ROBOT SKILL LEARNING THROUGH HUMAN DEMONSTRATION AND INTERACTION

By

YE GU

Bachelor of Engineering in Electrical Engineering
Harbin Institute of Technology
Harbin, Heilongjiang
2007

Master of Science in Engineering Management
University of Minnesota Duluth
Duluth, Minnesota
2010

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2015

ROBOT SKILL LEARNING THROUGH HUMAN

DEONSTRATION AND INTERACTION

Dissertation Approved:

ACKNOWLEDGEMENTS

I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my family.

I would like to express my deepest gratitude to my advisor, Dr. Weihua Sheng, for guiding and supporting me over the years. You have set an example of excellence as a research, mentor and role model. Thanks for taking me on as a new graduate student who knew very little about Robotics. Thanks for your tremendous effort to my dissertation and papers.

I'd like to thank my committee members for all of their guidance through this process; your advices, discussion and feedback have been absolutely invaluable. Special thanks go to Dr. Crick for giving me precious professional advices and access to his lab equipment. It was a pleasure to work with you.

I would also like to thank my parents, my grandmother and all my family members. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to dedicate this thesis to my grandfather. You have been always motivating me especially when I was struggling. Although it has been years since you have passed, I still take your words with me, every day.

Name: Ye Gu

Date of Degree: MAY, 2015

Title of Study: ROBOT SKILL LEARNING THROUGH HUMAN DEMONSTRATION AND INTERACITON

Major Field: ELECTRICAL ENGINEERING

Abstract: Nowadays robots are increasingly involved in more complex and less structured tasks. Therefore, it is highly desirable to develop new approaches to fast robot skill acquisition. This research is aimed to develop an overall framework for robot skill learning through human demonstration and interaction. Through low-level demonstration and interaction with humans, the robot can learn basic skills. These basic skills are treated as primitive actions. In high-level learning, the complex skills demonstrated by the human can be automatically translated into skill scripts which are executed by the robot. This dissertation summarizes my major research activities in robot skill learning. First, a framework for Programming by Demonstration (PbD) with reinforcement learning for human-robot collaborative manipulation tasks is described. With this framework, the robot can learn low level skills such as collaborating with a human to lift a table successfully and efficiently. Second, to develop a high-level skill acquisition system, we explore the use of a 3D sensor to recognize human actions. A Kinect based action recognition system is implemented which considers both object/action dependencies and the sequential constraints. Third, we extend the action recognition framework by fusing information from multimodal sensors which can recognize fine assembly actions. Fourth, a Portable Assembly Demonstration (PAD) system is built which can automatically generate skill scripts from human demonstration. The skill script includes the object type, the tool, the action used, and the assembly state. Finally, the generated skill scripts are implemented by a dual-arm robot. The proposed framework was experimentally evaluated.

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Overview of Robot Skill Learning Approaches . . . 3

1.3.1 Programming by Demonstration . . . 4

1.3.2 Reinforcement Learning . . . 5

1.4 Related Works . . . 6

1.4.1 Learning from Demonstration . . . 6

1.4.2 Action Recognition . . . 9

1.4.3 Ways of Skill Demonstration . . . 12

1.5 Organization of the Dissertation . . . 13

2 Human Robot Table Lifting through PbD and Reinforcement Learning **15**

2.1 Motivation . . . 15

2.2 Problem Statement . . . 16

2.3 Experimental Setup . . . 16

2.3.1 Motion Capture System . . . 17

2.3.2 Calibration . . . 18

2.3.3 Imitation Learning Phase . . . 19

2.3.4 Reinforcement Learning . . . 22

2.4 Experimental Results . . . 24

2.4.1 Imitation Learning Results 24
2.4.2 Reinforcement Learning Results 25
2.5 Summary 28

3 RGB-D Sensor based Manipulative Action Recognition 30

3.1 Motivation 30
3.2 Problem Statement 31
3.2.1 System Setup 31
3.2.2 Formulation of the Problem 32
3.3 Methodology 33
3.3.1 Object Recognition 33
3.3.2 HMMs for Low-level Action Recognition 36
3.3.3 Bayesian Network for Modeling Object/action Dependency 37
3.3.4 HMM Modeling of Sequential Constraints 39
3.4 Experiments and Results 40
3.4.1 Experiment Procedure 40
3.4.2 Results 42
3.5 Summary 48

4 Fine Manipulative Action Recognition through Sensor Fusion 50

4.1 Motivation 50
4.2 Problem Statement 51
4.2.1 System Setup 52
4.2.2 Formulation of the Problem 54
4.3 Methodology 54
4.3.1 Object Recognition 55
4.3.2 Fusion Method 56
4.3.3 Bayesian Network for Modeling Object/action Dependency 58
4.4 Experiments and Results 59

