Shibaura Institute of Technology

Learning and Executing Everyday Task from Instruction Manual and Human Demonstration

A DISSERTATION SUBMITTED TO THE
GRADUATE SCHOOL OF ENGINEERING AND SCIENCE OF THE
SHIBAURA INSTITUTE OF TECHNOLOGY

by

PHAM NGOC HUNG

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF ENGINEERING

SEPTEMBER 2017

Abstract

Robots have been widely used in the industrial applications where they are often pre-programmed in a well-defined and controlled environment. With the significant improvements of robotic technology nowadays, many special-purpose robots are entering human daily life. Autonomous robots are becoming more and more skilled in performing human-scale manipulation tasks. However, everyday tasks at home demand much knowledge a robot needs to have. The main challenges facing the robots are (1) what actions the robot needs to perform in the task; (2) how to perform each action; (3) perceiving objects (identification, pose, location) for manipulation actions.

This dissertation presents a proposal for learning and executing everyday manipulation tasks which involve in operating objects or home appliances such as dispensing water from a water thermos pot, making a cup of coffee by a coffee maker, warming a lunch box by a microwave oven, ect. To solve this, firstly, the tasks are planned by the sequences of actions which are automatically acquired from instruction manual. Then, the knowledge about how to perform the action is obtained by learning from hand movement in human demonstration. A learning method of motion primitives from human hand movement using Dynamic Movement Primitives model has been implemented for generating movement trajectories adapted to new changes. In addition, this study also deployed the ability of perceiving objects for the robot to recognize object and estimate its 6-DOF pose which is used to manipulate that object. To confirm the proposal, the research in this dissertation takes into account an example scenario with the task 'dispensing water' from a water thermos pot.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor, Professor **Takashi Yoshimi**, for his guidance, support, and encouragement throughout my research at Robot Task and System Laboratory, Shibaura Institute of Technology. His continued support led me to the right way.

I would also like to extend my appreciation to the review committee members: Prof. **Tadahiro Hasegawa**, Prof. **Satoko Abiko**, Prof. **Masaomi Kimura**, and Prof. **Mihoko Niitsuma** for their advice and comments for my research. My sincere thanks to Professor **Makoto Mizukawa**, Professor **Yoshinobu Ando** for their advice and valuable lecturer what I attended.

My sincere appreciation is extended to all staff members of Shibaura Institute of Technology for their kind support during my stay in Japan.

I would also like to thank all members of Robot Task and System Laboratory. They are my good friends who shared with me for studying and enjoying the student life during my doctoral course. The enjoyable parties and summer camps are a part of my memories in Japan.

I would like to acknowledge the financial support from JICA (Japan International Cooperation Agency), AUN/SEED-Net for my doctoral course and my daily life in Japan.

Finally, on this dissertation, I would like to extend my deepest gratitude to my family for their love and encouragement during the years I stayed abroad for studying.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robots have been widely used in the industrial applications where they are often pre-programmed in a well-defined and controlled environment. With the significant improvements of robotic technology nowadays, many special-purpose robots are entering human daily life. Cognitive robots are becoming more and more skilled in performing human-scale manipulation tasks. Robots are expected to serve everyday task for human in daily life at home, especially for elderly, disabled person. However, everyday tasks at home demand much knowledge a robot needs to have. The main challenges are that the robot needs to determine the sequence of actions in the task, perform actions in the unstructured and dynamic environment, and manipulate with a variety of differing objects.

Numerous everyday tasks often involve in operating objects or home appliances, equipment such as water thermos pot, coffee maker, microwave oven, etc which are supplied with user manuals containing the instructions for certain tasks, for example dispensing some water, making a cup of coffee, cooking or warming a food, etc. This motivates to use instruction manuals as the source of knowledge to provide robots about everyday tasks. A task plan can be built from a sequence of actions achieving from instruction manual. However, only knowledge from instruction manuals is not enough for autonomously accomplishing the task. The knowledge about how to perform the action is not shown in the instruction manuals because they are common-sense knowledge for human. To overcome this challenge, learning from observing human demonstration is a possible solution to recognize how to perform an action for the robot. Figure 1.1 shows the example about actions which operate home appliances in everyday such as pick and place a cup, press a key on a water thermos pot, open a oven's door, push a button on microwave oven.

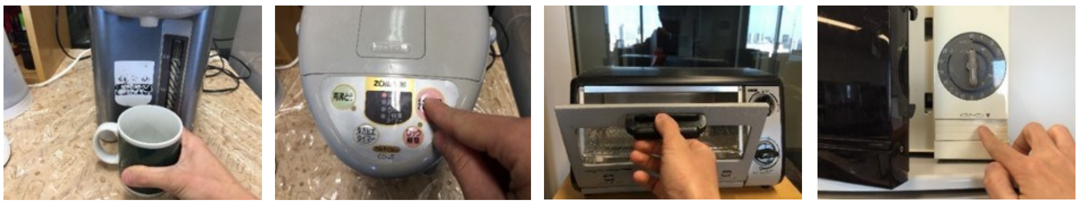


Figure 1.1: Example of actions that manipulate home appliances in daily life. From left to right: pick up a cup, press a key, open a oven's door, push a button

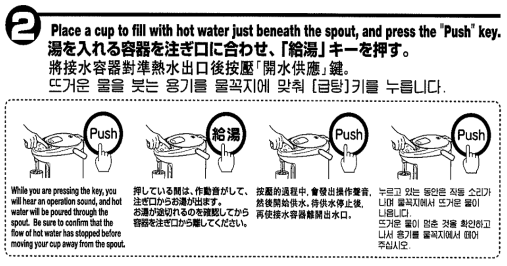
The motivation for the work described in this thesis is to develop a method for learning and executing the everyday tasks by providing the robots the knowledge on two key problems:
1. What actions the robot needs to perform in the task
2. How the robot perform the action

1.2 Example Scenario

Imagine an example scenario that a home service robot receives the command "dispensing water" or "making a cup of coffee". Given this command, it has to create a plan to achieve the desired goal, which is usually solved by planning, for example, searching for a sequence of actions and generating executable plan of each action that leads to the given goal state. However, doing this from human everyday tasks is still challenge to the capabilities of the intelligent robots nowadays. Solving this problem demands the robots the abilities to acquire actions in the task, perform actions, and perceive the object that the action operates.

A common way to build an executable plan for a robot task is generating manually the sequence of atomic actions from the initial state to the goal state to accomplish the task. This way is dependent on the specific task and not appropriate with the large number of tasks. Moreover, the more complex the task is, the more complex this gets. Instead of generating a robot plan in this way, we proposed to make use of existing description in instruction manuals of home appliances or equipment that explain how to perform a certain everyday task. The robot can use these instruction manuals to look up the sequence of actions the robot needs to perform in the task. After having read the instructions, the robot is autonomously provided a plan containing actions that it needs to perform. With the state-of-the-art techniques in natural language processing nowadays, it is feasible to equip for robots the capability of reading these instructions.

There are lots of instruction manuals equipped for home appliances which contain instructions for many everyday tasks. Figure 1.2 shows an examples of instructions taken from user manual of a coffee maker and a water thermos pot. These instructions are to guide a part of tasks 'how to dispense water' or 'how to make a cup of coffee'.



2 Place a cup to fill with hot water just beneath the spout, and press the "Push" key.
湯を入れる容器を注ぎ口に合わせ、「給湯」キーを押す。
將接水容器對準熱水出口後按壓「開水供應」鍵。
뜨거운 물을 붓는 용기를 물꼭지에 맞춰 [급탕]키를 누릅니다.

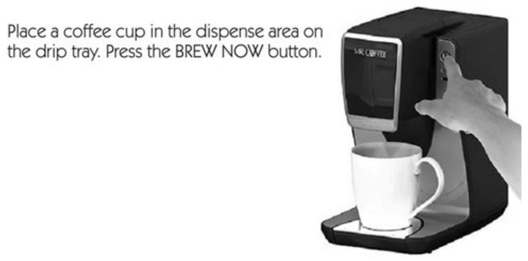
While you are pressing the key, you will hear an operation sound, and hot water will be poured through the spout. Be sure to confirm that the flow of hot water has stopped before moving your cup away from the spout.

押している間は、作動音がして、注ぎ口からお湯が出ます。
お湯が途切れるのを確認してから容器を注ぎ口から離してください。

按壓的過程中，會發出操作聲音，然後開始供水。待供水停止後，再使接水容器離開出水口。

누르고 있는 동안은 작동 소리가 나며 물꼭지에서 뜨거운 물이 나옵니다.
뜨거운 물이 멈춘 것을 확인하고 나서 용기를 물꼭지에서 떼어 주십시오.

(a) 'dispensing water' from a water thermos pot



Place a coffee cup in the dispense area on the drip tray. Press the BREW NOW button.

(b) 'making coffee' from a coffee maker

Figure 1.2: An example of instruction taken from instruction manuals of some common home appliances

However, the instructions only describe the overall series of actions on a very abstract level, which is not sufficient to actually enable a robot to perform the actions. One important reason is that they were written for humans and therefore lack of information about how to perform the action that a robot needs to be provided. Therefore, the knowledge about how to perform the action needs to be complemented from other knowledge sources such as from observations of human activities. Towards performing this example scenario, we will discuss how a robot could proceed to obtain the sequence of actions from instruction manual and how the robot perform the action from observing human demonstration.

1.3 Task, Action and Motion Primitive

In robotic research, some different terms can be used with equivalent meaning or some similar terms can be used with different meaning depending on the research context. The terms as 'task', 'action', 'motion primitive' are used frequently in this thesis will be explained as listed below.

Motion Primitive A motion primitive is a small unit of behavior that, for example, can be a part of a motion trajectory or a specific motion type such as "move the robot's end-effector from position A to position B".

Action An action consists of a sequence of motion primitives. For example, the action "pick up an object" consists of motion primitives: (1) reaching the object, (2) grasping the object, (3) withdrawing from object's location.

Task A task is composed by a sequence of actions. For example, the task "dispensing water" which is collected from instruction manual can be composed by three actions: (1) pick up a cup, (2) place the cup under the spout, (3) press the button. The task plan is dependent on the specific task.

1.4 Research Purpose

The purpose of the research conducted in this dissertation is make the robot to execute everyday manipulation task by automatically acquiring the sequence of actions what the robot needs to perform in the task from instruction manual of home appliances; and learning how to perform the action from hand movement in human demonstration using Dynamic Movement Primitives (DMP) model for generating sub-movements adapted to new changes in the execution of that action.

This is a method to build robot programs that avoids manually programming for each one specific task in static conditions. The idea is for the robot 'understand' what action the robot needs to perform in the task and 'learn' how to perform each action from human demonstration.

To manipulate the object, the robot should recognize that object identification, as well as object pose, location which also support the robot action to adapt with new changes such as the change of object's location in the action 'pick up an object'. Therefore, one work in this research is to implement and evaluate the method of object recognition and pose estimation using 3D object information for robot vision ability.

There are two original points in this research:

- The robot can acquire the sequence of action and related object from instruction manual for task planing without understanding the meaning of the task.
- The introduction of learning method of human hand movement using DMP model for robot action.

1.5 Research Overview

The main contributions of the work in this dissertation is developing a method to learn and execute everyday manipulation tasks by providing the robots the knowledge about the task from two sources of knowledge: instruction manual and human demonstration. In particular, the thesis solves three challenge problems: (1) acquiring the actions the robot needs to perform in the task; (2) learning how to perform the actions from human demonstration; and (3) perceiving objects for manipulation actions. We propose the solution for each problem as follows

- Proposal 1. Instead of planning the task manually, the task plan which contains the sequence of actions and the related objects is automatically obtained from instruction manuals.
- Proposal 2. The executable plan of each action can be achieved from observing human demonstrations (in general, human activities in daily life). This action execution plan is able to adapt to changes in dynamic environment such as the change of object location in manipulation action.
- Proposal 3. The robot needs to perceive about objects for manipulation actions, for example, object's identification, object's location, object's pose.

Figure 1.3 illustrates a conceptual design for robot executing everyday tasks. Firstly, the knowledge about a task includes the sequence of actions and the related object which can be acquired from instruction manual of home appliances.

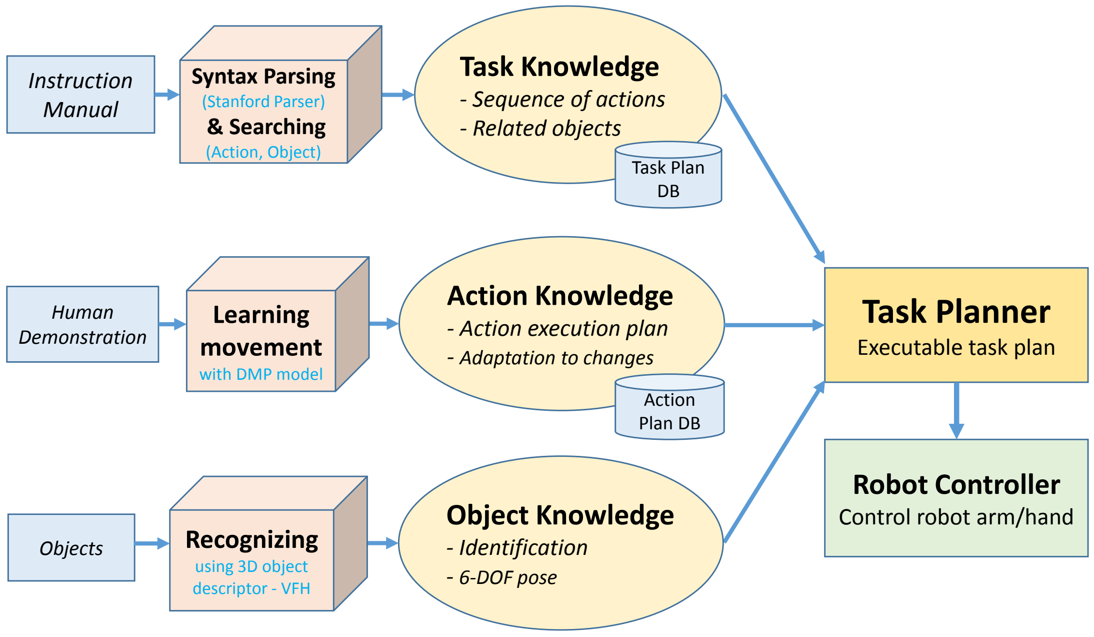


Figure 1.3: Conceptual design for everyday task executing robot

Secondly, the knowledge about how to perform the actions achieved by learning from human demonstration. A learning method of movement is applied for adapting to new changes which can come from dynamic environment such as the change of object's location, pose. Thirdly, the robots need to perceive the object that the action manipulates. This is done by providing the robot with the vision ability for object recognition and pose estimation. A task planner will gather these three knowledge and generate a executable plan for a robot controller which controls a robot arm to perform the task.

Figure 1.4 represents an overview of the research presented in this dissertation. The first work is to acquire the knowledge about the sequence of actions in the task from instruction manuals. In the second work, the knowledge about how to perform the action is acquired by learning from human demonstration. To scale the robot's ability in performing a large number of everyday tasks, actions can be encoded as building-block units forming a library of popular actions and they reused to generate different tasks. The third work is to deploy a solution for recognizing object and estimating pose which can be used in real time for object manipulation actions. We implement an application executing the task 'dispensing water' to validate the proposals.

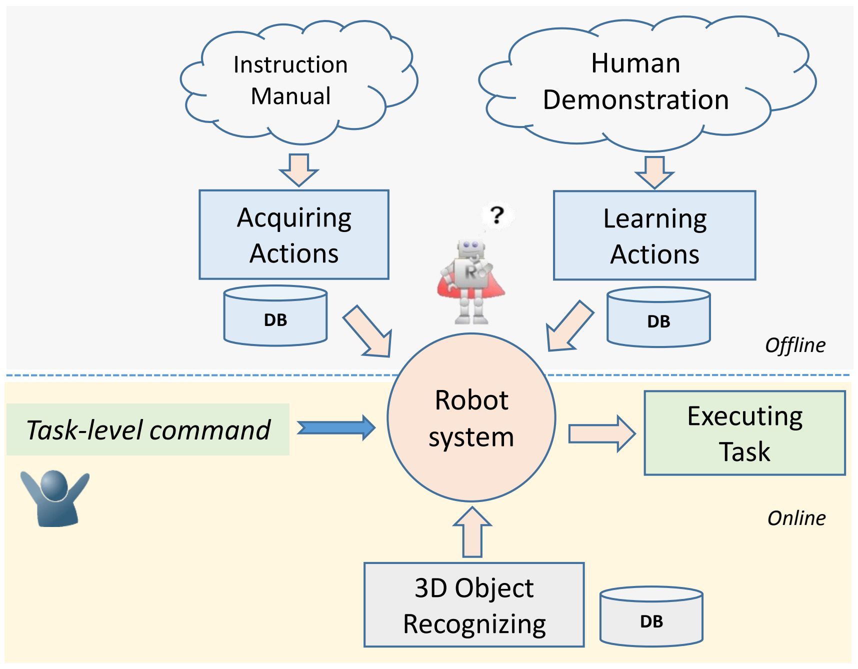


Figure 1.4: The overview of research works in this dissertation

1.5.1 Acquiring Actions from Instruction Manual

This work aimed at acquisition of actions and objects from instruction manual for automatically generating a task plan. The proposed method in this work includes two main steps:

- Parsing the grammar structure of instruction sentences. This step outputs the parse tree of each sentence.
- Searching on parse tree to extract actions and their following objects.

This work is presented in the Chapter 2.

1.5.2 Learning Action from Human Demonstration

The main objective of this work is to learn how to perform the action from observing human demonstration. After having a task plan from previous work, to automatic accomplish the task, the robot needs to refer to the knowledge about how to perform each action in that task plan. This knowledge can come from observation of hand movement in human demonstration of the desired action.

A learning method with the ability of adaptation to new changes was proposed. The key points in this work are:

- Recording human hand movement
- Segmentation of movement
- Learning movement with adaptation to new changes

This work is described in the Chapter 3. A scenario of the task 'dispensing water' is considered with the sequence of actions performing by a robot arm according to the proposed method.

1.5.3 Perceiving Object for Manipulation Actions

The robots need to perceive about the objects that they want to manipulate. The object information including identification, pose, location, size, etc are important for object manipulation actions. Recognizing the change of object's location in real time also help to adapt a robot action to new location of the object. In this work, a method of object recognition and pose estimation using a RGB-D camera is implemented to provide a vision ability for robot in manipulating objects. The method includes main steps:

- Retrieving point cloud data from RGB-D camera
- Segmenting and clustering object cluster
- Extracting feature using Viewpoint Feature Histogram
- Recognizing objects
- Estimating object pose

This work was described in Chapter 4

1.6 Thesis Organization

This thesis is organized into six chapters as below.

- Chapter 1. Introduction
- Chapter 2. Acquiring Actions from Instruction Manual
- Chapter 3. Learning Action from Demonstration
- Chapter 4. Perceiving Object for Manipulation Actions
- Chapter 5. Discussion
- Chapter 6. Conclusion and Future Work

Chapter 2

Acquiring Actions from Instruction Manual

This chapter introduces a proposal to acquire actions and objects from instruction manual to generate an executable plan of the everyday task for the robot.

2.1 Introduction

Everyday tasks at home such as dispensing water from a water thermos pot, making a cup of coffee by a coffee maker, cooking or warming food by a microwave oven, etc. are still hard for robots due to challenges in understanding the tasks about what actions need to perform in the task, how to perform that actions. These tasks can be decomposed into isolated actions such as pick up an object, place object to somewhere, press a button, turn a knob, open or close a cover, and so on. These actions are common and may be repeated many times in one or some tasks in daily life.

A service robot should be able to automatically plan manipulation actions to accomplish the task in domestic environments. If the service robot is ordered to serve a cup of water, it should be able to move to the water pot, pick up a cup and place under the spout, then press a button to pour out some water. The robot should be able to perform this sequence of actions and plan it by the robot itself. Therefore a method for automated task planning is essential for robots.

Automated planning is the process in which the robots generate artificially the sequences of actions to reach their goals. The planning is normally abstracted to a symbolic level, so that a symbolic planner can handle and solve the problem. Although solutions in the symbolic world can be found, it is still problem for map the used symbols back to the real world. For example, if a symbolic planner generates an action "pick up a cup" without considering the arm reachability and obstacles around the cup, the plan could not be performed in the real world, although its symbolic formulation is correct. Automated task planning for the service robots faces great challenges in handling dynamic domestic environments. There are two challenges that can be foreseen. Firstly, service robots are required to work in home environments, which are highly unstructured and present considerable uncertainties. For example, a cup may have been observed to be one location, yet might not be at the same location at a different time. Secondly, the construction of a sequence of actions for a service robot presents another challenge. Although a robot can detect an object using computer vision or other sensing technologies, it will not know when it should search for the object or how the object is linked to the task. The robot requires certain knowledge from human users. Nowadays, in practice, action sequences for robots are mostly hard-coded or predefined for certain scenarios. This method is rather inflexible as it requires manual amendments of source code in orders to reprogram the action sequence for each task.

The classical approach to solve a robot task is to generate a plan as a sequence of actions which leads to desired goal, taking into account the prerequisites and the effects of the actions. For common household tasks, the sequences of actions are unspecified and there is no complete description of the prerequisites and effects of actions. Moreover, the task is more complex, it is more difficult to get the executable plans. Instead of generating a task plan in a manual way, we proposed to acquire the sequence of actions in the task from instruction manual. The robots can look up the sequences of actions of a certain task in the instruction manual and transfer them into task plan. We proposed the method to extract action and object in a task instruction taken from instruction manual of home appliances or equipment. The method uses natural language processing technique for parsing the grammar structure of instruction sentences and searching algorithm to obtain action and object which will be used for building executable plans for the robot in next work.

2.2 Acquisition of Robot Knowledge

When an intelligent robot acts in home environment, it is inevitable to consider what tasks the robot is competent for and how the robot plans to accomplish the task. Instead of pre-programming all actions that robots might need, automatically acquiring the corresponding action knowledge is necessary to automatically plan the task.

There are more and more open-source knowledge resources being available including knowledge bases, ontologies, household appliance manuals. etc. Such open knowledge provides a new opportunity for intelligent robots to dynamically acquire the action knowledge. The challenge of translating open knowledge lies in the formalization of natural language [11]. Many researches provide a number of effective approaches so called semantic parsing, to translating natural language into formal expression, for example in [70], [32]. Kunze et al. [31] present a robotic system that translates the OMICS (Open Mind Indoor Common Sense) knowledge into a formal presentation. Tenorth et al [63] proposed extracting the action knowledge from the natural language instructions from the World Wide Web.

Efficiently understanding natural language instructions is important for intelligent robots. Many existing natural language instruction approaches either use simple language understanding or large corpora of hand-annotated training data to pair language with robot actions. Mericli et al. [36] allow users to specifiy a task program to be stored and executed on the robot. Their method for understanding natural language is done by keyword search and assumes certain words in a particular order. Cantrell et al [10] used semantic parsing to extract an action, pre- and post-world conditions for that action, and the entities involved. Natural language instruction can dictate a sequence of actions to a robot. Some approaches pair robot actions with language descriptions, then build models that map language instructions to action sequences in [38] and [61]. In [29] Klein and Manning relied on a well-trained initial parser for interpreting the sequence of actions from high-level instructions. Another approach enables a robot to learn a sequence of actions and the lexical items that refer to them from language instruction and dialog [57]. Jesse Thomason et al. in [65] introduced a dialog agent that understands human instructions through semantic parsing. They resolved the problem of ambiguities using a dialog manager, and incrementally learns from human robot conversations by inducing training data from user paraphrases.

Some researches emphasized on solving the problem of understanding and performance of everyday tasks by robots. Moritz Ternorth and et al. in [63], [62], [64], Daniel Nyga and Michael Beetz in [44] translated the task instructions from websites into the almost executable robot plans. These researches aim at building up a robot knowledge framework which shares the large amount of robot knowledge about every tasks, actions as well as related objects and world. Mario Bollini and et al. [6] described a method of interpreting and executing recipes with a cooking robot. The authors mapped from natural language instructions to robotics instructions by designing an appropriate state-action space. Dipendra K Misra and et al. [38] and [37] carried out grounding the natural language instructions with appropriate environment context and task constraints.

2.3 Extraction of Action and Object from Instruction Manual

For everyday tasks taken from instruction manuals, in order to build a task plan, we proposed a method to automatically acquire the sequences of actions in the task from instruction manual. The method uses syntax parsing of sentences to extract actions and flowing objects if applicable.

From the linguistic point of view, verbs denoting actions and nouns denoting objects are very important in a sentence because they can briefly convey the key message of the sentence. This is more exact with the sentences in the instruction manuals since they are often imperative statements that guide to perform certain actions. For example, the instruction take from the user manual of a water thermos pot "Place a cup to fill with hot water just beneath the spout, and press the 'Push' key" contains two important pair of actions and following objects 'place - cup' and 'press - key' which describe the actions need to be performed in this instruction. From this key point, a method of extracting action and object from instructions was proposed to provide robots necessary knowledge about what actions the robots need to perform in the instructions of a certain task.

