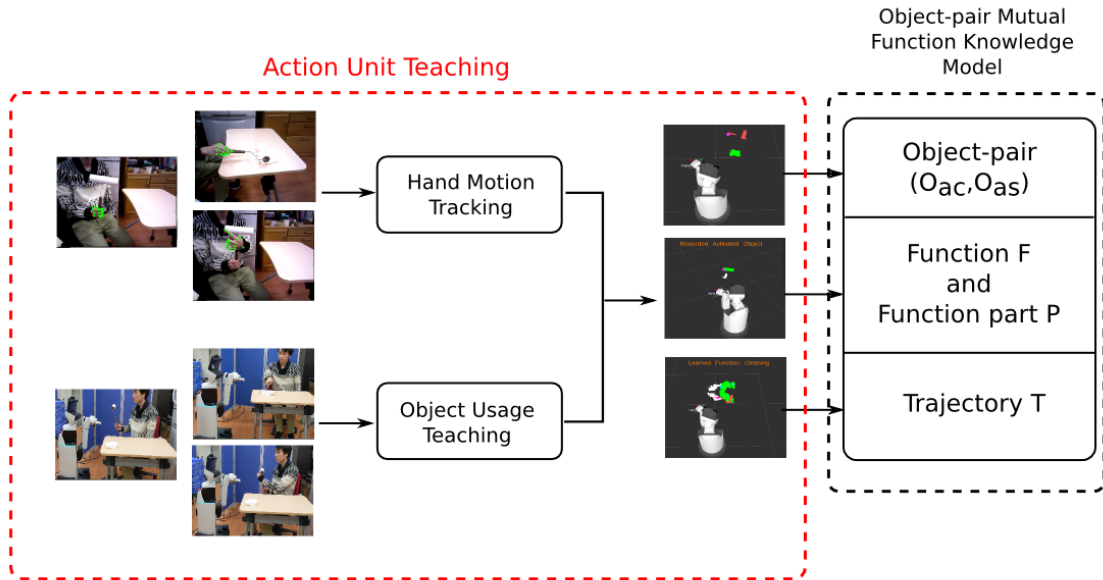


Figure 4.1: The position of the teaching component in the system

Chapter 3 presented an Object-pair Mutual Function Knowledge model that aims to address the scalability and generalizability issues found in existing methods. In order to extract the corresponding variables in the knowledge model, the system needs to extract the object-pair, the object's function part, and the object state trajectory from human demonstrations (Figure 4.1). However, we can hardly find a teaching approach that allows a system to extract these variables from human demonstrations. For acquiring the desired variables, this chapter presents two teaching approaches: the on-site teaching approach enables the robot to acquire variables from watching, while the virtual teaching approach is presented as a supplement.

## 4.1 Transparent Teaching

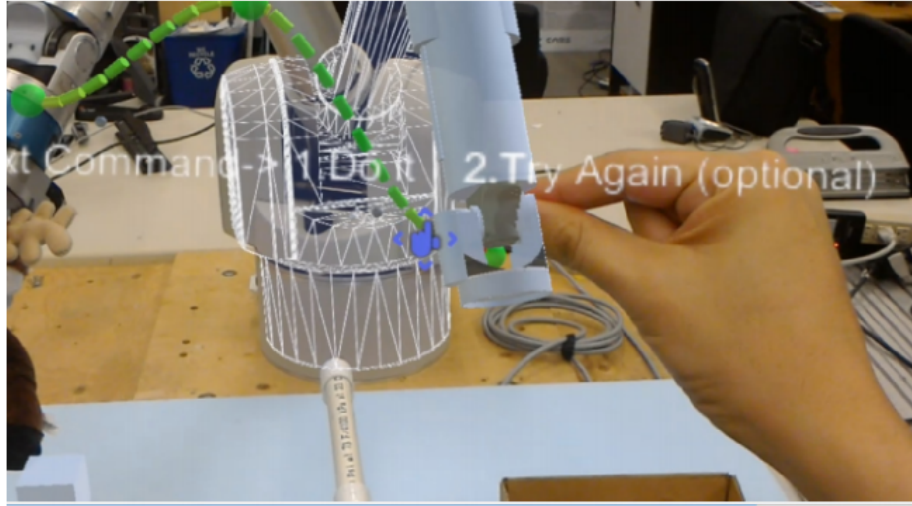
When teaching robots, explicit feedback is important because it can tell users what information has been acquired by the robot and whether a demonstration is good or bad. Here we introduce the term transparent teaching to describe a system that can help users understand what the robot has learned, in other words, a teaching system is transparent if the robot's learned behavior is predictable. For example, traditional industrial robot teaching systems are transparent because the extracted trajectories are shown to the users. On the other hand systems in [45, 46, 12] are not transparent because users can not predict the robot's behavior through the system outcomes (Figure 4.2). In this chapter, we propose a framework that can extract essential knowledge from human teaching, meanwhile the outcomes are transparent to users.

In the kinesthetic teaching process, the human teacher maneuvers the robot joint to desired poses then the system records the waypoints to get manipulation trajectories [144, 145, 146, 147, 148, 149]. When it comes to the visual demonstration, a robot usually acquires manipulation trajectories by tracking the motion of the human hand [18, 150] and reproduce the motion by mapping the human hand to the robot gripper. These teaching systems are transparent to users because they provide explicit outcomes (the waypoints) to show the manipulation plan.

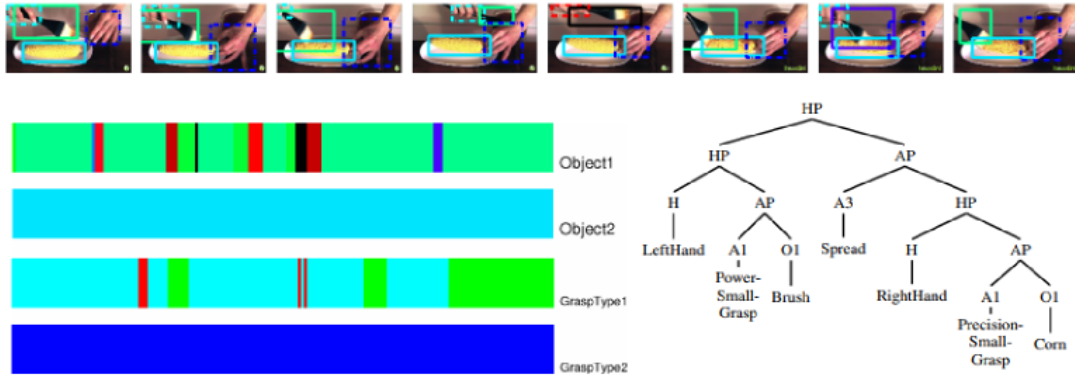
On the other hand, demonstration systems in [151, 152, 53] encode the visual features into parameters, including object states and manipulation motions, but the system outcomes are non-transparent to users. In summary, the previous transparent robot teaching system only considers how to move from the start pose to the end pose along the trajectory without noticing the object states, while the non-transparent teaching system can not provide knowable outcomes for demonstrators.

## 4.2 An On-site Teaching Approach

To acquire more meaningful information from visual demonstration while keeping the system transparent, this thesis presents an on-site visual demonstration framework.



(a)



(b)

Figure 4.2: Illustration of transparent teaching systems (a) An example of a transparent robot teaching system proposed in [149]. (b) An example of a non-transparent robot teaching system presented in [12]

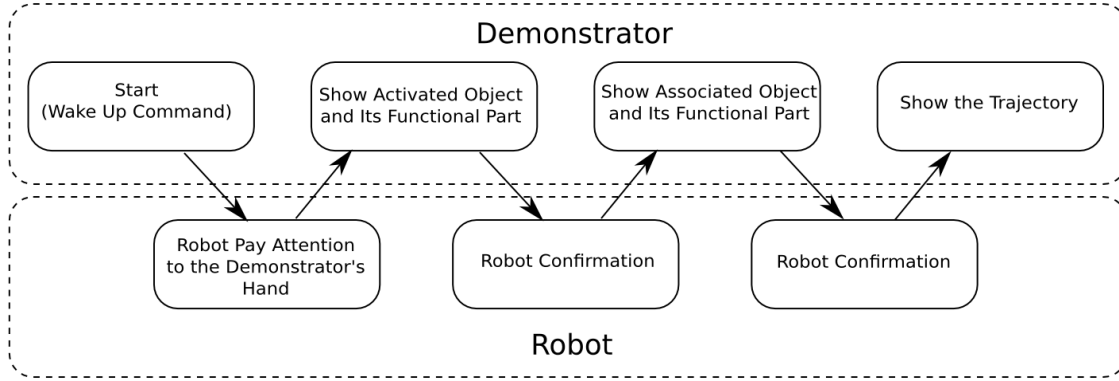


Figure 4.3: An overview of the on-site teaching approach.

### 4.2.1 Overview of the Teaching Process

An overview of the demonstration procedure is shown in Figure 4.3. This procedure starts with a wake-up command, this command can be some predefined sentences such as “I will tell you how to use this object”. Then the demonstrator will show the robot an activated object and its function part, the robot will observe the demonstrator’s hand and record the handheld object and its function part in the database, After confirmation of the first step, the demonstrator will show the robot an associated object. Similarly, the robot will give the demonstrator after a successful recording. Finally, the demonstrator will show the robot how to complete a function using this object-pair.

### 4.2.2 Human Hand Keypoint Estimation

Hand motion contains crucial information during a demonstration, the system needs to know about the key positions of the fingers in order to understand human instructions, therefore, a hand keypoint (or key pose) detector is required to achieve this function. In the past, demonstrators are required to wear gloves to improve the accuracy of the hand keypoint detector. Taking advantage of the development of computer vision and machine learning, many approaches [153, 154] have been proposed to extract hand keypoints from RGB images without extra effort (Figure 4.4).

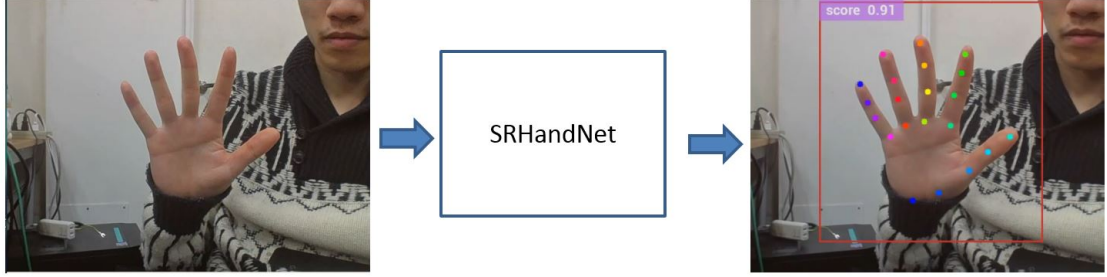


Figure 4.4: An example of the hand keypoint detection results provided by the SRHandNet [154].

SRHandNet [154] is a superior hand keypoint detector. This network has an encoder-decoder architecture, the most important feature of this network is cycle detection, that is, the network iteratively takes the hand region of interests (ROI) as feedback information to boost the performance of the hand estimation network. This cycle detection strategy enables the network to estimate keypoints of small hands and part occluded hands in RGB images with high accuracy.

### 4.2.3 Handheld Object Extraction

The handheld object extraction component is used to obtain the handheld object point cloud. This process can be done by setting a bounding box around the hand because we already got the hand keypoints' x-y-z coordinates. Specifically, the default size of the bounding box is  $0.3m \times 0.3m \times 0.3m$ , users can adjust the size of the bounding box based on their needs. The bounding box is set on the top of the index finger, therefore, the

demonstrator must ensure the target object is higher than his index finger.

---

**Algorithm 1:** Hand Stable Detection

---

**Data:** average hand keypoint positions in previous frame  $P_{prev}$  and current frame

$P_{current}$ , Timer  $T_{timer}$ , Hand Movement Threshold  $D_{threshold}$

1 Initialization:  $T_{timer} \leftarrow 0, handstableflag = False$ ;

2 **while** not hand stable flag **do**

3     Update  $P_{current}$ ;

4      $distanceD = |P_{current} - P_{prev}|$ ;

5     **if**  $D > D_{threshold}$  **then**

6          $T_{timer} \leftarrow 0$ ;

7          $P_{prev} \leftarrow P_{current}$ ;

8     **end**

9     **if**  $D < D_{threshold}$  **then**

10         **if**  $T_{timer} < 5second$  **then**

11              $P_{prev} \leftarrow P_{current}$ ;

12             update timer  $T_{timer}$ ;

13         **end**

14         **if**  $T_{timer} > 5second$  **then**

15              $handstableflag = True$ ;

16             return ;

17         **end**

18     **end**

19 **end**

---

To avoid capturing outlier points, the robot will wait some seconds until the demonstrator's hand is stable, this function is achieved by computing the distance between the neighboring frames of specified hand keypoints, if the distance is lower than a certain threshold (the threshold is set to 5cm in this thesis), the robot will wait for some second and detect again, if the distance is kept at a low level, we considered the demonstrator's hand is stable, then the robot will crop all the points within the bounding box, the hand stable detection algorithm is shown in Algorithm 1. A demonstration is shown in Figure 4.5.



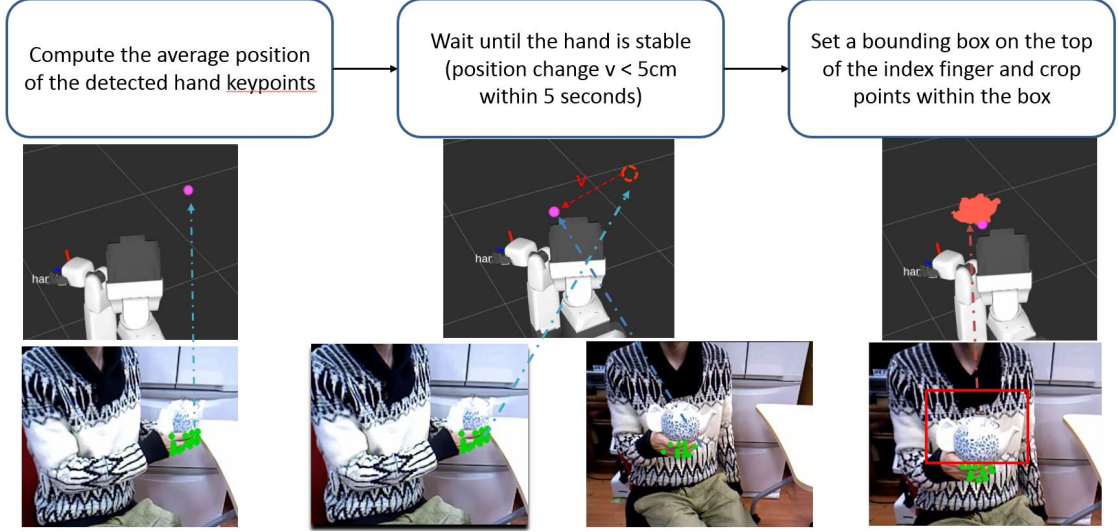


Figure 4.5: A demonstration of the handheld object extraction process.

#### 4.2.4 Extracting Object Function Part through Human-Object Interaction

Once the point cloud of the target object is obtained successfully, the robot will send a response to tell the demonstrator to continue to the next step. During the process, the demonstrator should keep holding the object, then points out the function part of the object using the index finger of the other hand. At the same time, the robot starts tracking the position of the index finger. Once the demonstrator starts moving the index finger to show the robot which part of the object is functional, the robot will record the moving path of the index finger. After the process is finished (the index finger becomes stable), a small bounding box is generated along the path, and points within the bounding box are labeled with functional points. A demonstration is shown in Figure 4.6.

#### 4.2.5 Trajectory Demonstration and Key Object State Extraction

Motion trajectory is widely used to tell a robot how to use an unknown object. In many previous demonstration systems, the trajectories are represented by a set of 2D or 3D points involving the object or hand's positions. In simple tasks such as pushing or

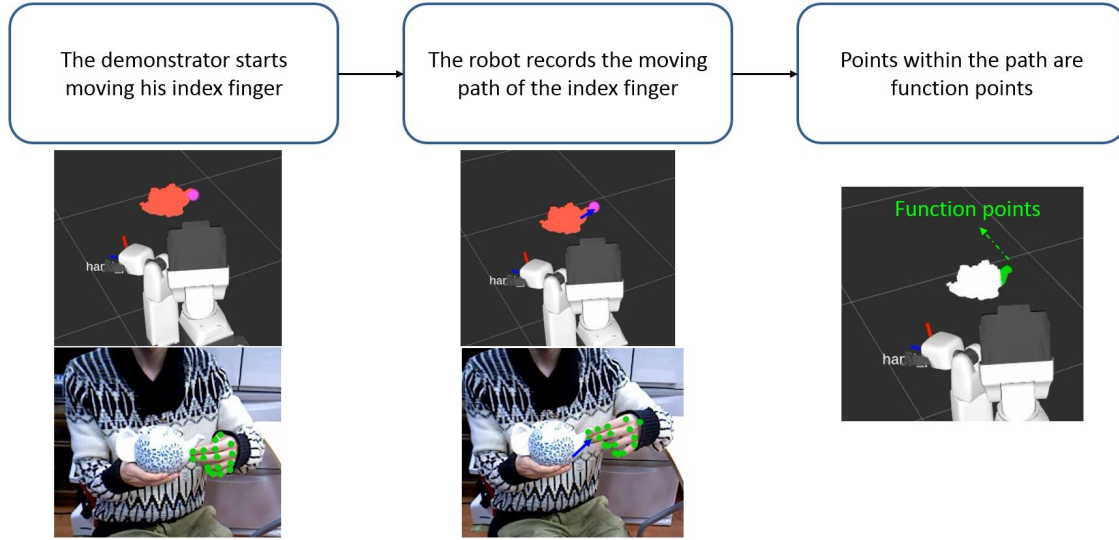


Figure 4.6: A demonstration of the function part extraction process.

reaching, using the position trajectory to guide a robot is enough. However, in more complex situations such as pouring water or cutting food, having the position information is insufficient, the robot also needs to know about the objects' poses.

The trajectory in this study is made up of a series of 6D poses which is described in Chapter 3. During the demonstration process, we predefine a workspace to remove the background, this workspace is a 1m x 1m x 1m cube on the top of the tabletop (the tabletop is not included). Similar to the previous process, the robot waits until the demonstrator's hand is stable, then puts a bounding box to get the handheld object. After that, the robot computes the centroid of the points with the bounding box and put an initial 6D pose ( $cx, cy, cz, 1, 0, 0, 0$ ) on the centroid, where  $cx, cy, cz$  are the  $x, y, z$  coordinates of the centroid. As the demonstrator starts moving his hand, the robot records the point cloud of the handheld object in every frame until the demonstration is finished (the demonstrator's hand becomes stable again). Finally, the robot applies the ICP algorithm [59] to calculate the transformation between each frame and the neighboring frame to obtain a sequence of 6D poses.

The system then computes the transformation between the first point cloud in the trajectory and the function labeled point cloud obtained in the function part extraction step

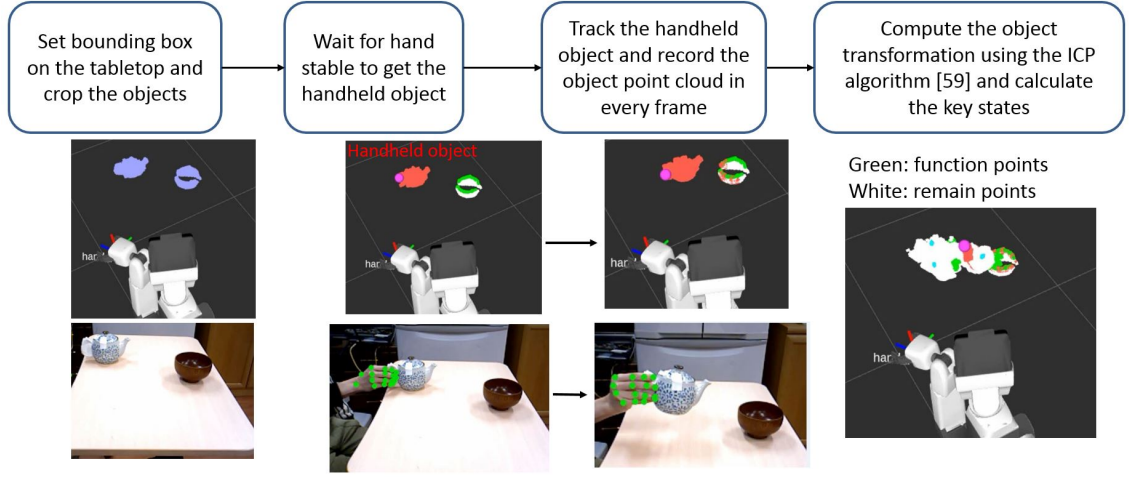


Figure 4.7: A demonstration of the trajectory extraction process.

to get a final 6D pose trajectory with the function labeled point cloud. Based on the 6D pose trajectory, the robot extracts three key states according to the equation described in Chapter 3. A demonstration is shown in Figure 4.7

## 4.3 Experiments

We conduct three experiments to illustrate how the proposed framework acquires manipulation knowledge from a visual demonstration. In these experiments, the demonstrator teaches the robot: (1) how to use a sticky roller to collect dirty tissue for cleaning; (2) how to use a knife (for convenience sake, we used a hacksaw instead) to cut a pudding; (3) how to perform a pouring function using a teapot and a bowl. The demonstration objects are shown in Figure 4.8.

### 4.3.1 Acquiring Manipulation Knowledge from On-site Teaching

In the cleaning demonstration, the whole process is shown in Figure 4.9. At the very beginning, the robot waited for the demonstrator, once the human hand was found, the robot started tracking the human hand and waited until the hand became stable. After

Tasks	Pouring	Cleaning	Cutting
Objects			

Figure 4.8: Objects used in the on-site teaching. In this thesis, we used a hacksaw instead of a knife for the cutting task.

that, the robot acquired the point cloud of the sticky roller. Then the demonstrator started pointing out the function part of the sticky roller and the robot started tracking the demonstrator's index finger. Once the demonstrator's hand was stable again, the function part of the sticky roller was extracted and shown to the demonstrator.

Since the dirty tissue does not have a specific function part, the robot can acquire the point cloud of the dirty tissue during trajectory demonstration. In this stage, the robot first segmented the point clouds on the tabletop and waited for the demonstrator, once the demonstrator grasped the sticky roller (color in red), the robot automatically labeled the other point cloud (the tissue) as an associated object (color in green), then the robot recorded the trajectory of the stick roller when the demonstrator performing the task. Finally, the robot computes the transformation between the labeled stick roller point cloud and the grasped object and applies the transformation to the rest of the trajectory to get a 6D pose trajectory.

In the cutting demonstration, the demonstrator teaches the robot how to use a knife (for convenience sake, we used a hacksaw instead) to cut a pudding, the whole process is shown in Figure 4.10. Similar to the cleaning demonstration, the robot first waited until the hand became stable. After that, the robot acquired the point cloud of the knife. Then the demonstrator started pointing out the function part of the knife and the robot extracts the blade part of the knife. Finally, the robot acquired the point cloud of the

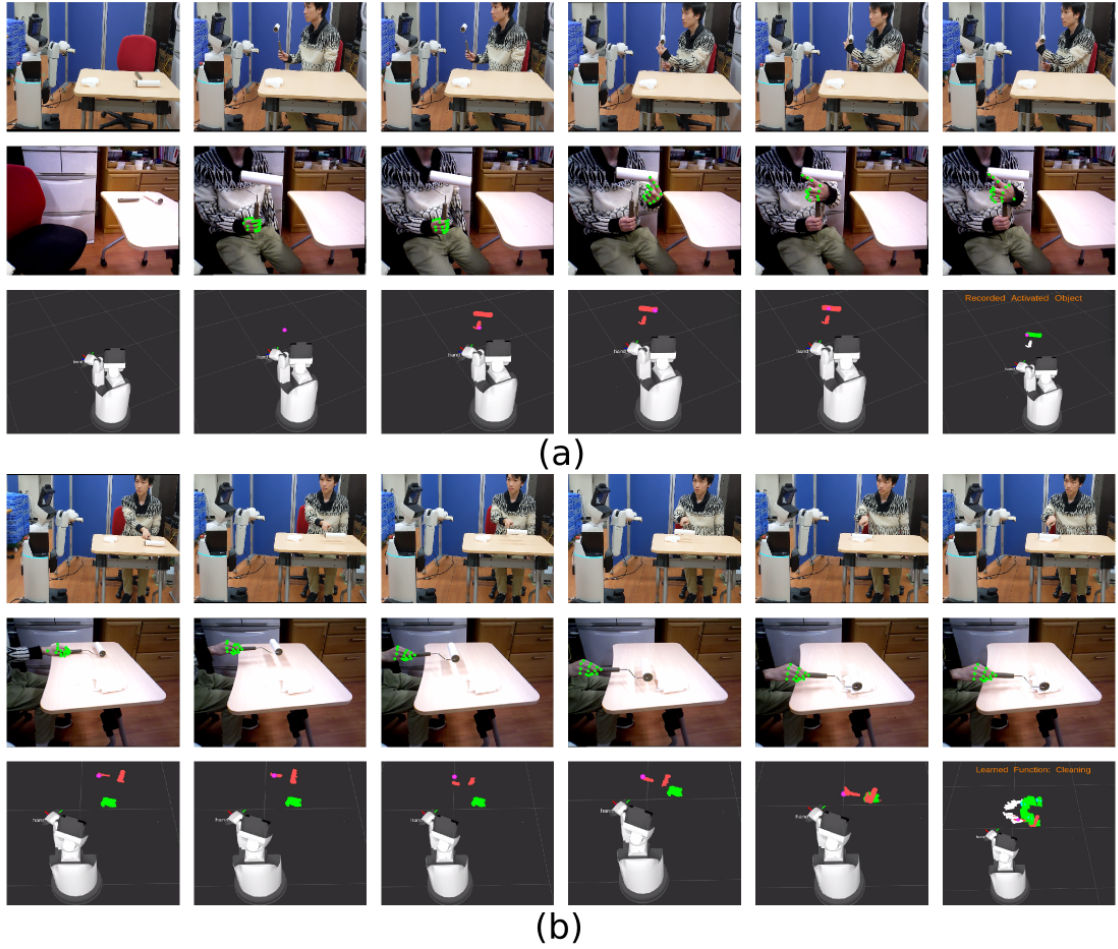


Figure 4.9: Example of the cleaning demonstrations. (a) The demonstrator is teaching the robot which part of the sticky roller is functional. (b) The demonstrator is teaching the robot how to use a sticky roller to clean a piece of dirty tissue.