4.4.1 Experiment Setup 59
4.4.2 System Evaluation 60
4.5 Summary 68

5 Automated Complex Assembly Skill Acquisition through Human Demonstration 69

5.1 Motivation 69
5.2 System Overview 71
5.2.1 PAD System Setup 71
5.2.2 Formulation of the Problem 71
5.3 Methodology 73
5.3.1 Assembly State Estimation 74
5.4 Experiment and Results 76
5.4.1 Object Recognition 76
5.4.2 Action Recognition 78
5.4.3 Assembly State Estimation 80
5.5 Summary 86

6 Implementing Assembly Skills on the Baxter Robot 87

6.1 System Overview 87
6.1.1 Hardware Setup 88
6.1.2 Calibration 88
6.1.3 Object Tracking 91
6.2 Basic Action Learning 93
6.3 Script Implementation 97
6.3.1 Experiment Procedure 97
6.3.2 Results 98
6.4 Summary 100

7 Conclusions and Future works 102

7.1 Conclusions 102

7.2 Future Works 104

List of Tables

3.1 Object and action dependencies. Darker color represents lower probability. 40
3.2 The confusion matrix of object recognition 44
3.3 Recognition accuracy of the low-level HMMs 44
3.4 Recognition accuracy of the Bayesian model 45
3.5 Recognition accuracy of the high-level HMM 47
3.6 The profiles of the subjects 47
3.7 Recognition Accuracy Comparison for MSRDailyActivity 3D dataset 48

4.1 The action, tool and part sets. 60
4.2 The object recognition accuracy. 61
4.3 The feature selection output 61
4.4 The list of recognition approaches. 62
4.5 Recognition accuracy of the decision-level fusion method 66
4.6 Recognition accuracy of the feature-level model 66
4.7 Recognition accuracy of the Bayesian model 67
4.8 The profiles of the subjects 67

5.1 The object recognition accuracy. 79
5.2 Recognition accuracy of the HMMs 79
5.3 Recognition accuracy of the Bayesian model 80
5.4 The procedure of the assembly task 83
5.5 The registration performance 85

6.1 Successful rate of script implementation. 100

List of Figures

1.1 The overall learning framework. 3

2.1 Experimental setup. 17

2.2 The frames used in the experimental setup. 18

2.3 The block diagram of the imitation learning phase. 19

2.4 State definition for the reinforcement learning. 23

2.5 Trajectory encoding and generalization. 25

2.6 (a)-(d) Replaying the generalized trajectory. (e)-(h) Reproducing the generalized trajectory in an unknown table pose. 26

2.7 (a) Random exploration learning performance. (b) Guided exploration learning performance. 27

2.8 Trajectory of the robot's movement and human's movement during lifting the table. 27

2.9 Snapshots of human robot performing the table lifting task. (a)-(d) the robot lifting the table up. (e)-(h) the robot putting the table down. 28

3.1 Breakfast scenario. The objects in the scene, from left to right are bowl, milk box, coffee can, cup, cereal box. 32

3.2 The multi-level action recognition framework. 34

3.3 Setup for 3D object model modeling. (a) the camera view. (b) the object recording setup. 35

3.4 Two different views of the 3D model of a coffee can. 35

3.5 Human skeleton tracking. 36

3.6 Data segmentation in recognition phase. 37
3.7 The hierarchical probabilistic model. 38
3.8 The constraints in the upper level HMM. 39
3.9 Real time recognition procedure. 41
3.10 Object recognition results 1. The objects from left to right are bowl, milk box, coffee can, cup, cereal box. (a) bowl, coffee can, cup and cereal box have been recognized. (b) bowl, milk box, cereal box have been recognized. 42
3.11 Object recognition results 2. (a) Camera view. The goal is to recognize the coffee can in the middle of the three objects. (b), (c) Recognition result with occlusion and similar object in the scene. (d) Recognition result in cluttered scene. 43
3.12 Action recognition results with object constraints. Results of 11 actions are shown here. The Bayesian network output is action object pair. The object type is not shown in the figure. The legend RER,REY,RSR,RSP represent the angle of right elbow roll, right elbow yaw, right shoulder roll and right shoulder pitch respectively. 45
3.13 Action recognition results with object constraints and sequential constraints. Results of 7 actions are shown here. The high-level output is action object pair. The object is cup. The legend RER,REY,RSR,RSP represent the angle of right elbow roll, right elbow yaw, right shoulder roll and right shoulder pitch respectively. . . . 46
3.14 Recognition performance on different subjects. 47