The extraction of information such as subjects and verbs from sentences is a technique in natural language processing. This technique is often applied in summarizing document or enriching knowledge for conceptual ontology. Delia Rusu and et al. in [15] presented the methods to extract subject-predicate-object triplets from English sentences by using four different well-known syntactical parser including Stanford Parser, OpenNLP, Link Parser, and Minipar. However, the extraction algorithms determined only one pair of verb and object in each sentence. This leads to ignorance of other action verbs if the sentence combine more than one action verb. In natural language sentence, action an object are identified based on the part-of-speech (POS) of words in the grammatical structure. Accordingly, the extraction method of action and object includes three steps. Firstly, parsing the syntax structure of a sentence to achieve parse tree, then editing the parse tree to remove the indirect action if it exists, finally, searching on each parse tree to determine action and object based on POS tags that are labeled in the parse tree. The flow of this method is shown in figure 2.1

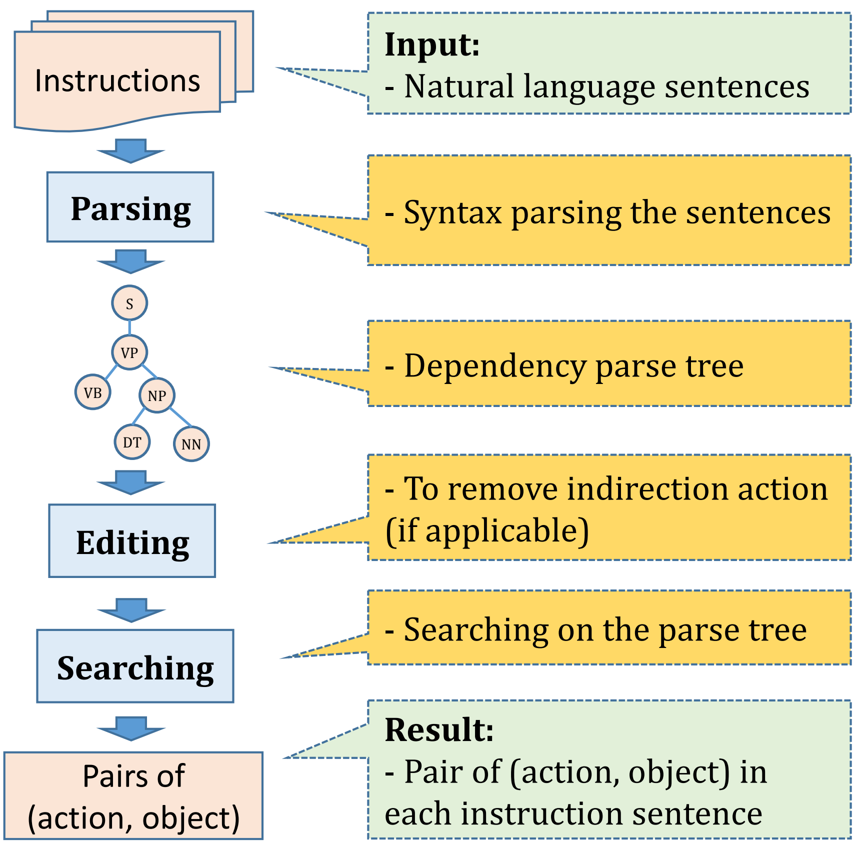

Instructions
Input:
- Natural language sentences
Parsing
- Syntax parsing the sentences
S
VP
VB
NP
DT
NN
- Dependency parse tree
Editing
- To remove indirection action (if applicable)
Searching
- Searching on the parse tree
Pairs of (action, object)
Result:
- Pair of (action, object) in each instruction sentence

Figure 2.1: The flow of method extracting action and object from instruction sentences

2.3.1 Parsing Syntax Structure

In the first step, syntax parsing, the input data which are natural instruction sentences in text files are parsed to determine the syntax structure of each sentence. There are some syntax parsing tools satisfying this situation. This study applied a state-of-the-art parsing tool, Stanford Parser, which is a well-known PCFG (Probabilistic Context Free Grammar) parser working out the grammatical structure of sentences [29]. This parser generates the dependence parse trees. The parse trees are labeled with part-of-speech (POS) tag for their words and phrases depending on their grammatical structure. Stanford dependence parse tree is represented in text form as an example follows:

```
(ROOT
  (S
    (VP
      (VP (VB Place)
        (NP (DT a) (NN cup)
          (S
            (VP (TO to)
              (VP (VB fill)
                (PP (IN with)
                  (NP (JJ hot) (NN water)))
                (PP
                  (ADVP (RB just))
                  (IN beneath)
                  (NP (DT the) (NN spout))))))))
        (CC and)
        (VP (VB press)
          (NP (DT the) (JJ Push) (NN key))))))
```

Figure 2.2 shows the parse tree of given example in visualization. The leaf nodes are POS tags that associate with words in the sentences, non-leaf nodes are POS tags associate with phrases in grammatical structure of the sentences. Example of POS tags are NN for a noun, VB for a verb, DT for a determiner, NP for a noun phrase, PP for a preposition and so on. This statistical parser still makes some mistakes, but commonly it is evaluated working rather well and used widely in natural language processing.

2.3.2 Removing Indirect Action

Instruction statements may be a simple grammatical pattern consisting of one action verb and one object noun or may combine more than one action. In the case of the sentence with more than one action, a certain action may not require to perform, so-called indirect action in the sentence. For example, in the example of above given sentence, the action 'fill' (with hot water) is an indirect action which does not require to perform. Hence, this kind of actions need to remove from extracted result. This situation mainly occurs when the instruction sentence contains a dependence clause that can be omitted without changing the key meaning of the sentence. As shown in figure 2.2, the action 'fill' is located in a sub tree with the node 'S' which is a dependent clause and can be removed. From this crucial point, the solution for removing indirect action is to prune the sub tree of dependent clause from the parse tree of original sentence before having next step to search pairs of action and object in the parse tree.

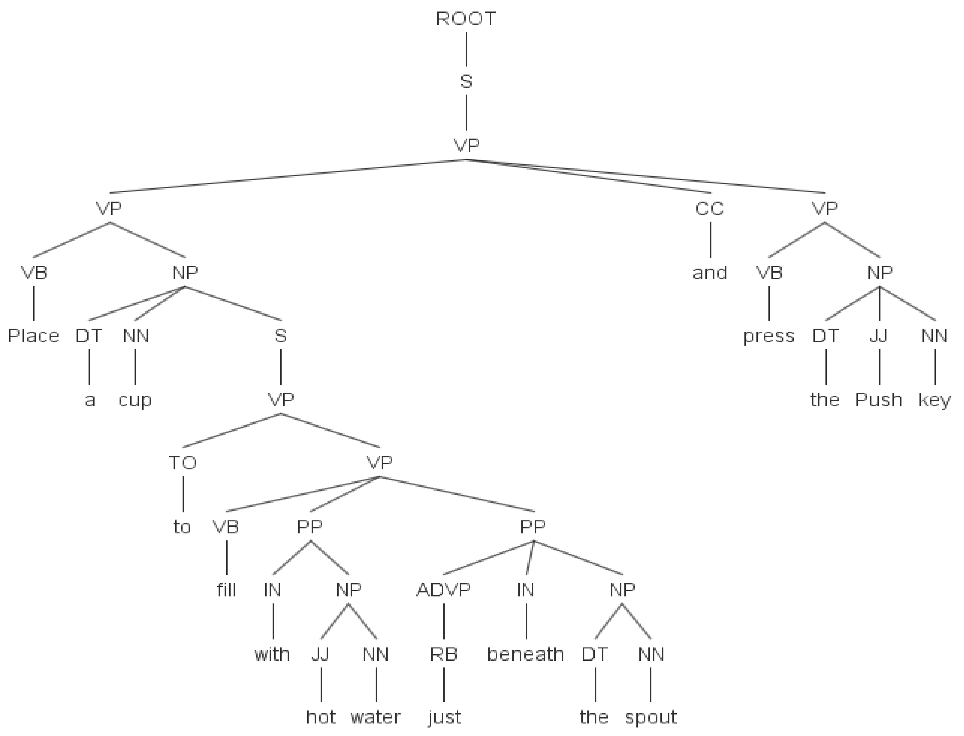


Figure 2.2: The parse tree of a given example is outputted by Stanford Parser

To this end, we exploited tree surgeon function which is known as TSurgeon tool [34] built in Stanford NLP package. It provides the way of tree editing based on the set of operations that are applied to tree locations which are matched to a tregex pattern. This is similar to regular expression for tree matching. The condition to identify the sub tree of a dependent clause is that its root node is labeled with 'S' tag and is dominated by a VP sub tree. The tregex pattern (regular expression) is "S=node ≫ VP". Then, the surgery operation 'prune' is executed to remove this sub tree from original parse tree. This procedure is illustrated as in figure 2.3. Figure 2.4 depicts the parse tree after pruning to remove a sub tree of dependent clause of given example. The pairs of action and object then are found in this edited tree. In case there is no dependent clause in a sentence matching with the above condition, the parse tree will not change.

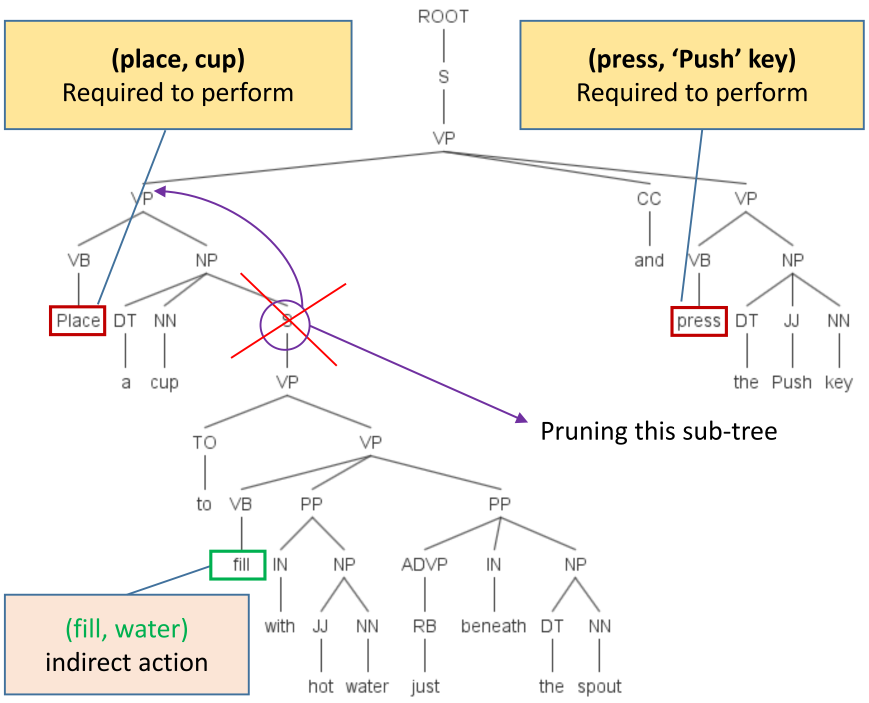


Figure 2.3: The edited parse tree after pruning the sub tree of dependent clause

2.3.3 Searching for Action and Object

The second step of the extraction method is searching for actions and objects on each parse tree. After achieving the parse tree and removing the sub tree of dependent clause if it exists. The actions and objects are found based on part-of-speech tags labeled in each parse tree. The searching procedure identifies all possible pairs of action and object. To this end, the searching algorithm

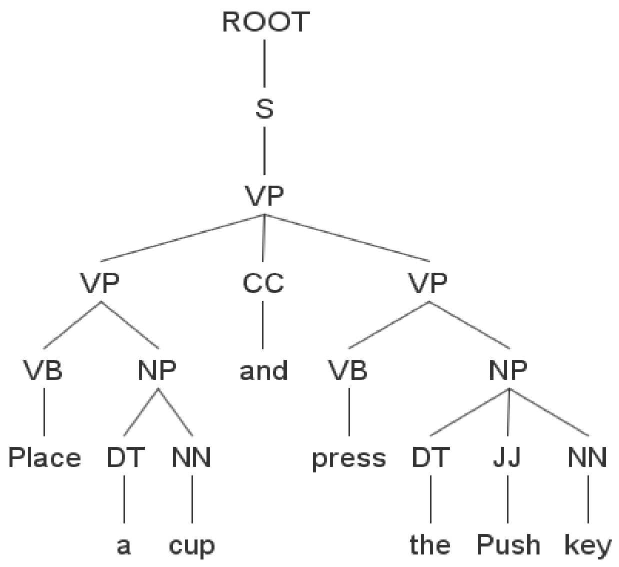


Figure 2.4: The edited parse tree after pruning the sub tree of dependent clause

traverses sequentially each action verb and its following object noun in the parse tree. Firstly, one verb is searched in the first VP sub tree and assigned as first action. With each action verb found in the VP sub tree, an object is assigned by noun following it by searching in siblings sub trees of that VP sub tree. Object is assigned as last noun that is found firstly in sibling sub trees "NP" or if not it is assigned as the first noun found in sibling sub trees "PP".

In given example, the instruction sentence is parsed into a VP tree at top-level as shown in figure 2.2. After pruning the sub tree of dependent clause, the edited parse tree is obtained as shown in figure 2.4. The searching procedure is implemented on this edited tree. The first verb found in the first VP sub tree 'place' is assigned as first action. Then, its following noun 'cup' which is found in the NP sibling sub tree is assigned as the object that accompanies with the action 'place'. Next, the second pair of action 'press' and object 'key' is determined in same way. The searching procedure is repeated until the end of each parse tree to obtain all possible pairs of action and object.

2.3.4 Action-Object Extraction Algorithm

From the steps as mentioned above, the extraction algorithm of action and object from instruction sentences is built as follows:

Algorithm 1: Pseudocode of Action and Object Extraction

```
Function getParseTree(sentence)
    word_list ← tokenize the sentence ;
    parse_tree ← parse the world_list ;
Function pruneDependenceTree(parse_tree)
    edit_tree ← pruning the dependence sub tree ;
Function extractActionObject(parse_tree)
    VP_subtree ← the top level VP sub tree of parse_tree ;
    for each verb_leaf_node found in VP_subtree do
        action ← word of verb_leaf_node ;
        ao_pair ← append action ;
        siblings ← siblings subtrees of verb_leaf_node ;
        for each sib_tree in siblings do
            object ← last noun in "NP", "PP" sib_tree ;
            if object found then
                break;
            object ← first noun in other sib_tree ;
        ao_pair ← append object
    ao_list ← append ao_pair ;
```

In this algorithm, each extracted object is assigned by a single noun. However, in some cases, better result can be achieved if an object is possibly identified by a noun phrase which provides more detailed information, for example, 'coffee cup', 'Push key', 'Lock/Unlock key'. In order to assign an object by a two-noun phrase, the searching procedure can be improved by finding and storing all nouns in a noun phrase and assigning two last ones for object if it exists.

2.4 Task Planning

After obtaining the sequence of action and following object if applicable, the task plan is built by assembling the series of these actions. A task plan is simply represented by:

$$T = \{< Action_i,\ Object_i >\}\ i = 1, 2, ..., n \tag{2.1}$$

The pair of $< Action_i, Object_i >$ is extracted from the task instructions taken from instruction manual. In next step, each action is built an executable plan for the robot.

2.5 Experiment and Result

We collected test data including natural language sentences of many different tasks taken from home appliance instruction manuals such as making coffee by a coffee maker, dispensing water from thermos pot, cooking something by microwave oven, etc. These task instructions are stored in text files as input data. We implemented the program of proposed algorithm to extract all possible pairs of action and object in instructions from each input text file. The extracted results are evaluated by comparing manually the pairs of action and object with respective to each described in the primary sentence to decide the accuracy of outputs. We tested two examples of extracted results corresponding to two input texts.

In the first example, the input texts consist of only two sentences guiding a simple task 'dispensing water' as described text 1, a principle task described in many water thermos pot user manuals. In this case, all extracted results are correct as shown in the table 2.1. Objects are also assigned by two nouns, if any, including a main noun and an auxiliary which provides more specific information on that object.

Text 1. Instructions of 'dispensing water' task taken from a water thermos pot user manual

1. Press the 'Lock/Unlock' key once.
2. Place a cup to fill with hot water just beneath the spout and press the 'Push' key.

Text 2. Instructions of 'cooking some food' by microwave oven

1. Push the button and open the door
2. Place the food on the turntable or on the grill jack
3. Select the function and cooking time required
4. Close the door and press the Start button.

Table 2.1: Extracted result of action and object from task instruction 'dispensing water'

Sentence No.	Action	Object
1	'Press'	'Lock/Unlock', 'key'
2	'Place'	'cup'
	'press'	'Push', 'key'

Table 2.2: Extracted result of action and object from task instruction 'cooking food' by microwave oven

Sentence No.	Action	Object
1	'Push'	'button'
	'Open'	'door'
2	'Place'	'food'
3	'Select'	'function'
4	'Close'	'door'
	'Press'	'Start', 'button'

In the second example, the input text is the instructions from an user manual of microwave oven for the task 'cooking food'. The extracted result of action and object is shown in table 2.2.

In the third example, the input texts are the instructions of a more complex task, which combine many operations taken from a coffee maker user manual as describe in text 3. The extracted results as represented in table 2.3 contain 6/10 instruction sentences obtaining the correct pairs of action and object (Sentences No. 1, 4, 7, 8, 9, 10). Four sentences (2, 3, 5, 6) with complex grammar structure return incorrect or empty actions due to the mistakes in syntax parsing by the Stanford parser.

Text 3. Instructions of 'How to operate' task taken from a coffee maker user manual.

1. Place the Brewer on a flat surface, remove protective sheet and plug into outlet.
2. Do not remove or puncture the foil lid of the K-Cup portion pack.
3. Lift front facing of the brewer to insert K-Cup.
4. Place chosen K-Cup into the K-Cup Assembly Housing.
5. Lower the front facing completely and firmly to close the Lid and puncture the K-Cup portion pack.
6. Depress the water marking button and the hot water tank will open automatically.
7. Fill the Hot Water tank with filtered or bottled water up to the FILL LEVEL indicator.
8. Close the Hot Water Tank cover.
9. Place a coffee cup in the dispense area on the drip tray.
10. Press the BREWNOW button.

Table 2.3: Extracted result of action and object from task instruction 'making a cup of coffee'

Sentence No.	Action	Object
1	'Place'	'Brewer'
	'remove'	'sheet'
	'plug'	'outlet'
2	'remove'	''
	'puncture'	'pack'
3	''	''
4	'Place'	'K-cup'
5	''	''
6	'open'	''
7	'Fill'	'water', 'tank'
8	'Close'	'tank', 'cover'
9	'Place'	'coffee', 'cup'
10	'Press'	'button'

The accuracy of extracted results depends on those of both syntax parsing tool and the searching procedure of proposed method. Stanford parser applied in this study works with very high accuracy but still has errors when the instruction sentences are formed by complex grammatical structures.

We did many experiments with other input texts, and showed here two examples. From these examples, we can confirm the availability of the processing steps of proposed method, and pointed out different cases of instruction sentences. We will consider a statistic evaluation with a larger number of data based on the category of grammatical patterns such as simple grammatical pattern consisting of only one pair of action and object or consisting of more than one pair of action and object; grammatical pattern containing a dependence clause; complex grammatical patterns including negative pattern, passive pattern, pattern with several actions but pairs of action and object are not corresponding.

Chapter 3

Learning Action from Human Demonstration

3.1 Introduction

In chapter 2, the actions what the robot needs to perform in a task can be achieved from instruction manual. In order to perform each action , the robot needs to be provided an executable plan. The knowledge about how to perform the action is necessary. The work in this chapter deals with the problem of planning action for the robot by learning from demonstration provided by human. The goal is to generate appropriate motion patterns for the robot (robot's end-effector and gripper in particular) to perform a desired action.

Learning from Demonstration (LfD) [1] (also known as Programming by Demonstration (PbD), Imitation Learning) aims to enable robots to autonomously perform new behaviors with the ability of adaptation to the new changes. This paradigm allows to transfer the knowledge about how to perform behaviors from an expert teacher to a robot through the use of demonstrations. This approach takes the view that an appropriate robot controller can be derived from observations of a human's own performance.

The ability of the robot to interpret the demonstration is an essential part of robotic systems employing learning by demonstration. Many methods have been proposed for teaching robots through demonstrations, depending on different levels of complexity of the robot's sensory capabilities. Although each approach has its own particularities, they can be classified into two main categories: learning by observation and learning from experience.

Learning by observation includes techniques based on the robot observes passively the teacher's performance, and attempting to reproduce the observed behavior. These techniques allow robots to gather external sensory information, in most cases from a camera. This requires using complex computer vision techniques to translate the teacher's actions. In learning by observation, the robot has to face with the major challenge of accurately perceiving the teacher's demonstration which is dependent on the ability of observation and noise in the real world domains. In addition, the robot must also be able to interpret the demonstrations and map them to its own capabilities accounting for differences in body structure with respect to the teacher. In this work, we rely on observed human demonstration to transfer the knowledge about task and actions from a teacher to the robot.

Learning from experience includes approaches which require the robot participates actively in the demonstration, performing the actions along with the teacher and experiencing it through its own sensors. From the physical guideline of the teacher, the robot is able to record internal information such as joint angles or positions relative to its own body. This approach is often appropriate for humanoid robots.

In learning from observing, the demonstration of a human expert is recorded and then reproduced by a robot. If the environment does not change during the robot's performance then learning to imitate the demonstrated trajectory of teacher is sufficient. However, in many cases the actions that the robot should learn are influenced by the state of the environment. Repeating exactly an observed movement for the robot is not realistic in a dynamic environment. For example, in the action 'pick up an object', if the robot only records exactly the trajectory of a particular instance of a demonstration, it would not be able to get the object in a dynamic environment when the object's location is changed. Therefore, learning a demonstrated movement should have the ability of adaptation to new changes such as changing the goal, scaling of time. A learning method that allows to adapt with new changes is essential.

Most of the actions in everyday tasks are performed by hand. Learning robot action from observing hand movement in human demonstration is feasible. Many methods have been deployed to model the movements and reproduce them for robot accurately and stably. Dynamic Movement Primitives (DMPs) framework was used for encoding and and regenerating movement trajectories for the robot in many studies, for example in [45], [35], [48], [42]. DMPs can be used to represent the movement trajectory in end-effector space (task space), which encodes robot trajectories from start position to goal position. The DMPs model has the generalization ability. The generated movement can be adapted to some new changes such as the change of new goal. In addition, this representation can generate continuous robot movements and has robustness to perturbations. DMPs framework can also be further extended to have capabilities such as joint limits avoidance and obstacle avoidance [45].

The hand movement of manipulation actions is complex. To learn each simple unit of movement by using DMPs framework, the observed movement should be segmented into simple units which are called motion primitives. This segmentation plays an important role to describe each motion primitive by DMPs models. The complex hand movements not only include translation motion from start position to goal position but also contain the motion of operating object such as grasping, releasing object by closing/opening hand fingers. These movements are signals to control position and orientation the robot's end-effector and the state of robot's gripper. The DMPs framework enables to combine multiple control signals so that the movement trajectory adding the data of orientation and gripper's state is feasible for encoding by DMPs models. In other hand, the complex movements for performing the actions by robot can be generated by sequencing motion primitives which were encoded as generative DMPs models. The idea is that common motion primitives, encoded by DMPs models, like low-level building-blocks, can form to a library of motion primitives. Each motion primitive is labeled accordingly to its DMPs models. The complex movement of a robot action can be composed by sequencing motion primitives from this library.

3.2 Background and Related Work

3.2.1 Learning from Demonstration

3.2.1.1 An Overview

The term 'learning' in the context of this study is in a general sense. It implies the ability of a robot system to autonomously acquire new skills, to adapt to changes in the environment and/or tasks, and to improve its performance over time. Instructions, experiences, demonstrations provided by a teacher may facilitate the learning process. The learning process here is not completely equivalent to the concept of machine learning but different machine learning techniques can be used as tools for the learning method. The level of learning is subject to the abilities that are desired from the system. The literature on learning from demonstration does not make a clear distinction among these various levels. Generally, two popular approaches are learning tasks and learning skills or actions.

In Learning from Demonstration (LfD), a teacher demonstrates a task or action to a robot, then in such a way that the robot is able to reproduce that task or action. A robot has ability of reproducing exactly certain behaviors as demonstrated but it is difficult to adapt to changed environment or conditions which come naturally to human. Although the idea of learning from demonstration at first seems simple, it faces many challenges [43]:

- The sensing abilities of robots are limited and different from human perception. What is the best way to give demonstrations to robots so that it can maximize knowledge transfer ?
- The robot's embodiment is different than a human's. For example, the motion of human arm and fingers is still difficult to be compatible with the configuration of robot arm and fingers. What matching mechanism is needed to create the mapping between teacher's actions and the robot's own sensory-motor capabilities ?
- Learning is incremental process. This means that certain knowledge could only be learned when an appropriate foundation knowledge already existed. What could a robot learn and what are the required capabilities for learning it ?

An overview of learning from demonstration and its classifications can be found in the survey article [3] and the book chapter [5]. These are some terms including Learning from Demonstration, Programming by Demonstration and Learning by Imitation, which are typically used as synonyms in most literature with some slight differences.

Simple methods of learning from demonstration topic such as teach-in and playback were developed from many year ago. In such scenarios, the teacher moves the robot arm manually or by teleoperation and a sequence of exact positions and orientations are recorded. Then the robot imitate exactly the movement as recorded. Although these methods are successful for some tasks, they still suffer from drawbacks. For example, this kind of imitation is only suitable for predefined work spaces and processes. Nowadays, advanced approaches in learning from demonstration enable to overcome this limitation to make robots operate in unstructured environments such as households.

3.2.1.2 Interface for Demonstration

The interface used to provide demonstrations influences how the information is acquired and transmitted. Different types of interfaces can be distinguished to two major categories:

- **Directly Recording Human Motions** This interface can use any of various different motion tracking systems, based on vision or wearable motion sensors, markers. The teacher performs demonstration on his own. The data is recorded either observed by external sensors like a camera or by sensors attached to the teacher. These methods have the advantage in that they allow the human to move freely in natural ways, but require good solutions to the corresponding problem. The observed data does not match with the actions of the robot. Therefore, a mapping between the different embodiment must be employed. Typically, this is done by an explicit mapping between human and robot joints, but can be quite difficult if the robot differs greatly from the human.