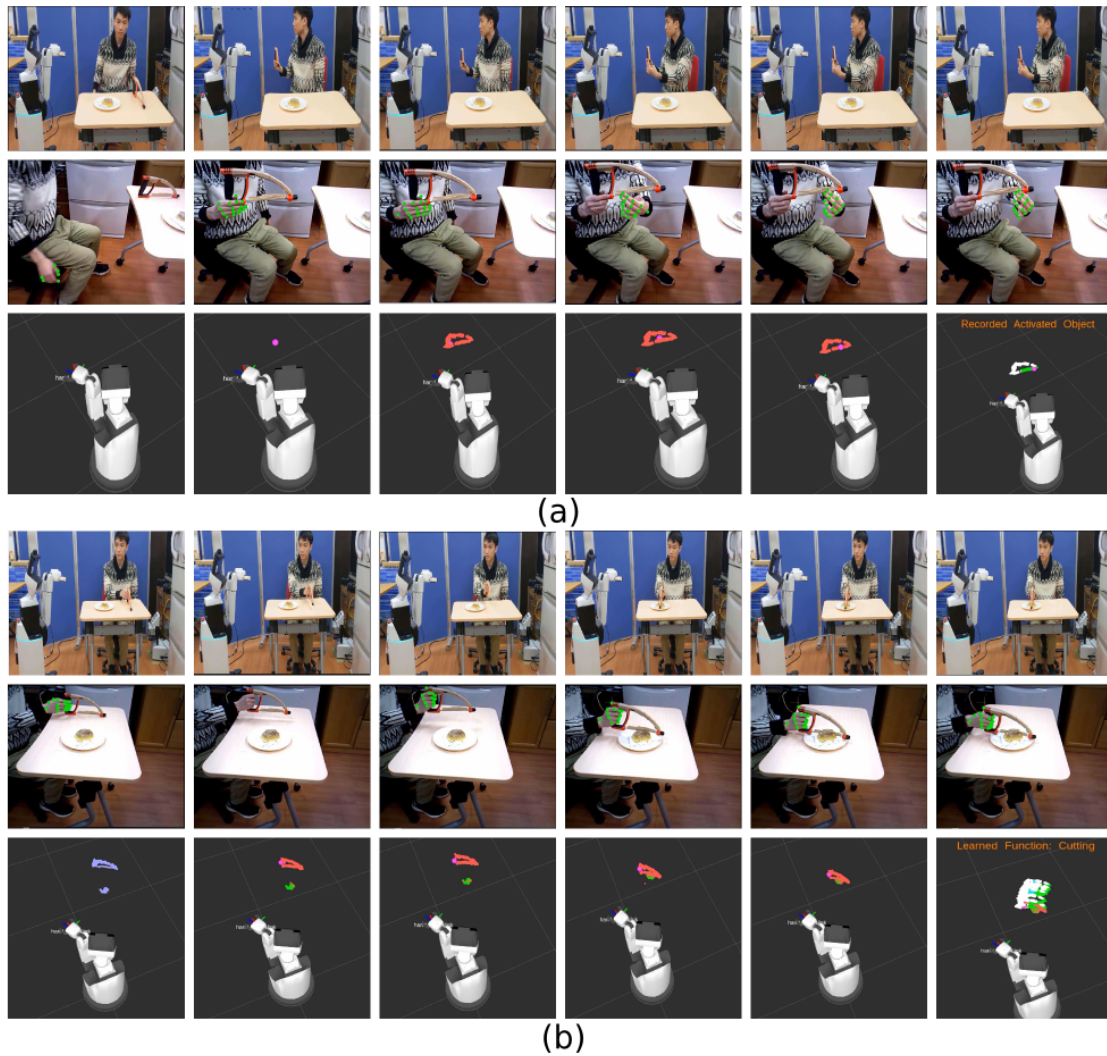


Figure 4.10: Example of the cutting demonstrations. (a) The demonstrator is teaching the robot which part of the knife is functional. (b) The demonstrator is teaching the robot how to use a knife to cut a pudding.

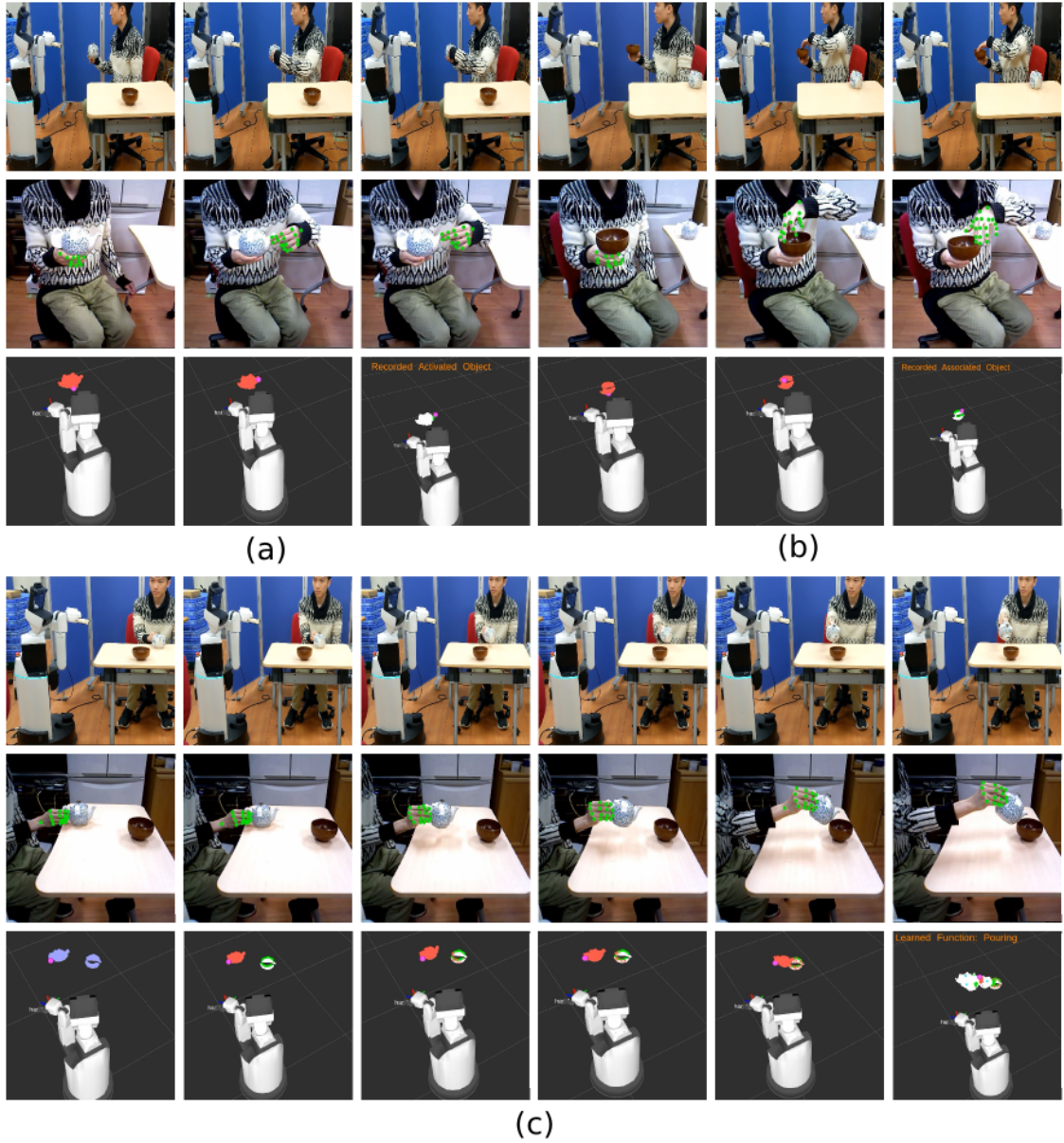


Figure 4.11: Example of the pouring demonstrations. (a) The demonstrator is teaching the robot which part of the teapot is functional. (b) The demonstrator is teaching the robot which part of the bowl is functional. (c) The demonstrator is teaching the robot how to perform the pouring task.

pudding during trajectory demonstration and recorded the trajectory of the knife to learn the cutting function.

In the pouring demonstration, the demonstrator teaches the robot how to perform the pouring function using a teapot and a bowl. The process is shown in Figure 4.11. Different from the cleaning and cutting demonstration, in the pouring function, both the activated object and the associated object have the function part, therefore the demonstrator had to show the robot the function part of the teapot and the function part of the bowl. After that, the demonstrator provided a trajectory to show how to pour from a teapot to a bowl. Finally, The system extracted the trajectory and computed the corresponding key object states.

## 4.4 Virtual Teaching

The previous section has already shown how to collect samples from on-site demonstrations. However, the on-site demonstration tool requires users to prepare relevant objects in the real world. Furthermore, the on-site demonstration tool has difficulty in estimating object grasping pose due to keypoint detection failures. As a supplement, this thesis also presents a virtual teaching method that allows users to teach robots without using real objects.

### 4.4.1 Synthetic Object Point Cloud Generation

As 3D CAD models are widely used in digital films and games, nowadays 3D CAD models can be easily obtained from the Internet. For example, ModelNet [155] is one of the large-scale 3D CAD model datasets which contains 127,915 CAD models within 662 object categories, its subset, the ModelNet40 is widely used for performance evaluation. There are also other well-known datasets like ShapeNet [156], Berkeley 3-D Object Dataset [157], NYU depth v2 [158], SUN3D [159], etc. The ShapeNet contains roughly 3,000,000 models and 220,000 models are classified into 3,135 categories.

To obtain point clouds from objects, we can find some datasets such as the RGB-D object dataset [160]. we can also sample points from the CAD models mentioned above.



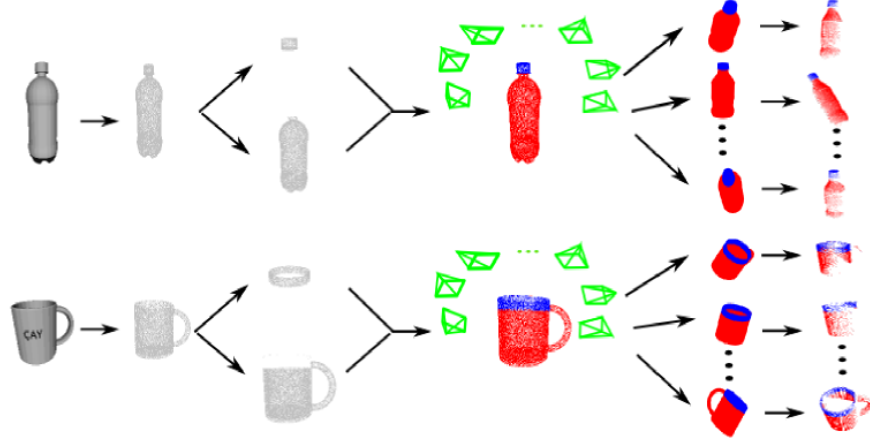


Figure 4.12: A demonstration of how to generate point cloud data from CAD models. Notice that CAD models on the left side are fully visible, while the synthetic point clouds on the right side are partly occluded.

However, it should be noted that the scanned point cloud which is captured from the real world are usually partly occluded, but the point clouds that we directly sample from the CAD models' surfaces are fully visible. If a learning algorithm is trained on the fully visible point clouds and implemented in detecting partly occluded point clouds, the model performance may become very poor. Therefore, we applied a point cloud processing pipeline to transfer fully visible point clouds into partly occluded point clouds with the following steps:

- 1) Sample 100,000 points uniformly from the model surfaces for every CAD model.
- 2) Scale the sampled point clouds to real object sizes.
- 3) Segment function parts of the point clouds and assign labels to every point.
- 4) Put virtual cameras on various viewpoints to generate partly occluded point clouds from a fully visible point cloud using the z-buffer algorithm [161] from the OpenGL library [162]. The z-buffer algorithm removes the invisible points by comparing the depths at each pixel position on the projection plane.

5) Generate partly occluded point cloud samples in the desired view points.

A demonstration is shown in Figure 4.12, notice that the invisible points are removed.

#### 4.4.2 Graphical User Interface Annotation Tool for Virtual Teaching

Data annotation is a time-consuming process, an easy to use annotation tool will save us much time. In the image processing field, there are some powerful annotation tools such as LabelMe [163] and VoTT [164] that help users attach labels to an image efficiently. When it comes to point clouds, there are not any publicly available annotation tools that meet our needs, therefore, we decided to create one for ourselves.

The virtual teaching tool is driven by the QT library and the Librviz library [165]. The former is a well-known widget toolkit for building graphical user interfaces, while the latter provides an interactive marker function that allows us to add 6D labels in 3D space. The interface of our annotation tool is shown in Figure 4.13, it contains a display/interactive zone and a function panel.

In the display/interactive zone, the desired point cloud data is shown, also, a user can change the viewpoint through the zoom-in/out operation, the rotation operation, and the translation operation.

The function panel has two areas. The first area has a series of data processing, including cropping Region of Interest (ROI) from the background, downsampling, and removing outlier points. When processing real-world point cloud frames, we do not want background points because these points contribute little to the system performance but consume many computing resources. A simple and straightforward way to solve this problem is to create a bounding box by setting x-y-z coordinates to get interest points from the point cloud frames. Similarly, downsampling reduces the number of points thus provide convenience to the following process.

The second area is related to labels. We adopted the interactive marker from Librviz to represent the 6D manipulation pose for visualization and operation. By clicking the “add marker” button or the “remove marker” button, a user can add or remove a marker in the display/interactive zone. The user can also adjust the scale of the marker by setting

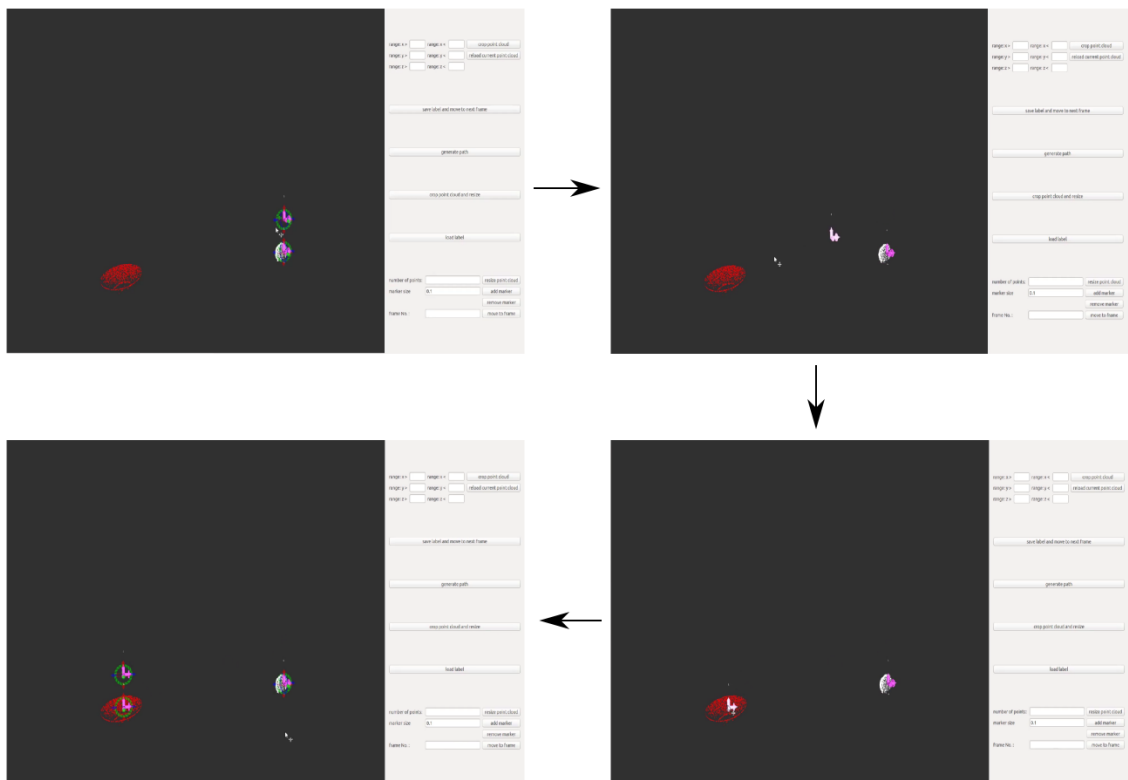


Figure 4.13: An example of using the virtual teaching tool to add demonstration samples of placing an apple (colored in white) on the plate (colored in red).

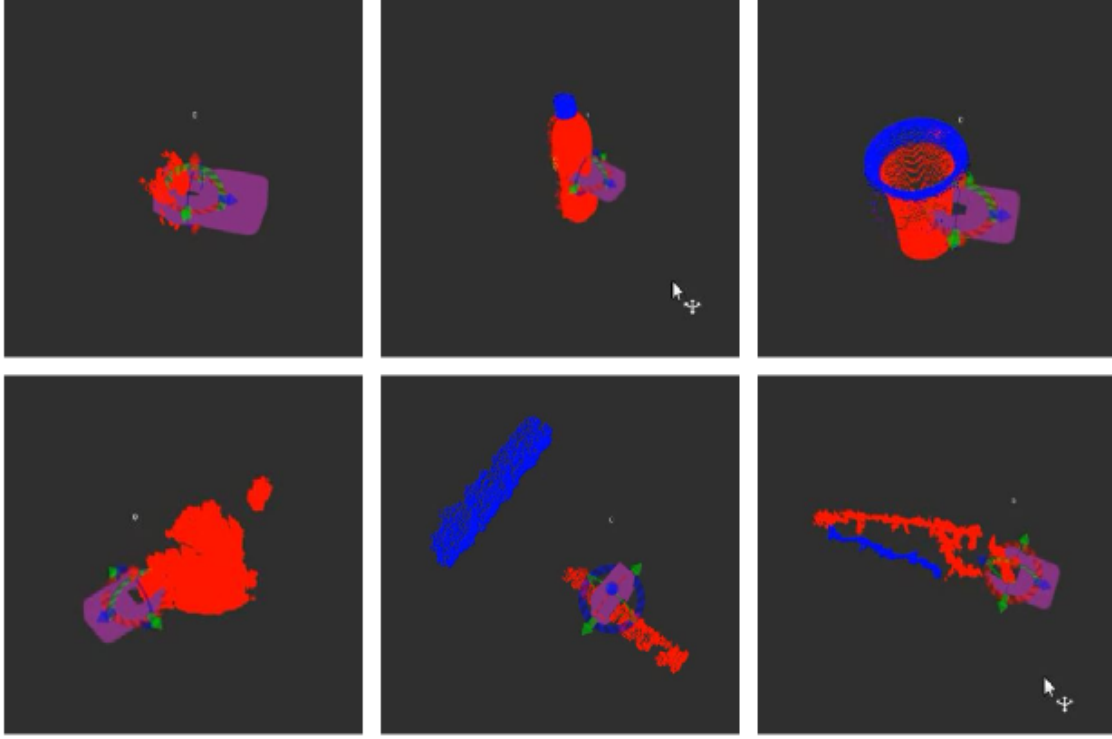


Figure 4.14: An example of using the virtual teaching tool to teach the robot how to grasp different objects.

a scale factor. The position and orientation of the marker can be changed by dragging the arrows or circles.

This tool allows users to add 6D pose labels for grasping pose annotation and key states annotation, for function part segmentation, we can use some point cloud segmentation tools like CloudCompare [166].

The virtual teaching tool can also be used to annotate object grasping poses, some examples are shown in Figure 4.14. When teaching the grasping pose of an object, users only need to drag the gripper marker (the purple marker) to the desired location then adjust the marker's pose. The examples shown in Figure 4.14 includes the grasping pose of an apple, a bottle, a cup, a teapot, a sticky roller, and a saw (from left to right, up to down).

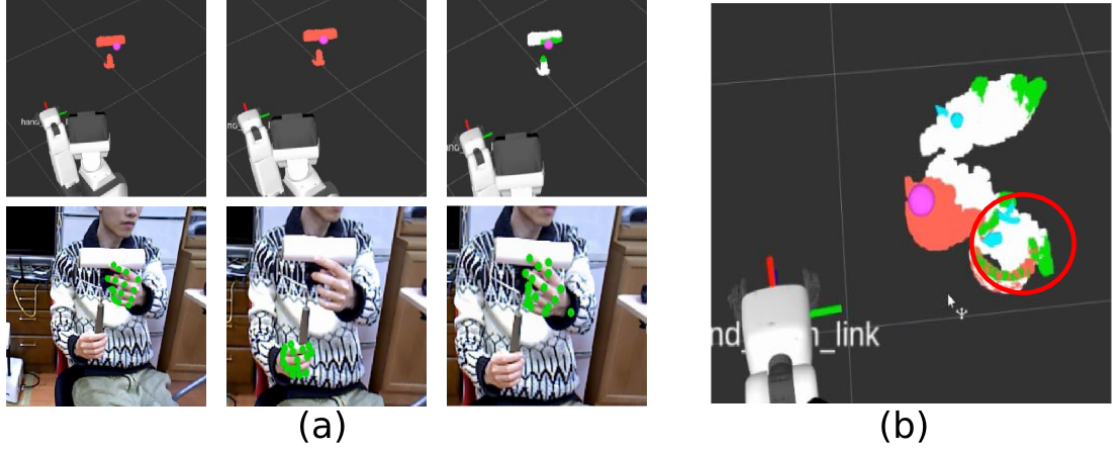


Figure 4.15: An example of the failure cases in the function part extraction (a) and trajectory extraction process (b).

## 4.5 Discussion

During the on-site teaching, failures may occur in the function extraction process and the trajectory extraction process, an example is shown in Figure 4.15. In Figure 4.15 (a), the hand keypoint detector lost the keypoints of the demonstration hand during the function part extraction process, since the results are shown on the screen, the demonstrator was waiting for the hand keypoint detector to get the hand keypoint back, after 5 seconds, although the detector has detected the demonstration hand keypoint again, the system considered the demonstration is finished because the position of the demonstrator's index finger did not move within 5 seconds. In Figure 4.15 (b), the ICP algorithm fails to calculate a correct transformation for the last pose (circled in red), and this alignment error led to failure. In these cases, users can remove the fail samples and do the demonstration again. Also, using the virtual teaching tool does not have these problems because users can select the desired function area by clicking the mouse and putting key object states to the desired places. However, an on-site teaching tool is expected to provide a better user experience, and thus we will improve this component in future work.

## 4.6 Summary

This chapter first introduced the concept of transparent teaching, which is able to output explicit feedback so that users know what has been learned by the robot. To extract manipulation knowledge from visual demonstration, this chapter presents an on-site teaching approach, which has three main components, the handheld object extraction components can extract the object held by human hand. The object function part extraction component can extract the object's function part. The manipulation trajectory extraction component allows the robot to extract a 6D pose trajectory of the grasped object. These variables are necessary for the Object-pair Mutual Function Knowledge model for manipulation task learning and reproducing. This chapter also presents a virtual teaching approach that allows users to provide visual demonstration samples from a user interface. Demonstration samples from this chapter will be used in the following chapter for learning and reproducing.

## 第5章

# **Learning Manipulation Knowledge through a Few Visual Demonstrations**





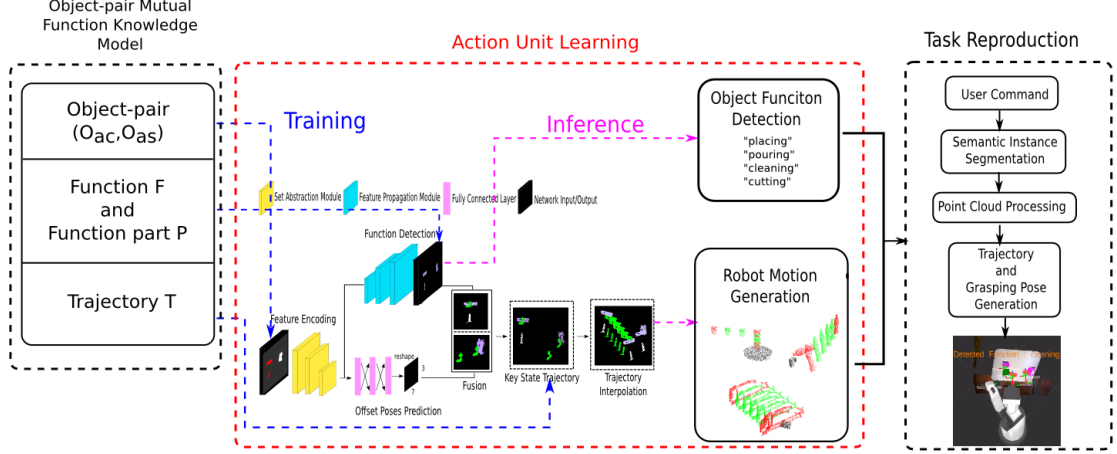


Figure 5.1: The position of the learning component in the system.

Chapter 4 presented an on-site teaching approach and a virtual teaching approach that allow the system to obtain variables for the Object-pair Mutual Function Knowledge model proposed in Chapter 3. Given the visual demonstration samples, the next and most important step is how to learn manipulation knowledge from them (Figure 5.1). Generally, giving more demonstrations for training can improve the system robustness against variations, but requiring more demonstration samples will increase the user burden. To solve this problem, this chapter introduces a framework that enables robots to learn manipulation knowledge from a few visual demonstrations.