4.1 The setup of the Portable Assembly Demonstration (PAD) system. 51
4.2 Finger model created. (a) the placement of the markers. (b) finger model created. . 53
4.3 Force sensor setup. (a) the force sensors on the finger tips. (b) the Xbee module for wireless communication. 53
4.4 The flowchart of the action recognition system. 53
4.5 The two-level action recognition framework. 55
4.6 Fusion approaches. (a) feature-level fusion method. (b) decision-level fusion method. 57
4.7 The Bayesian Model for action recognition. 58

4.8 The parts and tools. From left to right, the top row shows saw, G clamp, drill, and wood planner. The middle row shows hammer, screw driver, and wrench. The bottom row shows screw, red base board, long base board, and bolt. 60
4.9 Results of board recognition, where the recognized objects are highlighted in green. (a) recognition of the red base board with occlusion from a bolt. (b) recognition of the red base board in a cluttered scene. 61
4.10 Recognition results of the HMMs using left arm motion features. 62
4.11 Recognition results of the HMMs using left finger motion features. 62
4.12 Recognition results of the HMMs using left finger force features. 63
4.13 Recognition results of the decision-level fusion method. 63
4.14 Recognition results of the feature-level fusion method. 63
4.15 Recognition results of the decision-level fusion method + Bayesian model. 65
4.16 Recognition results of the feature-level fusion method + Bayesian model. 65
4.17 Recognition performance on different subjects. 67

5.1 The hardware setup of the PAD system. 71
5.2 Coordinate system defined by the 6 markers in [1]. 74
5.3 State estimation for a two-step assembly task. M_1 and M_2 are the 3D models of the base board and the bolt. CM_1 is the 3D model of the assembled part after the first action. CM_2 is the 3D model of the assembled part after the second action. 74
5.4 The parts and tools. From left to right, the top row are saw, G clamp, drill, and wood planner. The bottom row are hammer, screw driver, wrench, bolt, screw, nut and red board. 76
5.5 Extraction of the based board's points from the scene point cloud. (a) and (b) show the scene point cloud with two different rotation angles of the Lazy Susan. (c) and (d) show the extracted points of the base board. 77
5.6 Results of tool and part recognition, where the recognized objects are highlighted in red. (a) recognition of the screw driver. (b) recognition of the hammer. (c) recognition of the base board. (d) recognition of the screw. 78

5.7 Results of board recognition, where the recognized objects are highlighted in green. (a) recognition of the board with occlusion from a bolt. (b) recognition of the board with occlusion from another board. (c) recognition of the board in cluttered scene 1. (d) recognition of the board in cluttered scene 2. 78

5.8 Accuracy of the rotation estimation with respect to the portion of point cloud kept. Different colors denote registration from the models with different intial angles. The angular difference between the input model and the target model of each registration is 30 degree. 81

5.9 Registration results. All the figures show the downsampled 3D models without color information. (a) the input 3D model. (b) the target model. (c) registration result of ICP. (d) registration result of SAC-IA. (e) registration result of our approach. ((c) - (e): Red ones indicate the registration results. Yellow ones indicate the target model.) 82

5.10 The actions used in the experiment. (a) inserting the bolt. (b) hammering the bolt. (c) wrenching the bolt. (d) aligning the nut to the hole on the base board. (e) inserting the screw. (f) screwing the screw. 82

5.11 The parts and the associated model. (A) red board. (B) bolt (C) nut (D) parts pose (CP1) after action 1. (E) parts pose (CP2) after action 2. (F) parts pose (CP3) after action 3. (G) parts pose (CP4) after action 4. (H) parts pose (CP5) after action 5. (I) parts pose (CP6) after action 6. (a) - (i) are the models of (A)-(I). 83

5.12 Registration results. (a) register ${M_1}^1$ and ${M_2}^1$ to CM_1 after action 1 “inserting”. (b) register ${M_1}^2$ and ${M_2}^2$ to CM_2 after action 2 “hammering”. 84