  Some studies use this type of interface for gathering the hand movements of human demonstration. Skoglund et al. in [58] used external marker mounted on a data glove to record the hand motions of 'pick and place' actions for teaching a manipulator. Ren Mao et al. in [35], Oikonomidis et al. in [22] used marker-less hand tracker which can reliably track a skeletal hand model. This approach was based on a 3D hand model, which has the advantage of accurate estimation of hand pose. The 3D data in real world is captured from Kinect FORTH tracking system using the hand model based method.

- **Kinesthetic Teaching** The teacher teleoperates the robot or moves the robot limbs physically (kinesthetic teaching). All joint states of the robot are recorded using sensors attached the joints. This results in good data quality and is not affected by the corresponding problem. However, one main drawback of this method is that the human must often use more of their own degrees of freedom to move the robot than the number of degrees of freedom they are trying to control. For example, the human must use both hands to move a few robot fingers. In both cases, teleoperation as well as kinesthetic teaching, a robot with dozens of joints is difficult to handle.

3.2.1.3 How to Solve LfD

Approaches in the topic of LfD results in different views on the learning and the representation of tasks. The most differing of the representations among learned robotic skills are symbolic encoding (high-level) and trajectory encoding (low-level). Symbolic encoding describes the task with a sequence of already known action primitives, like reaching object. The trajectory encoding represents a task as a sequence of sensor data such as positions, velocities and accelerations.

Learning high-level action composition

Learning complex tasks, composed of a combination of individual motions, is the goal of LfD. A common approach is to first learn models of all of the individual motions, using demonstrations of each of these actions individually [14], and then learn the right sequencing/combination in a second stage either by observing a human performing the whole task [16], [58] or through reinforcement learning [39]. However, this approach assumes that there is a known set of all necessary primitive actions. This may be true for specific tasks, but so far it does not exist a database of general purpose primitive actions, and it is unclear whether the human activities can really be reduced to a finite library.

Another method is to observe the human demonstrate the complete task and to automatically segment the task to extract the primitive actions which may then become task-dependent, [30]. The main advantage is that both the primitive actions and the way they should be combined are learned in one pass. One issue that arises is that the number of primitive tasks is often unknown, and there could be multiple possible segmentation which must be considered [18].

In the studies at [12], [13], a high-level approach of LfD was presented. The learning method proposed in these publications focuses on extracting an abstract description of the task from multiple demonstrations. The actual movements are generated using a trajectory planner.

Low-level learning of individual motions

Individual motions/actions (e.g. picking up a cup, pressing a button) could be taught separately instead of all at once. The teacher would then provide one or more examples of each sub-motion apart from the others. If the learning proceeds from the observation of a single instance of the motion/action, it is called one-shot learning [69]. Examples can be found in [41] for learning locomotion patterns. Different from simple record and play, here the controller is provided with prior knowledge in the form of primitive motion patterns and learns parameters for these patterns from the demonstration.

Multi-shot learning can be performed in batch after recording several demonstrations, or incrementally as new demonstrations are performed [33]. Learning generally performs inference from statistical analysis of the data across demonstrations, where the signals are modeled via a probability density function, and analyzed with various non-linear regression techniques from machine learning. Popular methods nowadays include Gaussian Process, Gaussian Mixture Models, Support Vector Machines.

of gestures by imitation.

The works in the studies at [56], [55] and [8], [9] was successfully done by using a parametrized trajectory to encode the whole task. These approaches use a regression method to generalize over multiple demonstrations. They do this in both the joint space (positions of joints) and the 3D task space (Cartesian coordinates calculated using robot kinematics). At task reproduction, several constraints such as the distance to the objects of the scene, are respected. The reproduction process generates a whole position-only trajectory which is executed using a standard controller. Thus interaction with the environment is impossible, feedback at execution time can not be considered.

A more low-level approach on learning movements is presented in [54], [25]. These works use dynamical systems to encode a movement. The main advantage of dynamical systems is their adaptation. In such systems the next state is always calculated from the current state by applying the fixed rule. Regression methods are used to approximated the non-linear part of these dynamical systems. Learning is possible with a single demonstrated trajectory [54] or by generalization over multiple [19]. Improving a trajectory with reinforcement learning has also been done successfully for example in [47]. This encoding of a trajectory is point attractive, which means a trajectory is represented in terms of a fixed start and goal state. Unfortunately, this limits the field of application to tasks with a predefined goal position. In order to overcome this limitation it is possible to build a sequence of these motion primitives to represent more complex tasks. This bases on the assumption that complex movements can be described by small units of action.

3.2.2 Learning Movements using Dynamic Movement Primitives

To facilitate automatic planning of actions with the data recorded from demonstrations, some frameworks of learning from demonstration are selected. The goal is to generate appropriate motion patterns for the robot. A popular framework for the representation and learning of movements with dynamical systems is known as Dynamic Movement Primitives (DMPs), originally introduced in [25]. The DMPs framework has advantage of adaptive learning to new changes of environment or the own movements. Furthermore, this framework has also been improved by modified DMPs versions which have advanced abilities such as generalization in 3D task space, joint limits avoidance and obstacle avoidance, for example in publications in [20], [21], [46], [45].

3.2.2.1 Dynamic Movement Primitives

The dynamic movement primitives was proposed by Ijspeert et al. in [25], [26] and then was developed over many years with improvements in [27], [53], [24]. In

DMPs framework, a captured movement can be described by a set of differential equations which is from a mathematical model of a dynamical system. It has the advantages in generating smooth movements with the perturbation that can be automatically corrected by the dynamics of the system. It addresses the flexibility and does not rely on time. Moreover, it allows the capability of generalization because the characteristic that the equations are formulated in way adapting to a new goal by changing goal parameter. In [60] F. Stulp et al. proposed an approach using DMP and a probabilistic model-free reinforcement learning algorithm to learn motion primitives.

The DMP framework enables to learn a movement trajectory from one reference sample. It represents a movement as a time evolution of a nonlinear dynamical system. Then it can reproduce the movement and optionally adapt to different configurations by changing the goal parameter. The formulation of a standard DMP model originates from a second order linear dynamical system (like a spring-damper system model) which is stimulated with a nonlinear forcing term, uses a set of differential equations:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \tag{3.1a}$$

$$\tau \dot{x} = v \tag{3.1b}$$

where $x(t)$ denotes a sample trajectory starting at start position $x(t_0) = x_0$ towards to goal position $x(t_f) = g$. K and D involve the inherent dynamics of the second order linear system, as the spring constant and the damper coefficient, respectively. K is chosen in advance to meet the desired velocity response of the system. D is chosen to satisfy that the dynamic system be critically damped and thus reach the goal position without overshoots. A suitable choice is $D = 2\sqrt{K}$. The forcing term $f(s)$ is an arbitrary non-linear function which is used to adapt the response of the dynamic system to an arbitrary complex movement. In [27], Ijspeert et al. proposed a suitable choice for $f(s)$ as a sum of M weighted exponential basis functions:

$$f(s) = \frac{\sum_{i=1}^{M} w_i \psi_i(s)}{\sum_{i=1}^{M} \psi_i(s)} s \tag{3.2}$$

where $\psi_i(s)$ are Gaussian functions defined as

$$\psi_i(s) = exp(-\frac{1}{2\sigma^2}(s - c_i)^2) \tag{3.3}$$

Parameters c_i and σ_i define the center and the width of the ith basis function, while w_i are the adjustable weights used to obtain the desired shape of the trajectory [42]. The variable s is a phase variable and defined by the canonical system:

$$\tau\dot{s} = -\alpha s \tag{3.4}$$

where α is a pre-defined constant. This variable evolves exponentially from 1 to 0. It used to remove the direct dependency on time of the forcing term $f(s)$ and also to weight the forcing term to continuously shift towards a purely goal-attracted system [48]. The dynamical system as described above was called as a transformation system. Figure 3.1 sketches the DMPs model: the canonical system drives the nonlinear function $f(s)$ which perturbs the transformation system.

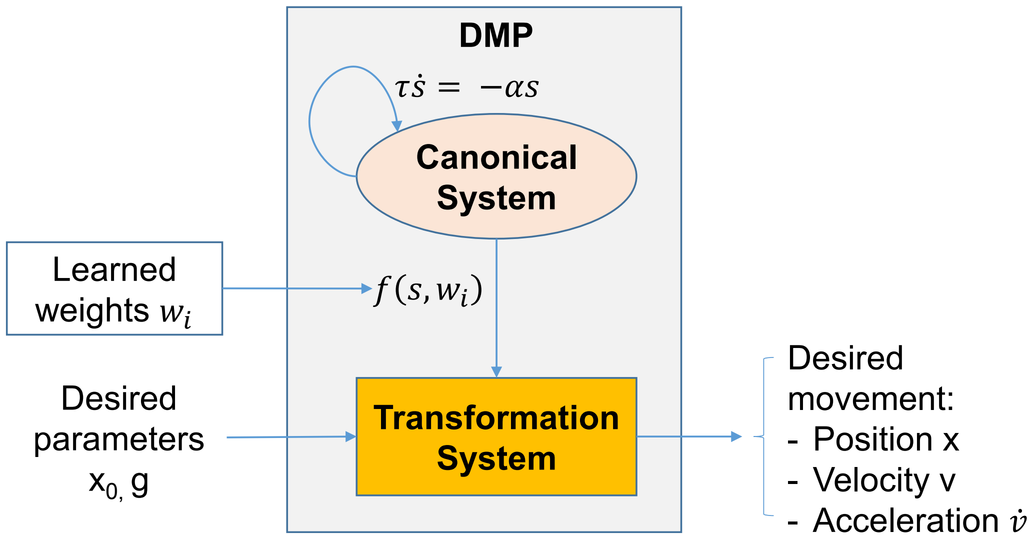


Figure 3.1: Illustration of a one-dimensional DMP model

3.2.2.2 Learning and Executing DMPs

The DMP model as described above can be used to learn a movement trajectory. The parameters w_i in (3.2) are adapted through the learning process so that the nonlinear function $f(s)$ forces the transformation system to follow the observed trajectory $x(t)$.

The first step of learning procedure is to initialize values to the parameters of the system. K and D is set beforehand responding to changes in the goal parameter. τ is the time constant and should be set to the duration of the sample trajectory $\tau = t_f - t_0$. The factor α in the canonical system (3.4) determines the decay rate of the phase variable, which is appropriate chosen to ensure that s will evolve to 0 at $t = \tau$.

Once these values are initialized, in the next step, the desired values of the forcing term is computed by extracting it from (3.1a).

$$f_{des}(s) = \frac{-K(g - x) + Dv + \tau\dot{v}}{g - x_0} \tag{3.5}$$

Then the values of the sample trajectory $x = x(t)$, which is recorded from human demonstration and its derivatives $v = \tau\dot{x(t)}$ and $\dot{v} = \tau\ddot{x(t)}$ are computed for each time step $t = t_0, ..., t_f$ and inserted in the (3.5). x_0 and g are set to $x(t_0)$ and $x(t_f)$, respectively. Next, the canonical system in (3.4) is integrated. $s(t)$ is computed with an appropriately adjusted temporal scaling τ by: $s(t) = exp(-\frac{\alpha}{\tau}t)$.

With these arrays of desired values $f_{des}(s)$, the appropriate centers and widths of the basis exponential functions in (3.2) can be set, and the weights w_i in (3.2) is found by minimizing the error criterion by least squares $J = \sum_s (f_{des}(s) - f(s))^2$, which is a linear regression problem. Figure 3.2 summarizes the process of learning phase of DMPs model.

The results of learning phase are learned weights w_i. The movement plan is generated by reusing the learned weights w_i. The desired start position x_0 and the goal position g are selected as required by the movement. The canonical system is reset by assigning the phase variable $s = 1$. By replacing the learned parameters w_i, adapting the desired movement duration τ, evaluating $s(t)$, and computing the nonlinear function $f(s)$, the desired trajectory is obtained from integrating the transformation system in (3.1a). Figure 3.3 summaries the process of executing phase of DMPs model.

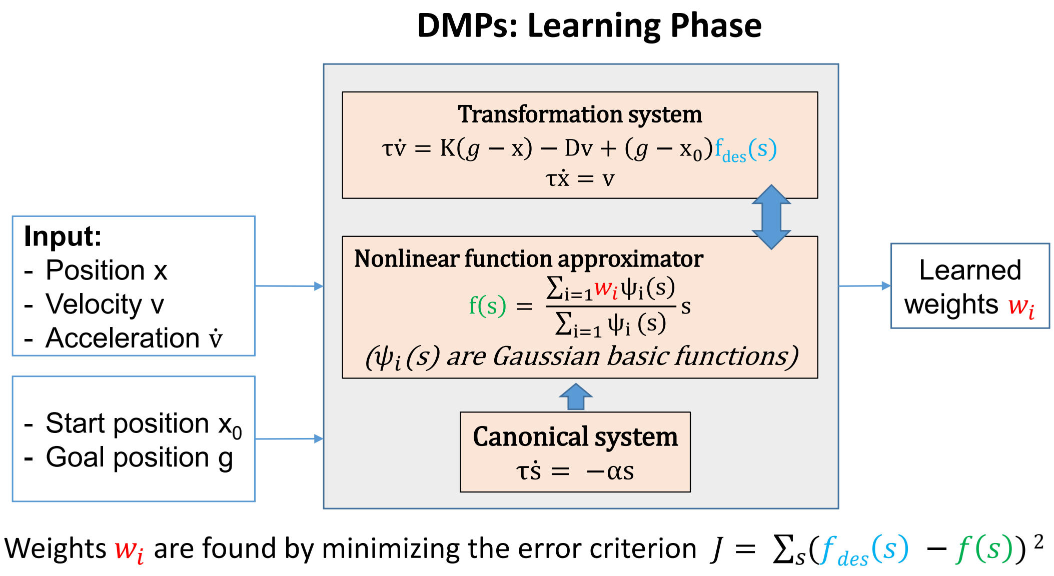

DMPs: Learning Phase
Input:
- Position x
- Velocity v
- Acceleration $\dot{v}$
- Start position x_0
- Goal position g
Transformation system
$\tau\dot{v} = K(g - x) - Dv + (g - x_0)f_{des}(s)$
$\tau\dot{x} = v$
Nonlinear function approximator
$f(s) = \frac{\sum_{i=1} w_i \psi_i(s)}{\sum_{i=1} \psi_i(s)} s$
($\psi_i(s)$ are Gaussian basic functions)
Canonical system
$\tau\dot{s} = -\alpha s$
Learned weights w_i
Weights w_i are found by minimizing the error criterion $J = \sum_s (f_{des}(s) - f(s))^2$

Figure 3.2: The summary of learning phase of DMPs model

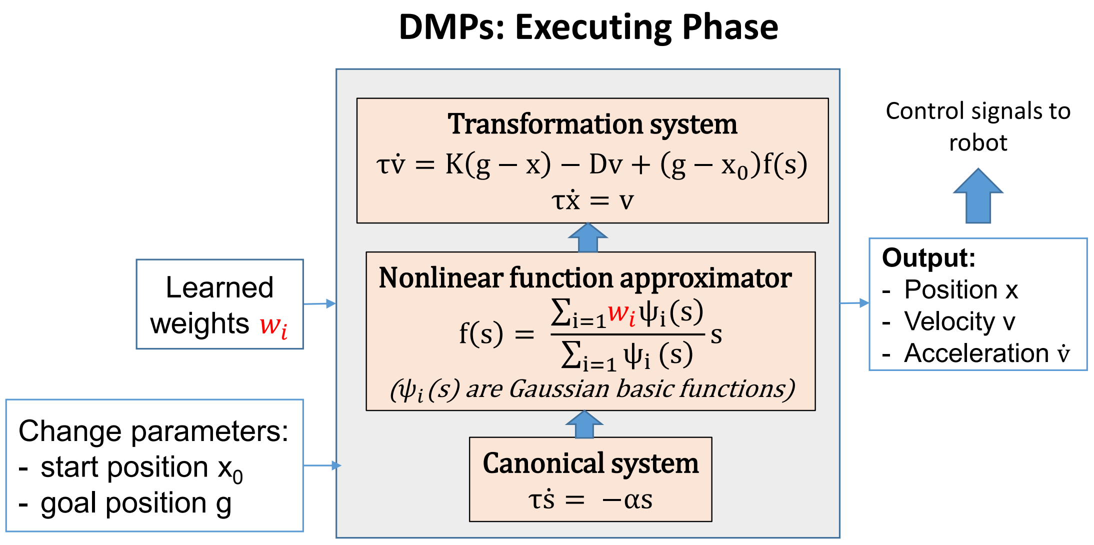


Figure 3.3: The summary of executing phase of DMPs model

3.2.2.3 Characteristics of DMPs

DMPs framework has the characteristics which are advantages when applying this framework for low-level learning of movements to generating total movement for the action.

- **Multiple Degrees of Freedom** Using DMPs for real world applications like moving a robot arm requires them to encode more than one-dimensional movements. This can be obtained by using one transformation system for each dimension of the data.
- **Sequence DMPs** DMPs are intended to be single basic units of movement. To allow more complex movements it is possible to sequence several DMPs.
- **Online Adaptation** The goal position g in Equation 3.1a can changed at any time of the execution. In the Learning for Demonstration context, online adaptation is the crucial advantage of using DMPs to encode a movement.
- **Movement Recognition** The similarity of two movements can be determined by comparing the weights w_i of two movements with each other.

3.2.2.4 Improvements of DMPs

The original formulation of DMPs as presented in the previous section, has some drawbacks with the adaptation to new goals. When the changes of the goal are extremely small, the movement between start and goal position does not adapt as expected. The work in [21] presents a modified version of DMPs which is not affected by this drawbacks. While the transformation system is changed to the following formulation, the canonical system stays the same as in equation (3.4).

$$\begin{aligned} \tau\dot{v} = K(g-x) - Dv - K(g-x_0)s + Kf(s) \\ \tau\dot{x} = v \end{aligned} \tag{3.6}$$

The function f(s) is the same as defined in the original formulation (Equation (3.2)). But the function f is not multiplied by $(g - x_0)$ any more. This helps to prevent the problems with movements which start and end at the same position

($g = x_0$). It also makes the adaptation to only slightly changed goals more optimal. The most important characteristic of this improved formulation is that it generalizes to new goals well. The third term $K(g - x_0)$ is required to avoid jumps at the beginning of the movements [45]. Learning and propagating DMPs is achieved with the same procedure as before, except that the target function $f_{des}s$ is computed according to

$$f_{des}(s) = \frac{\tau\dot{v} + Dv}{K} - (g - x) + (g - x_0)s. \tag{3.7}$$

This new formulation are used in our implementation in the next section.

3.3 Learning Action from Hand Movement in Human Demonstration

3.3.1 The Approach

Hand movement in human demonstration of desired action is tracked by a motion tracker. There was numerous methods to capture human motions depending on the different scenarios of application. Many methods are based on motion tracking system using vision or other wearable motion sensors for directly recording human motions. To capture hand motions, some approaches using external markers or special equipment such as Data Gloves were proposed in [7], [59], [24]. In [23], [35], the authors used a marker-less hand tracker which can reliably track a skeletal hand model. This approach based on a 3D hand model, which has the advantage of accurate estimation of hand pose. The 3D data in real world is captured from Kinect FORTH tracking system using the hand model based method. To deal with the generation of movement for robots from observed data, some approaches have been proposed. The works at [17], [4], [66] solved the problems of trajectory generation from a database of demonstrated movements. In [4], the authors proposed to learn demonstrated movements for a robot by using Hidden Markov Models. In [7], S. Calinon et al. also proposed an approach based on Hidden Markov Model and Gaussian Mixture Regression for learning and reproducing of gestures by imitation. Forte and et al. in [17] applied Gaussian process regression for multiple sample trajectories to learn task parameters.

We proposed a method for learning and reproduction of a demonstrated action. This approach is based on a single demonstrated trajectory from hand movements. The movement of a manipulation action is segmented into movement primitives. Then movement primitives are encoded by DMPs with the ability of adaptation to new changes. A sequence of DMPs is used to generate the whole movement of desired action. The adaptation to changes in the environment is made possible by the ability to change the goal position of a DMPs. In order to know how the goal position should be changed, a reference object must be identified. Tracking the position of target object enables the system to update the goal position online, which in turn makes it possible to move objects while the action is performed.

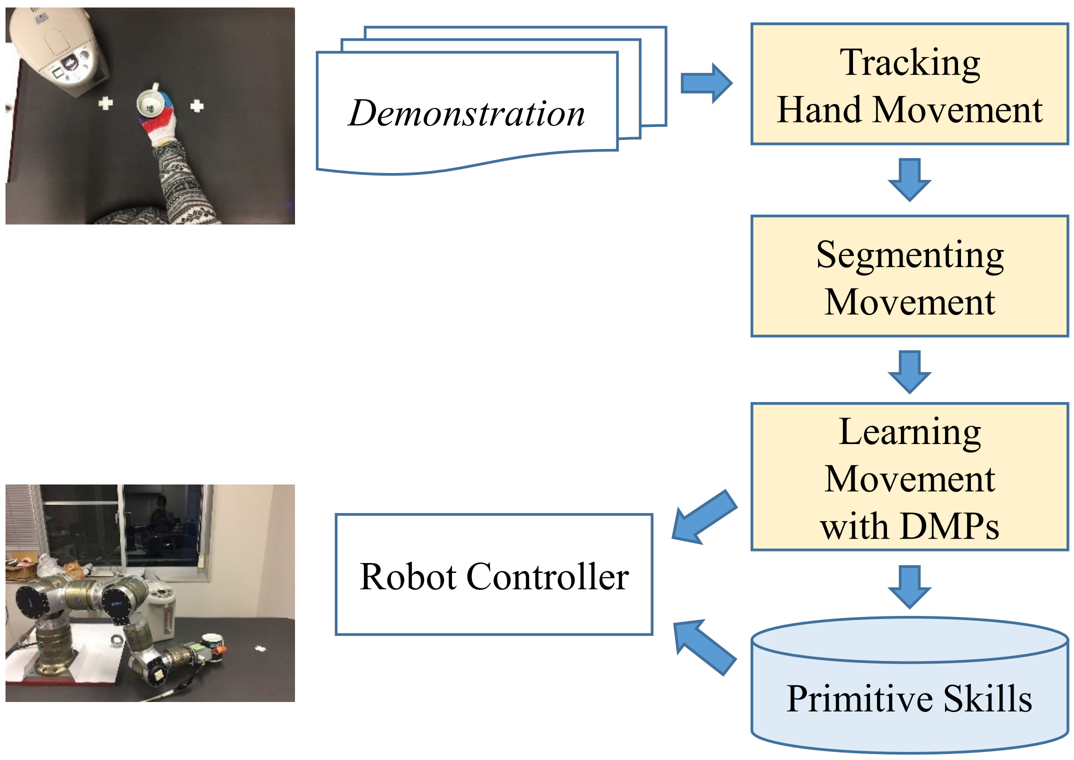


Figure 3.4: The flow of proposed method for learning action from hand movements in human demonstration with Dynamic Movement Primitives (DMPs)

Figure 3.4 shows the architecture of proposed method including three main steps:

- Recording Hand Movement
- Segmentation of Movement

- Learning Movement with DMPs

3.3.2 Recording Hand Movement in Human Demonstration

3.3.2.1 Hand Movement Tracking Solution

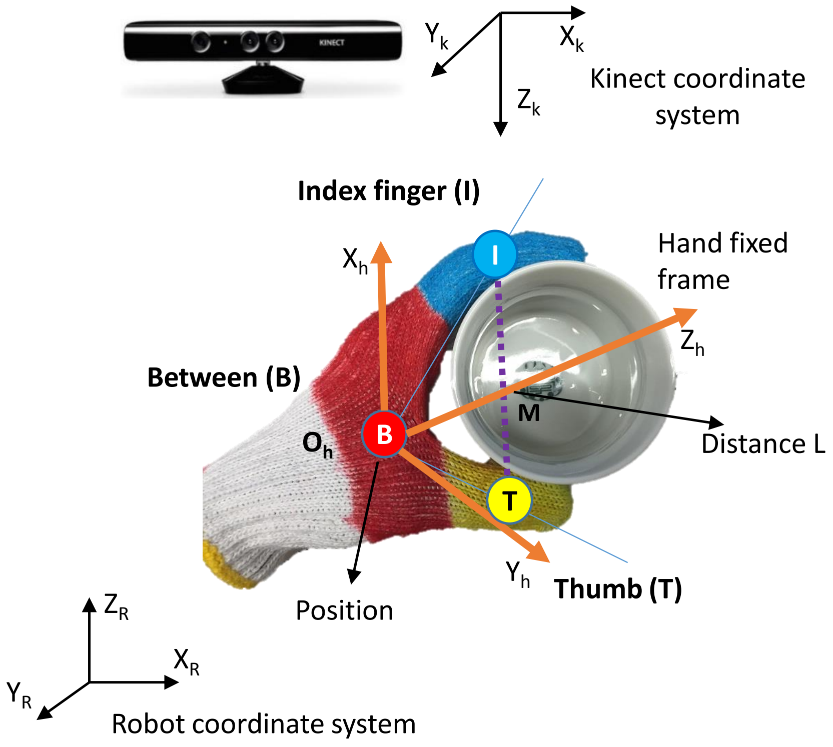


Figure 3.5: The hand motion tracker using Kinect camera with a color-marker glove

We designed a hand motion tracker by using a Kinect camera to track the movement of human hand wearing a color-marker glove. Figure 3.5 describes this motion tracker. Kinect camera is mounted in a fixed coordinate system which is Kinect's coordinate system. To have the best view, it is mounted on the top of working space for observing the hand movement of a human wearing the color-marker glove and demonstrates a desired action. The color-marker glove has three different color parts on the thumb, the index finger and the part between them. When the human wears this glove and demonstrates the action, the color markers are captured continuously by Kinect camera. We define three important points

I, T, B which are center points of color parts on the index finger, the thumb and the part between them, in respectively. Kinect camera captures both color image and depth image at the same time in data streaming sequences with a sequence of timestamp. The center points of color markers are detected in the image color by applying image processing algorithms with color image. Then, their 2D coordinates together with their depth values obtaining from Kinect's depth sensor are projected to world coordinate system to archive their three dimensional (3D) positions. The series of these image processing steps is shown as in figure 3.6. The detail of steps are summarized as follows:

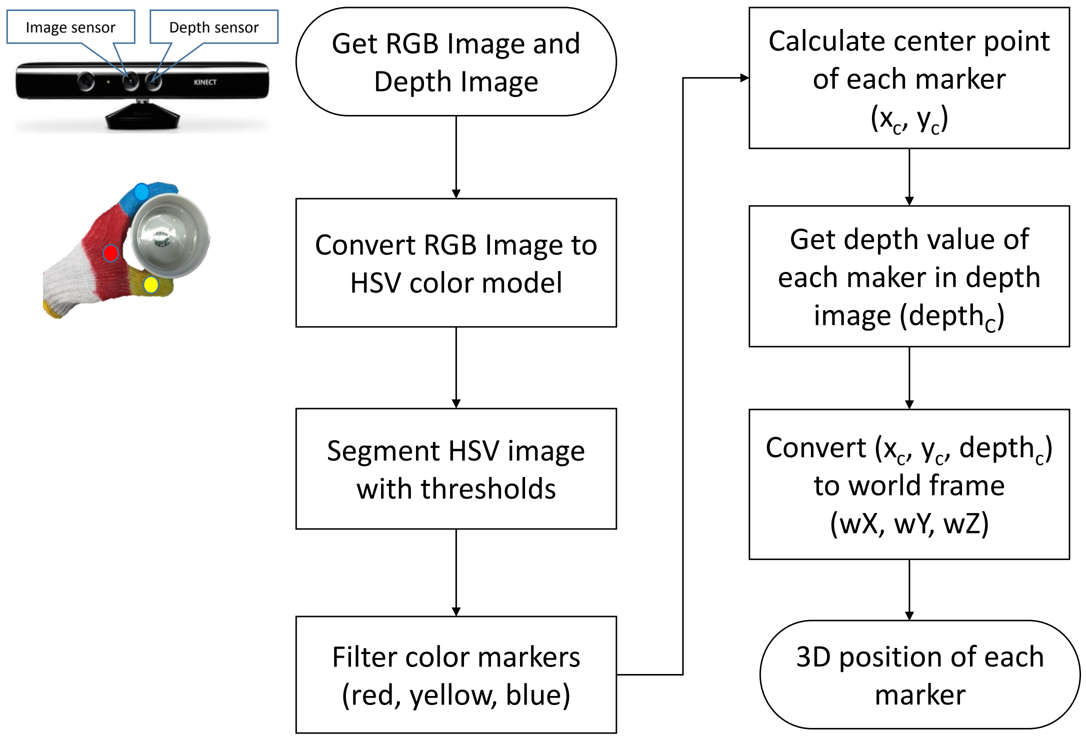


Image sensor
Depth sensor
KINECT
Get RGB Image and Depth Image
Convert RGB Image to HSV color model
Segment HSV image with thresholds
Filter color markers (red, yellow, blue)
Calculate center point of each marker (x_c, y_c)
Get depth value of each maker in depth image $(depth_c)$
Convert $(x_c, y_c, depth_c)$ to world frame (wX, wY, wZ)
3D position of each marker

Figure 3.6: Image processing steps for tracking color markers on the glove

1. Getting both color image and depth image from Kinect sensor simultaneously. Both images must be same size and be aligned (registration).