## 5.1 Manipulation Learning through Imitation

Previous imitation learning methods require off-line programming [39] or creating 3D geometrical models for the manipulation objects and thus these methods are not suitable to be integrated into a robot teaching system. On the other hand, reinforcement learning based methods [53, 52] take RGB image frame sequences as inputs, which is difficult for learning due to two reasons: the first reason is the influence brought by the background. When performing manipulation tasks, only a small part of pixels contains information about the target objects, while the rest of the pixels are meaningless. The second reason is

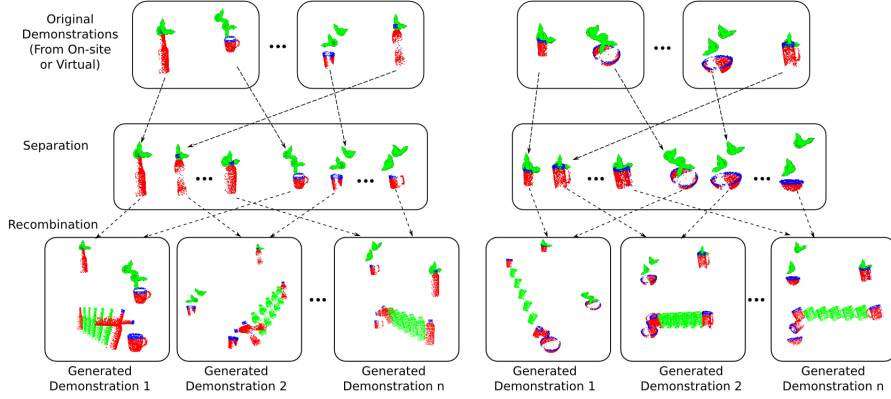


Figure 5.2: A demonstration of the data auto-augmentation component.

the object positions in the image coordinate are absolute, and thus the learning algorithm requires a large number of demonstrations to handle the position variation in a single manipulation task. Furthermore, traditional data augmentation techniques can not change the target object’s position in images, therefore, the users must manually change the target objects’ relative positions in every demonstration, which brings a large burden for the user.

Instead of directly imitating the demonstrator’s action, the proposed method achieves imitation by reproducing the target object’s state trajectory. However, if we simply use the whole demonstration trajectory for task reproduction, the object position changes will lead to failure. In the following section, we present a data processing and augmentation framework to handle the position variation problem.

## 5.2 Demonstration Data Processing and Separation Augmentation

Through previous steps, the system has obtained one or a few demonstration samples for a certain manipulation task. However, if a system performs a task by simply following the demonstration trajectory, it will fail to handle position variations. Instead of asking the user to provide more demonstration samples, an automatic data augmentation process is more preferred.

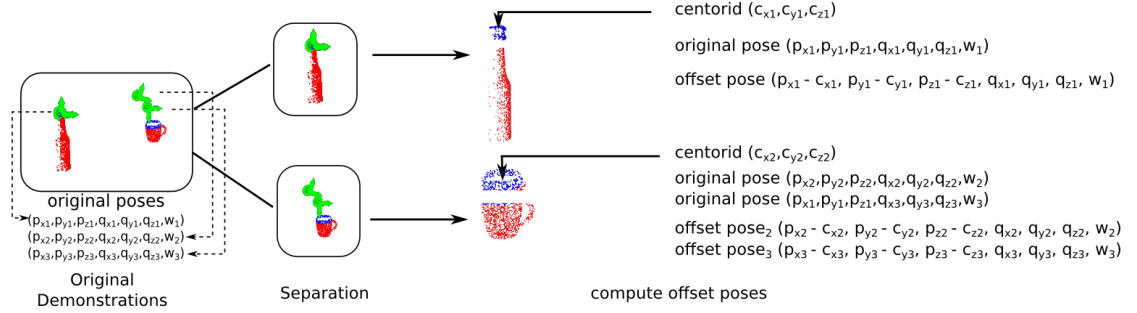


Figure 5.3: Illustration of the pose trajectory assignment and separation process.

Data Augmentation can be used to reduce user effort when collecting data, and thus widely used in computer vision [167] and speech recognition field [168], the data augmentation methods for RGB images include flipping, cropping, rotation, translation, random erasing [169], creating synthetic data [170], and so on. When it comes to robot manipulation teaching, we can hardly find existing approaches for data augmentation. Since the idea of data augmentation is to increase the number of samples by reusing and modifying some variables of the existing data. An overview of the proposed data augmentation framework is shown in Figure 5.2.

### 5.2.1 6D Pose Trajectory Separation and Assignment

As the point clouds and state trajectories acquired by demonstrations are presented in absolute coordinates, the first step is to convert the absolute coordinates to relative coordinates [45, 171]. Concretely, the system computes the centroid of the function part of the object being grasped and assigns the first offset pose to this object, similarly, the rest two offset poses are assigned to the other object. the offset poses are calculated by subtracting the centroids of the function parts of the corresponding objects from the positions of the corresponding poses. An illustration is shown in Figure 5.3, in this example, the original demonstration sample is pouring from a bottle to a cup, so the first step is assigning the first pose to the bottle and the rest two poses to the cup, then separation the object-pair into two parts and compute their corresponding offset poses.

### 5.2.2 Object-pair Recombination

In the recombination stage, according to the function, the system randomly select objects to make up object-pairs (if the selectable object in a category is more than one), meanwhile, the system assigns random translation to each of the object point clouds. This step allows the system to generate new demonstrations with different object combinations and relative positions. The range of the random factor in the x-y-z axes are  $[0.2, 1.0]$ ,  $[-0.7, 0.7]$ ,  $[0, 1.0]$  separately, this range is selected to match the robot's operation space.

Also, when adding a new object category for a known function, users can reuse a part of the trajectory. For example, the robot has ready known how to pour from a bottle to a cup, when the users want to pour from a teapot to a cup, the only thing they need to do is adding a start state to the mouth part of the teapot, while the rest states can be reused.

## 5.3 Manipulation Knowledge Learning on Augmented Visual Demonstration Samples

Given the augmented data, a learning algorithm needs to solve two main problems: (1) according to the object-pair input and the activation signal, the learning algorithm must predict the function and the function part of the input object-pair. (2) The learning algorithm must generate a suitable offset pose trajectory to achieve the function while handling the position variation. To solve these problems, the following section presents an Object-pair Mutual Function Knowledge Acquisition Network.

## 5.4 Object-pair Mutual Function Knowledge Acquisition Network

The Object-pair Mutual Function Knowledge Acquisition Network is based on the encoder-decoder architecture (Figure 5.4), the encoder network has three set abstraction modules [172, 173] to extract point cloud features. In the first set abstraction module, the number of points in the sampling layer is set to 256, the number of neighboring points

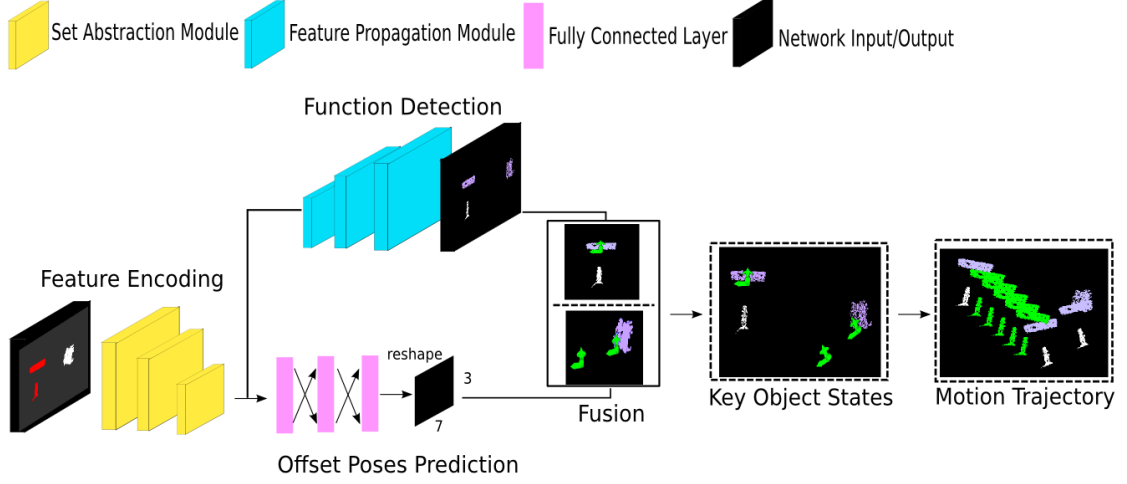


Figure 5.4: Illustration of the Object-pair Mutual Function Knowledge Acquisition Network.

in the grouping layer (nsample) is set to 32 with the radius of 0.02, the numbers of neurons of multi-layer perceptron (MLP) network inside the PointNet layer are set to 16, 16, 32 separately. The parameters of the second set abstraction module are (npoint = 128, nsample = 64, radius = 0.05, MLP = [32, 32, 64]), and the parameters of the third set abstraction module are (npoint = None, nsample = None, radius = None, MLP = [64, 128, 256]).

Similar to the convolutional layer’s mechanism, the set abstraction module is used to encode input point clouds into higher-level features. It takes  $N \times (d + c)$  matrixes (point clouds) as inputs, where  $N$  is the number of points,  $d$  represents the coordinate dimensions and  $c$  represents the extra point features such as colors or surface normals. Its outputs have the shape of  $\bar{N} \times (d + \bar{c})$ , where  $\bar{N}$  is the number of subsampled points and  $\bar{c}$  represents encoded feature vectors. Generally, a set abstraction module is composed of a Sampling layer, a Grouping layer, and a Pointnet layer.

- **Sampling layer.** The sampling layer is designed to gather a subset from an input point cloud with the iterative farthest point sampling algorithm. The main idea of the iterative farthest point sampling algorithm is described as follows:

---

**Sampling  $k$  points from a point set  $P \in (P_0, P_1, \dots, P_n)$ ,  $k < n$ . (python code can be found in Appendix A)**

- 1) Randomly pick a start point  $P_{s0}$  from the point set and put it into the sampled points list, computer the distances between the start point and the remaining points.
- 2) Find the point  $P_{s1}$ , which has the farthest distance to  $P_{s0}$ , put  $P_{s1}$  into the sampled points list. For the remaining points, computer their distances to  $P_{s0}$  and  $P_{s1}$  separately and find the farthest point to  $P_{s0}$  and  $P_{s1}$ .
- 3) Repeat step 2) until all  $k$  points are found.

The subset generated the iterative farthest point sampling coverage of the entire point set better than random sampling. The elements within the subset are treated as centroids of local regions.

- Grouping layer. The function of the grouping layer is constructing local region sets by searching  $K$  neighboring points around the centroids which are sampled by the sampling layer.
- PointNet layer. Given the centroids and their corresponding local regions, the PointNet layer is able to learn features around the centroids and integrate them into higher-level features.

The decoder network consists of two branches: a function detection branch and an offset pose prediction branch.

### 5.4.1 Function Detection Branch

The function detection branch (Figure 5.5) is composed of three feature propagation modules. The feature propagation module interpolates values via average weights from  $k$  nearest neighbors and adding skip links from the lower level set abstraction module.

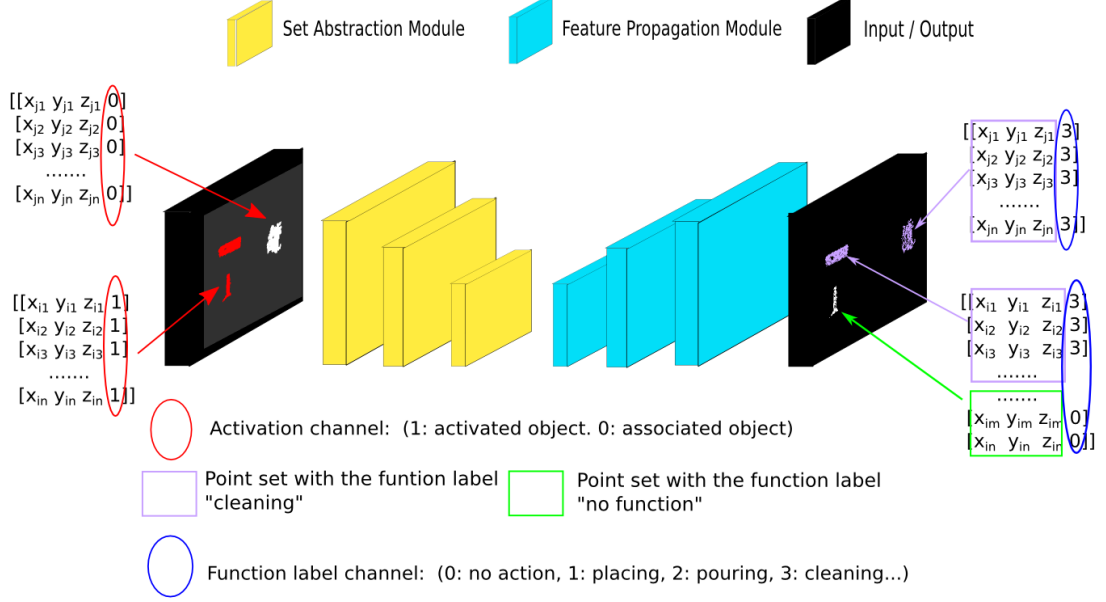


Figure 5.5: Illustration of the function detection branch.

The parameters of the three feature propagation modules are (MLP1 = [64, 64], MLP2 = [64, 32], MLP3 = [32, 32, 32]). After the feature propagation module is a 1D convolutional layer which has 128 neurons, following by a dropout layer with a dropout rate of 0.5, the last layer is another 1D convolutional layer that outputs a tensor with the shape (2400, n). Where 2400 is the number of points and n is the number of classes. In the inference phase, we can use the Argmax function in the Numpy library to get each point's label, then concatenate the x-y-z coordinate with the label to form a 4-channel matrix.

### 5.4.2 Offset Pose Prediction Branch

The offset pose prediction branch (Figure 5.6) consists of three fully connected layers, the numbers of neurons in these layers are [128, 64, 3x7], where 3 is the number of poses and 7 represents an offset pose (3 values for position and 4 values for orientation in quaternion form).

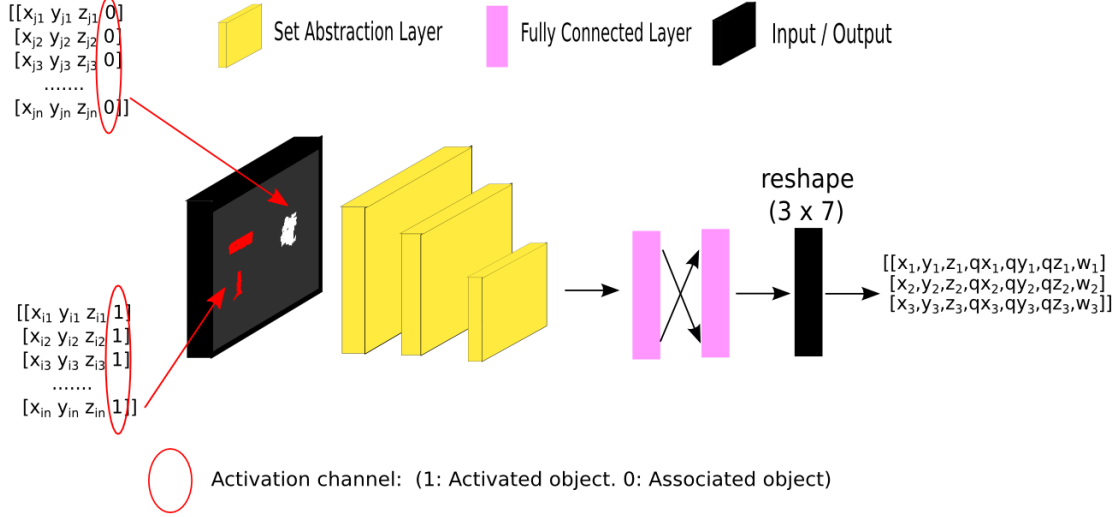


Figure 5.6: Illustration of the offset pose prediction branch.

### 5.4.3 Fusion

The data fusion module is responsible for recovering the offset states and deciding which object should be manipulated according to the affordance output. For instance, the scene in Figure 5.4 contains a sticky roller and a piece of dirty tissue, the output function in this example is “cleaning” because the sticky roller is activated, this means the robot can use the sticky roller to clean the dirty tissue. Therefore the module puts the start state on the sticky roller and puts the rest states on the dirty tissue, notice that the position in the start state is calculated by adding up the position of the first offset pose and the centroid of the head of the sticky roller, similarly, the second and the third states are computed by adding up the rest offset poses and the centroid of the dirty tissue, the equation can be written as follows:

$$P_{recover}(x, y, z, qx, qy, qz, w) = P_{offset}(x + c_x, y + c_y, z + c_z, qx, qy, qz, w) \quad (5.1)$$

$$C(x, y, z) = \frac{\sum_i^N Point_i(x, y, z)}{N} \quad (5.2)$$



Where  $N$  is the number of points in the corresponding point set,  $C(x, y, z)$  is the centroid of the point set.

#### 5.4.4 Grasping Pose Detection

The network also accepts one object as input (Figure 5.7), in this case, the input point cloud's activation channel is set to ones. The function detection output is dependent on the input object. For example, if the input object is a sticky roller, the output function part is the head of the sticky roller and the function is “cleaning”, if the input object is an apple in the placing task, there is no function part and the function is “no function”. In particular, the system only takes the first output of the offset pose prediction branch results. When recovering the grasping pose, the system will compute the centroid of the whole object point cloud and add up the position of the first offset pose and the point cloud centroid.

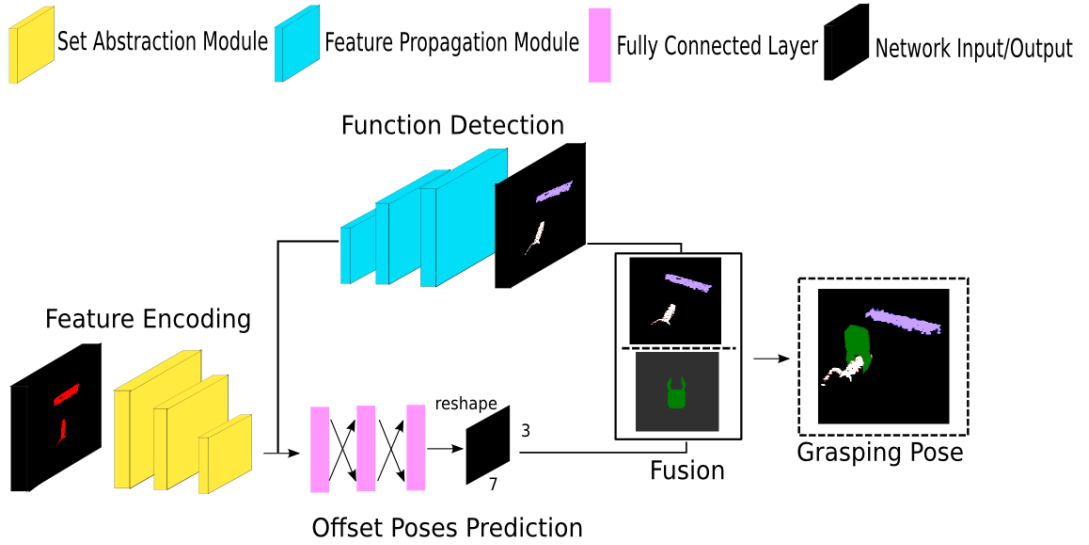


Figure 5.7: When the input only has one object, the network output is a grasping pose of the object.

### 5.4.5 Network Training

During the training phase, we apply the sparse softmax cross entropy (given by 5.3 and 5.4) to measure the classification loss

$$\text{softmax\_logits} = \frac{e^{z_i}}{\sum_k e^{z_k}} \quad (5.3)$$

$$\mathcal{L} = - \sum_k y * \log(\text{softmax\_logits}) \quad (5.4)$$

Where  $z_i$  is the  $i^{th}$  element in the logits, the number of elements in the logits is  $k$ , and  $y$  is the ground truth label.

While the regression loss is measured by the mean squared error:

$$\mathcal{L}_{MSE} = \frac{\sum_{i=1}^3 \sum_{j=1}^7 (\bar{y}_{i,j} - y_{i,j})^2}{3 \times 7} \quad (5.5)$$

Where  $\bar{y}_{i,j}$  is the predicted result and  $y_{i,j}$  is the ground truth label.

The total loss is given by  $0.1 * \text{classification loss} + 0.9 * \text{regression loss}$  because the classification loss is much larger than the regression loss. The training label for offset pose is computed by subtracting the corresponding part centroids from the state labels, and the training label for function detection is a (1, 2400) tensor which indicates each point's class. The number of epochs is set to 200. In each epoch, we randomly shuffle the training set then loop over all the training data with the batch size of 2. We use Adam optimizer with an initial learning rate of 0.0001 and decay exponentially by a factor of 0.7.

## 5.5 Experiments

To evaluate the network performance, we set up experiments to find out how many augmented samples are needed to learn the correspondence of the offset poses to the position variation and the correspondence of the offset poses to different functions.







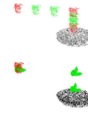

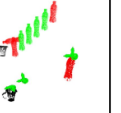
Demonstration objects	Tasks	Number of original demonstrations	Number of augmented demonstrations
 Apple  Plate  Bottle  Cup	Grasping	2  	200
	Placing	1 	100
	Pouring	2  	200
Total demonstrations		5	500

Table 5.1: The demonstration data used in the experiments. The augmented demonstration is generated by the virtual teaching approach described in Chapter 4.



(a)



(b)

Figure 5.8: The test objects used in the experiments. (a) five cups and five bottles which are used in the pouring tasks. (b) a toy apple, a bowl, and two plates used in the placing tasks.

### 5.5.1 Demonstration Data

The demonstration objects are shown in Table 5.1, including the point clouds of an apple, a plate, a bottle, and a cup, where the point clouds of the apple and the plate are given by the RGB-D Object dataset [160], the point clouds of the bottle and cup are generated from CAD models given by the ModelNet40 [155] dataset. The number of demonstrations provided by the user is five, includes grasping an apple, grasping a bottle, placing the apple on the plate, and two pouring water from the bottle to the cup from different directions. Each of the demonstrations is augmented to the number of 100, and the total number of augmented demonstrations is 500.

### 5.5.2 Test Objects

The test objects used in the placing task and the pouring task are shown in Figure 5.8. In the placing task, we prepared a toy apple, two different plates, and a bowl. In the pouring task, we prepared five bottles and five cups.

### 5.5.3 Evaluating the Number of Augmented Demonstrations Needed for Handling Position Variation

In the learning phase, we gradually increased the number of demonstrations to evaluate the number of augmented samples that needed for the Object-pair Mutual Function Knowledge Acquisition network against position variation, the original demonstrations include one grasping demonstration and two pouring demonstrations (one for pouring from the left-hand side and one for pouring from the right-hand side).

From Figure 5.9 we can see that using only one demonstration for training, the network always selects one direction in the pouring tasks, also, the generated pose trajectories are unusable due to collision (The transformed bottle point clouds (colored in red) overlap with the cup point clouds). When the number of augmented demonstrations is increased to five samples for every original demonstration, the network outputs a feasible pose trajectory for one side, but still fails on the other side. As the number of augmented demon-

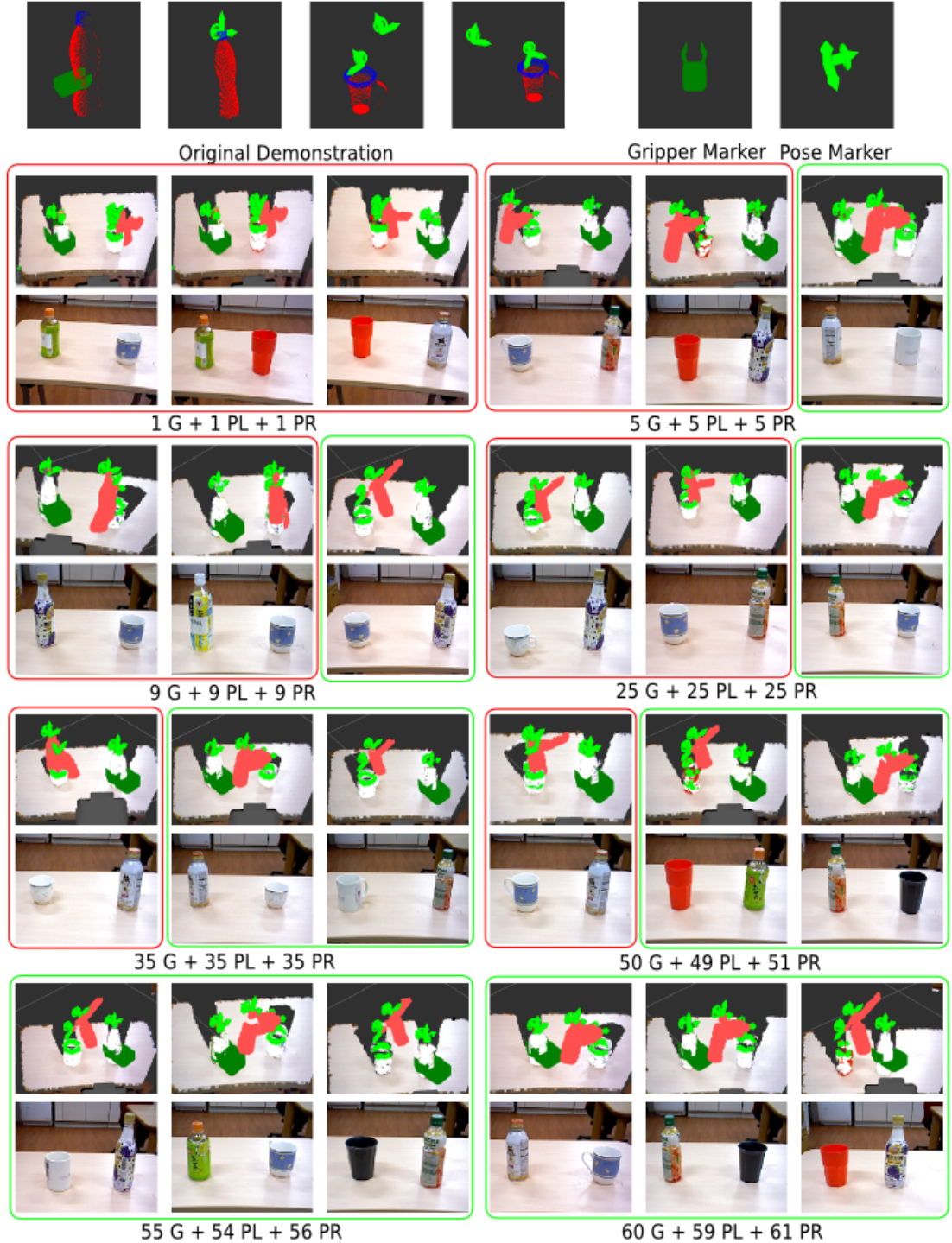


Figure 5.9: The number of augmented demonstration needed for handling position variation in a single task. The generated results within the red rectangle are infeasible while the generated results within the green rectangle are feasible. G means the number of augmented demonstration for grasping, PL means the number of augmented demonstration for pouring from the left-hand side, and PR means the number of augmented demonstration for pouring from the right-hand side.