5.13 Registration results. Register ${M_1}^5$ and ${M_2}^5$ to CM_6 after action 6 “screwing”. . . . 85

6.1 The assembly skill implementation procedure. 88

6.2 The robot implementation setup. (a) The Baxter robot [2]. (b) The workspace of the Baxter robot. 89

6.3 Camera views. (a) The Xtion color image view. (b) The left gripper camera view. . 89

6.4 Camera calibration results. (a) Robot and camera positions. (b) Relationship between the camera frame and selected frames of the Baxter robot. 90
6.5 Marker placement. (a) reflective markers on camera and object (hammer). (b) frames created using reflective markers. 91
6.6 The frame transform tree. The dash line represents the unknown relationship. . . . 92
6.7 The chessboard frame (CH). (a) the chessboard frame defined in the calibration method. (b) the placement of three markers which are used to create the same frame. 92
6.8 Tracking accuracy evaluation. The peaks are caused by the quick motion of the tracked target. 94
6.9 Human scaffolding for the hammering action. 94
6.10 Trajectory encoding and generalization 1. The figures on the left column represent the trajectories of x,y, and z positions of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances. 95
6.11 Trajectory encoding and generalization 2. The figures on the left column represent the roll, pitch, and yaw rotations of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances. 96
6.12 Reproducing the hammering action given the bolt pose. 97
6.13 The robot implements script 1 (inserting bolt). 99
6.14 The robot implements script 2 (hammering bolt). 99
6.15 The robot implements script 3 (inserting screw). 100
6.16 The robot implements script 4 (screwdriving). 100

Chapter 1

Introduction

In this chapter, first the motivation and objectives of our research are presented. Then an overview of existing robot skill learning approaches is provided. At the end of this chapter, related work is introduced.

1.1 Motivation

As more and more advanced robots enter our life, the demand for teaching robot complex skills increases [3, 4]. If a robot can replace a human worker doing complex assembly tasks, the labor cost can be reduced. Compared to the heavy-duty industrial robot manipulators employed in automotive industry, light weight arm (LWA) robots and dual-arm robots have appeared recently for small batch manufacturing. For example, Foxconn Technology Group has already deployed their own assembly robot called “Foxbot” in their factories [5]. Dual-arm robots have been developed by several companies that are mainly targeting small part assembly. These robots include Rethink Robotics’ Baxter robot [2], Kawada’s Nextage robot [6], etc. Currently these robots can only perform simple assembly tasks. Many of the complicated assembly processes still require human labors. Many of the complicated assembly processes still require human labors. Teaching a robot such delicate assembly skills through human demonstration can avoid lengthy robot programming, while no technical expertise is required for the operator.

Programming by demonstration can enable quick skill acquisition [7]. However, this is only true in the situation where the reproduction is sufficiently similar to that of the demonstration. To allow the robot to learn to perform a task again in a different situation, it appears important to combine PbD methods with other learning techniques. Reinforcement learning (RL) [8] is particularly suitable for this type of problem. It enables the robot to learn its own ways of acting, in order to improve its abilities. RL algorithms have already been successfully implemented in robotic applications such as swinging up and controlling an inverse pendulum [9], refining forehand and backhand tennis swing [10]. RL algorithms perform well in discrete environments of low dimensionality.

There are mainly two ways of demonstration: robot-based demonstration and human-based demonstration. Compared to robot-based demonstration, human-based demonstration is more convenient for the human operators since it avoids operating the robot. To extract important clues from the human demonstration, the following functions are required: accurate action and object recognition and robust action effect estimation. With the effect of the action available, the learned skills can be used in a robot task planning system.

1.2 Objectives

Our research goal is to build a robot skill learning framework for acquiring complex human skills. The overall learning framework we proposed is shown in Fig. 1.1. It contains both low-level skill learning and high-level skill learning. In low-level learning, both PbD and reinforcement learning are employed. These learned skills are treated as basic actions. In high-level skill acquisition, complex skills that usually involve multiple actions and the use of various objects need to be captured. The manipulated objects are recognized. With the object/action dependencies, the manipulative actions are recognized. 3D object models are created which can be used to estimate the spatial relationship between the objects. One example of high-level skill acquisition is assembly skills acquisition. The human will demonstrate the whole process of an assembly task. For example, the sequence of actions may include inserting a screw into a hole, screwing using a screw driver and etc. Finally, the learned skills are implemented on the robot through a planning