2. Converting RGB image to HSV color model.

3. Segmenting HSV image with each appropriate threshold. Removing segmentation noise using morphological operations (erosion and dilation).

4. Filter sequentially 3 color markers (red, blue, yellow). After this step, color markers are detected as areas on HSV image.

5. Calculating center coordination (x_C, y_C) of each detected color marker using the moments. Getting depth value at each color marker center to have $(x_C, y_C, depth)$.

6. Converting $(x_C, y_C, depth)$ to world coordinate to obtain 3D position (wX, wY, wZ) of each marker.

The 3D positions of points I, T, B are recorded during the demonstration corresponding to the recording frequency of Kinect camera. The data is captured as discrete points. Because these data are in the Kinect's coordinate system, they are transformed to Robot's coordinate system when using as control signals for the robot. This is easy by applying the coordinate system transformation. In addition, the tracked data may contain errors or be affected by noise. Therefore, we applied a moving average calculation to smooth the recorded trajectories of points.

Figure 3.7 shows the trajectories of 3D position of points B, I, T are tracked in the demonstration of the action 'pick up' a cup.

From the 3D position of three points I, T, B, the tracked data of hand motion is computed including hand position, hand orientation and distance between two fingers. The hand position is assigned by the 3D position of point B. The hand orientation is assigned by the orientation of a fixed frame $O_h x_h y_h z_h$ attached to the hand where the origin O_h is at point B; the axis $O_h z_h$ is identical with vector BM where M is middle of I, T; the axis $O_h y_h$ being right-hand perpendicular to $O_h z_h$ and satisfying $O_h y_h z_h$ is identical with the plane of B, I, T. the axis $O_h x_h$ being upside perpendicular to $O_h y_h z_h$. The orientation data of hand is obtained by the orientation of this fixed frame refer to the robot's base coordinate system. The distance between the index finger and the thumb is calculated by distance L in mm between point I and T.

Figure 3.8 shows the 3D trajectory of point B along with orientation vector $O_h Z_h$.

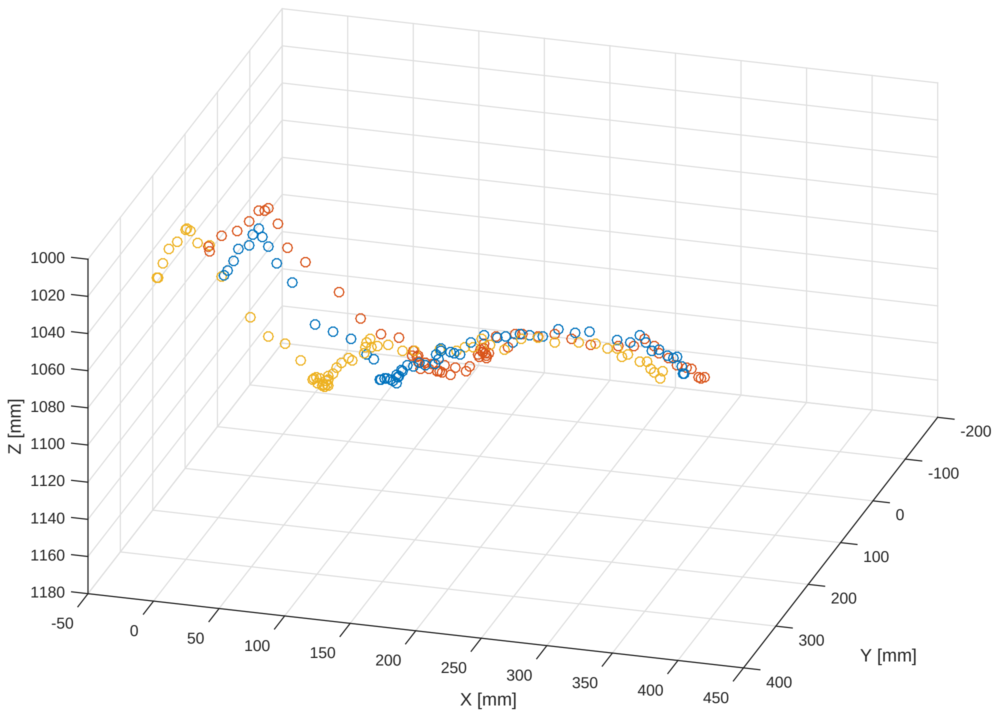


Figure 3.7: The 3D position trajectories of points B, I, T are recorded in the demonstration of the action 'pick up' a cup

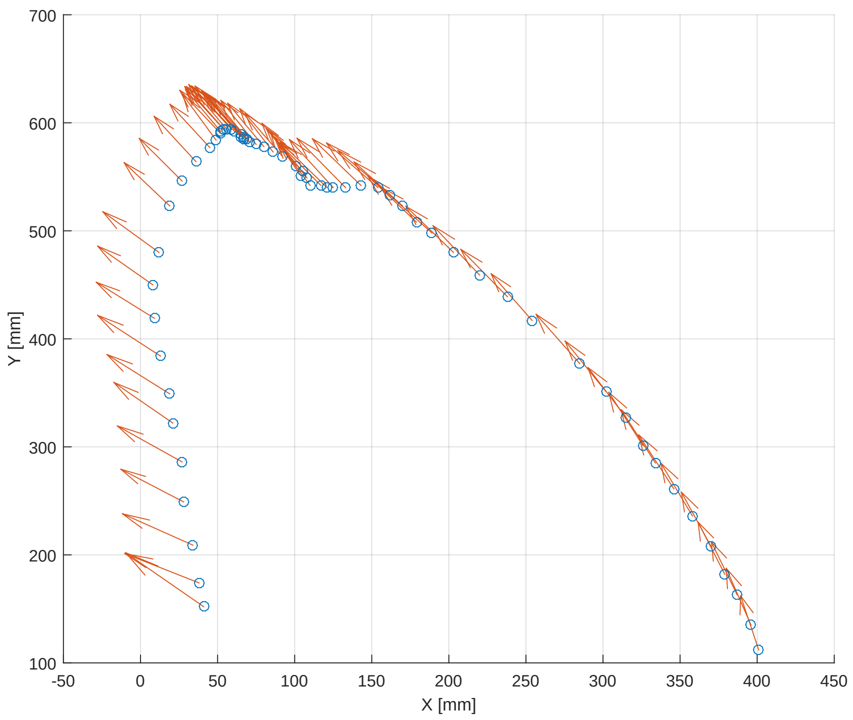


Figure 3.8: The trajectory of point B and orientation vector O_hZ_h, viewed in OXY plane

3.3.2.2 Hand Movement Data Description

By using the design and tracking solution as mentioned above, we can obtain desired tracking data including a set of discrete points captured by Kinect camera. The tracking data of hand motion for each demonstration of a manipulation action can be described as follows:

$$D = \{< pos_i,\ orient_i,\ dist_i >\}\ i = 1, 2, ..., n \tag{3.8}$$

where:
$pos_i = [p_{x_i}, p_{y_i}, p_{z_i}]$ is 3D position of hand (point B),
$orient_i = [x_{Z_i}, y_{Z_i}, z_{Z_i}, \alpha]$ is the orientation data of fixed hand frame which represents the hand orientation,
$dist_i$ is distance L [mm] between index finger and thumb.
pos_1 records the starting position of hand and pos_n records the position when the demonstration finished.

The table 3.1 shows a part of hand movement data which is recorded in the demonstration of the action 'pick up a cup'. The recorded data includes hand 3D position data (Pos_x, Pos_y, Pos_z), hand orientation data (Orient_x, Orient_y, Orient_z, Orient_angle) and fingers distance (Dist_L).

Table 3.1: A part of recorded data from demonstration of action 'pick up a cup'

Pos_x	**Pos_y**	**Pos_z**	**Orient_x**	**Orient_y**	**Orient_z**	**Orient_angle**	**Dist_L**
401	112	242	-0.221	0.944	-0.245	0.255	31
396	135	236	-0.281	0.918	-0.281	0.285	30
387	163	233	-0.305	0.894	-0.328	0.340	34
379	182	233	-0.315	0.878	-0.360	0.369	30
370	208	226	-0.340	0.889	-0.307	0.328	30
358	236	220	-0.365	0.874	-0.321	0.335	28
346	261	217	-0.395	0.865	-0.310	0.322	31
334	285	210	-0.393	0.874	-0.284	0.292	29
326	301	210	-0.393	0.874	-0.284	0.292	29
315	327	207	-0.445	0.835	-0.323	0.355	30

3.3.3 Segmentation of Movement

In movement learning from human demonstration, it is most common to teach constituent units of complex movements in isolation, before sequencing them into complex movements. This is realistic since demonstrations performed by humans can be decomposed into multiple different movement primitives. Thus, segmentation of observed movements plays an important role in learning movements for robots. A complex action is commonly segmented into simple movement units which are called motion primitives.

The human hand movements in manipulation actions are complex for robots. Depending on the types of different manipulation actions, the hand movements may contain: translation movement, rotation movement and fingers movement. Translation movement changes the hand position from a start point to a goal point. For examples, moving arm to approaching pose, moving arm to grasping pose, and moving arm away from object location are translation movements. Rotation movement makes the hand orientation change much around a axis. This movement occurs in actions such as 'turn a knob', 'open a bottle cap', 'open a door handle'. Fingers movement changes the distance between fingers when grasping or releasing objects. Recognition of movements is also a challenge for robots. In this study, we manually label for each simple units of movement (or motion primitives) which are segmented from a complex movement. Then robots can recognize what kinds of motion primitives to have appropriate executable plan for that motion primitives.

The result of movement segmentation is a sequence of motion primitives. The observed data of movement is segmented into segments of data. We present each motion primitive by a set M of recorded data which describes the data from start point to end point of that motion primitives.

$$M = \{< pos_s, orient_s, dist_s >, ..., < pos_e, orient_e, dist_e >\} \tag{3.9}$$

where $< pos_s, orient_s, dist_s >$ are position, orientation and finger distance of hand, respectively at the start point, and $< pos_e, orient_e, dist_e >$ are position, orientation and finger distance of hand, respectively at the end point in one motion primitive.

For most common human hand movements, it is reasonable to assume that the observed movement generally has three simple units. Firstly, reaching phase, during which the hand moves from a start position until it comes desired position such as a position in contact with the object for grasping that object or a position for releasing the object being hold in the hand. Secondly, manipulating phase, during which the hand conducts the movement to manipulate the object depending on the types of different manipulation actions such as grasping, releasing, turning (a knob), pressing (a button). Finally, withdrawing phase, which the hand moves away from object location after the manipulation is done.

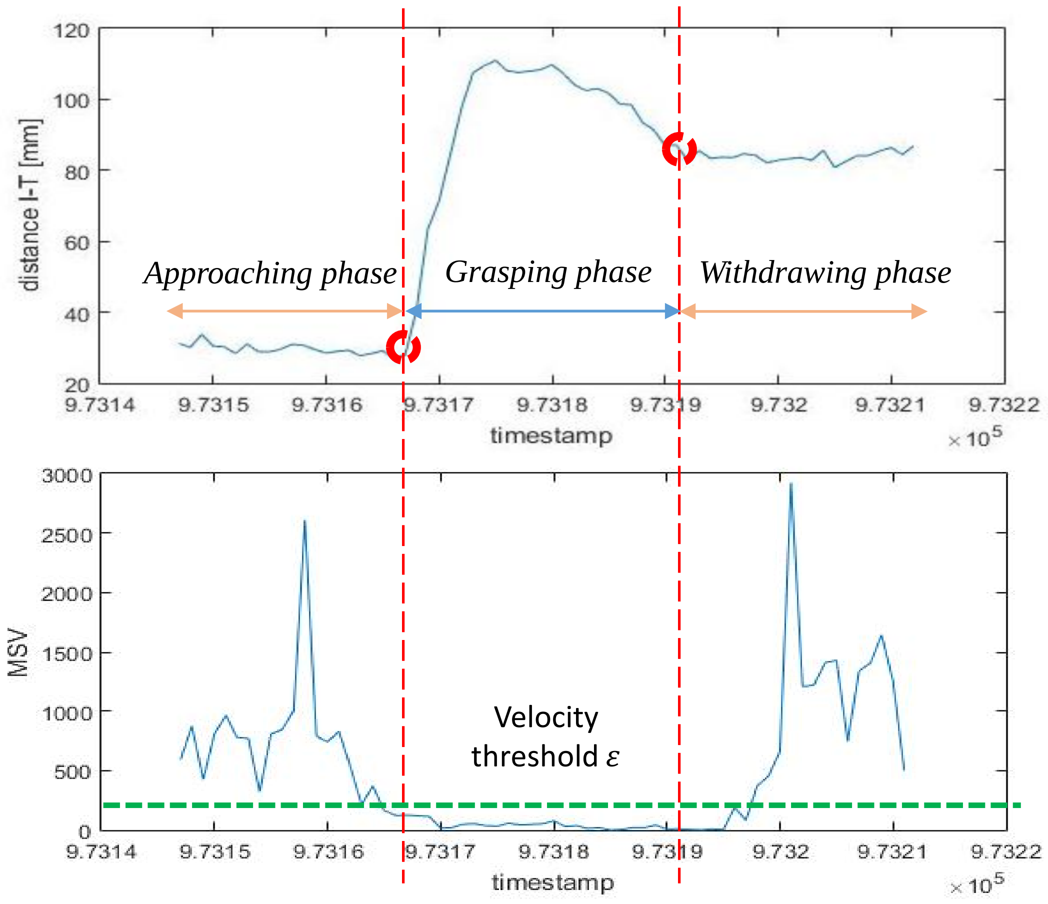


Figure 3.9: The segmentation technique based on two parameters distance L and hand velocity MSV

In order to segment a total movement, the detection of segmentation points is crucial. We proposed a segmentation technique using the velocity of hand motion and the change of fingers distance as two important parameters to detect segmentation points in the movement trajectories. First, the velocity of hand

Table 3.2: Segmentation Rules based on mean squared velocity (MSV) and fingers distance (L)

Hand Velocity (MSV).	Fingers distance L [mm]	Motion Primitives
$MSV > \epsilon$	$\Delta L \approx 0, L < L_1$	Reaching
$MSV < \epsilon$	$\Delta L > \rho$	Grasping
$MSV > \epsilon$	$\Delta L \approx 0, L > L_2$	Withdrawing

motion is computed by the mean squared velocity, which is the sum of squared velocity in 3D space, given by:

$$MSV(t) = (\frac{dx(t)}{dt})^2 + (\frac{dy(t)}{dt})^2 + (\frac{dz(t)}{dt})^2 \tag{3.10}$$

where:
$dx(t), dy(t), dz(t)$ are the 3D position differences,
dt is the time difference between two samples.

Mean squared velocity is used to determine stop points in hand motion trajectory. In natural demonstration, the velocity of hand motion decreases significantly to almost zero at the position of manipulating object such as grasping or releasing object. Therefore, by using a tiny threshold of velocity, the 'stop points' can be detected in a movement trajectory.

Figure 3.9 shows the values of distance L [mm] between two finger (upper grahp) in the hand velocity MSV (lower graph) in the demonstration of the actio 'pick up' a cup. There are two segmentation point to separate the movement into 3 parts: reaching phase, grasping phase and withdrawing phase. The segmentation rule to detect these segmentation points as shown in table 3.2.

3.3.4 Adaptive Learning of Hand Movement with DMPs

Imitating exactly an observed movement is unrealistic in a dynamic environment. Learning a demonstrated movement can be adapted to a new change like the change of goal of the movement. In other hand, an action can be performed by many different ways of movement trajectories. Therefore, one solution is to provide for the robot a generative model which can reproduce the movement to perform the action learned from observing human demonstration.

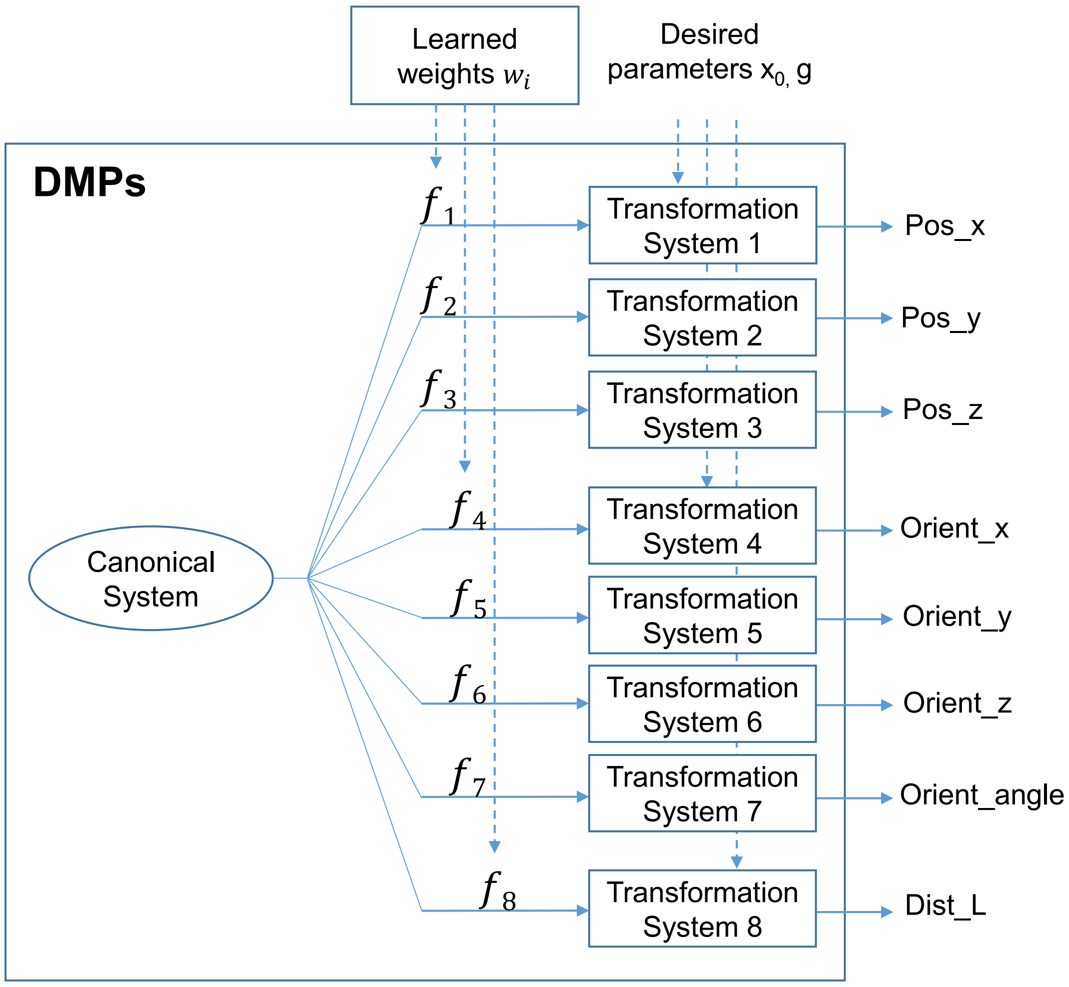


Figure 3.10: Sketch of a 8-dimensional DMPs to encode a motion primitive

The observed complex movements can be segmented in multiple simple movement units, each described as a motion primitive. In order to encode each unit of movement by DMP model, the ability of movement segmentation plays an important role. However, the complex movements often include not only translation motion from start point to goal point but also motion of object manipulation such as grasping or releasing the object. This leads to controlling both the translation movement of robot end-effector and the opening/closing state of robot gripper. The multiple degree-of-freedom characteristic of DMPs framework allows to extend this framework to describe the multi-dimensional data including the 3D position and the orientation of robot end-effector and the open/closing state of robot gripper.

The movement data of a motion primitive is encoded by multiple dimensional DMPs. We used a 8-dimensional DMPs to represent a motion primitive which is segmented from the recorded hand movement. The control signal includes 3D position of orientation of hand, the distance L between two fingers. Figure 3.10 illustrates the sketch of a 8-dimensional DMPs to represent a motion primitive.

In the section 3.2 we describe the background of DMPs for one dimensional trajectory. This model is extended for encoding a motion primitive from observed data of hand movement. We separate each degreee of freedom of the observed data and employ for each an individual transformation system of DMPs framework. In particular, the employed variables include the 3D hand position (x, y, z) in Cartesian coordination system, the hand orientation represented by (Orient_x, Orient_y, Orient_z, Orient_angle) and fingers distance (Dist_L) between the index finger and the thumb as shown in figure 3.10.

The control signals for a robot arm is generated respectively with the adaptation to the changes of start or goal point of movement.

A complete movement plan for robot actions will be generated by combining movement primitives which are represented by a set of learned DMPs. The DMPs are sequenced to build the whole movement. The successive DMP is started just after the preceding DMP has finished. This is straightforward since the boundary conditions of a DMP are zero velocity.

The complex movements of an action for robot can be generated by sequential composing of movement units represented by DMPs. To solve the problem of generalization for complex movements, one idea is that the DMP framework can be applied to build a library of movement primitives. Each of movement primitives, which was recorded from human demonstration, is represented by a DMP model, and labeled accordingly. The complex movements of a robot action will be composed by sequencing movement primitives from this library. In addition, a library of motion primitives containing learned DMPs for popular motion primitives which can be reused with the change of parameters such as start and goal position. With a library of motion primitives, the generation of complete movement for robot action will be a simple high level command to choose appropriate primitives, sequence them and set action specific required parameters. The adaption to new situations is satisfied by adjusting the start, the goal, and the movement duration.