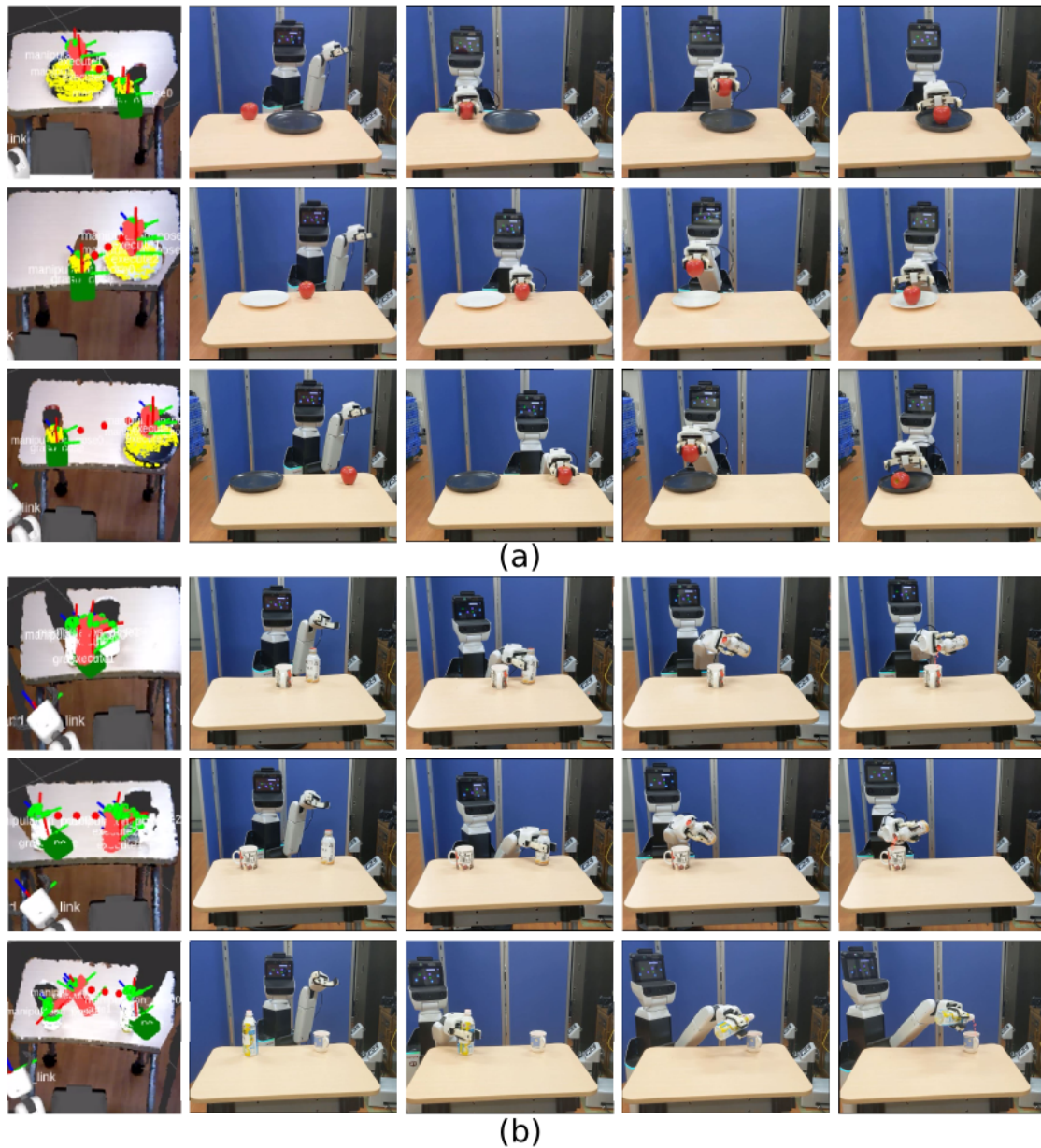


Figure 5.10: System outputs and robot execution of the placing tasks (a) and the pouring tasks (b). In each of the tasks, the objects are randomly placed. The first column shows the system outputs, where the yellow points have the label “placing”, the green points have the label “pouring”, and the white points have the label “no function”.



strations is increased gradually, from nine to fifty for each demonstration, we can see that the network is adapting to the two different pouring sides and the results are getting better. Finally, when the number of augmented demonstrations is larger than around fifty-five, the network has successfully learned to handle the position variation by selecting feasible pose trajectories.

#### 5.5.4 Test the Trained Network on New Instances with Different positions

Next, we use all of the 500 augmented demonstrations (includes grasping, placing, and pouring) to train the network and test the system performance in the real world environment. The results are shown in Figure 5.10. In the placing tasks (Figure 5.10 (a)), the toy apple and the plate were placed randomly, the system has successfully distinguished the placing function on the point clouds (the yellow points) and generated proper key object state trajectories to guide the robot to perform the placing tasks. In the pouring task (Figure 5.10 (b)), although the bottles and cups are placed in random positions and directions, the system has successfully distinguished the pouring function on the opening part of the bottles and the cups (the green points), then generated proper key object state trajectories to guide the robot to perform the pouring tasks.

From the results, we can see that augmented each demonstration to the number of 100 is enough for the proposed network to learn the correspondence of the offset poses to the function parts and the correspondence of the offset poses to the position variations. On the other hand, increasing the number of augmented demonstrations will not affect the network performance, but more demonstrations usually need more time for training the deep neural network.

#### 5.5.5 Network Training Time Evaluation

We then estimated the relationship between the number of demonstrations and the time needed for training the network until convergence. In particular, The network is trained on an NVIDIA GTX 1080Ti GPU. Since the pouring task requires at least 55 augmented

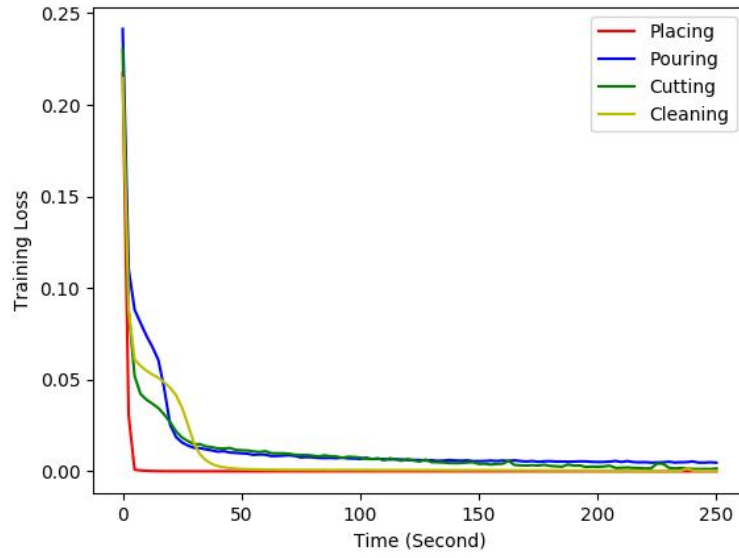


Figure 5.11: Network training time for a single task, each task has 200 augmented demonstrations.

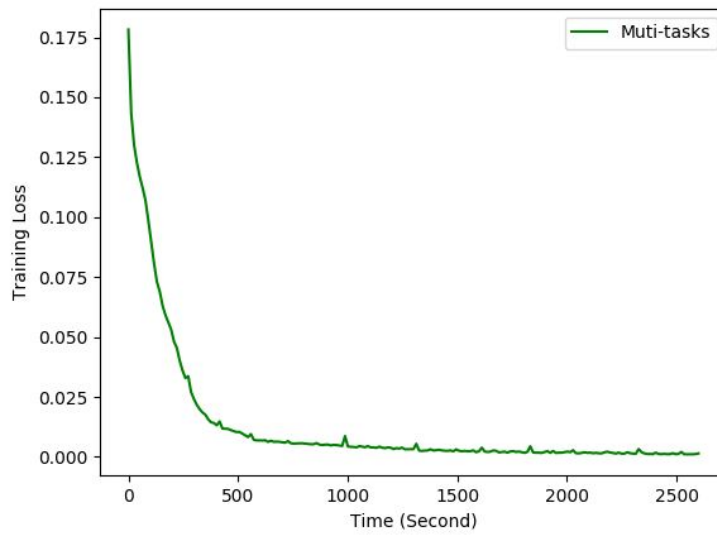


Figure 5.12: Network training time for multiple tasks (grasping, placing, pouring, cutting, and cleaning), the number of augmented demonstrations is 1,100.



demonstrations for handling position variation, we augmented the demonstrations to the number of 100 in each direction, and thus the total number of demonstrations in the pouring task is 200, to evaluate the training time for other tasks, we also provide the demonstrations of the placing task (place an apple to the plate), the cutting task (cut a pudding with a knife), the cleaning task (clean a piece of dirty tissue with a sticky roller), and augmented these demonstrations to the number of 200. From Figure 5.11 we can see that it takes no more than 250 seconds for training the network until convergence.

Finally, we prepared 11 demonstrations using the on-site or virtual teaching for the training time estimation, each of the demonstrations is augmented to the number of 100 (the total number of augmented demonstrations is 1,100). For convenience sake, here use the sign (placing, apple, plate) to represent the task “ placing an apple to the plate. The demonstrations include (grasping, apple), (grasping, bottle), (grasping cup), (grasping, teapot), (grasping, knife), (grasping, sticky roller), (placing, apple, plate), (pouring, bottle, cup) (left and right), (cutting, pudding, knife), (cleaning, dirty tissue, sticky roller), where the grasping, placing, and pouring demonstrations are given by virtual teaching, the cutting and cleaning demonstrations are given by on-site teaching. The result is shown in Figure 5.12, we can see that it takes around 2,000 seconds for training the network until convergence.

## 5.6 Summary

This chapter introduced a data augmentation framework for obtaining synthetic demonstration samples from the original samples. Furthermore, an Object-pair Mutual Function Knowledge Acquisition Network is presented to learn manipulation knowledge from the augmented samples. From the experiment results, we can see that given only one demonstration sample for different pouring directions, the network has difficulty in handling the object’s position changes. This problem can be solved by augmenting the samples to the number of around 55 for each direction with different object relative positions because it forces the network to notice the position changes of the objects. After that, the relationship between training samples and training time is estimated, it takes more epochs for the

network to learn the position change in the pouring task. It takes around 15 minutes for the GTX 1080Ti GPU to train on 800 demonstration samples to convergence. The trained model in this chapter will be used in the next chapter for manipulation task reproduction.

## **第6章**

# **A Framework for Manipulation Task Reproduction**



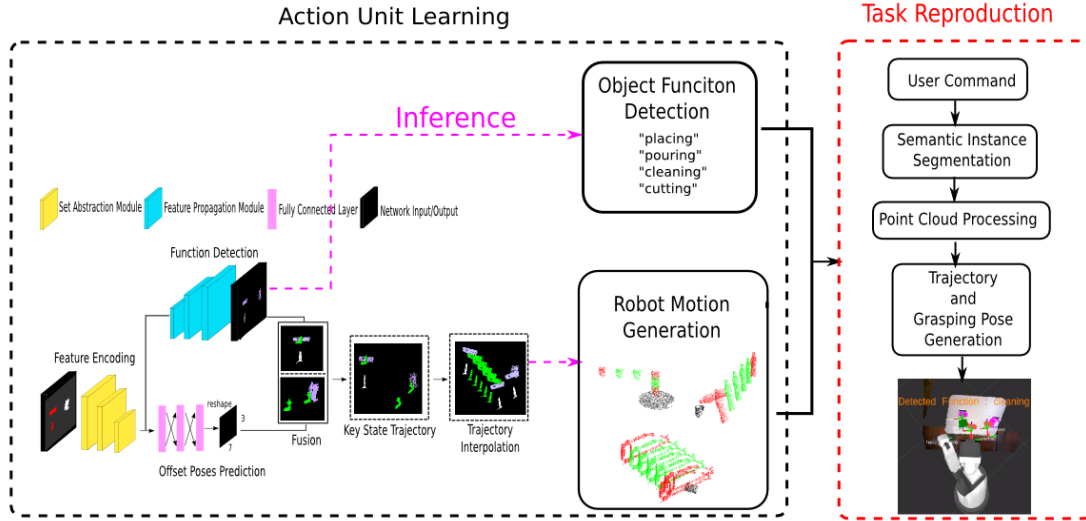


Figure 6.1: The position of the task reproduction component in the system.

Chapter 5 presents a data augmentation framework and an Object-pair Mutual Function Knowledge Acquisition Network that allow the system to learn manipulation knowledge with a few visual demonstrations given by human teaching. The trained network is then be used in the manipulation task reproduction phase (Figure 6.1). Given a user command, the task reproduction component needs to solve the problem of how to guide a robot to accomplish the corresponding manipulation task. Unlike existing methods that need to search specified instances in the environment, the proposed system is expected to find out suitable instances to perform the desired tasks. To achieve this goal, this chapter presents a reproduction framework that enables the system to detect different instances in the same category that can be used to perform the desired task, after that, it generates the corresponding point clouds as the input of the Object-pair Mutual Function Knowledge Acquisition Network. Finally, the framework generates the manipulation motions to guide a robot to accomplish manipulation tasks.

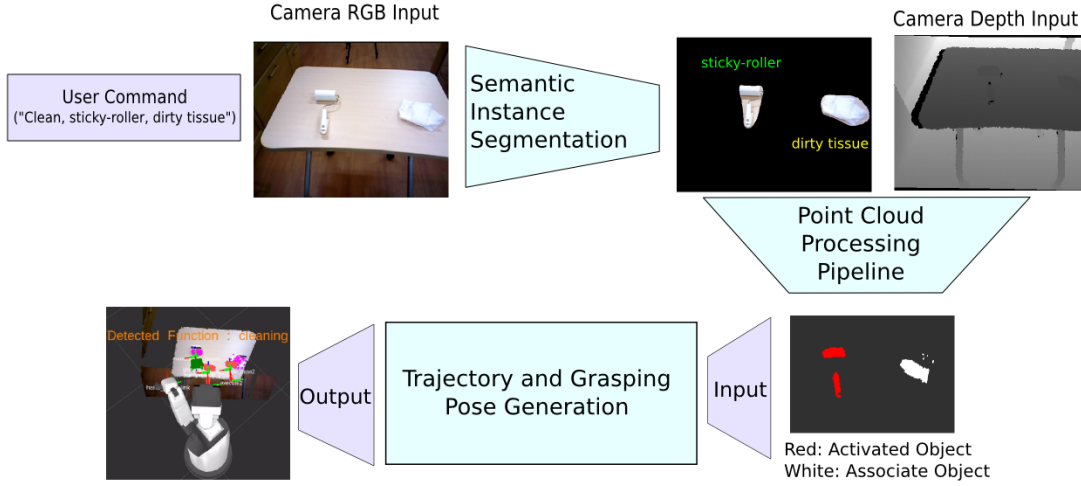


Figure 6.2: An overview of the task reproduction framework.

## 6.1 Instance Matching Based Task Reproduction

Most of the existing manipulation task reproduction framework are instance matching based methods, that is, the robot only knows how to use specified instances given in demonstrations, when the manipulation instances are changes, even though the target instance and the demonstration instance are in the same category, these methods require extra demonstrations or object information to accomplish the task.

For example, the method presented in [72] enables a robot to learn manipulation skills from one demonstration, they achieve this by applying the DAML[174] algorithm and meta-learned initial parameters which are trained with a large number of demonstration data. The results show that their method achieves success in handling position variation. However, it can only handle some simple tasks such as pushing, pick and place, furthermore, the manipulated objects must appear in the demonstration. On the other hand, manipulation approaches presented in [55, 175, 176, 62, 177] requires objects' 3D geometrical model or templates for 6D pose estimation and manipulation, these methods tend to fail when the target instances have different sizes or shapes to the templates.

## 6.2 Object Function Knowledge-Based Task Reproduction

Since the goal of this thesis is to reduce the user effort in robot teaching, recognizing the object function part enables the robot to apply the learned manipulation skill to as many instances as possible.

### 6.2.1 Overview of the Framework

The task reproduction framework is shown in Figure 6.2, the input of the framework is a user command that contains the desired function, and the target object-pair. According to the user command, the semantic instance segmentation component finds out the target objects from the environment and passes the mask of the target objects to a point cloud processor. Next, the point cloud processor convert computes the point clouds of the target objects and assign the activation channels to the activated object, then passes the result point clouds to the manipulation action generation module. Finally, the manipulation action generation module detects whether the target objects can be used to achieve the desired function, if nothing goes wrong, it provides a motion trajectory for execution.

### 6.2.2 Semantic Instance Segmentation

The semantic instance segmentation component allows robots to find out instances within the known category in the environment. Also, utilizing the semantic instance segmentation component for object detection can avoid the faulty generalization problem described in Chapter 3, since the system inputs are user commands and the semantic instance segmentation component only detects desired object categories in the environment. An example is shown in Figure 6.3, instead of detecting a specified bottle for the pouring task, this component allows the system to find out all bottles that can be used for the task, and thus can meet our need. The semantic instance segmentation component we used in this study is a Mask R-CNN that trained on the corresponding object categories.



Figure 6.3: Finding out all bottle instances using the trained Mask-RCNN [178].

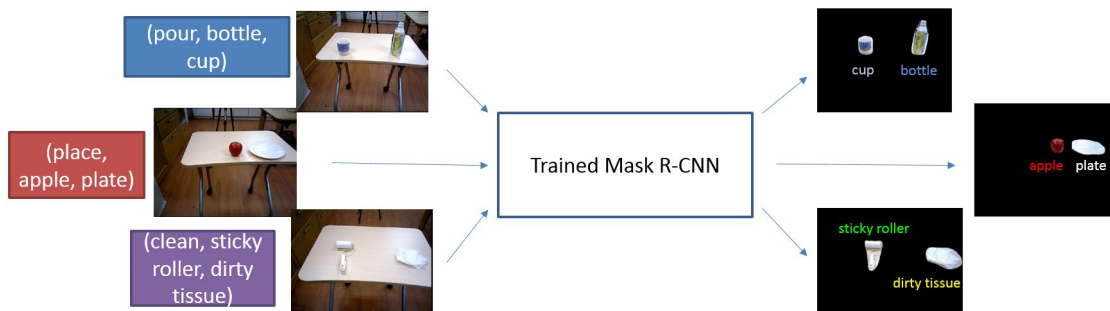


Figure 6.4: An illustration of the Mask R-CNN [178] outputs given different user commands.



Mask R-CNN [178] is one of the most successful deep learning architectures in the instance segmentation field. A standard Mask-RCNN architecture includes two components: The first component is a Region Proposal Network (RPN) [179], which produces candidate object bounding boxes effectively in terms of sharing features with the detection network; the second component an RoI Align layer which improves mask accuracy by aligning the extracted features properly. In this thesis, we use an existing Mask-RCNN model which is trained on the COCO dataset [51] with Inception V2 [180] backbone. The output masks of the model are binary images with a fixed size of 15 x 15 and the Non-maximum suppression threshold of 0.3, then the masks will be resized with their corresponding box offset values.

When adding new object categories to a trained Mask-RCNN model, users may need to provide a small number of images to fine-tune the trained Mask R-CNN model. Generally, this process will not take much time. In this thesis, we assume the Mask-RCNN model has already trained on the corresponding dataset and has the capability to detect the target objects.

Given the user commands and RGB images captured from the robot's vision sensor. The system calls the Mask R-CNN to obtain the objects' categories and their corresponding masks (see Figure 6.4). The desired object masks will be passed to the point cloud processing component for further processing to obtain the input point clouds with activation signals.

### 6.2.3 Point Cloud Processing Pipeline

The point cloud processing pipeline takes the object mask images and their corresponding depth images as inputs. Pixels within the region of interest will then be unprojected into 3D space to generate a point cloud. After that, we downsample the point cloud using a VoxelGrid filter with a leaf size of 0.006. Generally, the semantic instance segmentation component will bring some outlier points due to detection errors, an example is shown in Figure 6.5. In this example, the semantic instance segmentation component fails to segment out the sticky roller from the tabletop, if we use this result as the input

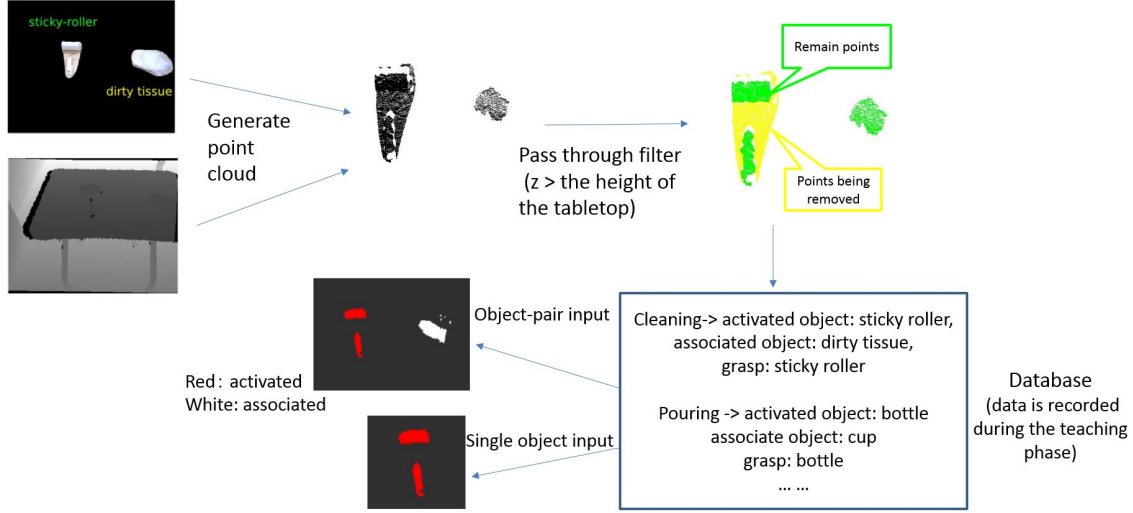


Figure 6.5: An illustration of the point cloud process pipeline.

of the proposed network, the network performance may consider the object belongs to an unknown category and thus perform very poor. To solve this problem, we apply a pass-through filter and set the  $z$  value to a little higher than the height of the tabletop. After that, a RadiusOutlierRemoval filter with the RadiusSearch parameter of 0.04 is applied to remove the outliers and the number of points is resized to 2,400. If the number of points is less than 2,400, the point cloud will be padded with zeros.

According to the task data recorded during the teaching phase, the component assigns the activation signal to the corresponding points to make up an object-pair point cloud. In particular, for points belong to the activated object, their activation channel will be set to one, for points belong to the associated object, their activation channel will be set to zero. After that, the object being grasped will be selected out and its activated channel will be set to one. For example, in the cleaning task, the activated object is a sticky roller and the associated object is a piece of dirty tissue. Therefore, the activation channel of points of the sticky roller is set to one, and the activation channel of points of the tissue is set to zero. Since the sticky roller is the being grasped object, it will be sent to the network alone for grasping pose detection.

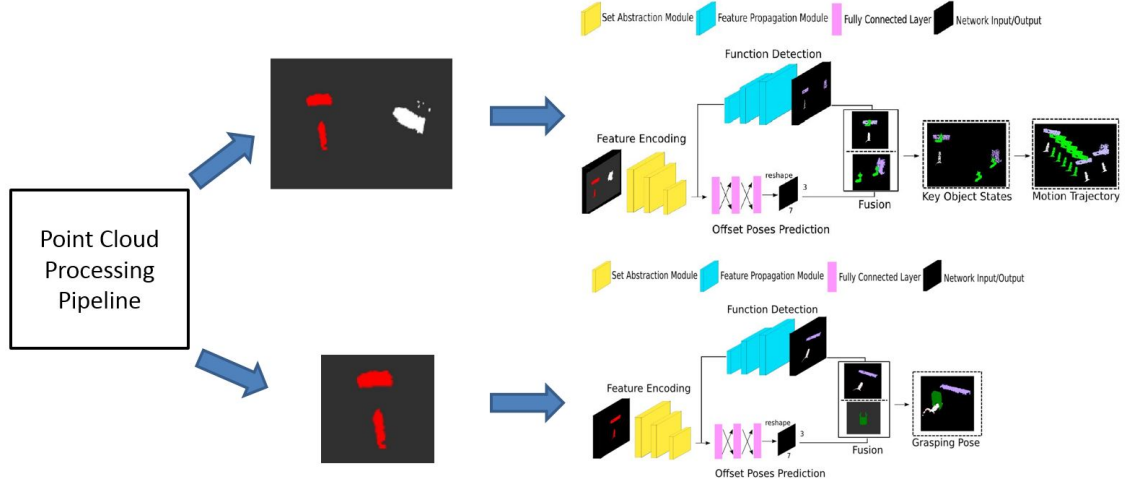


Figure 6.6: An illustration of the grasping pose and trajectory generation using the trained Object-pair Mutual Function Knowledge Acquisition Network.

### 6.2.4 Trajectory and Grasping Pose Generation

When generating trajectories and grasping poses, we used the Object-pair Mutual Function Knowledge Acquisition Network with trained weights described in Chapter 5. The network will be used twice in a manipulation task reproduction process. For the first time, an object-pair point cloud is sent to the network for generating key object pose trajectories. In the second time, a point cloud that only contains the object being grasped is sent to the network to detect a grasping pose. An illustration is shown in Figure 6.6.

The grasping pose and key object states are then be used to generate the key robot motion poses. A demonstration is shown in Figure 6.7. In particular, the system first computes the transformation between the grasping pose and the first key state, then it applies the transformation to the second and third key states to get the key robot motion poses. Using these poses, a path planner can generate a complete motion trajectory to guide the robot to perform a task, in free space, we can apply the linear interpolation to fill the rest of the motion waypoints.