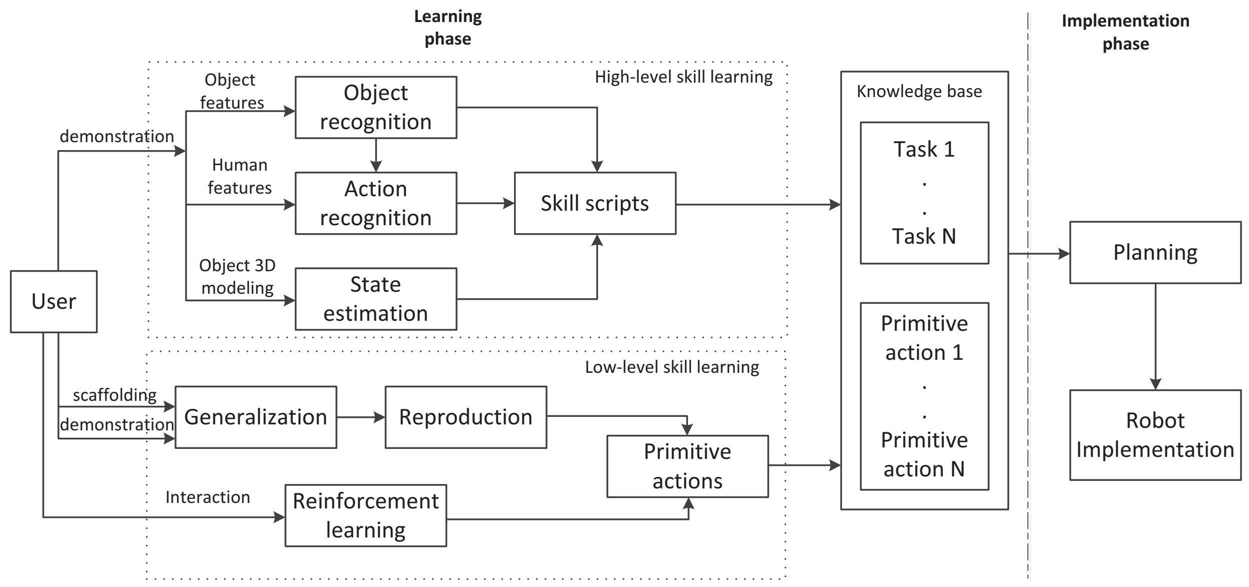

Figure 1.1: The overall learning framework.

procedure.

In summary, the proposed framework enables the following robot capabilities:

- Basic skills learning. With human demonstration, scaffolding, and interaction, the robot can learn basic actions like grabbing, approaching, lifting and etc. The learned actions are treated as basic skills for the high-level skill learning.

- Complex skills learning. With a Portable Assembly Demonstration (PAD) system, the skill scripts can be automatically extracted from the demonstration. It includes the action applied, the object involved and the relative position and orientation between objects.

- After learning both the basic actions and the complex skills, the robot can autonomously interpret and implement the generated skill scripts.

1.3 Overview of Robot Skill Learning Approaches

Robots can acquire skills through many different learning approaches, such as programming by demonstration, and reinforcement learning. The idea of these methods is inspired by how human learns. The most straightforward method humans use to learn is through imitation. Also we can learn by ourselves through trial-and-error since demonstrations are not always available.

1.3.1 Programming by Demonstration

The research on robot programming by demonstration [11] started about thirty years ago, and has been attracting more research interest during the past decade. The rationale for moving from tedious coding for robot programming to flexible demonstration-based skill acquisition is three-fold [12].

First, imitation learning offers an implicit means of training a machine, such that explicit and tedious programming of a task by a human user can be minimized or eliminated. Imitation learning is thus a “natural” means of interacting with a machine that would be accessible to people.

Secondly, it is a powerful mechanism to reduce the complexity of search spaces for learning. When observing either good or bad examples, one can reduce the search space for a possible solution, by either starting the search from the observed good solution (local optima), or conversely, by eliminating bad solutions from the search space. Imitation learning is, thus, a powerful tool for enhancing and accelerating learning.

Third, studying and modeling the coupling of perception and action, which is at the core of imitation learning, helps us understand the mechanisms by which the self organization of perception and action could arise. The reciprocal interaction of perception and action could explain how competence in motor control can be grounded in rich structure of perceptual variables, and vice versa, how the processes of perception can develop as a means to create successful actions.

PbD promises are thus multi-fold. On the one hand, it will make learning faster, in contrast to reinforcement learning or other trial-and-error learning approaches. On the other hand, one expects that the methods, being user-friendly, will enhance the application of robots in human environments. Recent progresses in the field, which we review in this chapter, show that the field has made a leap forward during the past decade toward these goals. In addition, we anticipate that these promises may be fulfilled very soon.

A number of key problems need to be solved to ensure such a generic approach for transferring skills across various agents and situations. One of the key problems is “What to Imitate” [12]. Current approaches to representing a skill can be broadly divided between two trends: a low-level representation of the skill, taking the form of a non-linear mapping between sensory and motor information, which we will later refer to as “trajectories encoding”, and, a high-level representation of the skill that decomposes the skill in a sequence of action-perception units, which we will refer to as “symbolic encoding”.

Another key problem is “How to reproduce a skill in a completely novel situation” [12]. The robustness of dynamic system control to perturbations occurring in uncontrolled environments has been demonstrated. However, it can happen that the dynamic system alone is not sufficient to handle any perturbation. In those cases, an exploration process is needed, during which the robot will search for new ways of accomplishing the task. Reinforcement learning is particularly suitable for this type of problem.

Learning through interaction is usually combined with PbD. The purpose is to detect and utilize “social cues” given implicitly or explicitly by the teacher during training. It has become the focus of a large body of work in PbD [13, 14]. Such social cues can be viewed as a way to introduce priors in a statistical learning framework which can speed up learning.