5

3.4 Experiments and Results

3.4.1 Experiment 1. Learning a movement adapted to new goal

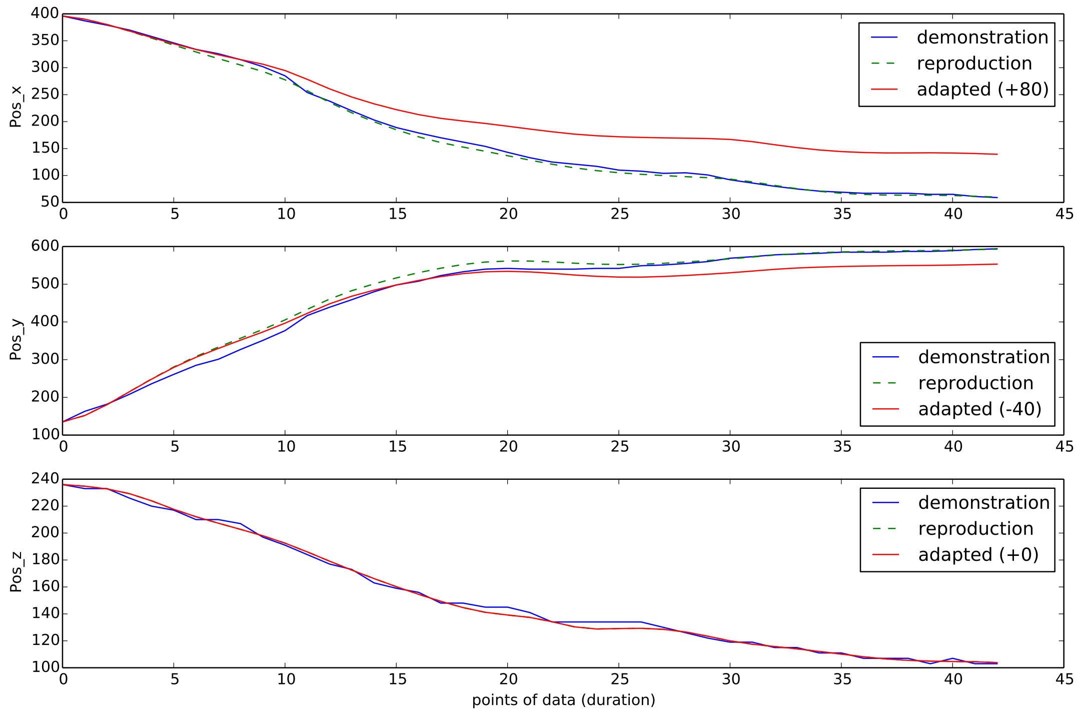


Figure 3.11: The 3D position data (x, y, z) of recorded demonstration (blue line), reproduction (dashed green line) and adapted (red line) with the change to new goal

Firstly, we implemented the procedure for learning hand movement with DMPs as described in section 3.3. We recorded the hand movement in demonstration of the action 'pick up a cup'. After segmenting, the hand movement is separeated into motion primitives including 3 phases: (1) approaching the cup; (2) grasping the cup; (3) withdrawing from the cup's location. The learning method with DMPs is implemented the motion primitive 'approaching the cup' to validate the generated movement trajectory with the adaptation to the change of goal position when the cup's location was changed.

Figure 3.11 depicts the 3D position data (x, y, z) of recorded demonstration (blue line), reproduction of movement without adaptation to new changes (dashed green line) and adaptive learned movement (red line) with the change to a new goal by adding an offset (+80, -40, 0) [mm]. These movement trajectories viewed in 3D space are shown in figure 3.12 which are represented in robot's coordinate system.

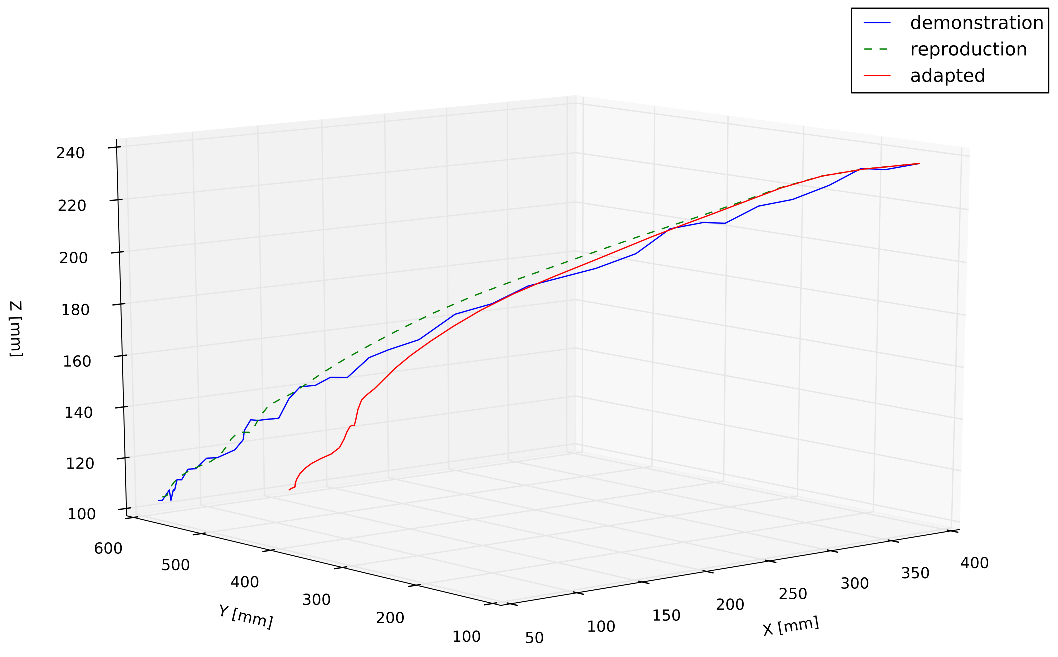


Figure 3.12: The 3D trajectories (in robot space) of recorded demonstration (blue line), reproduction (dashed green line) and adaptive learning (red line)

As can be seen from these figures, the trajectory of reproduction movement without the adaptive change imitates almost same as the trajectory of demonstrated movement. While the trajectory generated by adaptive learning has similar shape with demonstration trajectory. In addition, the generated movement is smoother than recorded movement.

Figure 3.13 shows the orientation data of recorded demonstration (blue line) and reproduction (dashed green line). The orientation with a change of desired orientation is not shown in this figure. However, the change of orientation is learned in the same way for position. The change occurs in some situations, for example, the change of object pose lead to changing the grasping pose (or hand pose) of a robot. Similar to position trajectory, the data of orientation reproduced is much smoother than recorded demonstration.

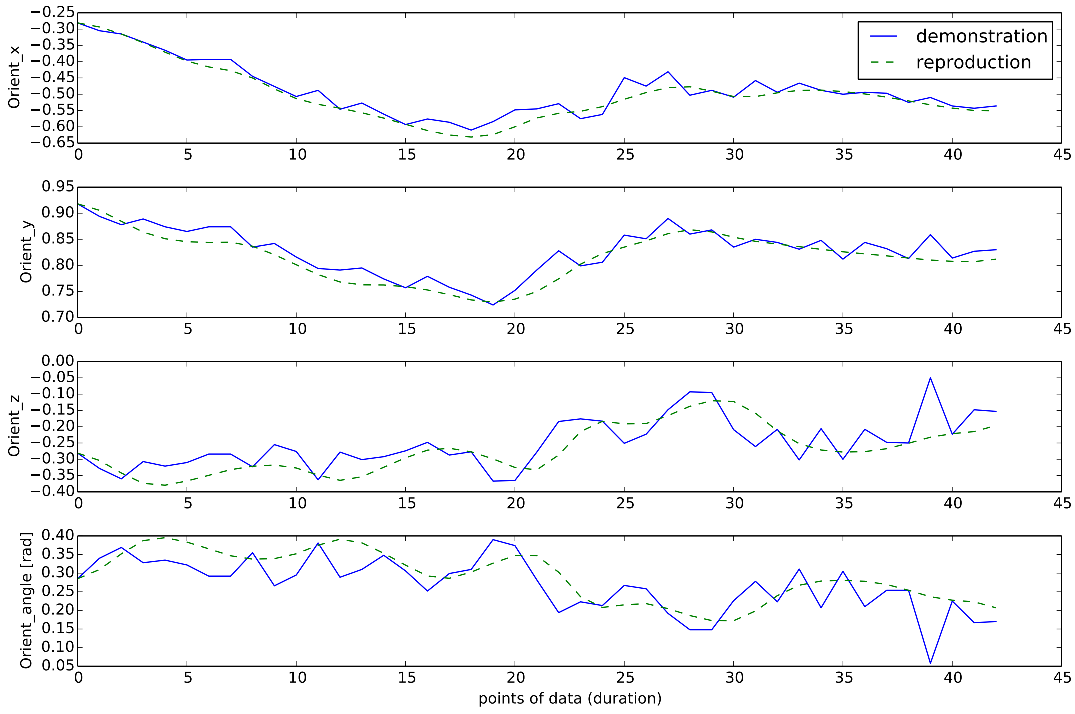


Figure 3.13: The orientation data (x, y, z, angle) of recorded demonstration (blue line), reproduction (dashed green line)

After generating the movement by using DMPs model, the generated movement data is used for control a robot arm and 2-finger hand. The robot used in this experiment is an 7-DOF (degree of freedom) robot arm Schunk LWA3 (Light Weight Arm) with a parallel gripper Taiyo ESG1-FS-2840 as shown in figure **??**.

In this experiment, we implement the action 'pick up the cup' in two cases. In first scenario, the robot perform the action with movement trajectory as demonstrated (without change of cup's location). Then, in second scenario, the robot perform once again the action with movement trajectory adapted to new location of the cup by adding an offset (+80, -40, 0) to have new location of the cup. Figure 3.15 shows photos of the robot arm LWA3 performing the action 'pick up the cup' with new location of cup. The robot performs the movement 'approaching

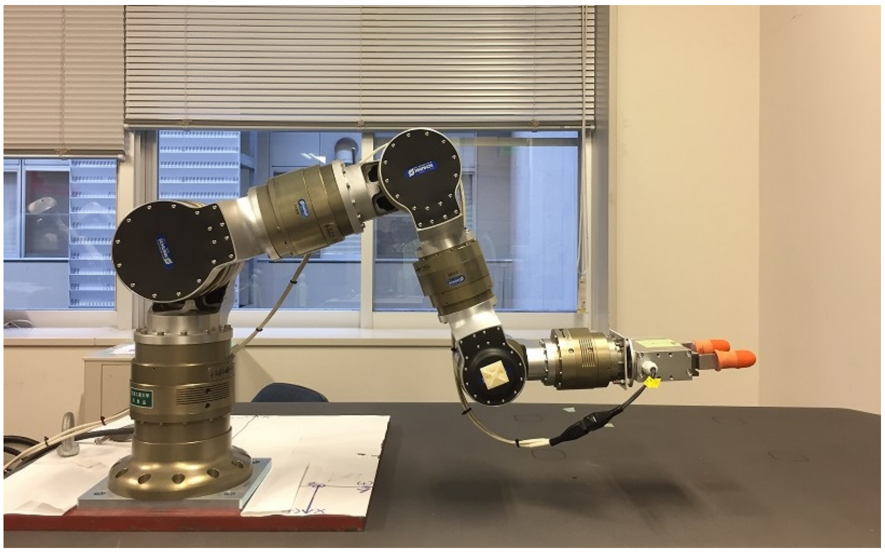


Figure 3.14: Robot arm Schunk LWA3 and parallel gripper used in the experiment

cup' with the generated movement data by DMPs model and the primitive skill 'grasping the cup' is manually programmed for this robot hand.

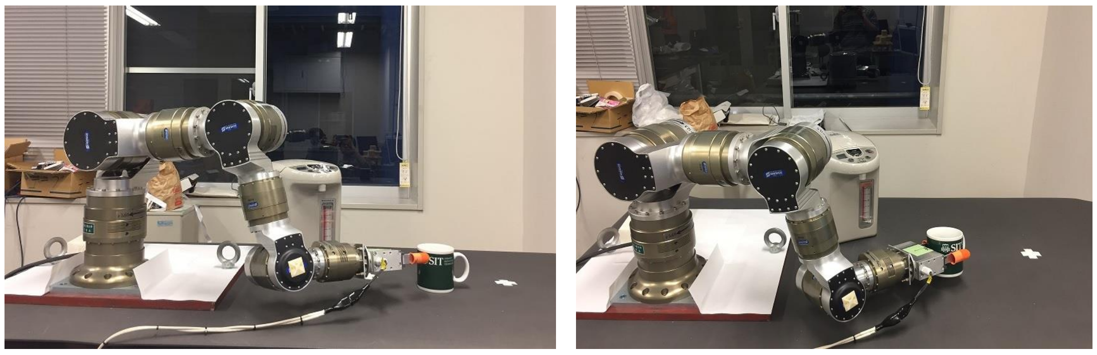
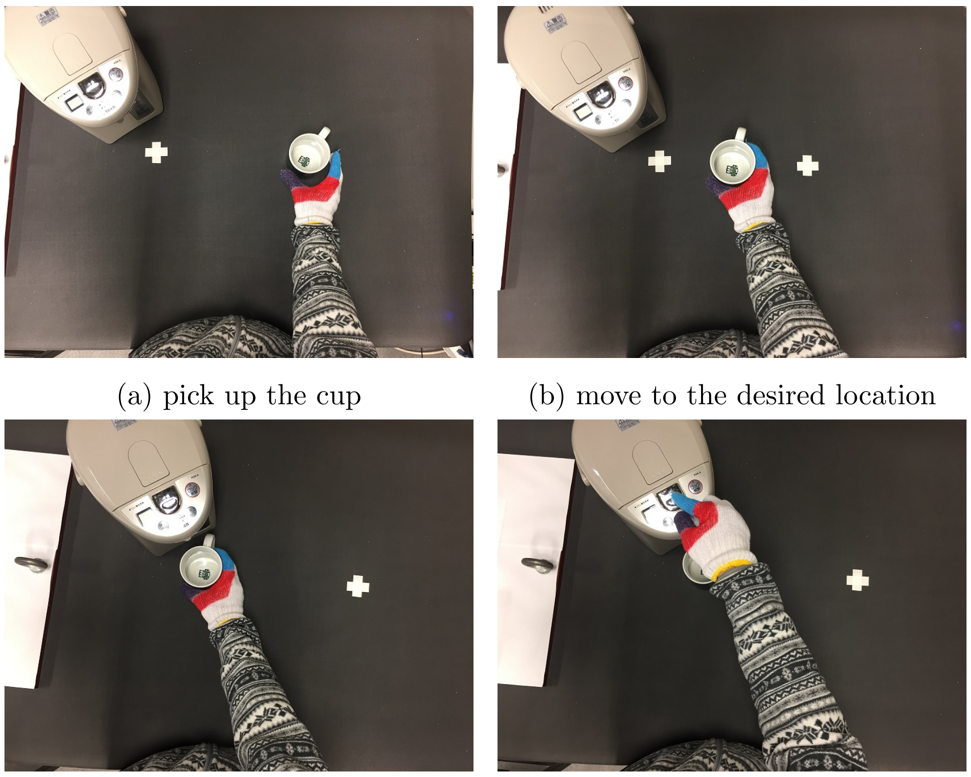


Figure 3.15: Robot arm LWA3 performs the approaching movement to the cup in new location (left photo) and grasping the cup (right photo) with 2-finger hand

3.4.2 Experiment 2. Executing the Task 'Dispensing Water'

In this experiment, we implement a complete task 'dispensing water' including the actions: picking up a cup, placing the cup under the spout, pressing a button on the water thermos pot. In order to perform these action by proposed method, the DMPs model is applied for sub-movements which follows the demonstrated


(a) pick up the cup (b) move to the desired location

(c) place the cup (d) press the button

Figure 3.16: Photos from (a)-(d): A human demonstrated the actions in the task 'dispensing water'

movement including approaching the cup, moving the cup to placing pose. For other movements, robot perform by primitive skills which are programmed for this robot arm and hand. The primitives skills are listed as below.

Firstly, the demonstration of the task "dispensing water" by a human is recorded. Figure 3.16 shows the actions in this demonstration including 'pick up a cup', 'place the cup under the spout' and 'press the button'. The demonstration is observed by Kinect camera mounted on the ceiling of working space. The observed data is transformed into robot's coordinate system.

Figure 3.17 shows the whole movement trajectory along with the orientation vector of hand movement of the demonstrated task. The figure 3.18 describes the movement trajectory is represented using DMPs model and segmented into sub-actions. The red symbols mark segmentation points where are critical points for segmenting sub-actions. The movement is segmented into segments of 'pick up a cup', 'place the cup under the spout', 'press the button', and 'pick up the cup (with some water)'. Each action includes motion primitives such as 'approaching', 'grasping', 'releasing','pressing'.

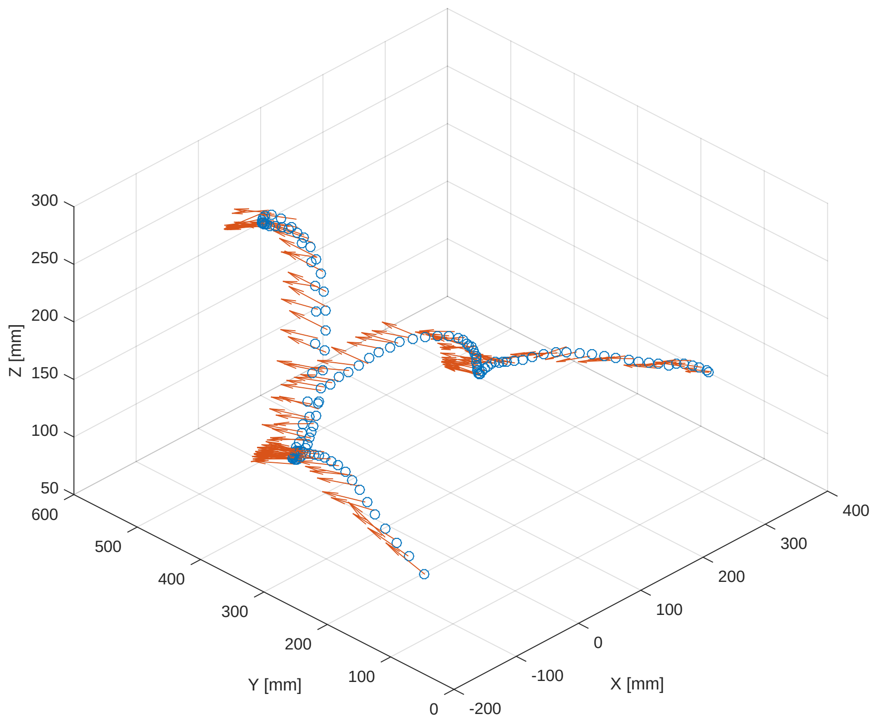


Figure 3.17: The 3D trajectory and orientation vector of hand movement acquired from human demonstration

In this experiment, we implemented the learning method with DMPs model for sub-movements which are following the movement trajectories in demonstrated task. For other primitives such as grasping, pressing which strictly depends on robot hand, we built as primitive skills for this robot. We proposed a collection of primitives skills which is necessary for perform the actions in the tasks by the robot. The primitive skills includes:

- FollowingTrajectory: To follow a movement trajectory from demonstration by using the generative model, DMPs model, the reproduced movement can adapt to a new desired goal of the trajectory.
- GraspingObject: To grasp an object by a parallel gripper. Assume that an appropriate grasp for the object is known and selected in advance.
- ReleasingObject: To release the object holding in the gripper.

- PressingButton: This skill is to move tip-point of gripper touching to a button with an appropriate force in a known amount of time. Assume that these parameters are known in advance. A vision ability helps the robot to recognize the position of contact point, for example, using a marker for the button.

- MovingToPos: To move the robot's end-effector to desired 3D position and orientation. This skill is not to follow demonstrated movement but to control freely the tip-point of end-effector to desired location when it is necessary to refine a goal position, for example before grasping an object or pressing a button.

In addition, this collection can be added more primitive skills, depending on the type of task. For instance, the skill RotatingObject is to rotate an object around a rotation axis with a desired angle. These primitive skills will be commonly used in many everyday tasks.

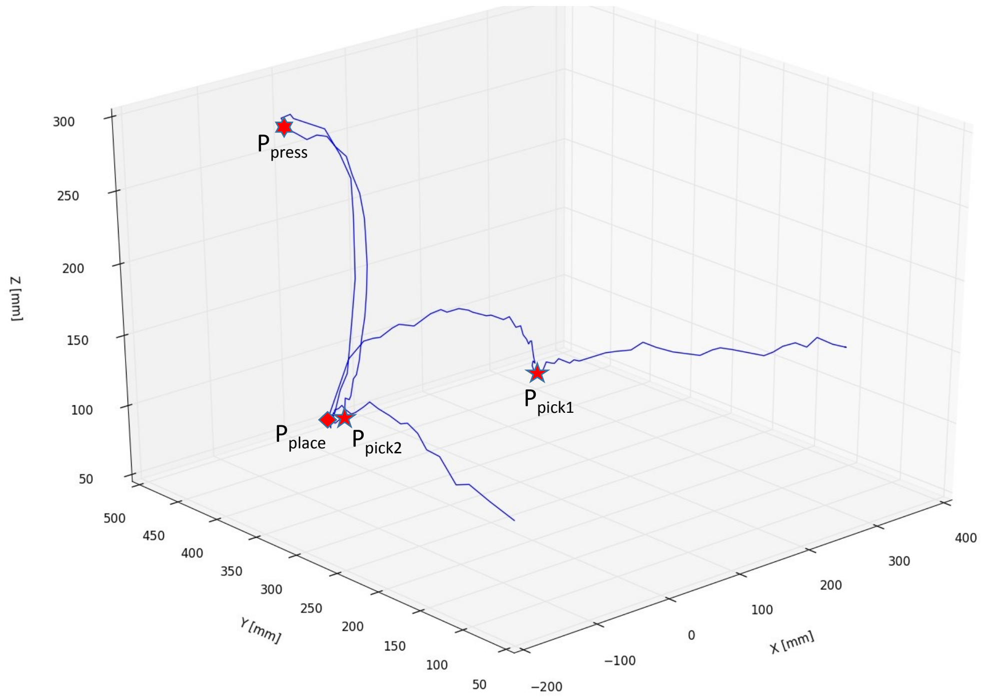


Figure 3.18: The movement trajectory of demonstrated task is represented using DMPs model. The red symbols mark segmentation points where are critical positions for grasping cup (P_{pick1} and P_{pick2}), releasing cup (P_{place}) and pressing button (P_{press}).

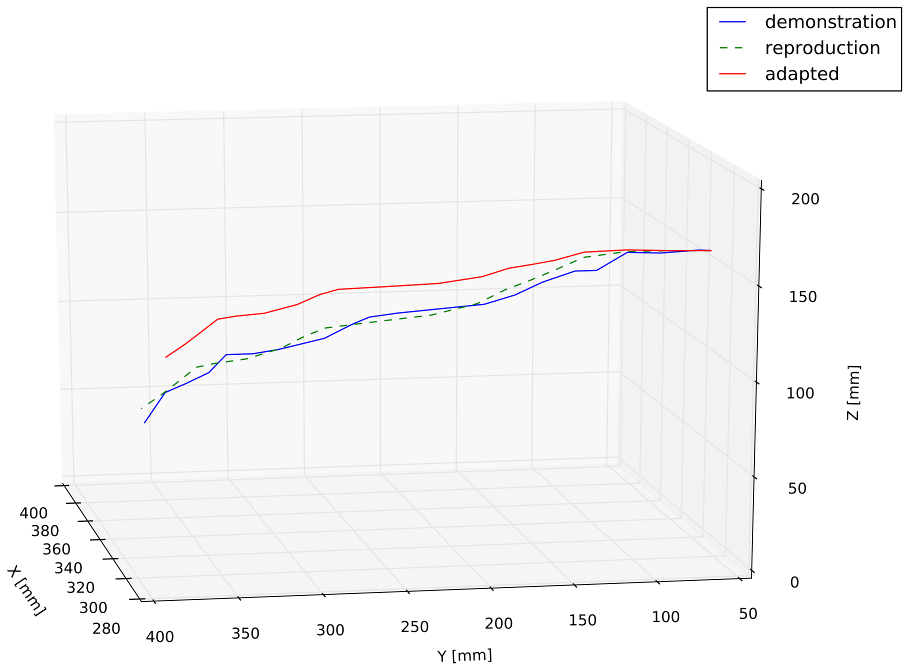


Figure 3.19: The trajectory of sub-movement "approaching object" is generated by DMPs model adapted to a new goal.

Figure 3.19 shows the movement trajectory of the sub-movement 'approaching the cup' which follows the recorded movement in the demonstration. The blue path is original demonstrated trajectory. The dash-green path is movement trajectory reproduced by DMPs model without change. The red path is movement trajectory generated by DMPs model with the adaptation to the change of goal by adding an offset (+60, -40, 0) [mm] to the original goal position. This adaptation is for the change of cup's location in comparison to the location in the observed demonstration.

The sub-movements of the actions in the demonstrated task are learned to generate adapted movements using DMPs framework. Then the task is automatically reassembled from the sequence of primitive skills. The control signals are generated to control the robot arm LWA3 including 3D position and orientation of its end-effector and the open/close movement of the gripper's finger. The control signal by position and orientation in task space is transformed to joint space by an inverse kinematic solution for this robot.

Figure 3.20 shows screen shots (from a-f) of the robot arm Schunk LWA3 performing the actions in the task 'dispensing water'. The sub-movements which follow the demonstrated movement trajectories such as 'approaching the cup' (a), 'approaching to placing location' (c) are generated using DMP model. The other motion primitives including 'grasping the cup' (b), 'releasing the cup' (d), and 'pressing the button' (e) are built as primitive skills for this robot arm (and robot hand).