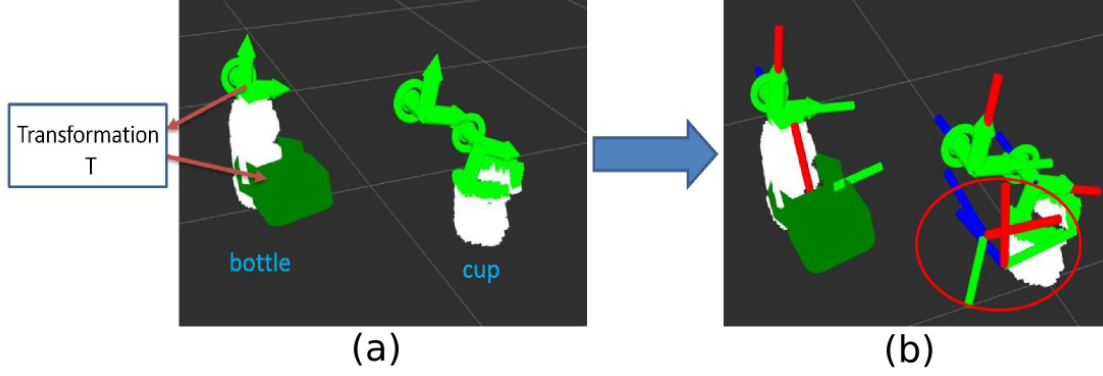


Figure 6.7: An illustration of generating robot motions using the output grasping pose and key object states. The system first computes the transformation between the grasping pose and the first key state, then it applies the transformation to the second and third key states to get the robot motion poses (6D poses within the red circle).

### 6.3 Experiments

When reproducing manipulation tasks, the capability of generalizing the learned knowledge to new instances without extra efforts is desired for a robot teaching system. To compare the performance between the instance matching based manipulation reproducing method and object function knowledge-based manipulation method in dealing with new instances, we consider testing the two methods in the placing and pouring task.

The instance matching based method is achieved through the Iterative closest point (ICP) algorithm [59], which is widely used in 6D object pose estimation. Specifically, given two point clouds (a source point cloud and a target point cloud) and an initial estimate of their transformation, the ICP algorithm will iteratively refine the transformation by selecting corresponding point pairs to minimize an error metric. This error metric is measured by the squared distance between points in each correspondence pair. The algorithm will finally output an optimal transformation to align the source point cloud to the target point cloud. When using this method, users need to create object templates during the demonstration, then apply the ICP algorithm to compute the transformation matrixes between the templates and the test point clouds until convergence. Therefore, if the target object is changed, users must provide a new 3D geometrical model or template. An

example of the point cloud alignment using the ICP algorithm is shown in Figure 6.8.

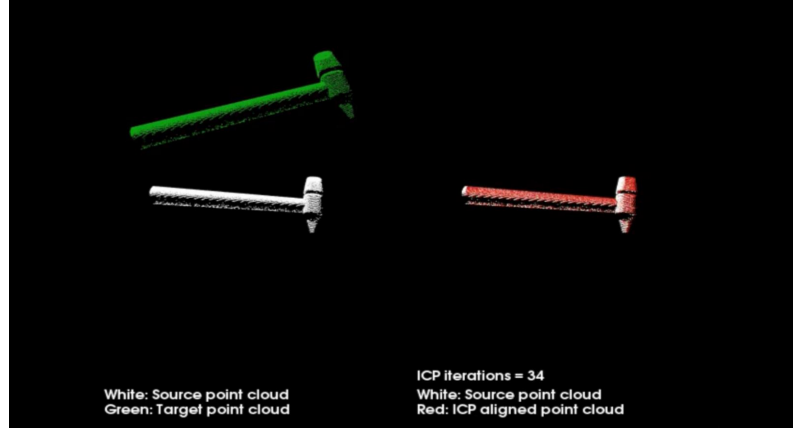


Figure 6.8: Point cloud alignment using the ICP algorithm.

When evaluating the instance matching based method, we replace the Object-pair Mutual Function Knowledge Acquisition Network in the trajectory generation stage and keep the grasping pose detection. The concrete implementation of the ICP approach is stated as follows: (1) Loop over all the relevant templates and apply the ICP algorithm to compute the transformation as well as fitness score (sum of squared distances from the source point cloud to the target point cloud) between the input object point cloud and the templates. (2) Find out the sample with the lowest fitness score, then apply the transformation to its corresponding state labels to generate a motion trajectory for the task.

The template models used for evaluating the ICP algorithm are collected from the ModelNet40 dataset[155], some examples are shown in Figure 6.9 and all of the models are shown in Appendix B. In addition, the template models are converted to partly occluded point clouds using the approach described in Chapter 4.

The Object-pair Mutual Function Knowledge Acquisition Network method does not use these templates, it only used the weights trained in Chapter 5. That is, the demonstration objects for the network are only one bottle and one cup.

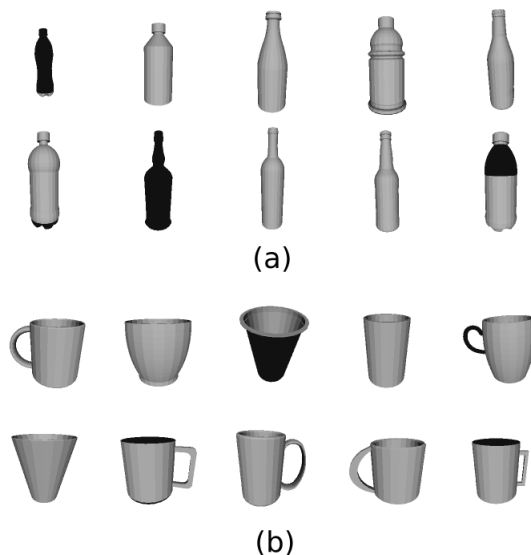


Figure 6.9: Some samples of the template models used in the ICP evaluation. (a) 3D CAD models of the bottles. (b) 3D CAD models of the cups.

## 6.4 Comparison of the Instance Matching Based Method and the Function Part Detection Based Method in Handling New Instances

During the evaluation, we use five bottles and five cups for testing, the test objects are shown in Chapter 5. In each trial, object pairs were randomly picked from the test objects. From Figure 6.10 we can see that the ICP algorithm fails in most of the pouring tasks due to two reasons: (1) the ICP estimation approach has difficulty in searching for a suitable template. (2) the ICP estimation approach outputs incorrect transformation due to alignment errors.

On the other hand, the object function knowledge base method first detects whether the target object has a known function part. If a known function part is detected, it can automatically choose a learned trajectory to accomplish the function.

Computing time is also an important factor in manipulation task reproduction. Figure 6.11 shows the Computing time of the two methods, the computing time is estimated on

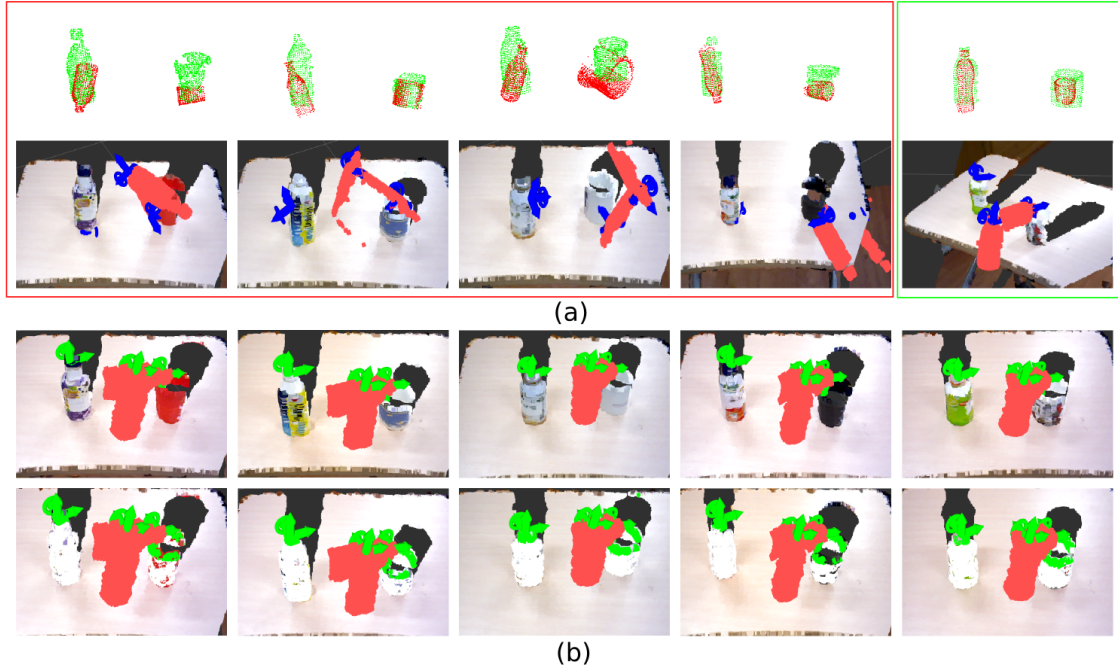


Figure 6.10: Applying different methods in new instances in the pouring task. (a) the point cloud alignment results using the ICP algorithm and the corresponding pose trajectories. (b) the trajectory outputs that are given by the Object-pair Mutual Function Knowledge Acquisition Network.

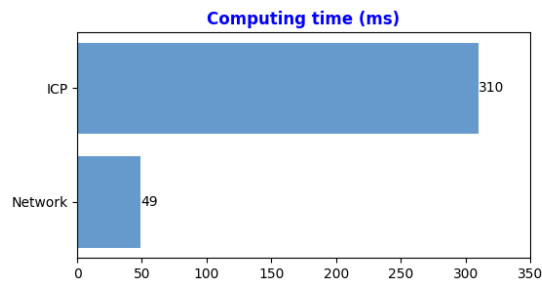


Figure 6.11: Computing time of different methods, the ICP algorithm takes around 310ms for one alignment, the network takes 49ms for one inference.

an Intel Core i5 CPU. The Object-pair Mutual Function Knowledge Acquisition Network is built upon the TensorFlow library [181]. Since the network does not need to search the closest templates, its computing time is fixed and around 49ms. The ICP method we used in the experiment is provided by the PCL library [182], the max iteration is set to 10 and it takes around 310ms for computing the fitness score and point cloud transformations for one template, as the number of templates increases, it will take more time. Therefore, an object function and learning base method is more suitable for a robot teaching system.

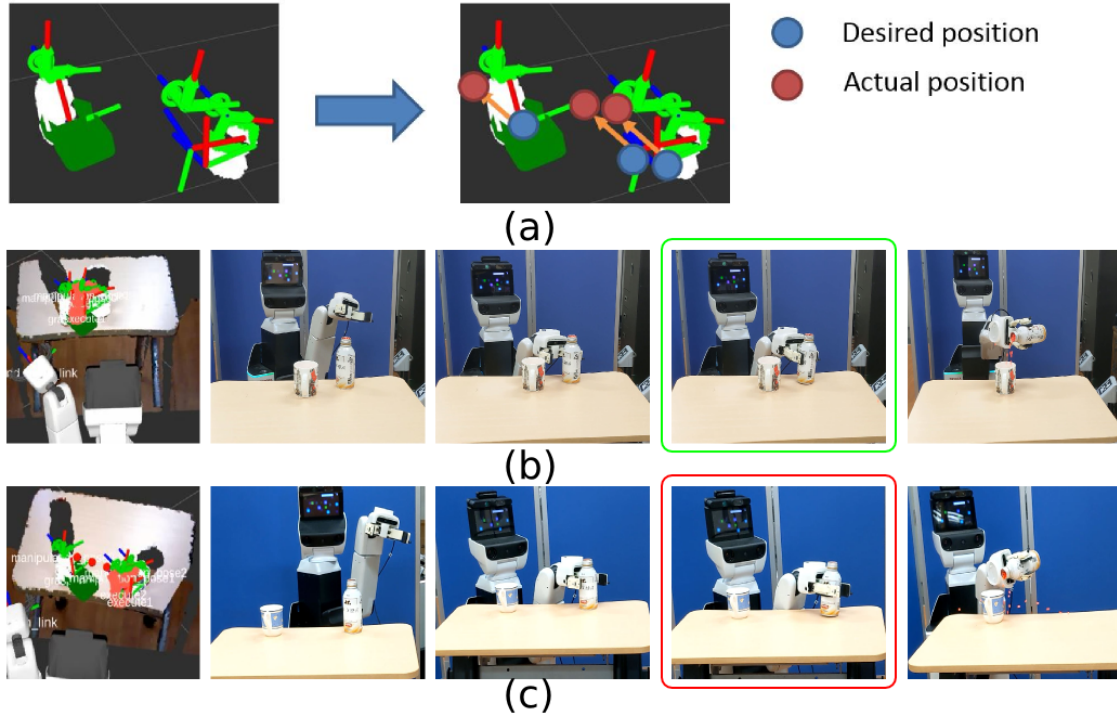


Figure 6.12: Although the proposed system has outputted valid trajectories for the pouring tasks, failure may occur due to the robot grasping deviation brought by the low level control. (a) a demonstration of the deviation between the desired grasping pose and the actual grasping pose. (b) The system outputted a valid trajectory and the task is success because the actual grasping pose is the desired grasping pose (green box), and the bottle is manipulated following the output trajectory. (c) The system outputted a valid trajectory but the task is failed due to the actual grasping pose is not the desired grasping pose (the bottle is pushed a little forward in the red box), this change causes the bottle deviates from the original trajectory during execution.



## 6.5 Discussion

When using the system outcomes to guide a robot to perform tasks, as the function part detection based method has successfully detected the function parts of the object pairs and generated a valid trajectory, the task should be successfully performed if the robot can act following the output trajectory. However, failure may occur when there is a deviation during grasping. In the example in Figure 6.12, a robot is going to grasp a bottle and then perform the pouring motion. From the first column in Figure 6.12 (b) and (c), we can see that the goal states of bottle point clouds are aligned to the centroid of the cup point clouds. However, in Figure 6.12 (c) the bottle is pushed a little forward after grasping due to the position error after the robot moved. This change causes the bottle deviates from the original trajectory during execution and leads to the task failure. In Figure 6.12 (b), the actual grasping pose is the desired grasping pose, and thus the robot can manipulate the bottle following the output trajectory. This failure only occurs when a task is very sensitive to the object's position.

## 6.6 Summary

This chapter has presented a framework for manipulation task reproduction. The input of the framework is a user command, according to the user command, the semantic instance segmentation component is able to detect objects in the desired category, the objects' masks are then passed to the point cloud processing component to obtain object-pair point cloud with activation signals. Finally, the processed point cloud is passed to the trajectory and grasping pose generation component. By comparing different trajectory generation methods, we found that applying the instance matching based method on new instances is difficult due to the difference of object shape and size, while the function part detection based method performs much better in handling new instances.



## **第7章**

# **Home-Assistant Robot Applications Using the Manipulation Teaching System with a Few Visual Demonstrations**



Chapter 3 presented an Object-pair Mutual Function Knowledge model that can represent the reusable knowledge in manipulation tasks. Chapter 4 presented two teaching approaches that enable the system to obtain visual demonstrations from human teaching. Chapter 5 proposed a data augmentation framework and an Object-pair Mutual Function Knowledge Acquisition Network that allows robots to learn manipulation knowledge from a few demonstration samples provided by users. Chapter 6 presents a framework for how to reproduce the manipulation task through user command inputs. This chapter will use all these components to achieve the application of manipulation teaching with a few visual demonstrations in an everyday scene.

## 7.1 Robot Platform

The robot which we use to evaluate the robot teaching system is called HSR. The HSR, which is shorted “Human Support Robot” is designed and built by the Toyota company. Aiming at assisting in activities of daily living, the HSR robot is designed for general purpose.



Figure 7.1: An overview of the HSR robot.

### 7.1.1 Hardware

The core components of the HSR robot are an autonomous mobile base, a liftable torso, a single arm with a parallel gripper, and various kinds of sensors, the robot is shown in Figure 7.1

- The Autonomous mobile base is in the shape of a cylinder with a maximum speed of 0.22m/s. A built-in bumper enables the mobile base to detect collisions and avoid injuries. Furthermore, a built-in six degrees of freedom Inertial Measurement Unit (IMU) and a top-mounted LiDAR enable the HSR robot to fulfill the navigation and collision avoidance functions.
- The liftable torso allows the robot to stretch out its spine to reach a higher place, the stroke of the liftable torso is 690mm. In the middle of the torso is a robot arm with a maximum payload of 1.2kg. Inside the arm are an absolute rotary encoder and a six-axis force and torque sensor, which enables the HSR robot to execute complex manipulation tasks such as groping.
- The end-effector of the HSR robot is a parallel gripper, the maximum opening width of 135mm and the maximum grasping force of 40N. A vacuum suction cup is attached to one of the fingers of the gripper, with a maximum suction force of 5N, this vacuum suction cup enables the HSR robot to suck up thin objects like papers and photos.
- The HSR's head is composed of a microphone, an RGB-D camera, a display, and a stereo camera. These sensors enable the HSR robot to capture external signals such as sound and image. The HSR robot has two core processors, a 4th Gen Intel Core i7 CPU, and an embedding NVIDIA Jetson TK1 GPU that allow the robot to run some complex algorithms such as detecting objects via deep learning networks.
- Finally, the total DOF of the HSR robot is 11, where the base has 3 DOF, the arm has 3 DOF, the wrist has 2 DOF, the hand has 1 DOF and the head has 2 DOF, Table 7.1 shows the motion range of each joint, and Figure 7.2 shows the relationships between different links and joints.

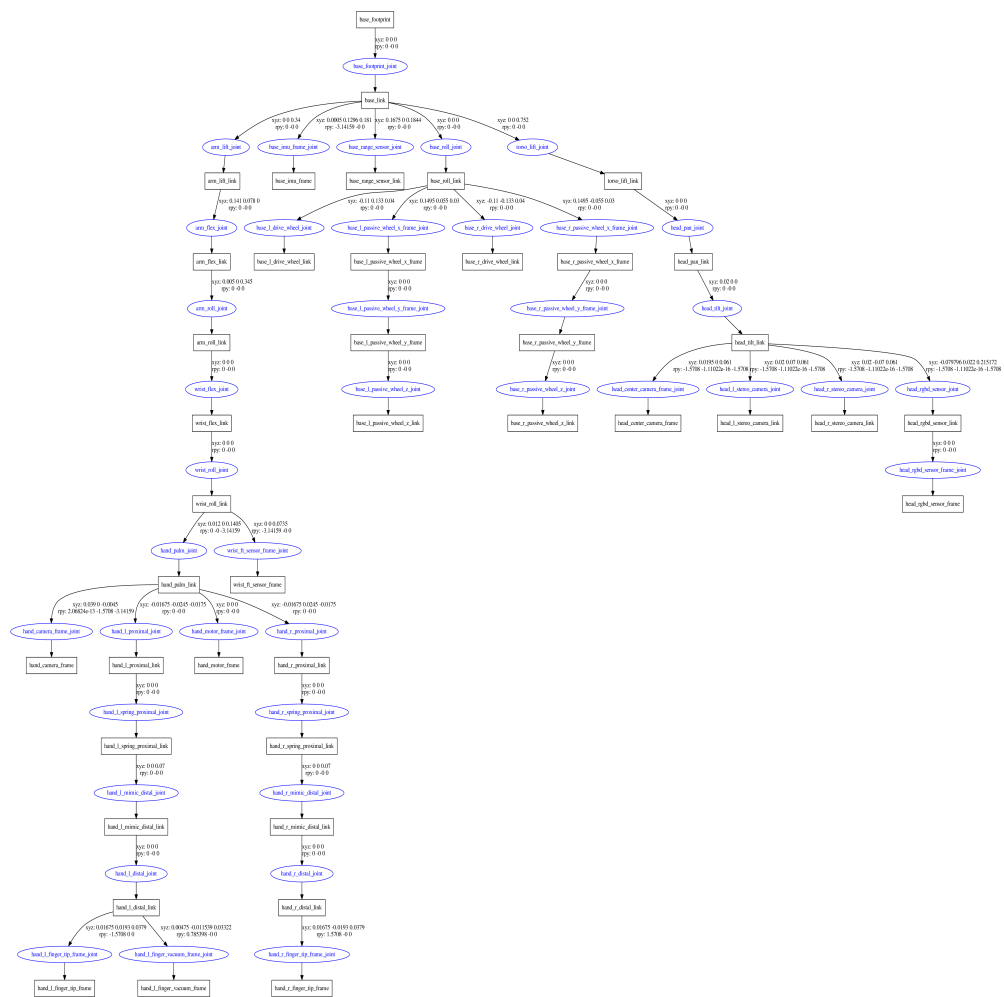


Figure 7.2: Relationships between different links and joints.

Joint Name	Type	Limit(-)	Limit(+)	Axial Direction
base_roll_joint	continuous	-	-	z
base_l_drive_wheel_joint	continuous	-	-	y
base_r_drive_wheel_joint	continuous	-	-	y
arm_lift_joint	prismatic	0 [m]	0.69 [m]	z
arm_flex_joint	revolute	-2.617 [rad]	0 [rad]	-y
arm_roll_joint	revolute	-1.919 [rad]	3.665 [rad]	z
wrist_flex_joint	revolute	-1.919 [rad]	1.221 [rad]	-y
wrist_roll_joint	revolute	-1.919 [rad]	3.665 [rad]	z
hand_motor_joint	revolute	-0.105 [rad]	1.239 [rad]	x
head_pan_joint	continuous	-3.839 [rad]	1.745 [rad]	z
head_tilt_joint	prismatic	-1.570 [rad]	0.523 [rad]	-y

Table 7.1: Joint specification of the HSR robot.

### 7.1.2 Middleware

Developers are allowed to drive the HSR robot via the Robot Operating System (ROS) API or python API, we used the former in this thesis.

ROS is an open-source robotics middleware that provides services designed for a heterogeneous computer cluster, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

The main characteristic of the ROS is that it provides an efficient mechanism for communication between different components. The runtime graph of ROS is a peer-to-peer network of processes, every component is represented as a node in the graph, and every node can talk to each other. In the publish-subscribe messaging pattern, the messages sent from a publisher are called topics, and a node who needs the messages can subscribe to the corresponding topics. Moreover, the ROS also provides a service and client messaging pattern to meet different needs. An example of a ROS runtime graph is shown in Figure 7.3.

Here we have the “affnet2\_tf” node, the “affordance\_detection” node, and a tf topic, which is flowing from the “affnet2\_tf” node to the “affordance\_detection” node.



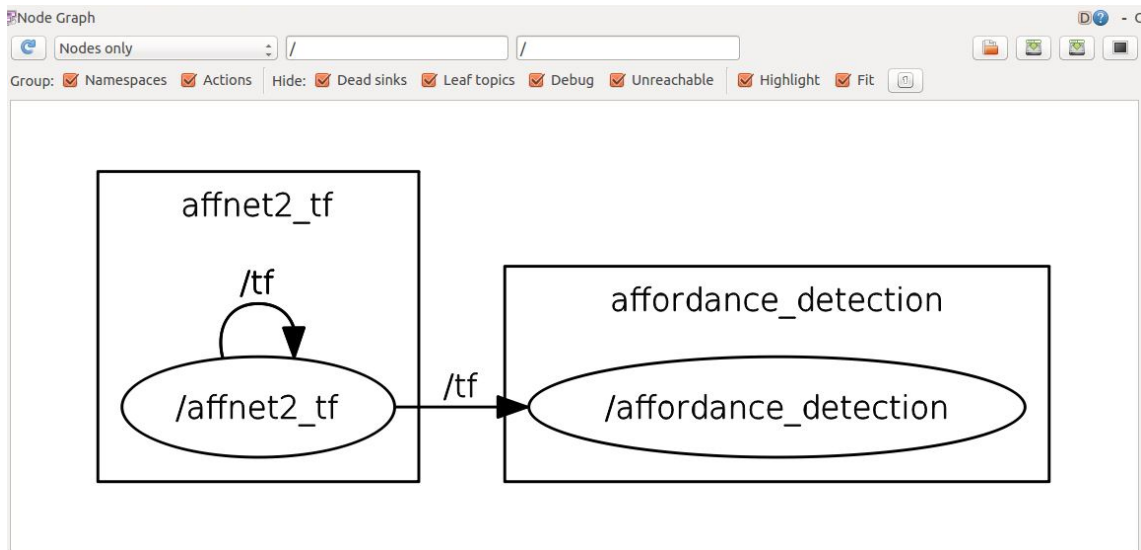


Figure 7.3: An example of a ROS runtime graph.

According to Quigley [183]. The main idea of ROS can be summarized as 5 keywords.

- **Peer to Peer**
- **Tool base**
- **Multi-lingual**
- **Thin**
- **Free and Open-source**

Among them, Tool based is a key concept to make the complex system easy to maintain. ROS allows the user to create a wide variety of tools, such as measuring bandwidth utilization, graphically plot message data, etc.

### The ROS packages used in this thesis

- **Std-msgs** involves common message types such as Bool, Int, Float, etc. Users also create their message types to fit their applications.

- **TF** is a library for coordinate transformation management. All coordinates in ROS have a parent frame and child frame. A tree structure is built by the connection of all these coordinate frames.
- **Rviz [165]** is a 3D visualization tool for displaying sensor data and state information from ROS. It has some default visual components such as the robot model, map, and coordinate. It also allows us to write a plugin to visualize our data, such as trajectory, position, etc.
- **PCL-ROS** provides interfaces to bridge the ROS message type and Point Cloud Library (PCL) data type. It also provides many functions for users to handle 3D geometry processing problems in ROS.
- **Toyota HSR's Python and ROS Interface** is a library provided and maintained by the Toyota company. This library provides many interfaces to allow users to drive the HSR robot using some complex algorithms such as whole-body control.