1.3.2 Reinforcement Learning

Reinforcement learning is learning how to map situations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics: trial-and-error search and delayed reward, are the two most important features of reinforcement learning [15].

Reinforcement learning is different from supervised learning, most current research in machine learning, statistical pattern recognition, and artificial neural networks. Supervised learning is learning from examples provided by a knowledgeable external supervisor [16]. This is an important type of learning, but alone it is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. In uncharted territories where one would expect learning to be most beneficial an agent must be able to learn from its own experience [17].

One of the disadvantages of reinforcement learning is that as the dimension of the learning space grows, the computation cost will grow exponentially. Combined with PbD, which has already offered a solution, the search space can be decreased dramatically.

1.4 Related Works

Related work is divided into three parts. First, PbD approaches are surveyed. Then a review on action recognition is presented. Finally, an overview of the ways of skill demonstration is given.

1.4.1 Learning from Demonstration

Current approaches to representing a skill can be broadly divided into two trends: a low-level representation of the skill, taking the form of a non-linear mapping between sensory and motor information, which we will later refer to as "trajectories encoding", and, a high-level representation of the skill that decomposes the skill into a sequence of action-perception units, which we will refer to as "symbolic encoding". Some related works are listed below. Work in low-level PbD encodes human movements in either joint space, task space or torque space. The encoding may be specific to a cyclic motion [18], a discrete motion [19], or to a combination of both. The most promising approaches to encoding human movements are those that encapsulate the dynamics of the movement into the encoding itself. Several of these methods are highlighted below.

Calinon *et al.* [20] presented a probabilistic architecture for solving generically the problem of extracting the task constraints through a PbD framework and for generalizing the acquired knowledge to various situations. They extend this approach to a more generic procedure handling simultaneously constraints in joint space and in task space by combining directly the probabilistic representation of the task constraints with a simple Jacobian-based inverse kinematics solution.

Two 5-DOFs Katana robots [21] from Neuronics are used for the experiment. Each motor is equipped with encoders which allows the user to move the robot manually while registering joint angle information.

Calinon *et al.* [22] proposed an approach to teaching incrementally human gestures to a humanoid robot. The learning process consists of first projecting the movement data in a latent space and encoding the resulting signals in a Gaussian Mixture Model (GMM). The demonstrations can be provided in various modalities. The robot observes a human user demonstrating the gesture while wearing motion sensors. On the other hand, the coach helps the robot move correctly its limbs by kinesthetic teaching.

Ude *et al.* [23] used spline smoothing techniques to deal with the uncertainty contained in several motion demonstrations performed in a joint space and in a task space. In [24], using different demonstrators ensures variability across the demonstrations and quantifies the accuracy required to achieve a Pick & Place task. The different trajectories form a boundary region that is then used to define a range of acceptable trajectories.

In [25], the robot acquires a set of sensory variables while a human demonstrates a manipulation task of arranging different objects. At each time step, the robot stores and computes the mean and variance of the collected variables. The sequence of means and associated variances is then used as a simple generalization process, providing respectively a generalized trajectory and associated constraints.

Dynamical systems also offer a particularly interesting solution to an imitation process aimed at being robust to perturbations. It is robust to dynamical changes in the environment. The first work to emphasize this approach was that of Ijspeert *et al.* [10], who designed a motor representation based on dynamical systems for encoding movements and for replaying them in various conditions. The approach combines two ingredients: nonlinear dynamical systems for robustly encoding the trajectories, and techniques from non-parametric regression for shaping the attractor landscapes according to the demonstrated trajectories.

The main advantages of these trajectory-level approaches are that they offer generic representation of motion which allows encoding of very different types of signals. However it doesn't allow to reproduce complicated high-level skills.

On the other hand, a large body of work [14, 26] uses a symbolic representation of both the learning and the encoding of skills and tasks. This symbolic way of encoding skills may take several forms. One common way is to segment and encode the task according to sequences of predefined actions, described symbolically. Encoding and regenerating the sequences of these actions can be done using classical machine learning techniques, such as Hidden Markov Model (HMM).

In [27], an approach for teaching a humanoid robot is presented that will enable the robot to learn typical tasks required in everyday household environments. The first step concentrates on an analysis of human actions and action sequences that can be identified when watching a human demonstrator. Secondly, sensor systems are introduced which augment the robot's perception capabilities while watching a human's demonstration and the robot execution of tasks respectively. The main focus is then put on the knowledge representation in order to be able to abstract the strategies and to transfer them onto the robot system.