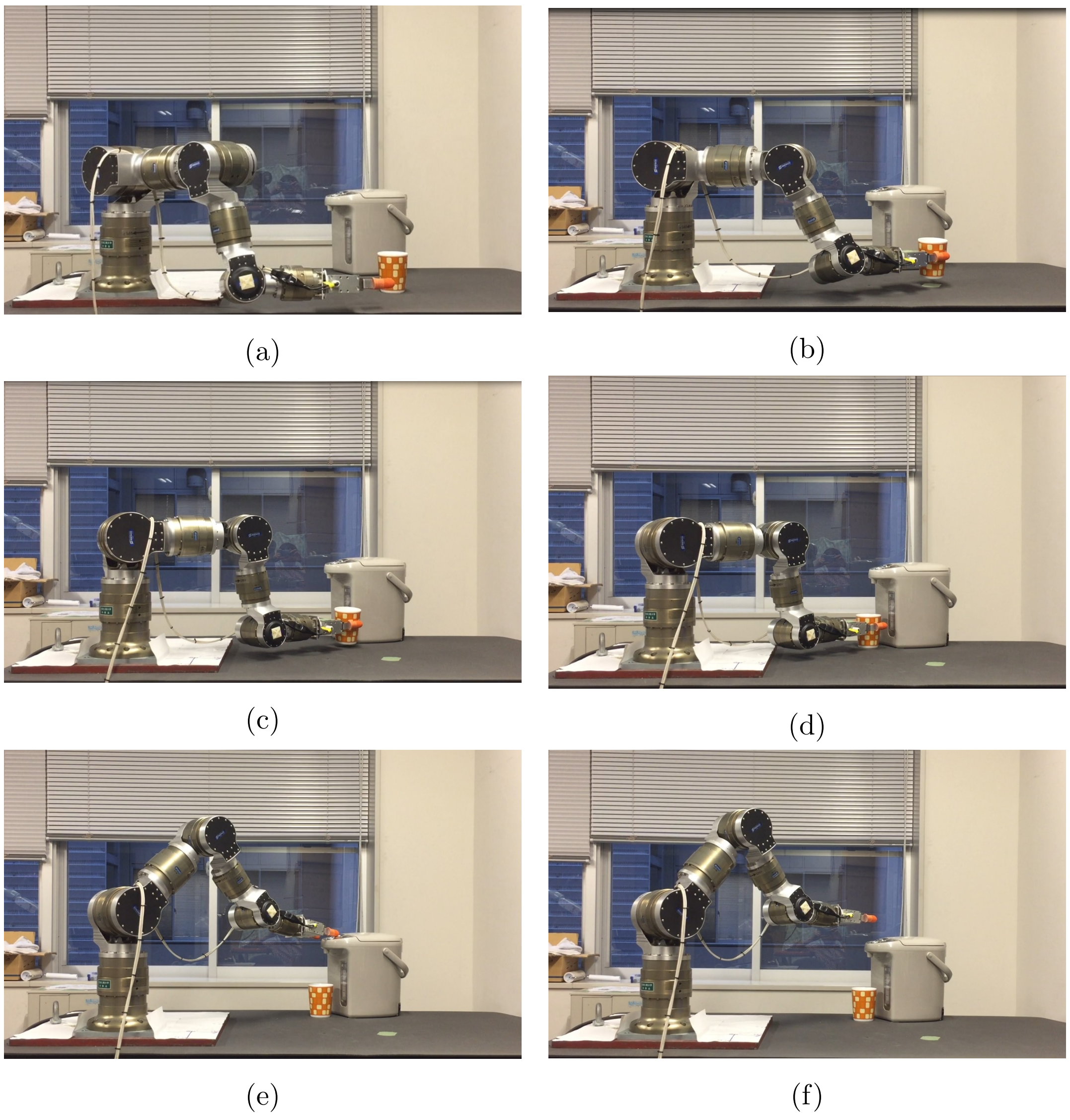
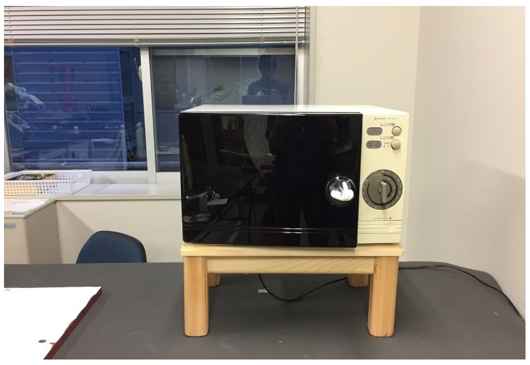


Figure 3.20: Screen shots from (a)-(f): Robot performs the actions in the task 'dispensing water' from a water thermos pot

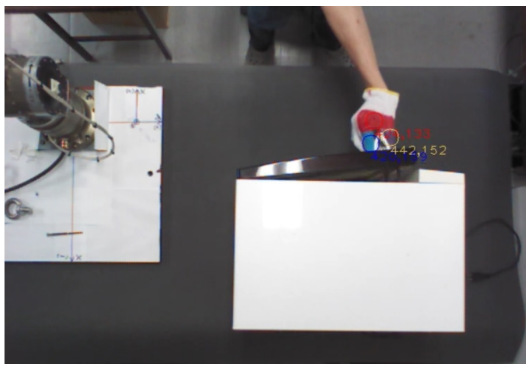
3.4.3 Experiment 3. Performing the Action 'Open a Microwave Oven's Door'

In this experiment, we implement the execution of the action 'open a door' such as a microwave oven's door by learning from human demonstration. In the movement which follows the demonstrated movement trajectory like approaching the cup in the task 'dispensing water', the generated trajectory is without constrains such as obstacle avoidance, robot's joints limitation. With these conditions, the movement trajectory also is generated by a simple method, solving the inverse kinematic solution from start configuration (position and pose of robot end-effector) to desired configuration without taking into account the shape of movement trajectory. However, in the movement like 'open a door', the movement trajectory is required for the robot to move the door along with a certain trajectory. Therefore, the reproduction of demonstrated movement by DMPs model is effective in this situation.

Firstly, the hand movement in demonstration of the action 'open a microwave oven's door' is recorded as shown in figure 3.21.



(a)



(b)

Figure 3.21: Experiment for recording the hand movement in demonstration of the action open microwave oven's door

Figure 3.22 depicts the movement trajectory of points B, I, T in this action which is transformed to the robot's coordinate system. From this data, the hand movement is assigned by the movement trajectory of point B and orientation of hand frame. Figure 3.23 shows the movement trajectory (point B) along with orientation vector $\vec{Z_h}$ of the action 'open a microwave oven's door'.

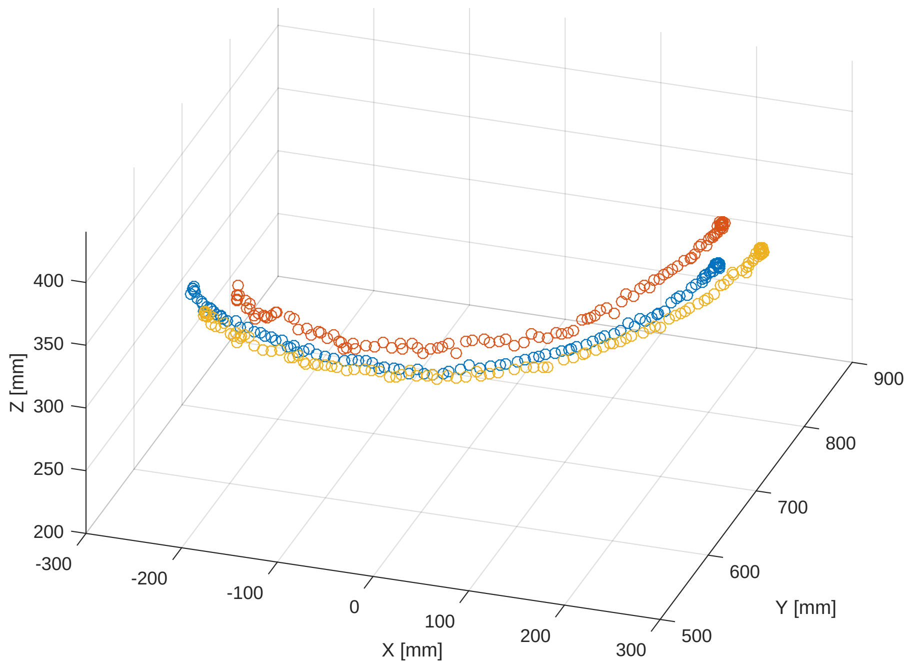


Figure 3.22: The 3D position of point I, B, T in the demonstration of the action open microwave oven's door.

The movement trajectory obtained from demonstration then is learned with DMPs model to generate the reproduction of movement trajectory. In this action, the reproduction trajectory is imitated the same as demonstrated trajectory without the change of goal position.

Figure 3.24 shows these movement trajectory after reproduction with DMPs model. These movement data will be used to control the robot arm's end-effector.

The movement trajectory is transformed to the robot arm's coordinate system which has the origin point at the robot base. However the limitation of working space of the robot arm when mounting on the flat surface such as ground or a desk surface is different in comparison with the working space of human arm. This correspondence problem can be avoided if the robot arm mounted horizontally like human arm.

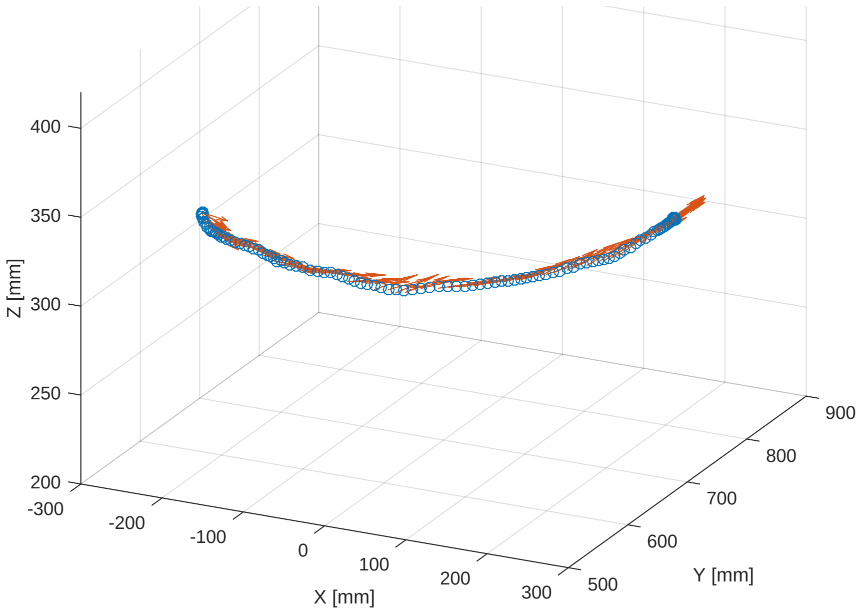


Figure 3.23: The movement trajectory (point B) along with orientation vector O_hZ_h.

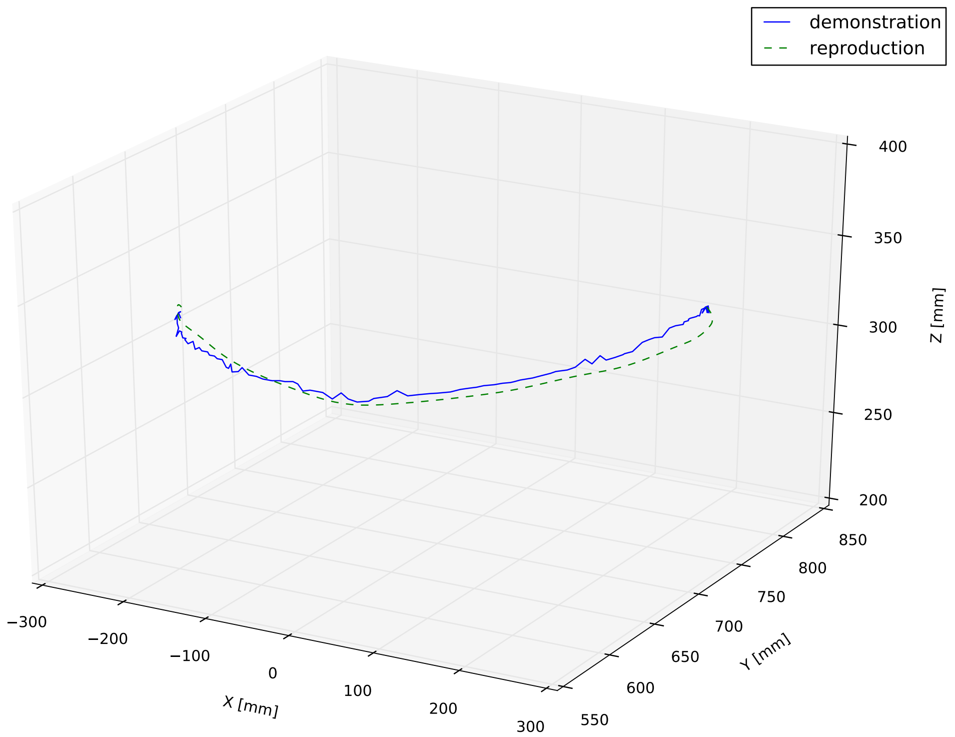


Figure 3.24: The movement trajectory of the action 'open microwave oven's door'. The blue path is demonstration, the dashed green path is reproduced by using DMPs model.

Chapter 4

Perceiving Object for Manipulation Actions

4.1 Introduction

Robots manipulating in human environment often be working with uncertainty due to their limited vision of a changing world. A perception system has the potential to reduce this uncertainty and enable robust autonomous manipulation. The perception becomes one of the key components of a robotic system that operates in a dynamic, unstructured environment. This work considers the problem of robot perception in domestic settings where, in order for the robot to be able to detect and manipulate objects in the environment, the robust perception is one important challenge demanding to be solved.

Humans use visual feedback extensively to plan and execute actions. However, panning and execution is not a well-defined one-way stream: how we plan and execute actions depends on what we already know about the environment we operate in, what we are about to do, and what we think our actions will result in.

Everyday manipulation tasks demand object manipulation as grasping, releasing, pushing are very popular. These actions require attention to different attributes in the environment such as the object's location, object's pose and object's identification. Many methods of object recognition and pose estimation were interested in both computer vision and robotics communities. In general, there are two main categories of methods for object recognition: 2D-based methods and 3D-based methods. The 2D-based methods often extract key points from an image. Then, a descriptor is evaluated for each key point, usually based on its surrounding pixels, and the descriptors are saved in a database for using in matching stage. These methods have the limit of the lack of spatial information. The 3D-based methods can overcome this limits. With the introduction of many types of 3D sensor devices, the 3D-based techniques for detecting, recognizing and pose estimation has been studied and applied in many applications.

This work is to investigate the state-of-the-art technique for 3D object recognition and pose estimation for developing a perception system for robot by a RGB-D camera introduced by Microsoft. We address the problem of a robot grasping 3D objects with known 3D shape from their projections in 3D point cloud. A cutting edge technique is using a 3D object descriptor Viewpoint Feature Histogram which was introduced in [51]. We take the advantage of several tools provided by the open source library, Point Cloud Library (PCL) [52] to implement the method. The PCL provides tools for retrieving the 3D point cloud from single view RGB-D image, segmenting objects from background and each other, computing the 3D feature of object cluster and matching to recognize object, and estimating the pose of object.

4.2 Background and Related Work

4.2.1 Point Cloud Data from RGB-D Camera

There are various types of vision sensors which can be equipped for robots to "see" the environment such as LIDAR, RGB camera, stereo cameras, structured light cameras, or a combination of these. With the advent of low-cost and open source SDK (software development kit), such RGB-D cameras as Microsoft Kinect or ASUS Xtion has become popular. These cameras contain a standard RGB color sensor and a structured light sensor that is capable of measuring the depth in an image. By means of an algorithm based on the color data and depth information, 3D image is generated as a set of 3D points known as a point cloud. Figure 4.1 shows an example of point cloud image taken with Microsoft Kinect.

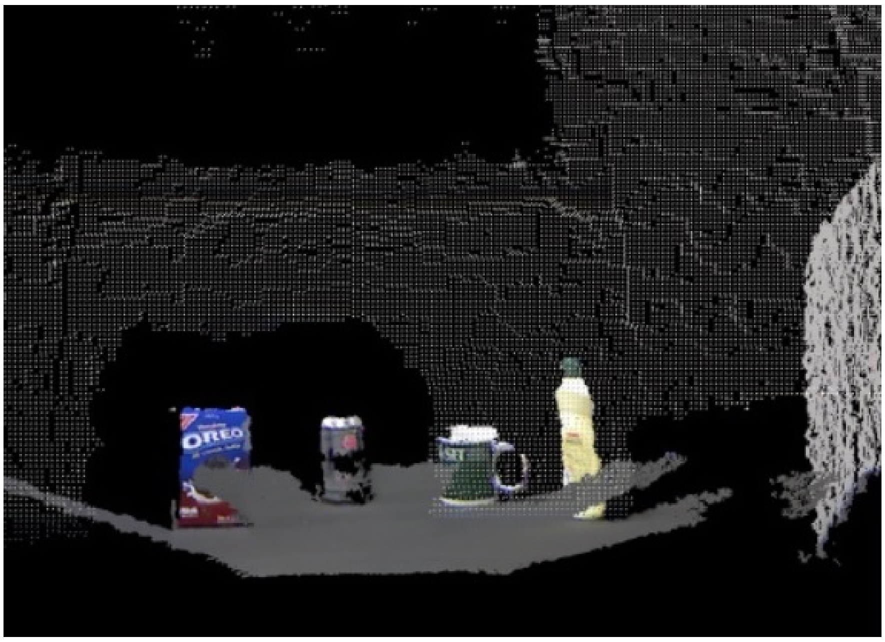


Figure 4.1: An example of point cloud containing objects from RGB-D camera

4.2.2 Object Descriptors

The algorithm to identify a 3D object uses the 3D feature descriptors of object by quantifying its geometric properties such as the surface normal directions, surface curvature, point distribution, etc. There are two main kinds of descriptors, based on how they quantify the geometric properties, local descriptors and global descriptors. The local descriptors describe the geometry at localized areas in the point cloud. They are based on extracting key points from the 3D-object model. Because each local descriptor corresponds to only one individual point, there are as many local descriptors as extracted key points to effectively describe an object. This requires much computational cost for local descriptors calculation. To address this issue, the local descriptors are generally only computed at considered key points for the specific descriptors used. The local descriptors do not require complete view to provide a sufficient description for object so that they can handle occlusion better than the global descriptors.

Unlike the local descriptors, global descriptors do not require key points detection but use all the points present to describe the overall shape of the point cloud. Therefore, it requires only one descriptor needs to be calculated to provide a sufficient description. This results in reduced computational cost compared with local descriptors especially in the matching stage. The description of an object with full view is better than the one that object is partially occluded. Although, the global descriptors do not handle occlusion well, they typically handle noise better than local descriptors. To compute a global descriptor for an object, the object cluster has to be isolated from the whole point cloud by a process known as segmentation of the point cloud. A global descriptor for object recognition and pose estimation are more suitable for real time applications than local descriptors because they are faster and more robust against noise.

4.2.3 Viewpoint Feature Histogram

Viewpoint Feature Histogram (VFH) is a global descriptor from the state-of-the-art to achieve real time 3D object recognition and pose estimation. This descriptor has advantages of high accuracy in the presence of noise and small computation time. The VFH descriptor is based on the surflet-pair relation which encodes the geometric properties of the object's surface by using the surface normals [67].

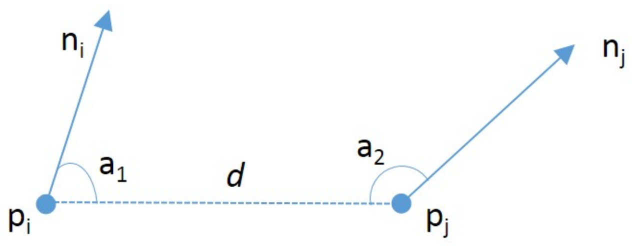


Figure 4.2: Surflet-pair Relation between two points p_i and p_j

Given, p_i and p_j are two arbitrary points of an object's point cloud, ni and nj are their corresponding surface normals. The surflet-pair relation describes the Euclidean distance between pi and pj, the angles between each surface normal and the connecting line of the two points as shown in figure 4.2. There are many local and global descriptors that use the surflet-pair relation such as the Point Feature Histogram (PFH) in research by R.B. Rusu and et al [49] and the Fast Point Feature Histogram (FPFH) [50], and the Ensemble of Shape Functions (ESF) proposed by Walter Wohlkinger and Markus Vincze in [68].

The VFH descriptor was developed in [51] and initially composed of two components, the viewpoint component and the extended Fast Point Feature Histogram (FPFH) component. To make this descriptor more robust, the third

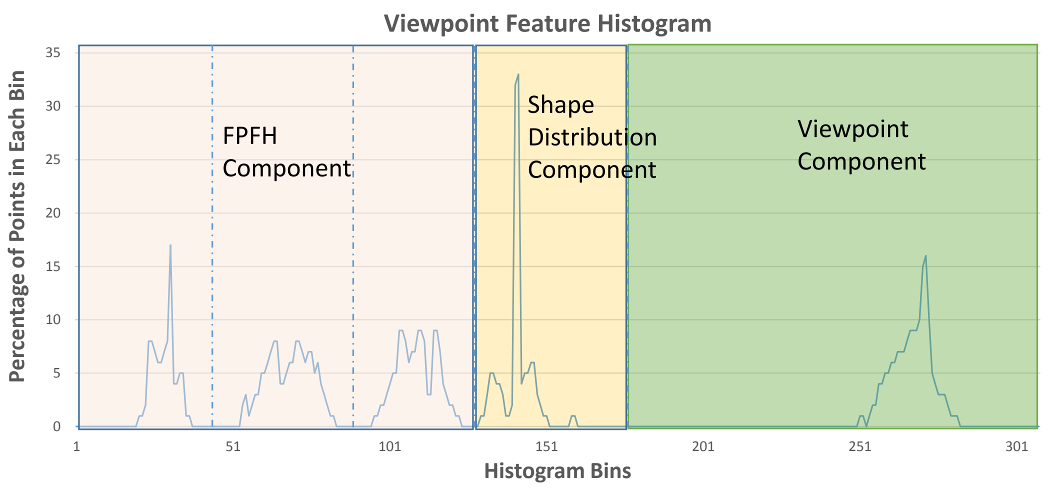


Figure 4.3: An example of Viewpoint Feature Histogram

component, Shape Distribution Component (SDC) was added. The first one, viewpoint component features the direction of surface normals relative to the direction of the camera's central viewpoint. The angle between estimated normal at each point cluster and the viewpoint vector is measured and stored in a 128 bin histogram. The second one, extended FPFH component characterizes the relation of surface normals and the centroid of the point cluster. This component is calculated by measuring pan, tilt and yaw angles of the normal at each point in the cluster and stored in three histograms with 45 bins for each one. The third one, SDC was originally not a part of VFH descriptor but was developed as an extension of the VFH descriptor taken from Cluster View Point Histogram (CVFH) descriptor introduced by Aitor Aldoma and Markus Vincze in [2]. The SDC evaluates the distribution of points around the centroid by measuring their distance from the centroid. Therefore, this component enables the descriptor to distinguish objects with a similar size but different point distribution. These distances are stored in a 45 bin histogram. Combining these three histogram components together will form the full VFH consisting of 308 bins (128 bins for the viewpoint component, 45 bins for each extend FPFH pan, tilt and yaw, and 45 bins for the SDC). Figure 4.3 shows an example of VFH descriptor.

However the VFH descriptor includes only geometrical information and does not take into account color information. Therefore, it cannot distinguish between two objects which have the same shape but different colors. In addition, the accuracy of VFH descriptor is closely related to the estimation of the centroid's position. Therefore, in case the object is partly occluded, the centroid is not correctly estimated, which may affect the recognition result. To overcome this problem, a feature descriptor that combines geometry and color information, the Viewpoint oriented Color-Shape Histogram (VCSH) was introduced in [40].

4.3 Object Recognition and Pose Estimation

This section presents the method using VFH descriptor to recognize an object and estimate its pose from a RGB-D image recorded by Kinect sensor. The method includes two phases: training phase and testing phase. The training phase is to build a database which contain the VFH descriptors of trained objects from many different views. The flow of this training phase is describe as in figure **??**. Then the testing phase is to recognize the object by matching its most similar models from the database and estimate its pose. The flow of this method as shown in figure 4.5

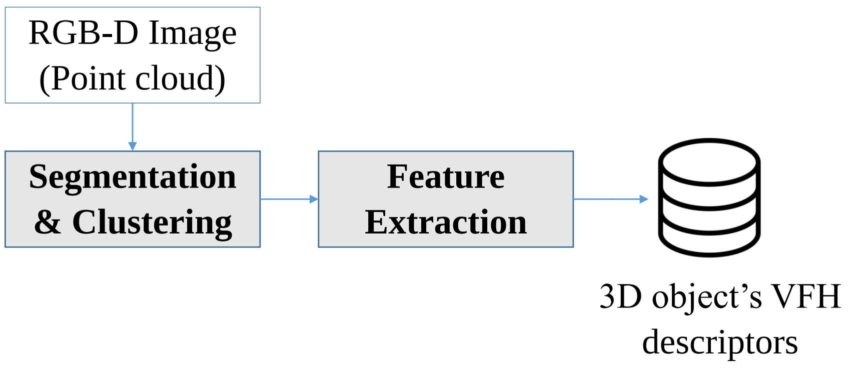


Figure 4.4: The training phase of method to recognize object and estimate its pose

In the training phase, because each viewpoint corresponds to an object pose with respect to the camera, we need to generate as many viewpoints as possible to be able to recognize object with different poses and provide more accurate pose estimation. There are several ways to generate a large number of synthetic views of an object such as manually scan 3D model of object by turning the cameras view around the object, or turning the object to collect views of objects with known orientation, or use a software to virtually scan 3D models of the object.