## 7.2 Integration of the Manipulation Teaching System on the Robot System

Figure 7.4 describes the details of integrating the proposed system on the HSR robot in the task reproduction phase. Firstly, the HSR robot captures an RGB image and its corresponding depth image from its head-mounted RGB-D camera. Then these two images are sent to the Mask R-CNN server and get the target objects' masks. With the intrinsic matrix given by an extra calibration process, the point cloud preprocessor module described in Chapter 6 unprojects the 2D image pixels into 3D points via the equation:

$$P_x = (u - c_x) * P_z * \frac{1}{f_x} \quad (7.1)$$

$$P_y = (v - c_y) * P_z * \frac{1}{f_y} \quad (7.2)$$

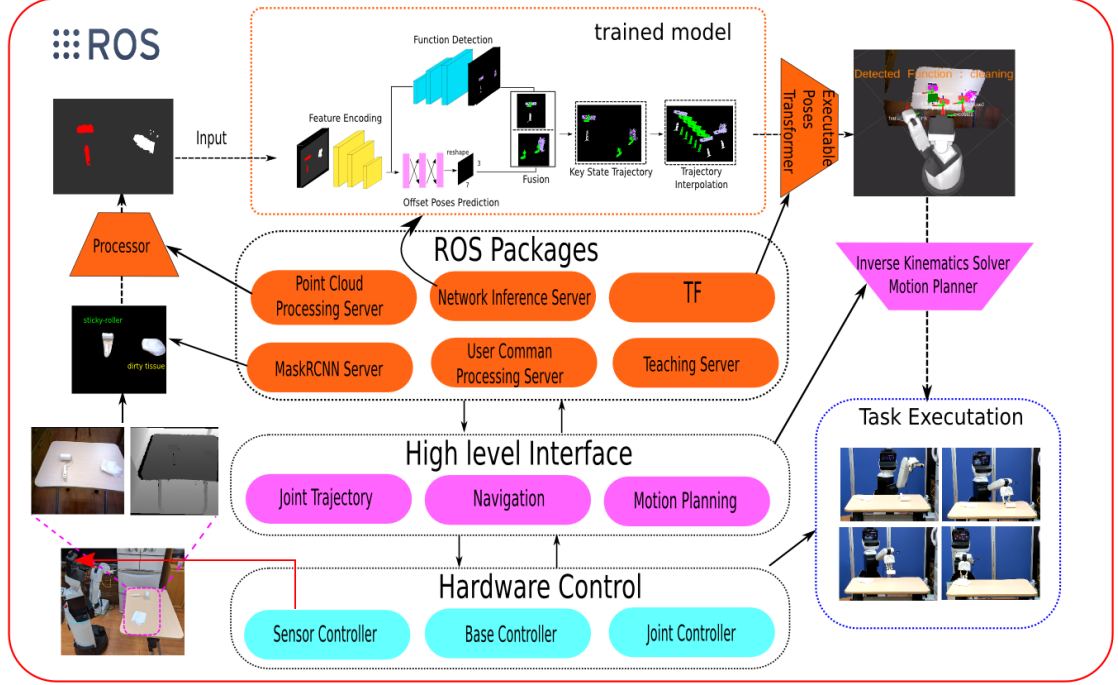


Figure 7.4: An illustration of integrating the proposed system on the HSR robot in the task reproduction phase.

Where  $(P_x, P_y, P_z)$  is the 3D coordinate of the point,  $P_z$  is given by the depth image,  $(u, v)$  is the corresponding pixel in the RGB image,  $c_x, c_y, f_x, f_y$  are elements in the camera intrinsic matrix.

At the same time, this module transforms the point cloud's reference coordinate from the robot's RGB-D camera link to the robot's base link for the convenience of subsequent processing.

After that, the point cloud processing module assigns an activation signal and sends the processed point cloud to the trained Object-pair Mutual Function Acquisition Network to get the key object state trajectory and grasping pose. In addition, we assume the generated grasping pose is consistent with the HSR's tool frame, which means this pose can be directly used for the HSR's grasping without further transformation.

An illustration is shown in Figure 7.5. Given the motion trajectory, the motion planning package within the ROS middleware will generate the robot movement, and the inverse

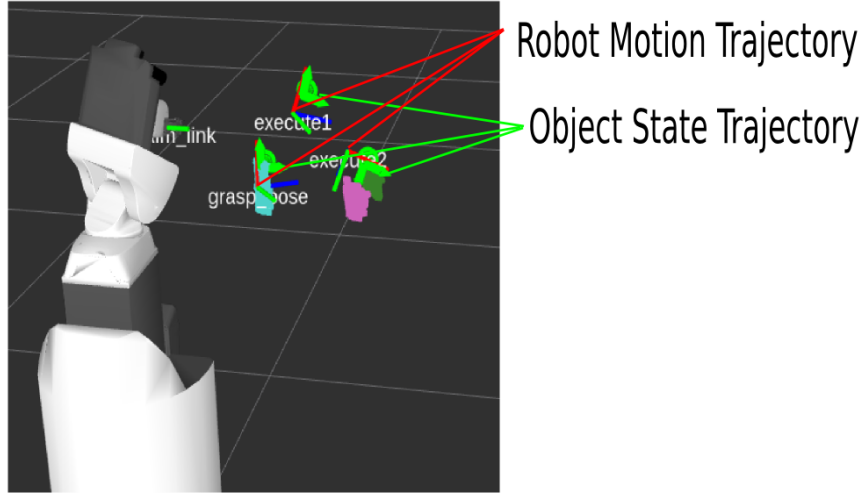


Figure 7.5: Transform object state trajectory into robot motion trajectory. The green markers represent the object states, while the 6D coordinates represent the robot executable poses.

kinematic solver will find out corresponding joint angles for every movement. These control signals are then transferred to the hardware level control packages to drive the HSR robot to execute a series of actions.

## 7.3 Teaching a Robot New Manipulation Knowledge in Daily Life

### 7.3.1 Task Description

The goal of this thesis is to reduce the user effort in robot teaching, in this section, we consider teaching a robot with a few demonstrations to increase its manipulation knowledge for human daily living assistance. At first, the robot only knows how to placing objects on plates and pouring liquid. The demonstration data includes the grasp pose of different kinds of objects, placing an apple on a plate, and pouring liquid from a bottle into a cup (the pouring task has two demonstrations, one for pouring from the left-hand side and one for pouring from the right-hand side). The robot learns these demonstrations and

applied the knowledge to the test objects. After that, the demonstrator teaches the robot not to pour water into an upside-down cup by providing a negative sample. Next, the demonstrator teaches the robot how to use a sticky roller for cleaning and a “knife” (for convenience sake, we use a hacksaw instead) for cutting through on-site demonstrations. During the experiments, objects will be randomly selected from the test objects shown in Figure 7.6.









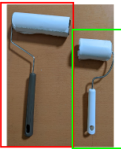













Tasks	Placing	Pouring (Include Abnormal State)	Cutting	Cleaning
<b>Objects</b>  Red: Demonstration Green: Test Blue: Used in Demonstration and Test    From RGB-D Object dataset [160]      From ModelNet dataset [155]	 	 Normal State      Abnormal State 	 	 
<b>Demonstrations (Number of Samples) [Original : Augmented]</b>	 [1 : 100]  [1 : 100]  [1 : 100]  [1 : 100]	 [1 : 100]  [1 : 100]  [1 : 100]  [1 : 100]	 [1 : 100]  [1 : 100]	 [1 : 100]  [1 : 100]

Figure 7.6: Demonstration data for the placing, pouring, cutting, and cleaning tasks. The objects in the red boxes are demonstration objects, the objects in the green boxes are test objects, the objects in the blue boxes are used in demonstration and test. The total number of original demonstrations is 12, including grasping an apple, grasping a teapot, grasping a cup, placing an apple to a plate, grasping a bottle, pouring from the bottle to the cup from two directions, no function for the upside-down cup, grasping a knife(hacksaw), cutting a pudding using a knife (on-site teaching), grasping a sticky roller, cleaning a piece of dirty tissue using a sticky roller (on-site teaching).

### 7.3.2 Objects

The demonstration objects in the placing task include an apple and a plate (from the RGB-D object dataset [160]), a cup (from the ModelNet40 dataset [155]) and a teapot. The test objects in placing task include a toy apple, a teapot, a bowl, three plates, five cups and five bottles (Table 7.2) The demonstration objects in the pouring task include two bottles and two cups, one of the cup and bottle are used for abnormal state demonstration. The test objects in the pouring task include five bottles, five cups and five bowls (Table 7.3). The demonstration objects in the cutting task include a knife (hacksaw) and a pudding, the test objects in the cutting task include the same pudding and two different knives. The demonstration objects in the cleaning task include a sticky roller and a piece of dirty tissu, the test objects in the cleaning task include the same tissue and a different sticky roller.

Demonstration Objects	Apple	Plate	Cup	Teapot			Total
Number	1	1	1	1			4
Test objects	Toy Apple	Teapot	Bowl	Plate	Bottle	Cup	Total
Number	1	1	1	3	5	5	16

Table 7.2: The number of the demonstration and test objects in the placing task.

Demonstration Objects	Bottle	Cup		Total
Number	2	2		4
Test objects	Bottle	Cup	Bowl	Total
Number	5	5	5	15

Table 7.3: The number of the demonstration and test objects in the pouring task.

### 7.3.3 Demonstrations

The total number of original demonstrations in this experiment is 12, including grasping an apple, grasping a teapot, grasping a cup, placing an apple to a plate, grasping a

bottle, pouring from the bottle to the cup from two directions, no function for the upside-down cup, grasping a knife(hacksaw), cutting a pudding using a knife (on-site teaching), grasping a sticky roller, cleaning a piece of dirty tissue using a sticky roller (on-site teaching). Each demonstration is augmented to the number of 100, and thus the total number of augmented demonstrations is 1,200.



Figure 7.7: The HSR robot is performing the placing task using different object-pair combinations. First row: the robot picks up a teapot and puts it on the plate. Second row: the robot picks up an apple and puts it into a bowl. Third row: the robot picks up a cup and puts it on the plate. Fourth row: the robot picks up a bottle and puts it on the plate.

### 7.3.4 Learning Manipulation Knowledge from Virtual Teaching

The demonstration samples in this section include one pose trajectory of placing an apple to a plate and two pose trajectories of pouring from a bottle to a cup (one for pouring

from the left-hand side and the other one for pouring from the right-hand side). Here, the placing and pouring trajectories, the grasp poses of a bottle, a teapot, a cup, an apple, a sticky roller, and a knife are given by virtual teaching. The demonstration samples are shown in Figure 7.6. For convenience sake, we use the sign (place, apple, plate) to represent the user command “placing an apple to the plate”.

In the placing tasks, the demonstration sample of placing is augmented to the number of 100, the user commands are (place, teapot, plate) (place, apple, bowl) (place, cup, plate), and (place, bottle, plate). The activated object is the plate or the bowl in this case, and the other object is the associated object. From Figure 7.7 we can see that although we only gave the robot one placing demonstration of (place, apple, plate), the robot has successfully applied the placing skill for different object pairs.

In the pouring tasks, the demonstration sample of pouring is augmented to the number of 200 (100 for left and 100 for right), and the total number of augmented demonstrations is 900 (100 for placing, 200 for pouring, 600 for grasping different objects). From Figure 7.8 we can see that the robot has successfully generalized the pouring skill from (pouring, bottle, cup) to (pouring, different shapes of bottles, different shapes of cups) and (pouring, different shapes of bottles, different shapes of bowls). The demonstration point clouds are generated from CAD models (Figure 7.6). In the reproduction phase, although the cups and bowls have different shapes and the objects are placed in different positions, the system still generated valid trajectories to guide the robot to perform the pouring task successfully.

### 7.3.5 Learning to Detect Object in Unexpected State with Negative Sample

Next, the demonstrator gave the robot a cup that is placed upside-down, since the robot has not learned how to deal with an upside-down cup, it wrongly tried to pouring liquid into the upside-down cup (Figure 7.9 (a)). To avoid this undesired behavior, the demonstrator gave the robot a negative sample and augmented the sample to 100 using the proposed framework. Concretely, when the object-pair contains a bottle and an upside-down



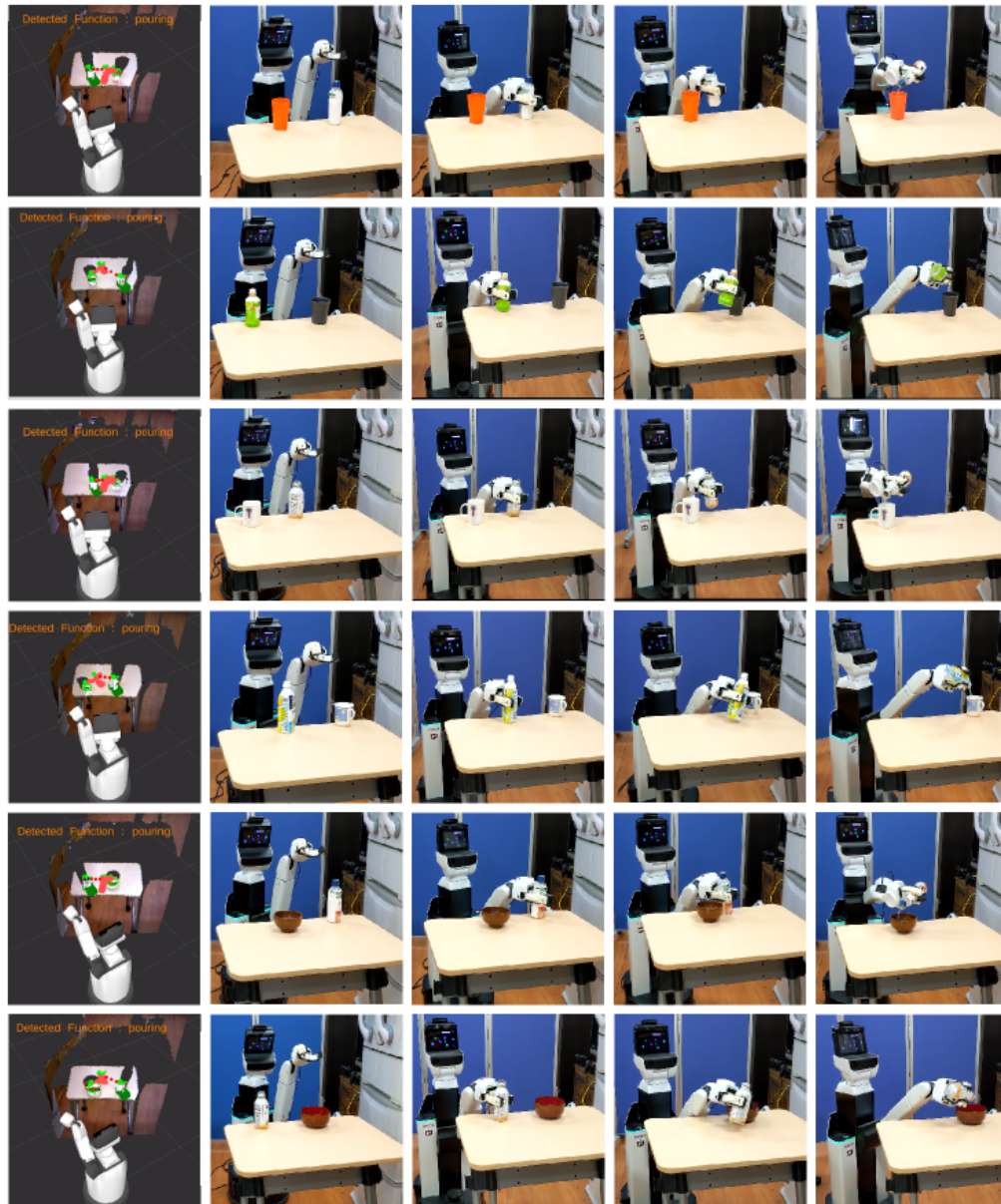


Figure 7.8: The HSR robot is performing the pouring task. The demonstration point clouds are generated from CAD models (Figure 7.6). In the reproduction phase, although the cups and bowls have different shapes and the objects are placed in different positions, the system still generates valid trajectories to guide the robot to perform the task successfully.

cup, we set the robot to output “no function” and set the pose trajectory to all zeros to represent an invalid trajectory.

After retrained the network, the demonstrator first gave the robot a cup in the upside-down state. As the robot detected “no function”, the demonstrator turned over the cup and the robot detected again. After that robot demonstrator gave the robot another upside-down cup, and the robot detected “no function” again. The result shows that the robot has successfully recognized the cup in different states after learning (Figure 7.9 (d)).

### 7.3.6 Learning Manipulation Knowledge from a Few On-site Demonstrations

Next, the demonstrator taught the robot how to use a sticky roller to collect a piece of dirty tissue. The demonstration was given by on-site teaching. In the teaching stage of the cleaning task, the demonstrator showed the robot which part of the sticky roller is functional and how to use the sticky roller to sticky the dirty tissue. The demonstration sample is augmented to the number of 100. After retrained the network, the demonstrator gave the robot another sticky roller that has a different size from the demonstration one. From Figure 7.10 we can see that the robot has learned how to use a sticky roller to clean a piece of dirty tissue using a new sticky roller.

Finally, the demonstrator used the system to teach the robot how to cut a pudding. In the cutting task, the demonstrator showed the robot how to use a knife (for convenience sake, we used a hacksaw instead) to cut a pudding by providing the function part of the knife and a trajectory for performing the cutting task. The demonstration sample is augmented to the number of 100. After retrained the network, the demonstrator gave the robot another two knives that have different shapes and sizes from the demonstration one. From Figure 7.11 we can see that the robot has learned how to cut a pudding using new knives.

In this experiment, the results show that the proposed system has the ability to generalize the learned manipulation skills to new instances through a few demonstrations. In the placing task, the number of original demonstration samples is five, four samples

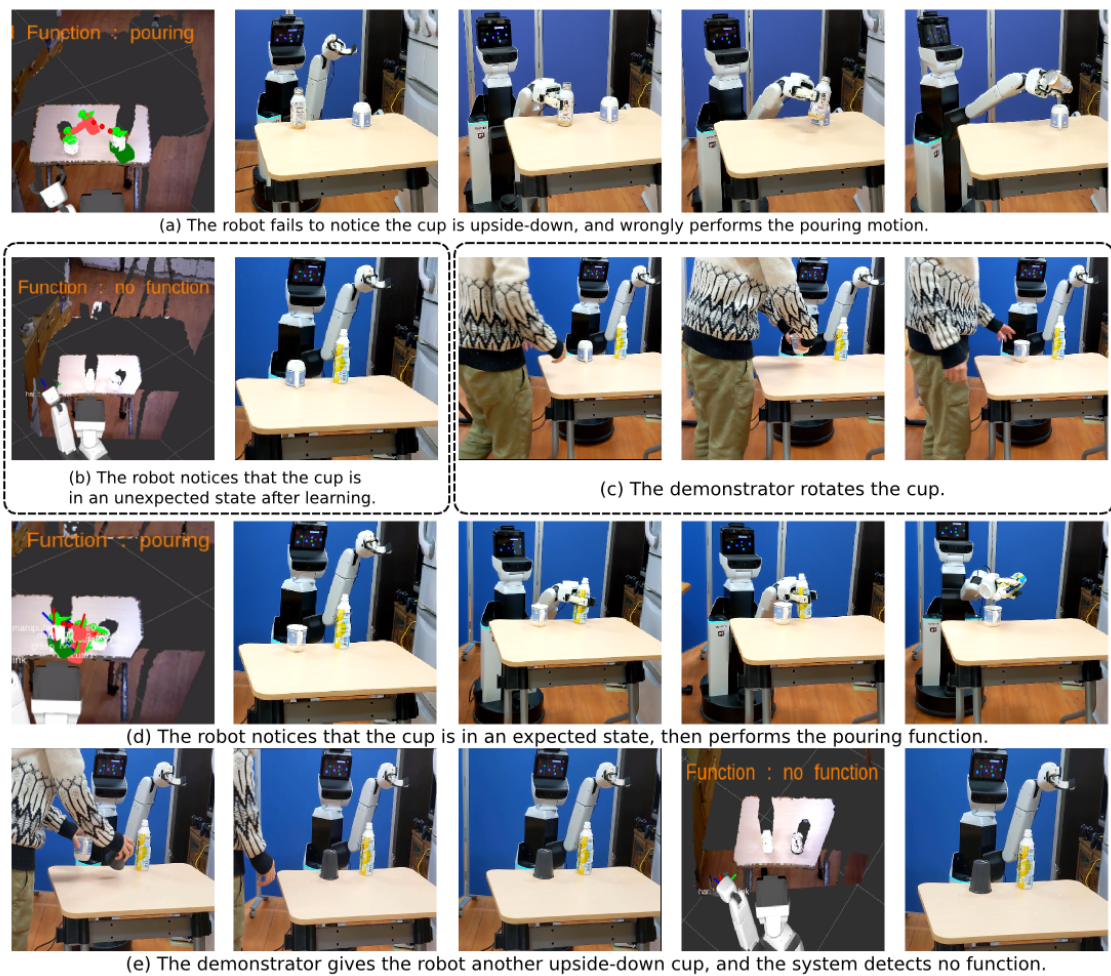


Figure 7.9: The HSR robot learns to distinguish different object states. (a) The robot fails to notice the cup is upside-down. (b) The robot notices that the cup is unexpected, and thus the robot does not take any action. (c) The demonstrator rotates the cup. (d) The robot notices that the cup is in an expected state then performs the pouring function. (e) The demonstrator gives the robot another upside-down cup, the system detects no function and the robot does not take any action.

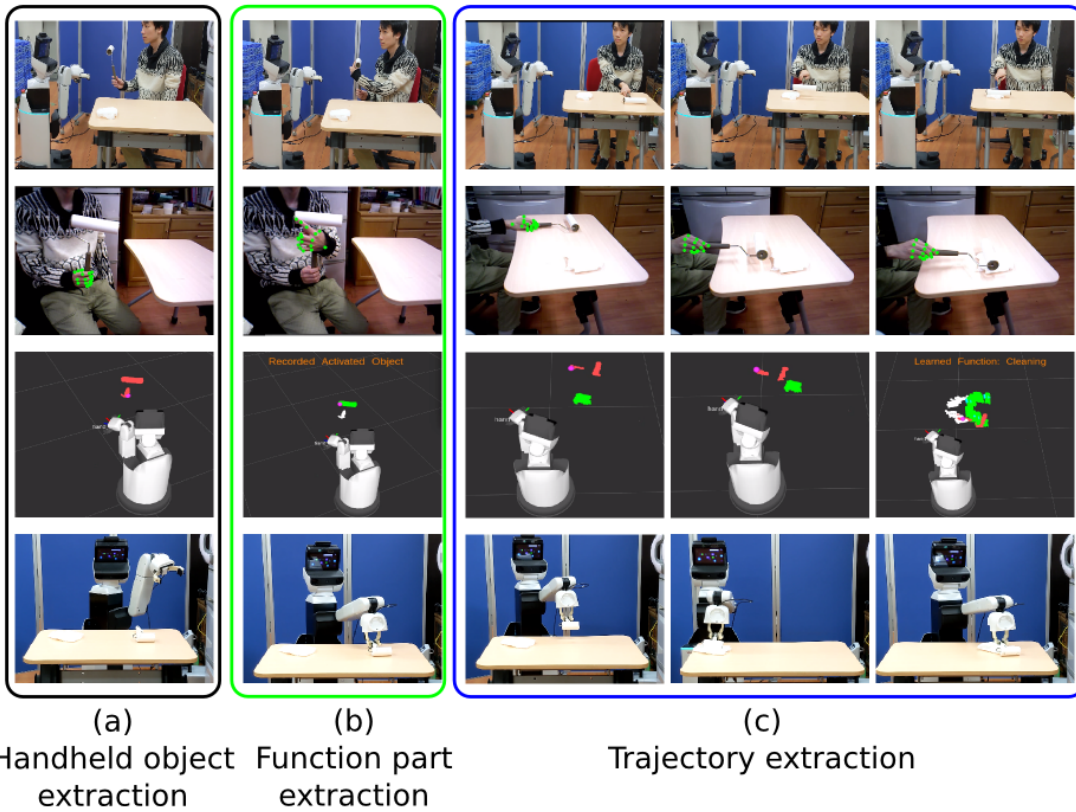


Figure 7.10: The HSR robot learns how to use the sticky roller for cleaning from an on-site demonstration. (a) The demonstrator is showing the robot the sticky roller. (b) The demonstrator is showing the robot the function part of the sticky roller. (c) The demonstrator is showing how to use the sticky roller to clean a piece of dirty tissue.



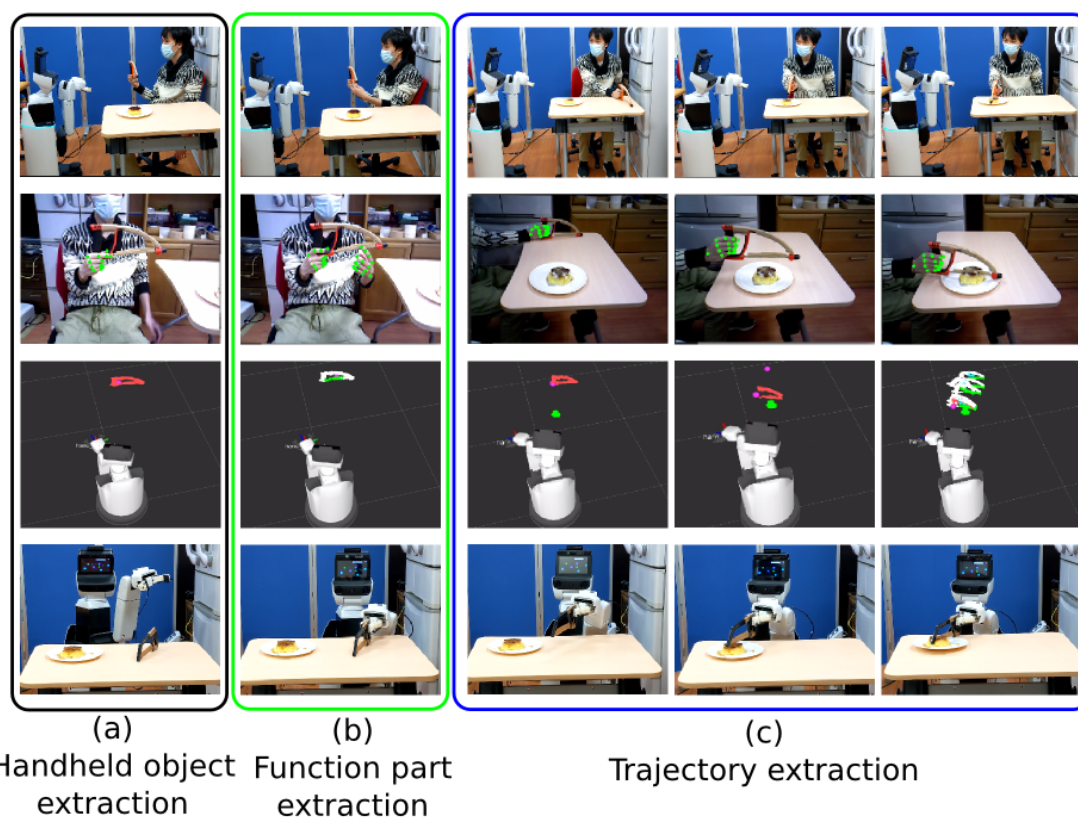


Figure 7.11: The HSR robot learns how to cut a pudding from an on-site demonstration. (a) The demonstrator is showing the robot the knife. (b) The demonstrator is showing the robot the function part of the knife. (c) The demonstrator is showing how to use the knife to cut a pudding.

for grasping different kinds of objects and one sample for placing, each of the sample is augmented to the number of 100, therefore the total number of demonstration sample for learning the pick and place task is 500. The robot has successfully picked the apple, the cup, the bottle, the teapot, and placed them on the plate. In addition, the robot was able to pick the apple and place it in the bowl.