Ekvall *et al.* [28] proposed a novel method for learning robot tasks from multiple demonstrations. Each demonstrated task is decomposed into subtasks that allow for segmentation and classification of the input data. The demonstrated tasks are then merged into a flexible task model, describing the task goal and its constraints. The two main contributions of their paper are the state generation and constraints identification methods. They also present a task level planner, which is used to assemble a task plan at run-time, allowing the robot to choose the best strategy depending on the current world state.

Saunders *et al.* [29] described an approach of robot task learning based on studies of social animals where two teaching strategies are applied to allow a human teacher to train a robot by moulding its actions within a carefully scaffolded environment. Within these environments, sets of competences can be built by building state/action memory maps of the robot interaction within that environment. These memory maps are then polled using a k-nearest neighbor based algorithm to provide a generalized competence.

The main advantage of these symbolic approaches is that high-level skills (consisting of sequences of symbolic cues) can be learned efficiently through an interactive process. However, because of the symbolic nature of their encoding, the methods rely on a large amount of prior knowledge to predefine the important cues and to segment those efficiently.

Also, as the PbD techniques develop, traditional ways of guiding/teleoperating the robot were progressively replaced by more user-friendly interfaces, such as vision [27], data gloves [30], laser range finder [31] and kinesthetic teaching which has already been introduced above.

1.4.2 Action Recognition

In recent years, vision-based human action recognition has found many applications, such as human-computer interaction, surveillance, etc. [32]. The use of human action recognition in robotics is a new research frontier. As robots start entering human environments, there is a great need for robots to acquire new skills in an efficient way.

There are generally three steps for vision based action recognition: feature extraction, model training and classification. Local spatial-temporal features such as histogram of gradient (HOG) [33], histogram of optical flows (HOF) [34], 3D scale-invariant feature transform (SIFT) [35], and Cuboids [36] have been widely used for action recognition. In [37, 38, 39, 40], features are first extracted on images from a 2D sensor or point clouds from a 3D sensor. Then probabilistic approaches such as Hidden Markov Models (HMMs) [41] and Supporting Vector Machine (SVM) [42] are used for modeling and classification.

To further improve the accuracy, feature-level fusion has been applied to the features generated from the single sensors. In He's paper [43], a new feature fusion method for gesture recognition based on a single tri-axis accelerometer has been proposed. Both time-domain and frequency-domain features are extracted. Recognition of the gestures is performed using SVMs. The average accuracy results using the proposed fusion methods is 89.89% which is improved compared with the approach using only one of the features. Tran *et al.* [44] presented a real-time or online system for continuous recognition of human daily actions. HMMs are applied to model a joint of human posture features. The results show the fusion models outperform the baseline approach.

However, the capability to differentiate very similar actions mainly depends on the distinction of features. Therefore, more and more attention has been focused on how to utilize multiple sensor modalities to help improve action recognition. Correspondingly, the information fusion approaches have been applied. Thomas *et al.* [45] present a novel method for continuous activity recognition based on ultrasonic hand tracking and motion sensors attached to the users arms. Plausibility analysis is designed to fuse the decisions. Lukowicz *et al.* [46] present a technique to automatically track the progress of maintenance or assembly tasks using body worn sensors. The technique is based on a novel way of combining data from accelerometers with simple frequency matching sound classification. Zhang *et al.* [47] presented a framework for hand gesture recognition based on the information fusion of a three-axis accelerometer (ACC) and multichannel electromyography (EMG) sensors signals. A decision tree and multistream HMMs are utilized for decision-level fusion to generate a final decision. Chen *et al.* [48] presented a robust visual system that allows effective recognition of multiple-angle hand gestures in finger guessing games. Three Support Vector Machine (SVM) classifiers were trained for the construction of the hand gesture recognition system. The classified outputs were fused by proposed plans to improve system performance. The system presented can effectively recognize hand gestures, with an accuracy of over 93%, for different angles, sizes, and different skin colors. Wu *et al.* [49] proposed to take audiovisual characteristics of realistic human action videos into account in human action recognition. Effective features are identified from a large number of audio features with the generalized multiple kernel learning algorithm. The widely used spacetime interest point descriptors are utilized as visual features, and a support vector machine is employed for both audio- and video-based classifications. At the final stage, fuzzy integral is utilized to fuse recognition results of both audio and visual modalities. Bahrepour *et al.* [50] implemented classification techniques on wireless sensor nodes attached to patients body, online evaluation of the classification results on individual nodes, and fusing results of various nodes to resolve possible conflicts between sensor nodes and reach a consensus. The wireless body sensor network consists of a number of wireless sensor nodes that cooperatively monitor physical and physiological conditions of a person.