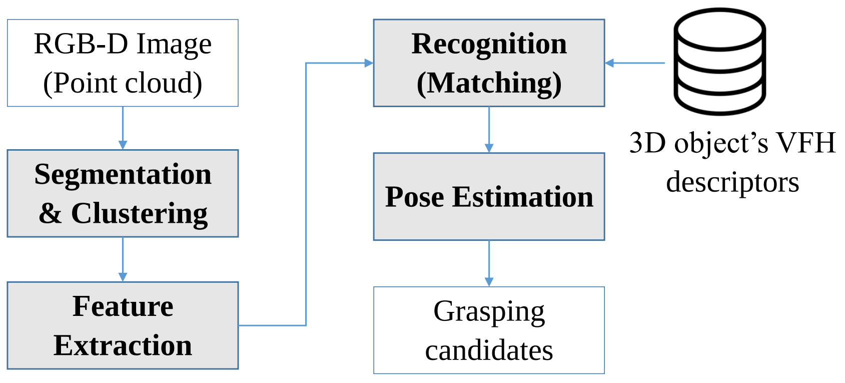

RGB-D Image (Point cloud)
Segmentation & Clustering
Feature Extraction
Recognition (Matching)
Pose Estimation
Grasping candidates
3D object's VFH descriptors

Figure 4.5: The testing phase method to recognize object and estimate its pose

After achieving the synthetic views of an object, a VFH descriptor for each view is estimated and stored with the object pose label into a database.

In the testing phase, the input is real color point cloud scene from an RGB-D image and the target is to identify different objects in the scene and estimate their poses. To this end, a segmentation of the scene need to be performed to isolate object clusters from each other and from the scene. Each point cluster will be a candidate for a detected object. After that, the global descriptor VFH is computed in feature extraction step for each cluster. Then, it is compared to the descriptors in the database of the training data. The object and its pose estimation result from finding the best matching by nearest neighbors in the database with respect to a certain metric. Finally, the pose estimation can be optimized by using Iterative Closest Point (ICP) algorithm. To implement algorithms in this method, the Point Cloud Library (PCL), an open-source library that allows 3D point cloud processing was used.

The raw point cloud image from camera contains a huge number of point data. This leads to a computational load for algorithms such as segmentation, feature descriptor estimation, matching and pose estimation. To satisfy the requirement of real time application like robotic grasping, a pre-processing step of raw point cloud data is useful to reduce the number of data samples. This can be done by downsampling the point cloud using a Voxel filter. The point cloud is broken into cube-shaped regions called voxels. This creates a voxel grid as a set of tiny 3D cubes in space. All the points inside a cube are replaced with a single point that is the average XYZ component of all those points. The voxel resolution is appropriately chosen depending on the accuracy of the raw point cloud.

4.3.1 Segmentation and Clustering

In the training phase, the object models are already clustered, so that each point cloud each point cloud of object corresponds to a single object captured from a known viewpoint. However, in the testing phase, the input is a real scene containing objects that are not separated from the background or from other objects. Therefore, before recognizing objects, we need to perform segmentation of objects from the scene and cluster the segmented point clouds to single object candidate to be recognized in next steps. Firstly, the large planar surfaces that could be tables, walls, floor, etc will remove from the point cloud. This is done by Random Sample Consensus (RANSAC) algorithm to find point clusters which fit to a plane model. Because the objects may also contain a planar surface, only planes with a number of points greater than a specific threshold are removed. Once the planar surfaces is eliminated from the scene, the remaining points are clustered into different clusters. The algorithm Euclidean Cluster Extraction (ECE) is utilized to locate and isolate the point clusters of objects. This algorithm is good for simple scenes with low clutter and when the objects are not occluded. This algorithm is also implemented in Point Cloud Library (PCL).

4.3.2 Feature Extraction

After achieving the point cluster of an object, the feature descriptor VFH is calculated with its components which are aforementioned. Firstly, the surface normal at each point of the object cluster is estimated. The technique to estimate a normal vector of the point cluster is determining an approximate plane which is tangent to the point by using all the points in a specific radius around that point. The normal vector at the given point is computed as normal vector of this plane. Next, each component of VFH is calculated based on the normal at each point. The viewpoint component is computed by collecting a histogram of the angles between the viewpoint direction (camera's viewpoint direction) and each normal. The second component, extended FPFH is computed by collecting the relative angles between the surface normals at each point to the surface normal at the centroid of the object cluster. The computation of these two component together the third component SDC was already implemented in Point Cloud Library. From that, we can obtain the VFH descriptor of an object cluster and use it for matching step to recognize the object.

4.3.3 Matching

Once the VFH descriptor of object candidate has been estimated, we can identify what the object is by matching its VFH descriptor with the others in the database of training dataset. The matching step is to search the most similar histogram in the database which matches to the histogram of object cluster's feature. The best possible candidates are determined by the smallest chi squared distance between the VFH descriptor and the one from the database. The nearest neighbor algorithm is applied to search for candidates in the database which are stored by kd-tree of trained dataset.

Ideally, the closest match is the correct view and will be able to provide the correct pose estimation. However, due to noise and measurement errors, the nearest match will not always produce a correct pose estimation. To overcome this issue, the algorithm can search multiple nearest matches in the database and then verify the result through hypothesis verification.

4.3.4 Pose Estimation

After the matches of object have been found, the pose of an object in the scene can be estimated. Firstly, an initial 6 DoF pose estimation of a recognized object is obtained through a centroid alignment with the Camera Roll Histogram (CRH) [2]. Then this rough pose is refined by using Iterative Closest Point (ICP) algorithm. ICP is used to minimize the difference between two point clouds. The ICP will iteratively align a target point cloud and a source point cloud as closely as possible. In this case, target point cloud is object in the testing data and the source point cloud is the recognized object in the training data taken from a certain viewpoint. After iterating a certain number of times of an error threshold has been satisfied, the pose of recognized object is estimated with the alignment by ICP algorithm.

4.4 Implementation and Experimental Results

We implemented the method of object recognition and pose estimation using functions of the C++ open source Point Cloud Library. Figure 4.6 shows the pipeline of the method implementation. The experimental setup with a Kinect

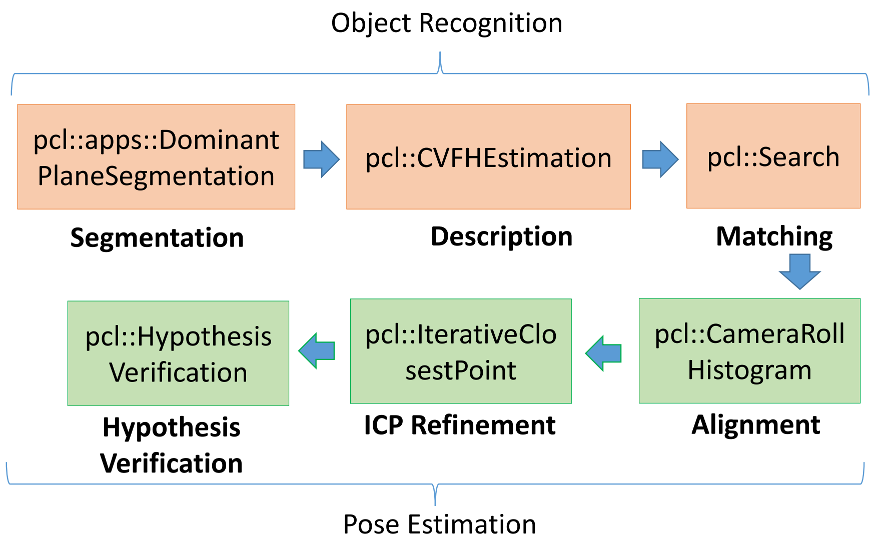


Figure 4.6: The pipeline of method implementation using Point Cloud Library functions

camera and testing objects as shown in figure 4.7. In training stage, we change the views of Kinect around the object to have different viewpoints of a trained object. The object cluster is segmented from the background, computed the VFH descriptors corresponding to each view and stored in the database. In the testing stage, the Kinect will see a real scene with some objects and objects are segmented before recognizing. The time consumption of the method will be measured for algorithms using in each step: segmenting step, matching step and pose estimation step. From that, we will evaluate the accuracy and performance of the method to apply for a robotic vision function in real time in our overall system.

Figure 4.8 shows four objects used for testing in the method. And the figure 4.9 shows the point cloud data of the scene with four objects captured by Kinect camera.

In the segmentation and clustering step, the clusters which are candidates of planes are segmented and removed. The clusters of object candidates are extracted separately. Figure 4.10 shows clusters of four object candidates viewed in one point cloud data (PCD) viewer.

After obtaining object clusters, the VFH descriptor is computed for each cluster from the current camera view. Figures 4.11, 4.12, 4.13, 4.14 show in turn

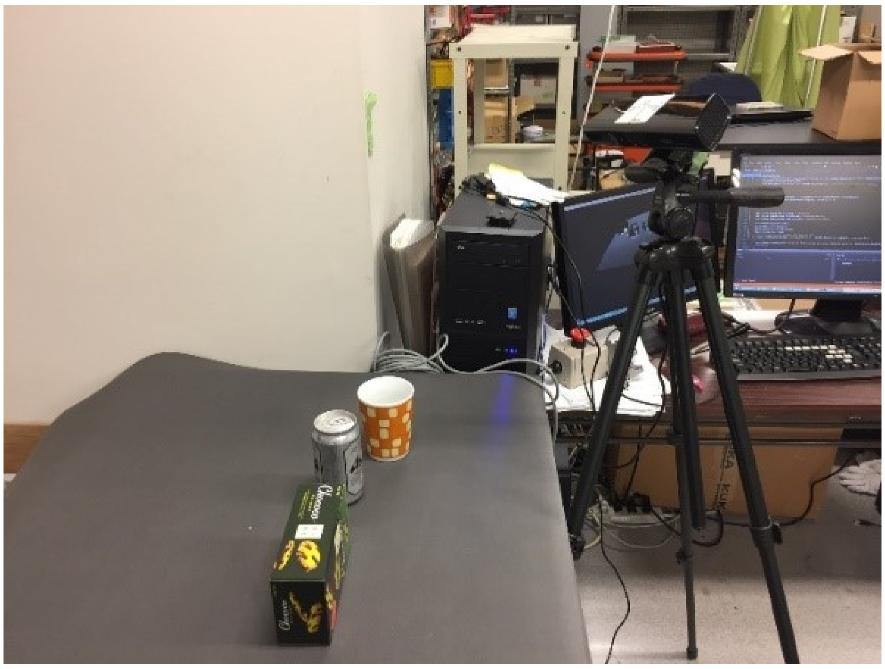


Figure 4.7: Setup of experiment with Microsoft Kinect and testing objects

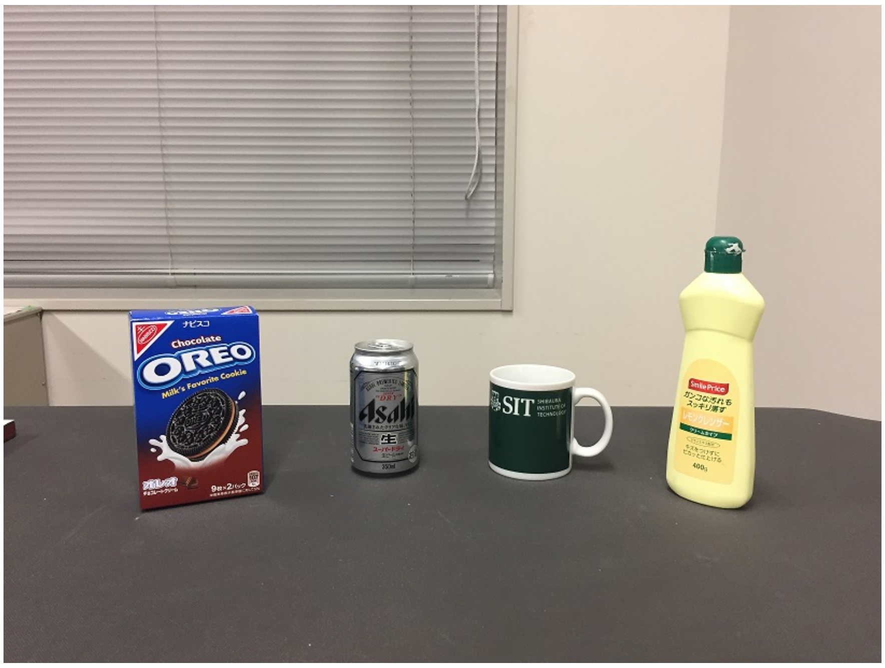


Figure 4.8: Four tested objects: box, can, cup, bottle

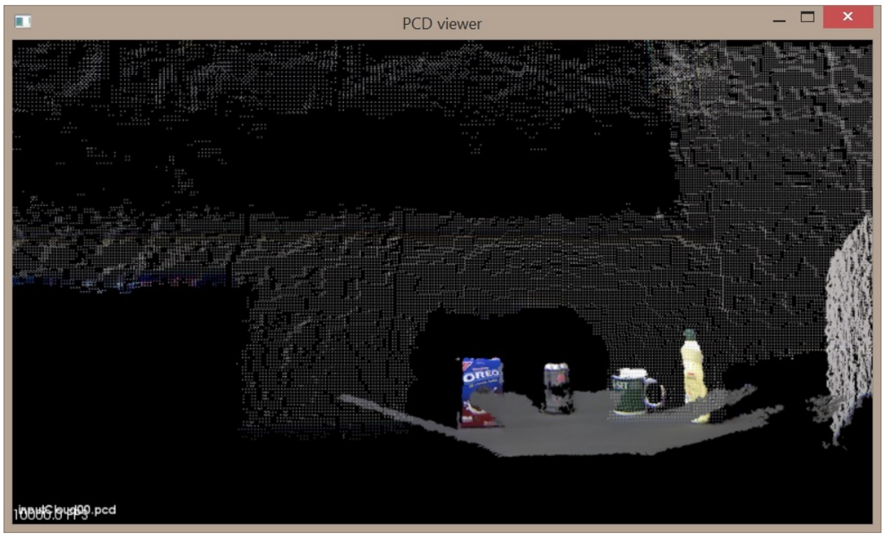


Figure 4.9: Point cloud data of a scene with tested objects captured by Kinect camera

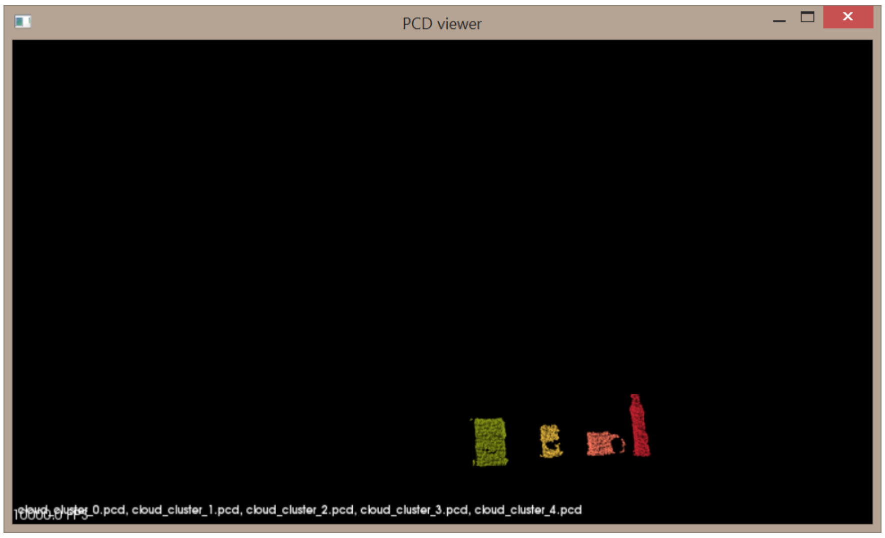

PCD viewer
cloud_cluster_0.pcd, cloud_cluster_1.pcd, cloud_cluster_2.pcd, cloud_cluster_3.pcd, cloud_cluster_4.pcd
10000.0 FPS

Figure 4.10: Result after planar segmentation and cluster extraction. Four object clusters is segmented and show in one viewer

the VFH descriptor of object clusters of object candidates for a cookie box, a cleaning bottle, a cup and a beer can, viewed in a histogram plotter of PCL.

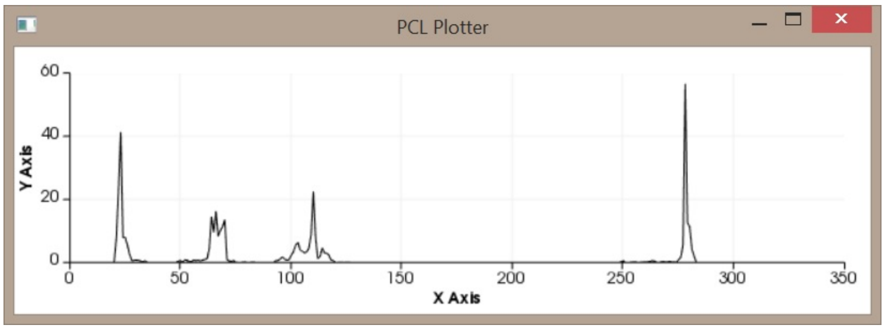


Figure 4.11: The VFH descriptor of the cluster for object candidate - a cookie box

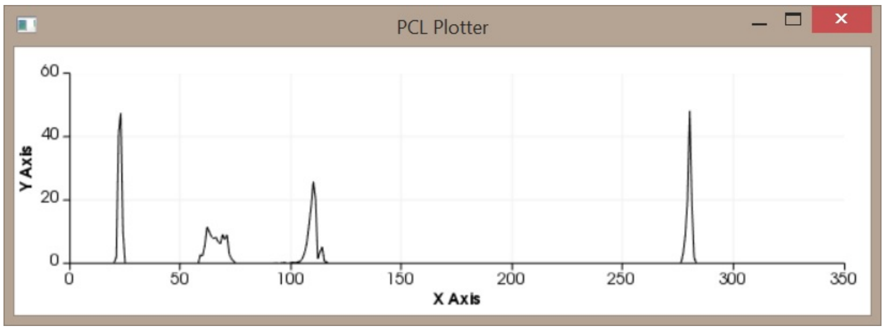


Figure 4.12: The VFH descriptor of the cluster for object candidate - a cleaning bottle

In matching step, the VFH descriptor of each object candidate is matched with the others in the database of training dataset. The most similar histogram in the database which matches to the histogram of object cluster's descriptor is identified by the nearest neighbor searching algorithm. Figure 4.15 shows three best possible candidates of the tested object, a cup, selected in turn by smallest chi squared distance between the VFH descriptor and the one in the database.

Finally, the pose of identified object is estimated as described in previous section. We measured the computation time of the steps by using functions of Point Cloud Library. The computation time is shown in table **??**. This measurement shows that the computation time of the method is suitable to apply for perceiving object in real time for object manipulation actions.

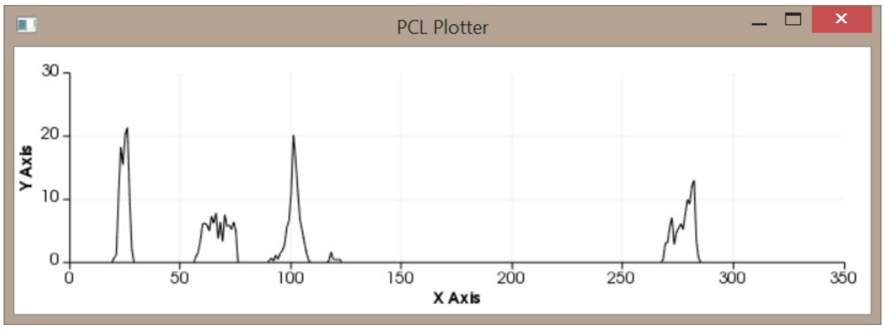


Figure 4.13: The VFH descriptor of the cluster for object candidate - a cup

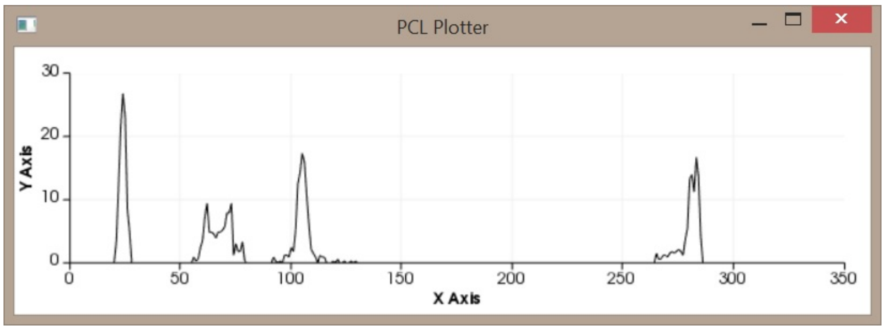


Figure 4.14: The VFH descriptor of the cluster for object candidate - a beer can

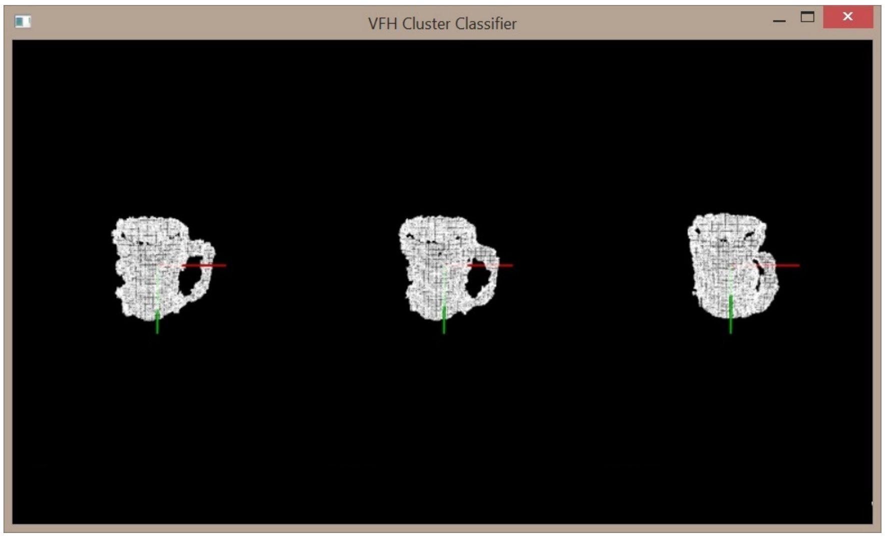


Figure 4.15: Three best matching candidates of the tested object - a cup, from the trained database

Table 4.1: The computation time of step in the method of object recognition and pose estimation

Step	**Time (ms)**
Segmentation	1102
VFH Computation	232
Matching	5
Pose Estimation	981

4.4.1 Applying for the Action 'Pick up a Cup'

The result of recognizing a cup and its pose will be applied for the action 'pick up a cup' as in the experiment in chapter 3. From the result of cup's 6-DOF pose, the cup's 3D location (x, y, z) and orientation (yaw, pitch, roll) are computed. When the robot recognizing the location and orientation of the cup was changed different from original demonstration, the robot will use DMPs model to generate the movement adapted with these changes for the movement 'approaching the cup'. The grasping pose of robot hand is chosen based on the orientation of the cup. The new goal position of movement is new location of the cup. Figure 4.16 shows a photo of the situation when the robot arm Schunk LWA3 with two-finger hand approaching to the cup with a new location.

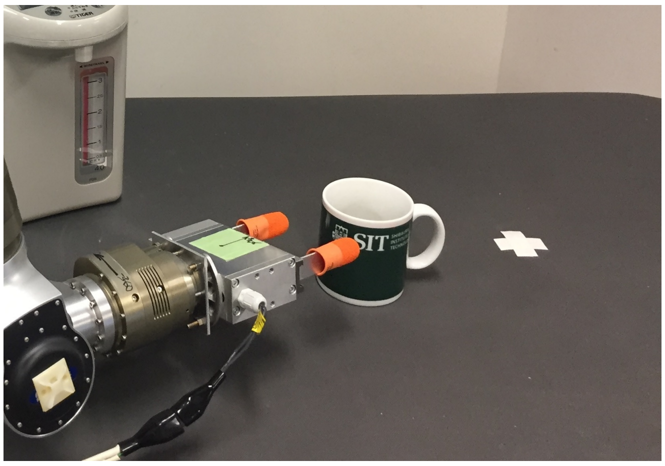


Figure 4.16: The robot arm and hand perform the movement approaching the cup in the situation after recognizing the location and pose of the cup

Chapter 5

Discussion

The research in this dissertation has solved several important proposals for executing everyday manipulation tasks by a robot. The new findings and contributions of the research is the method to the robot automatically acquire the actions and objects the task from instruction manual and the method using Dynamic Movement Primitives model to regenerate the movements learning from human demonstration for robot actions. However, there are also some limitations. This chapter discusses contribution as well as drawbacks of the proposed approaches.

5.1 Acquiring Actions from Instruction Manual

In the first proposal of this dissertation, a task plan including a sequence of actions was extracted from instruction manuals of home appliances. One advantage point of this proposal is that the task plan can be obtained from the sequence of action without understanding the meaning of task. In addition, the order of action in the task has already known from the instruction sentences. This is easier compared with some other methods which require understanding the meaning of instruction by robot or determine the order of action in the task.

However, the instructions collected from instruction manuals may lack of the action required in the task. For example, the instruction of the task 'dispensing water' may contain the actions: place the cup under the spout, press the 'Push' button and without the action pick up a cup to tell the robot what to do before having the cup in the robot's hand. Therefore in some cases, this kind of knowledge from the instruction manual is not enough for fully achieving a task plan.

One solution for reasoning the action which is absent in the task instruction is necessary for this problem.

In addition, in the experimental evaluation part of this proposal, we proved the effectiveness of the extraction method of action and object but simple by manually compare the results with the input data in candidate examples of input data. The accuracy of this method depends on the grammatical complexity of input instruction sentences which should be evaluated in statistic way with more input data.