To increase the robot's manipulation skill, we taught the robot how to pouring, cleaning and cutting using virtual or on-site teaching, the results show that given twelve original demonstrations and the corresponding 1,200 augmented demonstrations, the robot has learned all these manipulation skills and applied these skills to new instances successfully.

During the experiment, it takes us no more than an hour for providing the demonstrations (the virtual teaching takes around one minute for each demonstration, while the on-site teaching takes around 10 minutes for each demonstration). The network training time is around 35 minutes, after that, the robot is able to use the new instances to perform the corresponding tasks.

## 7.4 Teaching a Robot to Tidy a Room

To further show the benefits of the proposed robot manipulation teaching system, we also teach a robot how to tidy a room by picking up objects on the floor, the manipulation tasks in this experiment include picking up a small object and put it into a drawer, picking up a cloth and put it on the backrest of a chair, picking up a bag and put it on the seat of a chair, and picking up an umbrella and hang it on the backrest of a chair.

### 7.4.1 Objects

The demonstration objects are shown in Figure 7.12, include a drawer, a banana case, a bag, an umbrella, a cloth, and a chair, a total of six objects for demonstration. During the test, we use another drawer and an eyeglass case for the putting an object into a drawer task, we use another bag, another umbrella, another cloth for the placing a bag, hanging an umbrella, and placing a cloth tasks separately.





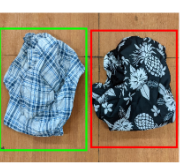

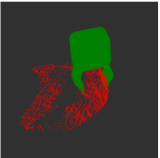
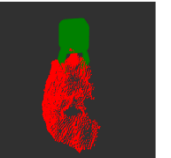
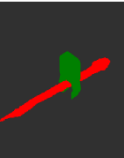
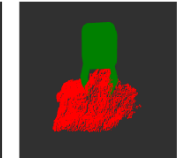
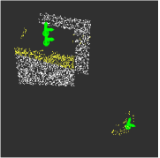
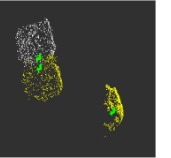
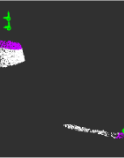
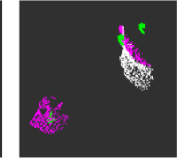
Tasks	Putting an object into a drawer	Placing a bag	Hanging an umbrella	Placing a cloth
<b>Objects</b> Red: Demonstration Green: Test Blue: Demonstration and Test (Chair)	 			 
<b>Number of Demonstration samples</b> [Original : Augmented]	 [1 : 100]	 [1 : 100]	 [1 : 100]	 [1 : 100]
	 [1 : 100]	 [1 : 100]	 [1 : 100]	 [1 : 100]

Figure 7.12: Demonstration data for the tidying task. The objects in the red boxes are demonstration objects, the objects in the green boxes are test objects, the chair is used for demonstration and test. The total number of demonstrations is eight, including grasping a banana case, grasping a bag, grasping an umbrella, grasping a cloth, putting the banana case into a drawer, putting the bag on the seat of a chair, hanging the umbrella on the backrest of a chair, and putting the cloth on the backrest of a chair. Each demonstration is augmented to the number of 100, and thus the total number of augmented demonstrations is 800.

### 7.4.2 Demonstration Samples

In this experiment, we used the virtual teaching tool to teach the robot how to deal with different tasks, the demonstration samples are shown in Figure 7.12, The total number of demonstrations is eight, including grasping a banana case, grasping a bag, grasping an umbrella, grasping a cloth, putting the banana case into a drawer, putting the bag on the seat of a chair, hanging the umbrella on the backrest of a chair, and putting the cloth on the backrest of a chair. Each demonstration is augmented to the number of 100, and thus the total number of augmented demonstrations is 800.

### 7.4.3 Learning to Tidy a Room with a Few Demonstrations

After training the network, we use the HSR robot to reproduce the task using new instances. The robot execution is shown in Figure 7.13. In the placing object into a drawer task, the system needs to detect the opening part of the drawer and guides the robot to put the eyeglass case into the drawer. In the placing a bag on a chair seat task, the input point cloud is a chair and a new bag, the system needs to detect the seat part of the chair and guide the robot to put the bag of the chair seat. In the hanging an umbrella on a chair backrest task, the system needs to detect the top edge of the backrest and the hook part of the umbrella, then guides the robot to perform the hanging motion. In the placing a cloth on a chair backrest task, the system needs to detect the top edge of the backrest and guides the robot to place the cloth on it. We can see that although the test instances are different from the demonstration instances and the user only provided eight demonstrations. The proposed system has helped the robot successfully perform all the above tasks.

## 7.5 Discussion

Through the above experiments, we have shown that the proposed system has the capability of learning new manipulation tasks from a few demonstrations provided by users. After training the Object-pair Mutual Function Knowledge Acquisition Network on the





Figure 7.13: The HSR robot is tidying a room. (a) The robot is picking up a small object and put it into a drawer. (b) The robot is picking up a bag and putting it on a chair seat. (c) The robot is picking up an umbrella and hanging it on a chair backrest. (d) The robot is picking up a cloth and putting it on a chair backrest.

	Behavioral Cloning Rahmatizadeh et al. 2018 [89]	Behavioral cloning Zhang et al. 2018 [52]	Behavioral Cloning Yu et al. 2018 [174]	Inverse Reinforcement Learning Finn et al. 2016 [128]	Instance Matching based Trajectory Imitation Welschhold et al. 2016 [18]	Function Part Detection based Trajectory Imitation (Ours)
Number of Demonstration for a Single Task	398~909	60~319	1	25-30	10	1~2
Learning Without Prior Manipulation Knowledge	✓	✓	✗	✓	✓	✓
Generalization to New Instances	✗	✗	✗	✗	✗	✓
Explainable Outcomes	✗	✗	✗	✗	✓	✓
Handling Position Variation	✓	✓	✓	✓	✓	✓
Multi-task Detection	✓	✗	✗	✗	✗	✓

Table 7.4: Comparison of proposed system with existing robot manipulation teaching approaches.

augmented samples, the proposed system is able to detect corresponding function parts of the given input object-pairs and handle the position variation. Finally, we have shown that the proposed function part detection base trajectory imitation method can guide the HSR robot to use new instances to perform manipulation tasks.

To evaluate the function part detection based trajectory imitation method proposed in the thesis against the other state-of-art approaches, a comparison is shown in Table 7.4. From the table, we can see that the behavioral cloning based imitation approaches proposed by Rahmatizadeh et al. [124] and Zhang et al. [52] require many human demonstrations for a single task to obtain state-action pairs for network training. On the other hand, although the approach presented by Yu et al. [174] only needs a few demonstrations for learning, they require prior manipulation knowledge, in other words, the system must be trained on the same task in previous. For example, in order to teach the robot the placing, pushing, and pick and place tasks with one demonstration, a user must provide 1,293, 640, and 1,008 demonstrations for meta-training on the same tasks. During experiments, their method is evaluated by changing the manipulated object on these tasks. Since the

performance of the behavioral cloning based approaches heavily rely on the amount of demonstrations, users have to provide a lot of demonstrations for training.

On the other hand, the inverse reinforcement learning based imitation method presented by Finn et al. [128] has cut down the number of demonstrations needed for training, they introduced a deep spatial autoencoder [127] that can extract visual features from image sequences that contain the robot motions for the cost function estimation. The cost function is then be used to obtain optimal policies for different tasks. However, the visual features extracted by the deep spatial autoencoder are easily affected by environment variations such as the change of lighting conditions. While the proposed method does not suffer this problem even the number of demonstrations given by users are a few. As a trade-off, the proposed method has sacrificed the dynamic response to the environment changes during the task reproduction process to enhance the user experience by cutting down the number of human demonstrations. A similar approach is proposed by Welschhold et al. [18], during teaching, their method extracts the demonstrator's hand pose trajectory and object pose trajectory. Their system then learns the correspondence of these two trajectories so that the robot can manipulate the demonstration object compliantly using an optimal trajectory. However, the learned trajectory is attached to the demonstration instances, and thus these trajectories can not be used in other instances with different shapes and sizes.

The approaches in [124, 52, 174, 128] also have the generalization problem because these methods take RGB images as inputs for network training, when the target objects' appearances are changed, their approaches will fail to find out the target objects. The proposed system solved this problem by introducing a semantic instance segmentation component to find out different instances in a category, then the Object-pair Mutual Function Knowledge Acquisition Network predicts the key object state trajectory to the function part of the object, this mechanism allows the proposed system to generate the learned manipulation skills to instances with different sizes and shapes.

Explainable demonstration outcomes are also important in a robot manipulation teaching system because users need to know what has been learned by the robot and what will the robot do with the target objects. In the behavioral cloning and inverse reinforcement learning based methods[124, 52, 174, 128], the demonstration outcomes are encoded as

network parameters that are implicit to humans, and thus the robot's behavior becomes unpredictable. This unpredictable behavior can lead to catastrophic failure. the multi-task detection capability is also desired for a system to handle multiple manipulation tasks, the trained network in [52, 174, 128] or the system in [18] do not have the task detection ability and thus these methods need extra training steps or codes to let a robot to perform different tasks. While the proposed system can handle multi-task by detecting functions on the target objects. Considering the criteria shown in Table 7.4, the proposed system has overperformed existing approaches by allowing the robot to handle variations in multiple manipulation tasks given only a few human demonstrations. This criterion is important to build a general user-oriented robot manipulation teaching system.

## 7.6 Summary

This chapter first introduced the HSR robot platform, as well as its hardware configuration and software configuration. then it showed the details of how to integrate the state trajectory learning model with other components to build a whole system that drives the HSR robot to execute multiple manipulation tasks. In the experiment, the demonstrator has taught the robot how to pick and place, pouring, cleaning and cutting with only eleven demonstration samples, while the previous methods require the user to provide at least one demonstration or template for every object. As a supplement, we also showed how to teach the robot to tidy the room by picking up different objects on the floor. In conclusion, the proposed system enables users to teach robots new manipulation skills with less effort. Furthermore, after learning, the robot is able to handle the position variation, the intra-category shape variation, and initial state variation. Finally, we compared the proposed function part detection based trajectory imitation method to the existing state-of-art behavioral cloning, inverse reinforcement learning, and instance matching based trajectory imitation methods, results show that the proposed system has overperformed existing approaches in general user-oriented robot manipulation teaching.

## 第8章

## Conclusion



## 8.1 Conclusion

This thesis has proposed a robot manipulation teaching system and a function part detection based trajectory imitation method to reduce user effort in robot teaching. In particular, the Object-pair Mutual Function Knowledge Acquisition Network is able to learn the correspondence of the offset poses to the position variation and the correspondence of the offset poses to different functions, while the proposed augmentation framework generates enough demonstrations from a few original samples for training the network. These two components enable a robot to learn new manipulation knowledge from a few demonstrations, these demonstrations can be given on-site interaction or virtual teaching. Furthermore, the demonstration samples given by users are automatically augmented by the system so that users do not need to act the same task many times for sample collection. Since the system adopts the object function knowledge-based method, when reproducing the manipulation skill, the system is able to generalize the skill to new instances without extra demonstrations.

As home-assistant robots enter the living environment, they will have more chances of learning the usages of different objects. By enabling robots to learn manipulation skills from a few demonstrations, the system allows general users to teach robots in an intuitive and simple way without the help of experts.

We summarize the main idea of each chapter as follows:

- Chapter 1 described the social background of robotics research, it explained why robot learning from human teaching is important for a robot to cooperate with human beings. It is difficult for robot programmers to preprogram all applications in daily lives, a robot teaching system enables general users who have less experience and knowledge in robotics to teach robots to perform the desired manipulation tasks, but the existing robot teaching system requires a lot of effort in teaching robot new manipulation knowledge. Therefore, the goal of this thesis is to build a robot teaching system that can reduce user effort in robot teaching by enabling robots to learn manipulation skills from a few human demonstrations and generalize the knowledge to new instances. The chapter then mentioned some background knowledge and an overview of the proposed framework. Finally,

it summarized the major contributions of this thesis and the main content of each chapter.

- Chapter 2 first reviewed the existing state-of-the-art approaches in the relevant research areas by analyzing the advantages and shortcomings of these approaches. For example, kinesthetic teaching and teleoperation are simple to implement, but they require the user to maneuver the robot joints. The robot learning-by-watching method is an ideal robot teaching method, but extracting visual features from observation is a challenging problem. The manipulation knowledge model determines what can be learned by the robot, the most widely used knowledge variables in previous methods are motion trajectory, object shape, and 6D poses, which are difficult to generate to new instances. Manipulation learning ability and generalizability are two important factors in robot manipulation teaching, the former describes how well a system learns from human teaching while the latter describes how well a system reproduces a manipulation task. After reviewing the current state-of-the-art approaches in relevant fields, this thesis proposed a function part detection based trajectory imitation learning problem which is positioned as middle manipulation learning ability but high generalizability.
- Chapter 3 explained the definition and representation of the Object-pair Mutual Function Knowledge model. This knowledge model has four elements,  $(O_{ac-as}, F, P, T)$ , where  $O_{ac-as}$  is the object pair,  $O_{ac}$  is the activated object that contains the main function  $F$ ,  $O_{as}$  is the associated object that cooperates with the activated object to complete a function  $F$ ,  $P$  indicates which parts of the objects are related to the function  $F$ , and  $T$  is the motion trajectory to tell a robot how to perform the task. Compared to the previous methods that the robot's manipulation behavior is triggered by specific object shapes, the proposed knowledge model enables robots to trigger their manipulation behaviors based on the object function part. This characteristic allows robots to apply their learned manipulation knowledge to new instances that have similar function parts. But this characteristic also brings the faulty generalization problem, we solved this problem during the task reproduction phase by applying a semantic instance segmentation component.
- Chapter 4 presented two teaching approaches that allow users to teach robots new manipulation knowledge in a transparent way. The on-site teaching approach has three com-



ponents: (1) the handheld object extraction component can extract the object held by the human hand. (2) The object function part extraction component can extract the object's function part through the human's hand gesture. (3) The trajectory extraction components can extract the grasped object's pose trajectory and the corresponding key states. This chapter then presents a virtual teaching tool that allows users to teach a robot without preparing real objects, the virtual teaching tool can also be used to handle situations that using hand gestures are difficult to teach. Previous transparent and visual teaching systems only provide human hand positions in 3D space, and thus the robot can only perform simple tasks by following specified trajectories. By integrating the hand keypoint detector into the proposed framework. The proposed system is able to extract rich visual variables such as object point clouds, object function parts, and object pose trajectories when performing a task.

- Chapter 5 presented a demonstration sample augmentation framework and an Object-pair Mutual Function Knowledge Acquisition Network. The data augmentation framework is able to generate a large number of demonstration samples with different relative object positions. These augmented samples are used to train the Object-pair Mutual Function Knowledge Acquisition Network against the position variation. From the results, we can see that in the pouring task, the network needs at least 55 augmented samples for learning the relationship between the object position and the corresponding pouring trajectory. This chapter also evaluated the relationship between the number of augmented demonstration and training time for convergence, from the results, we can see that around 15 minutes are needed to train the network with 800 augmented sample on a GTX 1080Ti GPU.
- Chapter 6 proposed a framework for manipulation task reproduction, this framework has three main components: (1) A semantic instance segmentation network that can detect target objects according to the user command inputs. (2) A point cloud processing pipeline that converts the target objects' image mask into point clouds and attaches activation signals. (3) A trajectory and grasping pose generation components that generate motion and grasp poses to guide the robot to reproduce tasks. This chapter also compared different

methods in dealing with new instances. When using the instance-based method for trajectory generation, due to misalignment the output trajectories can not be used to guide robots to perform complex tasks, otherwise, the tasks will fail. On the other hand, the object function knowledge-based method does not have this problem because it achieves its goal by detecting the object function part and automatically choose a proper trajectory based on its learned knowledge. Computing time is also important in the task reproduction phase, from the evaluation results we can see that it takes around 49ms for the object function knowledge-based method to generate trajectories through a deep neural network, while the instance-based method requires 310ms for one estimation using the ICP algorithm. As the number of templates increases, the instance-based method will take more time for the searching process. Therefore, the object function base method provides better performance when handling new instances.

- Chapter 7 first introduced the hardware and software of the HSR robot platform, as well as how to drive the robot via the ROS middleware. By integrating the proposed teaching system on the HSR robot, it is able to learn the “grasping”, “placing” and “pouring” task from virtual teaching, after learning, the robot is able to pick up teapots, cups, bottles and place them on plates, it also knows how to pick a toy apple and place it to a bowl. In the pouring tasks, the robot has successfully performed the pouring action using different bottles and cups, it also generalized the skill to pour from a bottle to a bowl, which is not provided in the demonstrations. As the system did not learn that pouring water into an upside-down cup is prohibited, the demonstrator has given it a negative sample. After augmentation and learning, the robot is able to distinguish the cup in different states. Finally, the demonstrator taught the robot how to “clean” a dirty tissue using a sticky roller and how to “cut” a pudding using a knife, after a one-time demonstration, the robot is able to perform the “cleaning” and “cutting” tasks using tools with different shapes. From the experiments, we can see that although the demonstrator provides eleven visual demonstrations, the robot has learned to use new instances to perform the corresponding manipulation tasks and handling the position variation, the intra-category shape variation, and the initial state variation. As a supplement, we also showed how to use the proposed system to teach a robot to tidy a room by picking up objects on the floor and placing these

Tasks	Placing (Table)	Pouring	Cutting	Cleaning
Number of Demonstration Objects	4	4	2	2
Number of Test Objects	16	15	3	2
Number of Trials	10	10	5	5
Number of Success	10	8	5	5
Tasks	Putting an object into a Drawer	Placing a Bag	Hanging an umbrella	Placing a cloth
Number of Demonstration Objects	2	2	2	2
Number of Test Objects	2	2	2	2
Number of Trials	5	5	5	5
Number of Success	5	5	5	5

Table 8.1: A summary of the tasks described in Chapter 7. The failure reason in pouring task is described in Chapter 6.5.

objects in proper places. A summary of the experiments is shown in Table 8.1, for each task, we have run multiple trials during testing. As the proposed system is able to predict the correspondence of the offset poses and function parts, the robot has achieved success in most of the tasks. However, due to the grasping deviation problem described in Chapter 6.5, failures may occur in the position sensitive tasks such as the pouring task. Finally, the chapter compared the proposed system to the current state-of-art approaches and results showed the proposed system outperforms other approaches in general user-oriented robot manipulation teaching.

## 8.2 Future work

This section describes some ways that the proposed approach can be improved in the future.

### Post Grasp Pose Recognition

In manipulation tasks, the change of object state after the robot grasped the object may cause failure. One way to solve this problem is to add a post grasp pose recognition which is used to correct the grasp pose after the robot gripper closed. Then we can recalculate the transformation between the real grasp pose and the start state, and apply the transformation to the rest states. The major challenge of this component is that after the robot gripper closed, the object is occluded by the robot gripper and the robot's head-mounted RGB-D camera cannot capture the complete point cloud of the object, therefore, computing this change is difficult.

### On-site Demonstration for Large Objects

Currently, the on-site demonstration framework requires the target objects must be handheldable. However, for large objects such as chairs and tables, we can only use the virtual teaching method. One solution to solve this problem is adding some specified gestures to create a bounding box for large areas, but this method may bring outliers points. Thus how to specify a large area using gestures and how to segment the desired points from the background is a challenging problem.

### Manipulation Goal Understanding

To achieve true imitation, robots have to understand humans' intentions behind their behaviors. However, existing imitation learning methods achieve the goal of understanding by predefining reward functions and apply the reinforcement learning method to learn

the reward functions. Since the reward functions are different in every manipulation task, it is a difficult problem to design a general reward to represent all the manipulation tasks.



## 参考文献





- [1] H. Inoue, S. Tachi, K. Tanie, K. Yokoi, S. Hirai, H. Hirukawa, K. Hirai, S. Nakayama, K. Sawada, T. Nishiyama, O. Miki, T. Itoko, H. Inaba, and M. Sudo. Hrp: Humaonid robotics project of miti. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2000.
- [2] K. Sasabuchi, H. Yaguchi, K. Nagahama, S. Hori, H. Mizohana, and M. Inaba. The seednoid robot platform: Designing a multipurpose compact robot from continuous evaluation and lessons from competitions. In *IEEE Robotics and Automation Letters (RA-L)*, 2018.
- [3] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent asimo: System overview and integration. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2002.
- [4] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase. Development of human support robot as the research platform of a domestic mobile manipulator. In *ROBOMECH Journal*, 2019.
- [5] K. Yamazaki, R. Ueda, S. Nozawa, M. Kojima, K. Okada, K. Matsumoto, M. Ishikawa, I. Shimoyama, , and M. Inaba. Home-assistant robot for an aging society. In *Proceedings of the IEEE*, 2012.
- [6] M. Kojima, K. Okada, and M. Inaba. Manipulation and recognition of objects incorporating joints by a humanoid robot for daily assistive tasks. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.
- [7] K. Okada, M. Kojima, S. Tokutsu, T. Maki, Y. Mori, and M. Inaba. Multi-cue 3d object recognition in knowledge-based vision-guided humanoid robot system. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.
- [8] K. Yamazaki, R. Ueda, S. Nozawa, Y. Mori, T. Maki, N. Hatao, K. Okada, and M. Inaba. System integration of a daily assistive robot and its application to tidying and cleaning rooms. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2010.

- [9] K. Okada, M. Kojima, S. Tokutsu, Y. Mori, T. Maki, and M. Inaba. Task guided attention control and visual verification in tea serving by the daily assistive humanoid hrp2jsk. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.
- [10] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2011.
- [11] M. Bollini. Following recipes with a cooking robot. *Master's thesis, MIT*, 2012.
- [12] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In *AAAI*, 2015.
- [13] S. Miller, J. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research (IJRR)*, 2012.
- [14] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen. Multi-sensor surface analysis for robotic ironing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.
- [15] S. Ambhore. A comprehensive study on robot learning from demonstration. In *International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020.
- [16] H. Urbanek, A. Albu-Schaffer, and P. van der Smagt. Learning from demonstration: repetitive movements for autonomous service robotics. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.
- [17] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots. Learning generalizable robot skills from demonstrations in cluttered environments. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.

- [18] T. Welschehold, C. Dornhege, and W. Burgard. Learning manipulation actions from human demonstrations. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [19] C.G. Atkeson and S. Schaal. Learning tasks from a single demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.
- [20] J. Jin, L. Petrich, M. Dehghan, Z. Zhang, and M. Jagersand. Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.
- [21] L. Rozo, P. Jimenez, and C. Torras. Robot learning from demonstration of force-based tasks with multiple solution trajectories. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [22] H. Dang and P. K. Allen. Robot learning of everyday object manipulations via human demonstration. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2010.
- [23] B. Reiner, W. Ertel, H. Posenauer, and M. Schneider. Lat: A simple learning from demonstration method. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [24] A. L. Thomaz and M. Cakmak. Learning about objects with human teachers. In *IEEE International Conference on Human-Robot Interaction (HRI)*, 2009.
- [25] N. Figueroa, A. L. P. Ureche, and A. Billard. Learning complex sequential tasks from demonstration: A pizza dough rolling case study. In *IEEE International Conference on Human-Robot Interaction (HRI)*, 2016.
- [26] E. Misimi, A. Olofsson, A. Eilertsen, E. R. ye, and J. R. Mathiassen. Robotic handling of compliant food objects by robust learning from demonstration. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.

- [27] H-I. Lin and Y-H. Lin. A novel teaching system for industrial robots. *Sensors*, Vol. 14, No. 4, pp. 6012–6031, 2014.
- [28] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Recent advances in robot learning from demonstration. In *Annual Review of Control Robotics and Autonomous Systems*, pp. 297–330, 2020.
- [29] J. Kober, J.A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. In *The International Journal of Robotics Research*, pp. 32:1238–1274, 2013.
- [30] B.D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. In *Robotics and Autonomous Systems*, pp. 57:469–483, 2009.
- [31] J. Miura and K. Ikeuchi. Task-oriented generation of visual sensing strategies in assembly tasks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.
- [32] A. Krizhevsky, Sutskever, Ilya, Hinton, and Geoffrey E. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.
- [33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [35] D.Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and et al. M. Lanctot. Mastering the game of go with deep neural networks and tree search. *Nature*, Vol. 529(7587), pp. 484–489, 2016a.

- [36] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, Vol. 550, pp. 354–359, 2017.
- [37] Openai five. <https://openai.com/blog/openai-five/>, 2018.
- [38] Y. Kuniyoshi. Emergence and development of imitation. In *International Workshop on Robotic Imitation IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.
- [39] R. Fukano, Y. Kuniyoshi, and A. Nagakubo. A cognitive architecture for flexible imitative interaction using tools and objects. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006.
- [40] S.R. Ahmadzadeh, R. Kaushik, and S. Chernova. Trajectory learning from demonstration with canal surfaces: A parameter-free approach. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 544–549, 2016.
- [41] S. Elliott, Z. Xu, and M. Cakmak. Learning generalizable surface cleaning actions from demonstration. In *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017.
- [42] T. Osa, K. Harada, N. Sugita, and M. Mitsuishi. Trajectory planning under different initial conditions for surgical task automation by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6507–6513, 2014.
- [43] S. Calinon, I. Sardellitti, and D.G. Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2010.
- [44] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. In *IEEE Transactions on Robotics*, pp. 27:943–957, 2011.

- [45] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: extracting reusable task knowledge from visual observation of human performance. In *IEEE Transactions on Robotics and Automation*, 1994.
- [46] Y. Kuniyoshi, M. Inaba, and H. Inoue. Seeing, understanding and doing human task. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1992.
- [47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [48] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision (ECCV)*, pp. 21–37, 2016.
- [50] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [51] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.
- [52] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [53] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning (CoRL)*, 2019.