5.2 Learning Action from Hand Movement in Human Demonstration

Using inverse kinematic is a simply way to generate a movement from start position to goal position in 3D space for the end-effector of a robot arm. In the action 'pick up a cup' (or pick up an object in general), the movement trajectory of sub-movement 'approaching the cup' can be simply generated by an inverse kinematic solution and it may have several different trajectories for this sub-movement. However, generating movement using Dynamic Movement Primitives (DMP) has some advantage points. Learning movement using DMP model can adapt flexibly the generated movement to other contexts, such as different goal position like approaching the cup when the cup's location was changed, scaling the duration of movement trajectory, or change of movement trajectory to differing coordinate system. Using DMP model also allows to generate smooth movement trajectory, generate not only position trajectory but also velocity or acceleration of movement which can be scaled in desired duration.

Another ability of learning movement using DMP model is avoiding obstacles even the obstacle appears suddenly in the movement trajectory. In this dissertation, the problem of generating movement with obstacle avoidance does not take into account. However, we can improve differential equations of DMP model to have this ability as the work in [45].

In the action 'open a door' (like a microwave oven's door), the generated movement trajectory demands to follow the movement trajectory which is record from human demonstration. This is because of the trajectory of door opening.

In this situation, the movement trajectory is generated (or reproduced) without changes. It means that the reproduction movement is almost same as demonstrated movement. And this can be easily done with DMP model but it does not with the inverse kinematic method. Inverse kinematic method may generate a movement trajectory which is not for door opening trajectory. Beside that, using DMP model still helps to generate movement trajectory of open door with new goal like a different opening angle of door.

One other discussing point is that the correspondence problem between human arm and robot arm as well as the limitation of working space in comparison between them. This problem has not been included in this dissertation. In the execution of the task 'dispensing water', this problem does not happen. But in the action open a microwave door, the limitation of working range of the robot arm when it is mounted on ground/table surface is much different from the one of human arm. This also lead to the joints limitation of the robot arm to perform the movement trajectory and it is difficult select a suitable coordinate system transformation to discriminate this problem. A robot arm mounted on a vertical base similar to human arm may help to avoid this difficulty.

In other hand, in this research, we used DMP model for movement trajectory in 3D space which then is solved into joints space to control joints of robot arm. However, DMP model can be applied for the movement data in joint space if we record the demonstration by kinesthetic teaching method (human directly guide the robot arm the movement and record its joint movement). The problem of joint limitation will be removed form this demonstration recording step.

The problem of segmentation of movement has not been completely solved yet in this dissertation. Our proposed method for segmenting movement depends on some conditions. The segmentation technique assumes that movement of the actions including three phases: approaching (to reach the object), manipulating (such as grasping, pressing, releasing, so on) and withdrawing. This may not apply for all actions. The segmentation technique also based on the parameters hand movement velocity and finger distance with appropriate thresholds however these thresholds are simply selected by trial and error method. In addition, the segmentation of movement also includes two levels. In low level, the movement of the action is segmented into motion primitives. In high level, the whole movement of a task is segmented into sub-actions. These concepts are understood as defined in chapter 1. One ideas for the problem of segmentation of movement in the task is determining task-critical events as work in [28].

5.3 Recognizing Object for Manipulation Actions

In adaptive learning of movement, we can input a new goal of movement and generated movement will be adapted to this new goal by DMP model. In the experiment in chapter 3, we assume that a new goal is already recognized by robot. One example, robot recognize the cup's location was changed to new location and the approaching movement is to new location of the cup. Thus, the work in chapter 4 is to provide the ability of recognizing an object such as a cup for robot. This is similar if robot hand needs to change to a new orientation of object. In that case, the object pose also should be recognized by robot's vision system. Therefore, a method for recognizing object and estimating its pose based on object 3D information is implemented in chapter 4. However, one of the challenge problems of this ability is building a database of common objects in daily life at home. In this dissertation, we tested with a few object in the experiment to verify the method and evaluate the computation time of the method to satisfy in real time for robot action. This function needs to be added to robot system for a fully testing of the proposals.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This research proposed the method for learning and executing everyday tasks at home from instruction manual and human demonstration. The final purpose is that the robot can perform a task by automatically acquiring the sequence of actions in the task from instruction manual and learning how to perform the action from human demonstration.

The first issue, acquiring actions from instruction manual, is solved by the proposal of extracting action and object from instructions which contain what action the robot needs to perform. The solution is using a syntax parser in natural language processing and the searching algorithm to determine the pairs of action and object from each instruction sentence.

The second issue, learning action from human demonstration, the robot system observed hand movement in human demonstration of the action, then by a learning method using a generative model - Dynamic Movement Primitives model - to generate movements for that action with the adaptation to new change such as the scale of duration of movement or the change of goal position of movement.

We conducted the experiment with the task 'dispensing water' from a water thermos pot. The sequence of actions in this task can be achieved from the instruction manual. The robot observed the demonstration of each action and then reproduced the movements for the action with using Dynamic Movement Primitives framework.

In addition, in order to provide the robot the perception about object for related manipulation actions in everyday tasks, for example, recognizing pose and location of object when performing the action 'pick up the object'. we implemented a method for recognizing 3D object and estimating its pose. The method uses point cloud data from 3D devices such as a Kinect camera and applying the cutting-edge descriptor - Viewpoint Feature Histogram descriptor.

6.2 Future Work

The actions in a task can be determined well from task instruction. Then these actions need to be mapped to the execution plan for each one. In this study, we mapped manually the action to its executable plan. However, a method for automatically mapping each action to the executable plan will provide the ability to full-autonomously accomplish the task by the robot. The method for this issue can be done by understanding the semantic meaning of the actions. For future work, a possible solution for this problem is to use conceptual knowledge (e.g. ConceptNet) to provide the robots for understanding the action meaning based on the relationship between action - action and action - object and/or the relationship of actions with the condition in the instructions.

The idea of learning action from human demonstration for executing everyday tasks can be enhanced by observing human activities in daily life and recognizing the actions in the tasks. These have been being studied in some researches but they are still challenges nowadays.

In order to generalize the method of learning action from demonstration for numerous everyday manipulation tasks, a library of basic actions or primitive skills for the robot can be built independently to the task. The executable plans of the actions in this library is encoded and labeled with respect to the actions. A complex task can be accomplished by reassembling the sequence of actions which are performed by selecting skills available in the library and combining with some new skills if it is necessary.

Appendix A

Inverse Kinematic for Robot Arm LWA3

We deployed a inverse kinematic solver using analytic method. This solver is based on the kinematic structure of Schunk LWA3 A.2 and an approximation for inverse kinematic analytic solution is selected as in A.1.

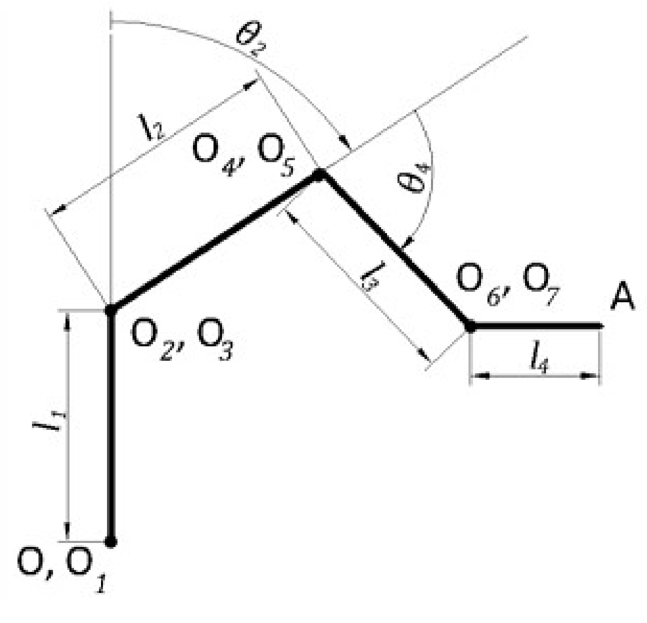


Figure A.1: An approximation of inverse kinematic analytic solution for Schunk LWA3

As the operation space may at most have 6 dimensions (3D position and 3D orientation), LWA3 robot is redundant and some measures need to be assumed. Usually redundancy is solved by specification of a secondary task for the robot. Such task can have a goal of avoiding singular configurations and joint limits, optimizing joint torques or providing decoupled force/position control of the robot. In this implementation, the secondary task for inverse kinematic solution is define by keeping the angle value of the joint 3 at zero and assigning only positive angle values to the joint 4. With these assumptions, the robot now can be modeled as 4 links, 3 of which are connected with revolution joints, and the last one (the end-effector) is attached with a spherical joint A.1. The input for an inverse kinematics task is the position of the end-effector (point $A(x_A, y_A, z_A)$) and its orientation as a unit vector of Z-axis of end-effector's coordinate frame $\overrightarrow{Z_7}$. From these we can easily find the desired coordinates of the point O_6:

$$\overrightarrow{OO_6} = \overrightarrow{OA} - \overrightarrow{Z_7}.l_4 \tag{A.1}$$

Next, the point O_6 is checked whether it is reachable. If $\left|\overrightarrow{O_2O_6}\right| > (l_2 + l_3)$, then the point is out of range and no solution exist. In the opposite case, by applying cosine theorem to the $O_2O_4O_6$ triangle, we can obtain rotation angles of the joint 1, joint 2 and joint 4:

$$\theta_1 = atan2(y_{O_6}, x_{O_6}) \tag{A.2}$$

$$\theta_2 = \frac{\pi}{2} - arccos(\frac{\left|\overrightarrow{O_2O_6}\right|^2 + {l_2}^2 - {l_3}^2}{2.l_2.\left|\overrightarrow{O_2O_6}\right|}) - arcsin(\frac{x_{O_6} - l_1}{\left|\overrightarrow{O_2O_6}\right|}) \tag{A.3}$$

$$\theta_4 = \pi - arccos(\frac{{l_2}^2 + {l_3}^2 - \left|\overrightarrow{O_2O_6}\right|^2}{2.l_2.l_3}) \tag{A.4}$$

The $\theta_3 = 0$ as because of the assumption, only the angles of joint 5 and joint 6 need to be computed to provide the desired orientation of the end effector. This task can be mapped to finding Euler angles $\alpha, \beta and \gamma$ that would rotate the coordinate frame $O_5X_5Y_5Z_5$ to the frame $O_7X_7Y_7Z_7$ where X_7 and Y_7 are arbitrary directed. The solution of this task is well known and we obtain:

$$\theta_5 = atan2(y_{Z_7^5}, x_{Z_7^5}) \tag{A.5}$$

$$\theta_6 = acos(z_{Z_7^5}) \tag{A.6}$$

where Z_7^5 is the vector $\overrightarrow{Z_7}$ observed from the coordinate frame $O_5X_5Y_5Z_5$.

After the desired angle values have been found, they should be validated against possible joint limits. If the test has been successfully passed, the calculated joint values are returned as output of inverse kinematics solver.

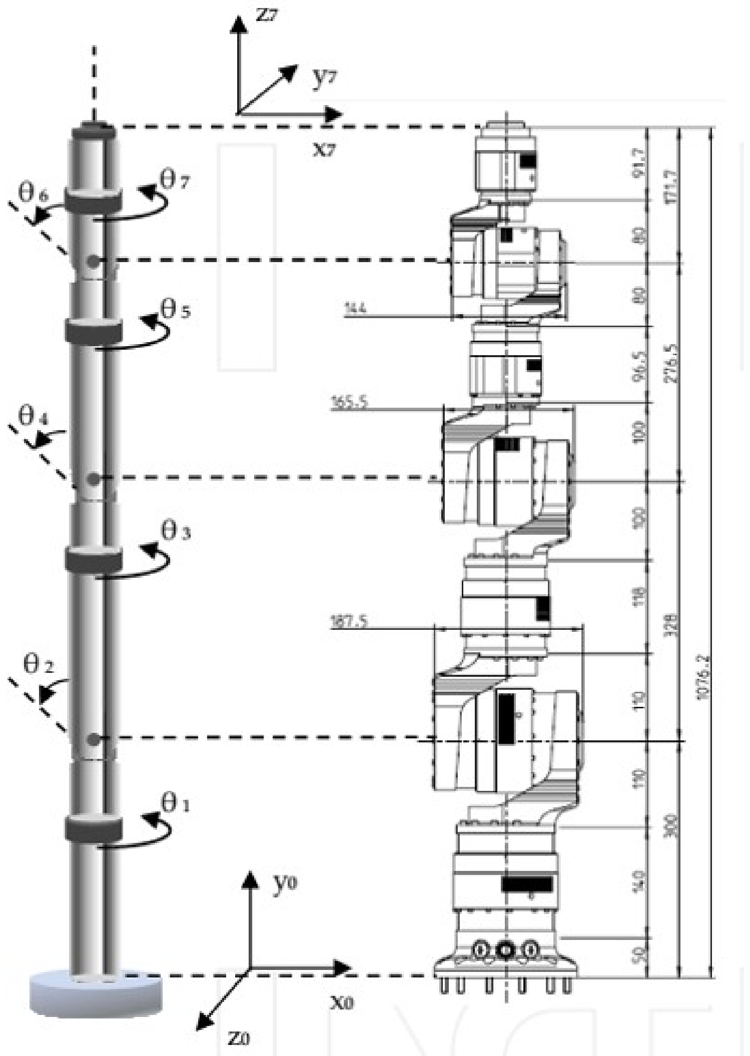


Figure A.2: The kinematic structure of the 7-DOF robot arm Schunk LWA3

Bibliography

[1] Aude Billard adn Daniel Grollman. Robot learning by demonstration. *Scholarpedia 8(12):3824*, 2013.

[2] Aitor Aldoma, Markus Vincze, Nico Blodow, David Gossow, Suat Gedikli, Radu Bogdan Rusu, and Gary R. Bradski. Cad-model recognition and 6dof pose estimation using 3d cues. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 585–592, 2011.

[3] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, **57**(5):469–483, 2009.

[4] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *I. J. Humanoid Robotics*, **5**(2):183–202, 2008.

[5] Aude Billard and Sylvain Calinon. *Handbook of Robotics Chapter 59: Robot Programming by Demonstration, Robotics*, chapter 59, page 48(21):13711394. , 2007.

[6] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics - The 13th International Symposium on Experimental Robotics, ISER 2012, June 18-21, 2012, Québec City, Canada*, pages 481–495, 2012.

[7] Sylvain Calinon, Florent D'halluin, Eric Sauser, Darwin Caldwell, and Aude Billard. Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression. *IEEE Robotics and Automation Magazine*, **17**(2):44–54, 2010.

[8] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning the statistical representation of a task and generalizing it to various contexts. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, May 15-19, 2006, Orlando, Florida, USA*, pages 2978–2983, 2006.

[9] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, **37**(2):286–298, 2007.

[10] Rehj Cantrell, Kartik Talamadupula, Paul W. Schermerhorn, J. Benton, Subbarao Kambhampati, and Matthias Scheutz. Tell me when and why to do it!: run-time planner model updates via natural language instruction. In *International Conference on Human-Robot Interaction, HRI'12, Boston, MA, USA - March 05 - 08, 2012*, pages 471–478, 2012.

[11] Xiaoping Chen, Jiongkun Xie, Jianmin Ji, and Zhiqiang Sui. Toward open knowledge enabling for human- robot interaction. *Journal of Human-Robot Interaction (JHRI)*, **1**(2):100–117, 2012.

[12] Richard Cubek and Wolfgang Ertel. Learning and application of high-level concepts with conceptual spaces and pddl. In *3rd Workshop on Planning and Learning*, number 2011, page 7683.

[13] Richard Cubek and Wolfgang Ertel. Conceptual similarity as a key to high-level robot programming by demonstration. In *ROBOTIK 2012 - Proceedings for the conference of ROBOTIK 2012, 7th German Conference on Robotics, 21-22 May 2012, International Congress Center Munich (ICM) in conjunction with AUTOMATICA.*, 2012.

[14] C. Daniel, G. Neumann, and J. Peters. Learning concurrent motor skills in versatile solution spaces. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3591–3597, Oct 2012.

[15] Bla Fortuna Marko Grobelnik Dunja Mladeni [Delia Rusu, Lorand Dali. Triplet extraction from sentences. In *Conference on Data Mining and Data Warehouses (SiKDD)*, Slovenia, 2007.

[16] RÜDIGER DILLMANN. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, **47**(2-3):109–116, 2004.

[17] DENIS FORTE, ANDREJ GAMS, JUN MORIMOTO, AND ALES UDE. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, **60**(10):1327–1339, 2012.

[18] DANIEL H. GROLLMAN AND ODEST CHADWICKE JENKINS. Incremental learning of subtasks from unsegmented demonstration. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pages 261–266, 2010.

[19] MICHA HERSCH, FLORENT GUENTER, SYLVAIN CALINON, AND AUDE BILLARD. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Trans. Robotics*, **24**(6):1463–1467, 2008.

[20] H. HOFFMANN, P. PASTOR, AND S. SCHAAL. Dynamic movement primitives for movement generation motivated by convergent force fields in frog. In *Adaptive Motion of Animals and Machines (AMAM)*, 2008.

[21] HEIKO HOFFMANN, PETER PASTOR, DAE-HYUNG PARK, AND STEFAN SCHAAL. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 2587–2592, 2009.

[22] NIKOLAOS KYRIAZIS IASON OIKONOMIDIS AND ANTONIS ARGYROS. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101.

[23] NIKOLAOS KYRIAZIS IASON OIKONOMIDIS AND ANTONIS ARGYROS. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101.

[24] AUKE JAN IJSPEERT, JUN NAKANISHI, HEIKO HOFFMANN, PETER PASTOR, AND STEFAN SCHAAL. Dynamical movement primitives: Learning

attractor models for motor behaviors. *Neural Computation*, **25**:328–373, 2013.

[25] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 1523–1530, 2002.

[26] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*, pages 958–963, 2002.

[27] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*, pages 1398–1403, 2002.

[28] Glatz Karl. *Adaptive Learning from Demonstration using Dynamic Movement Primitives*. Master's thesis, University of Applied Sciences, 2012.

[29] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430, 2003.

[30] Dana Kulic, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *I. J. Robotics Res.*, **31**(3):330–345, 2012.

[31] Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting people's common sense into knowledge bases of household robots. In *KI 2010: Advances in Artificial Intelligence, 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24, 2010. Proceedings*, pages 151–159, 2010.

[32] Tom Kwiatkowski, Luke S. Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1512–1523, 2011.

[33] Dongheui Lee and Christian Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Auton. Robots*, **31**(2-3):115–131, 2011.

[34] Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *In 5th International Conference on Language Resources and Evaluation*, 2006.

[35] Ren Mao, Yezhou Yang, Cornelia Fermüller, Yiannis Aloimonos, and John S. Baras. Learning hand movements from markerless demonstrations for humanoid tasks. In *14th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2014, Madrid, Spain, November 18-20, 2014*, pages 938–943, 2014.

[36] Çetin Meriçli, Steven D. Klee, Jack Paparian, and Manuela M. Veloso. An interactive approach for situated task specification through verbal instructions. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, pages 1069–1076, 2014.

[37] Dipendra Kumar Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*, 2014.

[38] Dipendra Kumar Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *I. J. Robotics Res.*, **35**(1-3):281–300, 2016.

[39] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *I. J. Robotics Res.*, **32**(3):263–279, 2013.

[40] Chiraz Nafouki. Object recognition and pose estimation from an rgb-d image. Technical report, Technical University Munich, Germany, 2015.

[41] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, **47**(2-3):79–91, 2004.

[42] Bojan Nemec and Ales Ude. Action sequencing using dynamic movement primitives. *Robotica*, **30**(5):837–846, 2012.

[43] Monica Nicolette Nicolescu. *A Framework for Learning from Demonstration, Generalization and Practice in Human-robot Domains.* PhD thesis, Los Angeles, CA, USA, 2003. AAI3103951.

[44] Daniel Nyga and Michael Beetz. Everything robots always wanted to know about housework (but were afraid to ask). In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 243–250, 2012.

[45] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, May 2009.

[46] P. Pastor, H. Hoffmann, and S. Schaal. Movement generation by learning from demonstration and generalization to new targets. In *Adaptive Motion of Animals and Machines (AMAM)*, 2008.

[47] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, **21**(4):682–697, 2008.

[48] Miguel Prada and Anthony Remazeilles. Dynamic movement primitives for human robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, 2012*, 2012.

[49] Radu Bogdan Rusu. *Semantic 3D object maps for everyday manipulation in human living environments.* PhD thesis, Technical University Munich, Germany, 2009.

[50] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe, Japan, May 12-17, 2009*, pages 3212–3217, 2009.

[51] Radu Bogdan Rusu, Gary R. Bradski, Romain Thibaux, and John M. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pages 2155–2162, 2010.

[52] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (PCL). In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 2011.

[53] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *IROS 2003*, pages 1–21. Max-Planck-Gesellschaft, October 2003.

[54] Stefan Schaal, Jan Peters, Jun Nakanishi, and Auke Jan Ijspeert. Learning movement primitives. In *Robotics Research, The Eleventh International Symposium, ISRR, October 19-22, 2003, Siena, Italy*, pages 561–572, 2003.

[55] Markus Schneider. *Learning from Demonstration with Gaussian Processes*. Master's thesis, University of Applied Sciences Ravensburg-Weingarten, 2009.

[56] Markus Schneider and Wolfgang Ertel. Robot learning by demonstration with local gaussian process regression. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*, pages 255–260, 2010.

[57] Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Yue Chai, and Ning Xi. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pages 89–97, 2014.

[58] A. Skoglund, B. Iliev, B. Kadmiry, and R. Palm. Programming by demonstration of pick-and-place tasks for industrial manipulators using task primitives. In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, pages 368–373, June 2007.

[59] F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *11th IEEE-RAS International Conference on Humanoid Robots*, pages 231–238, 2011.

[60] Freek Stulp, Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning to grasp under uncertainty. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 5703–5708, 2011.

[61] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, 2011.

[62] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *IEEE Robotics Automation Magazine*, **18**(2):58–68, June 2011.

[63] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pages 1486–1491, 2010.

[64] Moritz Tenorth, Alexander Clifford Perzylo, Reinhard Lafrenz, and Michael Beetz. Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. *IEEE Trans. Automation Science and Engineering*, **10**(3):643–651, 2013.

[65] Jesse Thomason, Shiqi Zhang, Raymond J. Mooney, and Peter Stone. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1923–1929, 2015.

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

[67] Eric Wahl, Ulrich Hillenbrand, and Gerd Hirzinger. Surflet-pair-relation histograms: A statistical 3d-shape representation for rapid classification. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM 2003), 6-10 October 2003, Banff, Canada*, pages 474–482, 2003.

[68] Walter Wohlkinger and Markus Vincze. Ensemble of shape functions for 3d object classification. In *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011, Karon Beach, Thailand, December 7-11, 2011*, pages 2987–2992, 2011.

[69] Yan Wu and Yiannis Demiris. Towards one shot learning by imitation for humanoid robots. In *IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3-7 May 2010*, pages 2889–2894, 2010.

[70] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *CoRR*, **abs/1207.1420**, 2012.

Publication List

Journal Articles

[P.1] Pham Ngoc Hung, Takashi Yoshimi, ”Extracting actions from instruction manual and testing their execution in a robotic simulation”, ASEAN Engineering Journal Part A, Vol.6, No.1, page 47-58, June, 2016.

[P.2] Pham Ngoc Hung, Takashi Yoshimi, ”Adaptive Learning of Hand Movement in Human Demonstration for Robot Action”, Journal of Robotics and Mechatronics Vol.29 No.5, 2017.

Conference Papers

[P.1] Pham Ngoc Hung, Takashi Yoshimi, Makoto Mizukawa, ”The proposal of the extraction method of primitive actions from instruction manuals for robot planning”, The Robotics and Mechatronics Conference, Kyoto, Japan, May 17-19, 2015.

[P.2] Pham Ngoc Hung, Takashi Yoshimi, ”Evaluation of the actions extracted from instruction manual and development of simulation environment for implementing robot actions”, AUN/SEED-Net Regional Conference for Computer and Information Engineering, Hanoi, Vietnam, October 1-2, 2015.

[P.3] Pham Ngoc Hung, Takashi Yoshimi, ”Extraction of actions and objects from instructions manual for executable robot planning”, Proceedings of International Conference on Control, Automation and Systems (ICCAS), pp. 881-885, Busan, Korea, October 13-16, 2015.

[P.4] Ngoc Hung Pham, Takashi Yoshimi, ”A proposal of extracting of motion primitives by analyzing tracked data of hand motion from human demonstration”, the 47th International Symposium on Robotics (ISR), Munich, Germany, June 21-22, 2016.

[P.5] P. N. Hung, Takashi Yoshimi, ”Extracting Motion Primitives by Tracking Hand Motions in Human Demonstrations”, IEEE International Symposium on Robot and Human Interactive Communication, New York, USA, August 25-31, 2016.

[P.6] P. N. Hung and Takashi Yoshimi, ”An approach to learn hand movements for robot actions from human demonstration”, In Proceedings of IEEE/SICE International Symposium on System Integration (SII), Sapporo, Japan, December 13-15, 2016.

[P.7] Pham Ngoc Hung, Takashi Yoshimi, ”RGB-D Image based object recognition and pose estimation for robot grasping”, The 11th South East Asean Technical University Consortium Symposium, Ho Chi Minh City, Vietnam, March 2017.

[P.8] Pham Ngoc Hung, Takashi Yoshimi, ”Programming Everyday Task by Demonstration using Primitive Skills for a Manipulator”, The 7th Annual IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (IEEE-CYBER 2017), Hawaii, USA, July 31 - August 4, 2017.

[P.9] Pham Ngoc Hung, Takashi Yoshimi, ”Programming Everyday Task using Primitive Skills and Generative Model of Movement Demonstrated by Human”, 2017 International Conference On Advanced Robotics and Intelligent Systems (ARIS 2017), Teipei, Taiwan, September 6-8, 2017.