- [54] S. Nozawa, M. Murooka, S. Noda, K. Okada, and M. Inaba. Description and execution of humanoid's object manipulation based on object-environment-robot contact states. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [55] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006.
- [56] M. Kojima, K. Okada, and M. Inaba. Learning by observation of furniture manipulation in daily assistive tasks for humanoid robots. In *International Conference on Intelligent Autonomous Systems*, 2008.
- [57] M. Murooka, S. Noda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Manipulation strategy decision and execution based on strategy proving operation for carrying large and heavy objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
- [58] M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Whole-body pushing manipulation with contact posture planning of large and heavy object for humanoid robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [59] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, No. 2, pp. 239–256, 1992.
- [60] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *International Conference on Computer Vision*, 2011.
- [61] K. Yamazaki, R. Ueda, S. Nozawa, Y. Mori, T. Maki, N. Hatao, K. Okada, and M. Inaba. Tidying and cleaning rooms by a daily assistive robot - an integrated system for doing chores in real world. In *Journal of Behavioral Robotics*, 2011.

- [62] K. Nagahama, K. Yamazaki, K. Okada, and M. Inaba. Hierarchical estimation of multiple objects from proximity relationships arising from tool manipulation. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012.
- [63] A. Myers, C. L. Teo, C. Fermuller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1374–1381, 2015.
- [64] A. Nguyen, D. Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. Object-based affordances detection with convolutional neural networks and dense conditional random fields. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 5908–5915, 2017.
- [65] T.-T. Do, A. Nguyen, I. Reid, D. G. Caldwell, and N. G. Tsagarakis. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *arXiv preprint arXiv:1709.07326*, 2017.
- [66] K. Chaudhary, X. Chen, K. Okada, and M. Inaba. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [67] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting object affordances with convolutional neural networks. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 2765–2770, 2016.
- [68] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. In *arXiv preprint arXiv:1505.07293*, 2015.
- [69] N. Ratliff, M. Zucker, JA. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.



- [70] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [71] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems (RSS)*, 2013.
- [72] T. Yu, P. Abbeel, S. Levine, and C. Finn. One-shot hierarchical imitation learning of compound visuomotor tasks. In *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [73] A. Lee, A. Gupta, H. Lu, S. Levine, and P. Abbeel. Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2015.
- [74] Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [75] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2017.
- [76] B. Stadie, P. Abbeel, and I. Sutskever. Third person imitation learning. In *International Conference on Learning Representations (ICLR)*, 2017.
- [77] C. Devin, P. Abbeel, T. Darrell, and S. Levine. Deep object-centric representations for generalizable robot learning. In *International Conference on Learning Representations (ICLR)*, 2018.
- [78] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectory and keyframes for kinesthetic teaching a human-robot interaction perspective. In *IEEE International Conference on Human-Robot Interaction (HRI)*, 2012.

- [79] J-P Hwang, S. H. Lee, and I. H. Suh. Behavior programming by kinesthetic demonstration for a chef robot. In *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011.
- [80] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. In *IEEE Transactions on Robotics*, 2008.
- [81] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss. Learning manipulation actions from a few demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [82] T. Tang, H-C Lin, Y. Zhao, Y. Fan, W. Chen, and M. Tomizuka. Teach industrial robots peg-hole-insertion by human demonstration. In *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2016.
- [83] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra. First-person teleoperation of a humanoid robot. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 997–1002, 2015.
- [84] J. I. Lipton, A. J. Fay, and D. Rus. Baxter’s homunculus: Virtual reality spaces for teleoperation in manufacturing. In *IEEE Robotics and Automation Letters*, pp. 179–186, 2018.
- [85] V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 657–663, 2015.
- [86] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. In *arXiv preprint arXiv:1708.03655*, 2017.
- [87] Dillmann R. Teaching and learning of robot tasks via observation of human performance. In *Robotics and Autonomous Systems*, 2004.

- [88] Scassellati B Hayes B. Discovering task constraints through observation and active learning. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2014.
- [89] Rahmatizadeh R, Abolghasemi P, Boloni L, and Levine S. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [90] Li Y, Song J, and Ermon S. Infogail: Interpretable imitation learning from visual demonstrations. In *Neural Information Processing Systems (NIPS)*, 2017.
- [91] Y. Kuniyoshi, H. Inoue, and M. Inaba. Teaching by showing: Generating robot command sequences based on real time visual recognition of human pick and place actions. In *Journal of the Robotics Society of Japan*, 1991.
- [92] Y. Kuniyoshi and H. Inoue. Qualitative recognition of ongoing human action sequences. In *International Joint Conference on Artificial Intelligence*, 1993.
- [93] SP. Chatzis, D. Korkinof, and Y. Demiris. A nonparametric bayesian approach toward robot learning by demonstration. In *Robotics and Autonomous Systems*, pp. 60:789–802, 2012.
- [94] A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Neural Information Processing Systems*, 2013.
- [95] N. Perrin and P. Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. In *Systems and Control Letters*, pp. 96:51–59, 2016.
- [96] M.A. Rana, M. Mukadam, S.R. Ahmadzadeh, S. Chernova, and B. Boots. Towards robust skill generalization: Unifying learning from demonstration and motion planning. In *Conference on Robot Learning (CoRL)*, pp. 109–118, 2017.

- [97] H. Ravichandar, S.R. Ahmadzadeh, M.A. Rana, and S. Chernova. Skill acquisition via automated multicoordinate cost balancing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.
- [98] H. Ravichandar, I. Salehi, and A.P. Dani. Learning partially contracting dynamical systems from demonstrations. In *Conference on Robot Learning (CoRL)*, 2017.
- [99] S. Manschitz, M. Gienger, J. Kober, and J. Peters. Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations. In *IEEE Robotics and Automation Letters*, pp. 3:926–933, 2018.
- [100] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.
- [101] H. Ravichandar and A. Dani. Learning position and orientation dynamics from demonstrations via contraction analysis. In *Autonomous Robots*, 2018.
- [102] J. Silvrio, L. Rozo, S. Calinon, and D.G. Caldwell. Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2015.
- [103] Y. Lin, S. Ren, M. Clevenger, and Y. Sun. Learning grasping force from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.
- [104] K. Kronander and A. Billard. Online learning of varying stiffness through physical human-robot interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.
- [105] M. Li, H. Yin, K. Tahara, and A. Billard. Learning object-level impedance control for robust grasping and dexterous manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

- [106] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [107] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, J. Schneider, P. Welinder, W. Zaremba, and P. Abbeel. Domain randomization and generative models for robotic grasping. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [108] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstration. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [109] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [110] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel. Learning robotic assembly from cad. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.
- [111] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-End Training of Deep Visuomotor Policies. *theJournal of Machine Learning Research (JMLR)*, 2016.
- [112] T. Lillicrap S. Gu, E. Holly and S. Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [113] A. Nair, P. Agrawal, D. Chen, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [114] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar. Learning robotic manipulation through visual planning and acting. In *Robotics: Science and Systems (RSS)*, 2019.

- [115] S. James J. Matas and A. J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning (CoRL)*, 2018.
- [116] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1999.
- [117] I. Bratko, T. Urbancic, and C. Sammut. Behavioural cloning of control skill. In *Machine Learning and Data Mining: Methods and Applications*, 1998.
- [118] Dean A. Pomerleau. Alvin: an autonomous land vehicle in a neural network. In *Advances in neural information processing Systems (NIPS)*, 1989.
- [119] M. W. Kadous, C. Sammut, and R. Sheh. Behavioural cloning for robots in unstructured environments. In *Advances in Neural Information Processing Systems Workshop*, 2005.
- [120] R. Sheh, B. Hengst, and C. Sammut. Behavioural cloning for driving robots over rough terrain. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011.
- [121] J. Monteiro, N. Gavenski, R. Granada, F. Meneguzzi, and R. Barros. Augmented behavioral cloning from observation. In *International Joint Conference on Neural Networks (IJCNN)*, 2020.
- [122] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*, 2018.
- [123] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning (CoRL)*, 2017.
- [124] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

- 
- [125] Andrew Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000.
- [126] P. Abbeel and Andrew. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.
- [127] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [128] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning (ICML)*, 2016.
- [129] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.
- [130] K. Yamazaki, K. Nagahama, and M. Inaba. A method of state recognition of dressing clothes based on dynamic state matching. In *IEEE/SICE International Symposium on System Integration*, 2013.
- [131] K. Yamazaki, T. Nishino, K. Nagahama, Kei Okada, and Masayuki Inaba. A vision system for daily assistive robots using character information in daily environments. In *IEEE/SICE International Symposium on System Integration*, 2013.
- [132] A. Saxena, J. Driemeyer, and A. Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research (IJRR)*, Vol. 27, No. 2, pp. 157–173, 2008.
- [133] Y. Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgbdimages: Learning using a new rectangle representation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3304–3311, 2011.

- [134] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1316–1322, 2015.
- [135] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. In *Robotics: Science and Systems (RSS)*, 2013.
- [136] S. Kumra and C. Kanan. Robotic grasp detection using deep convolutional neural networks. In *IEEE Conference on Intelligent Robots and Systems (IROS)*, pp. 769–776, 2017.
- [137] L. Trottier, P. Giguere, and B. Chaib-draa. Sparse dictionary learning for identifying grasp locations. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 871–879, 2017.
- [138] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi. A hybrid deep architecture for robotic grasp detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1609–1614, 2017.
- [139] M. A. Goodale and A.D. Milner. Separate visual pathways for perception and action. In *Trends Neurosci*, 1992.
- [140] F. Osiurak, Y. Rossetti, and A. Badets. What is an affordance? 40 years later. In *Neuroscience Biobehavioral Reviews*, 2017.
- [141] P. Bach, T. Nicholson, and M. Hudson. The affordance-matching hypothesis: how objects guide action understanding and prediction. In *Frontiers in human neuroscience*, 2014.
- [142] C. Nabeshima, Y. Kuniyoshi, and M. Lungarella. Towards a model for tool-body assimilation and adaptive tool-use. In *IEEE International Conference on Development and Learning*, 2007.



- [143] C. Nabeshima, M. Lungarella, and Y. Kuniyoshi. Timing-based model of body schema adaptation and its role in perception and tool use: A robot case study. In *IEEE International Conference on Development and Learning*, 2005.
- [144] S. Calinon, T. Alizadeh, and D. G. Caldwell. On improving the extrapolation capability of task-parameterized movement models. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013.
- [145] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [146] D. Constantinescu, C. Daniel, and E. Croft. Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. In *Journal of Robotic Systems*, 2000.
- [147] T. Zhang, M. Zhang, and Y. Zou. Time-optimal and smooth trajectory planning for robot manipulators. In *International Journal of Control, Automation and Systems*, 2020.
- [148] X. Gao, Y. Mu, and Y. Gao. Optimal trajectory planning for robotic manipulators using improved teaching-learning-based optimization algorithm. In *Industrial Robot: An International Journal*, 2016.
- [149] C. P. Quintero, S. Li, M. KXJ Pan, W. P. Chan, H.F. Machiel Van der Loos, and E. Croft. Robot programming through augmented trajectories in augmented reality. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [150] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012.
- [151] J. Kober and J. Peters. Learning motor primitives for robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2112–2118, 2009.

- [152] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth. Probabilistic model-based imitation learning. In *Adaptive Behavior*, pp. 21:388–403, 2013.
- [153] Z. Cao, T. Simon, S-E Wei, and Y. Sheikh. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [154] Y. Wang, B. Zhang, and C. Peng. Srhandnet: Real-time 2d hand pose estimation with simultaneous region localization. In *IEEE Transactions on Image Processing (TIP)*, pp. 2977–2986, 2020.
- [155] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015.
- [156] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. In *arXiv preprint arXiv:1512.03012*, 2015.
- [157] A. Janoch, S. Karayev, Y. Jia, J.T. Barron, M. Fritz, K. Saenko, and T. Darrell. A Category-Level 3-D Object Dataset: Putting the Kinect to Work. In *ICCV Workshop on Consumer Depth Cameras in Computer Vision*, 2011.
- [158] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *European Conference on Computer Vision (ECCV)*, 2012.
- [159] J. Xiao, A. Owens, and A. Torralba. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.
- [160] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

- [161] E. Catmull. A Subdivision Algorithm for Computer Display of Curved Surfaces. *PhD Thesis, Dept of Computer Science, University of Utah*, 1974.
- [162] D. Shreiner, G. Sellers, J. M. Kessenich, and B. M. Licea-Kane. *Programming Guide: The Official Guide to Learning OpenGL, Version 4.3, 8th*. ISBN:0321773039 9780321773036, 2013.
- [163] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 2007.
- [164] Vott. <https://github.com/microsoft/VoTT/>.
- [165] Librviz. [http://docs.ros.org/jade/api/librviz\\_tutorial/html/index.html](http://docs.ros.org/jade/api/librviz_tutorial/html/index.html).
- [166] Da vinci. <https://danielgm.net/cc/>.
- [167] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. In *Journal of Big Data*, 2019.
- [168] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, and E. D. Cubuk. SpecAugment: A simple data augmentation method for automatic speech recognition. In *INTERSPEECH 2019*, 2019.
- [169] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In *ArXiv e-print*, 2017.
- [170] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Bochoon, and S. Birchfield. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [171] K. Ogawara, J. Takamatsu, H. Kimurat, and K. Ikeuchi. Generation of a task model by integrating multiple observations of human demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

- [172] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *ArXiv preprint arXiv:1612.00593*, 2016.
- [173] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *arXiv preprint arXiv:1706.02413*, 2017.
- [174] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems (RSS)*, 2018.
- [175] C. Reed G., T. Lozano-Pere, and L. P. Kaelbling. Sampling-based methods for factored task and motion planning. In *International Journal of Robotics Research*, 2018.
- [176] K. Nagahama, K. Yamazaki, K. Okada, and M. Inaba. Manipulation of multiple objects in close proximity based on visual hierarchical relationships. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [177] K. Nagahama and K. Yamazaki. Learning from demonstration based on a mechanism to utilize an object’s invisibility. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [178] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [179] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards realtime object detection with region proposal networks. In *Neural Information Processing Systems (NIPS)*, 2015.
- [180] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *arXiv preprint arXiv:1512.00567*, 2015.

- 
- [181] A. Martn, B. Paul, C. Jianmin, C. Zhifeng, et al. Tensorflow: A system for large-scale machine learning. In *Symposium on Operating Systems Design and Implementation*, 2016.
- [182] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.
- [183] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.



# Acknowledgement

I would like to thank my supervisor Prof. Kei Okada for giving me the opportunity to conduct robotics experiment at the JSK lab with much freedom. I would like to thank him for his meticulous guidance and continuous support throughout my doctoral studies. Without his advice, this dissertation would not have been complete.

I would like to thank Prof. Inaba for his insightful advice that has improved my knowledge and skills throughout my doctoral studies.

I would like to thank Iori Yanokura for helping me with the final experimentation. I would like to thank Feiyu Zhao for his encouragement.

I would like to thank all the members in the lab for their friendly and kindness that making JSK a great place for studying.

Finally, I would like to thank my family for the love, encouragement and support I have gotten over the years. I would like to thank my fiancée, Cui Liyuan for her unconditional support in these years.

2020 年 11 月 1 日 李 梓佳

## Publication List

### Conference

**Zijia Li**, Kei Okada and Masayuki Inaba. Acquiring Mechanical Knowledge from 3D Point Clouds. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020) pp. 8065-8072. **(Best paper award on cognitive robotics finalist)**.

**Zijia Li**, Kei Okada and Masayuki Inaba. Searching a Suitable Keypoint Detection Network for Robotic Assembly. 2020 IEEE/SICE International Symposium on System Integration. (SII 2020) Hawaii, USA. pp. 377-383.

**Zijia Li**, Kei Okada and Masayuki Inaba. Affordance Action Learning with State Trajectory Representation for Robotic Manipulation. 2019 IEEE-RAS 19th International Conference on Humanoid Robots. (Humanoids 2019) Toronto, Canada. pp. 638-644.

**Zijia Li**, Chi Wun Au, Yohei Kakiuchi, Kei Okada and Masayuki Inaba. What am I doing? Robotic Self-Action Recognition. 2016 IEEE-RAS 16th International Conference on Humanoid Robots. (Humanoids 2016) Cancun, Mexico. pp. 165-170.



# **Appendix A**



Sampling a subset from a point set using the iterative farthest point sampling algorithm.  
(Python code)

```
import matplotlib.pyplot as plt
import math
import random

def create_point_cloud(n):
    temp = []
    for _ in range(n):
        temp.append([2 * random.random() - 1, 2 * random.random() - 1])
    return temp

def plot_points(points ,
                ax=None ,
                style={'marker': 'o', 'color': 'b'}, label=False):

    if ax == None:
        ax = plt.gca()
    for ind, p in enumerate(points):
        ax.plot(p[0], p[1], **style)
        if label:
            ax.text(p[0],
                    p[1],
                    s=ind,
                    horizontalalignment='center',
                    verticalalignment='center')

    ax.set_xlim(-1.1,1.1)
    ax.set_ylim(-1.1,1.1)

def distance(A,B):
    return (A[0]-B[0])*(A[0]-B[0])+(A[1]-B[1])*(A[1]-B[1])

def incremental_farthest_search(points , k):
    remaining_points = points[:]
```

```

solution_set = []
init_solution=random.randint(0, len(remaining_points) -1)
solution_set.append(remaining_points.pop(init_solution))

for _ in range(k-1):
    distances=[distance(p, solution_set[0]) for p in remaining_points]
    for i, p in enumerate(remaining_points):
        for j, s in enumerate(solution_set):
            distances[i] = min(distance[i], distance(p,s))
    max_dis = max(distances)
    idx = distances.index(max_dis)
    solution_set.append(remaining_points.pop(idx))
return solution_set

if __name__ == '__main__':
    cloud100 = create_point_cloud(100)
    fig = plt.figure(figsize=(5,5))
    plot_points(cloud100)
    plot_points(incremental_farthest_search(cloud100,10),
               style={'marker': 'o', 'color': 'r','markersize':12})

```

Output results:

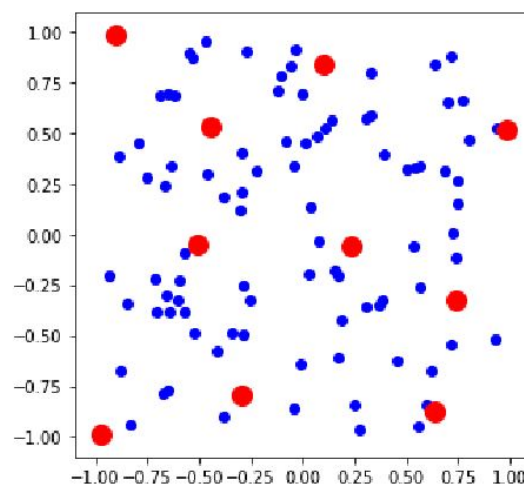


Figure 8.1: output results of the iterative farthest point sampling algorithm

## **Appendix B**



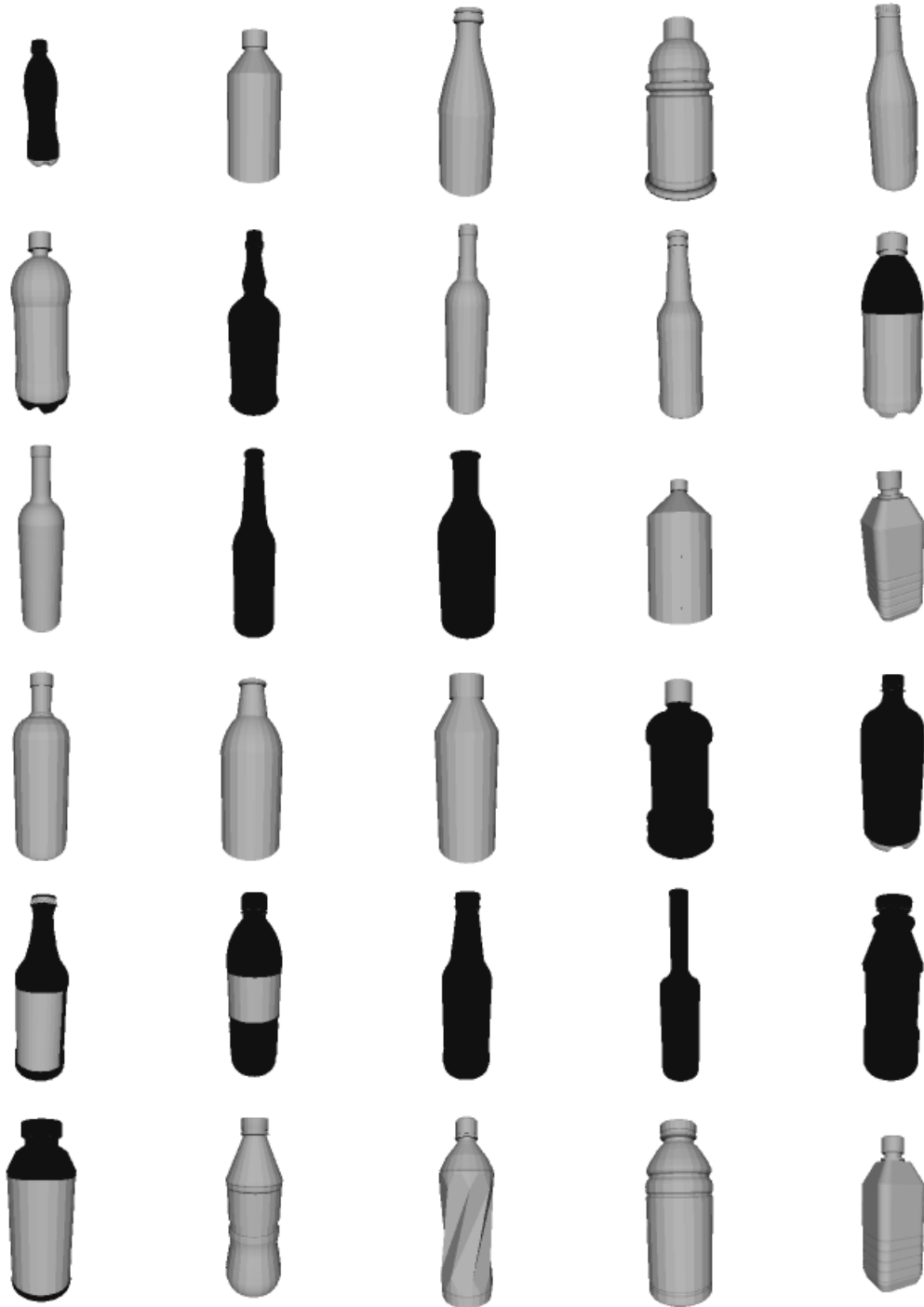


Figure 8.2: The 3D model of bottles used in chapter 6 for the ICP algorithm evaluation.

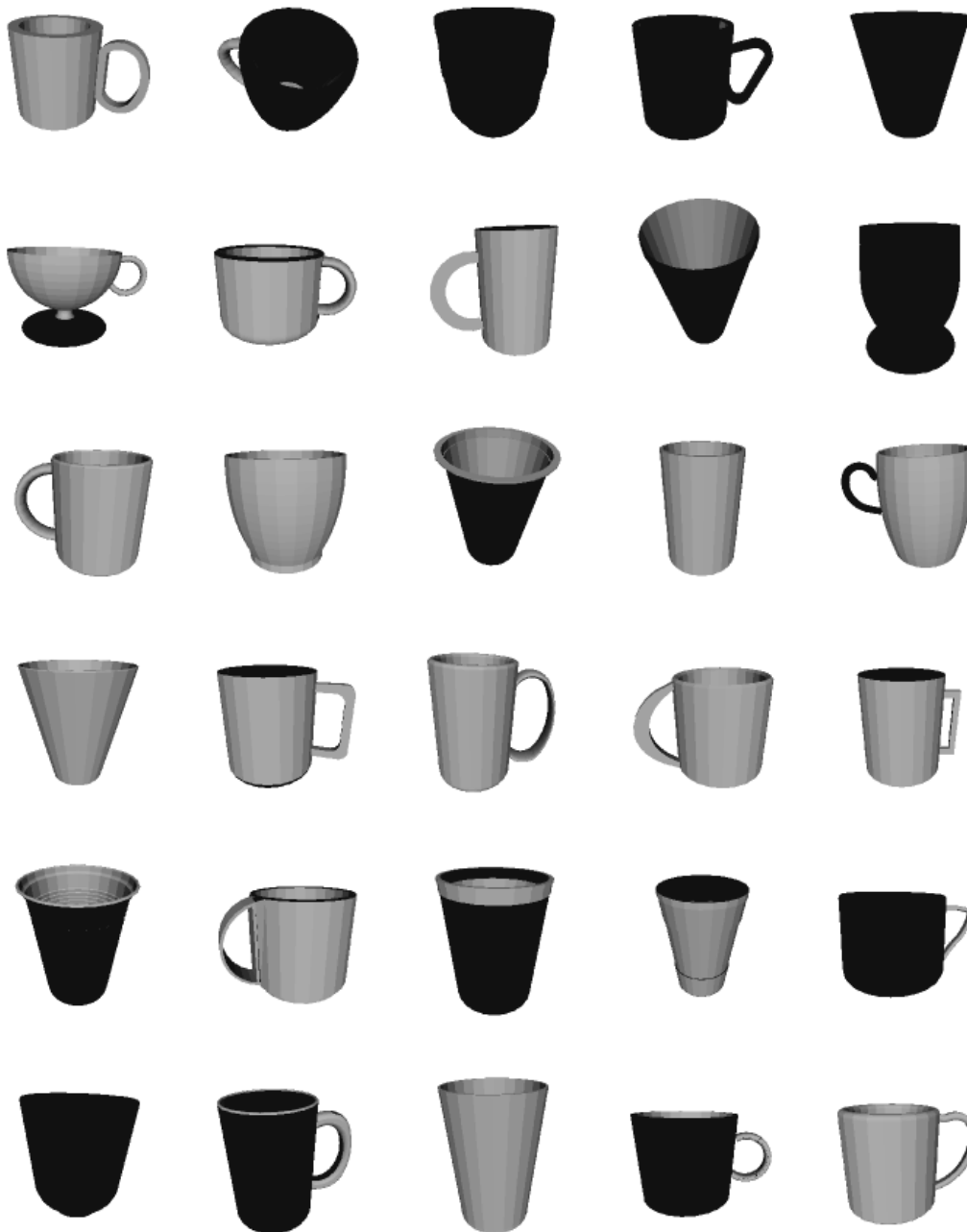


Figure 8.3: The 3D model of bottles used in chapter 6 for the ICP algorithm evaluation.





以上

1p ~ 182p 完

博士論文

令和2年12月04日提出

知能機械情報学専攻  
48177511 李 梓佳