The other way is to model contextual information. Kojima *et al.* [51] proposed a method for recognizing human actions and objects. With the recognition of the human pose, object and the detection of contact points, human actions can be inferred based on the prior knowledge. The limitation is that each object and pose correspond to only one action which results in a deterministic object and action correlation. Gupta *et al.* [52] presented an approach which unifies the inference process involved in object recognition and localization, action understanding and perception of object reaction. They use the object properties as prior for action recognition, while taking the advantage of object reaction to verify the object detection. Wilson *et al.* [53] presented a parametric Hidden Markov Model (PHMM) for human action recognition. They indirectly modeled the effect of object properties on human actions. Yao *et al.* [54] proposed a mutual context model to jointly model objects and human poses in human-object interaction activities. In their approach, activity classes, objects and human poses all contribute to the recognition and detection of each other. The model includes co-occurrence compatibility between activities, object and pose. Instead of considering the dynamics of actions in the 3D image, they mainly focus on pose detection in 2D images. Kjellstrom *et al.* [55] presented a method for categorizing manipulated objects and human manipulative actions in the context of each other. Their method employs conditional random fields (CRFs) and is able to simultaneously segment and classify human hand actions, as well as detect and classify the objects involved in the action. However, to segment hand from the 2D images, the appearance of the hand is compared to a large database (on the order of 10^5 examples) of synthetic hand views tagged with articulated pose and orientation which is computationally intensive.

Kojima *et al.* [51] proposed a method for recognizing human actions and objects. With the recognition of the human pose, object and the detection of contact points, human actions can be inferred based on the prior knowledge. The limitation is that each object and pose correspond to only one action which results in a deterministic object and action correlation. Gupta *et al.* [52] presented an approach which unifies the inference process involved in object recognition and localization, action understanding and perception of object reaction. They use the object properties as prior for action recognition, while taking the advantage of object reaction to verify the object detection. Wilson *et al.* [53] presented a parametric Hidden Markov Model (PHMM) for human action recognition. They indirectly modeled the effect of object properties on human actions. Yao *et al.* [54] proposed a mutual context model to jointly model objects and human poses in

human-object interaction activities. In their approach, activity classes, objects and human poses all contribute to the recognition and detection of each other. The model includes co-occurrence compatibility between activities, object and pose. Instead of considering the dynamics of actions in the 3D image, they mainly focus on pose detection in 2D images. Kjellstrom *et al.* [55] presented a method for categorizing manipulated objects and human manipulative actions in the context of each other. Their method employs conditional random fields (CRFs) and is able to simultaneously segment and classify human hand actions, as well as detect and classify the objects involved in the action. However, to segment hand from the 2D images, the appearance of the hand is compared to a large database (on the order of 10^5 examples) of synthetic hand views tagged with articulated pose and orientation which is computationally intensive.

1.4.3 Ways of Skill Demonstration

The ways to demonstrate skills to robots fall into two categories: robot-based demonstration and human-based demonstration. Robot-based demonstration uses robots as the demonstrator. This method mainly includes teleoperation [56] and kinesthetic teaching which is also called scaffolding [57]. On the contrary, human-based demonstration employs humans as the demonstrator. Compared to robot-based demonstration, human-based demonstration is more convenient for the human operators since they can just focus on the task itself and do not need to control the robot. Two kinds of sensors are widely used for collecting human demonstration data: wearable sensors and vision sensors. Wearable sensors include inertial measurement units (IMUs) [58] and data gloves [59, 60]. In robot learning, it is also called the “sensor on teacher” approach. Wearable sensors are usually used along with vision sensors that extract the information of the involved objects. Lee *et al.* [61] proposed a skill learning and inference framework for a Skilligent robot. Data gloves are used to demonstrate the task. Based on the framework, the robot learns and infers situation-adequate and goal-oriented skills to handle uncertainties and human perturbations. Kunze *et al.* [62] performed a manipulation task in an interactive physics simulation. By performing novel manipulation tasks in a virtual environment using a data glove, task-related information of the demonstrated actions can be directly accessed and extracted from the simulator. By using simulation, they avoid the practical problem of object and pose estimation.

On the other hand, vision-based human motion tracking systems are regarded as a natural way to capture demonstration since no sensors need to be attached to the demonstrator. Ahmadzadeh *et al.* [63] proposed a visuospatial learning approach which uses a simple algorithm and minimum prior knowledge to learn a sequence of operations from a single demonstration. Song *et al.* [64] developed a testbed for learning by demonstration from natural languages and a Kinect sensor, with the initial domain of kitchen activities. In their work, only a single large-size object is involved, which can be recognized easily. Additionally, there are research works specifically on vision-based assembly skill learning from human demonstration. Hovland *et al.* [65] proposed an approach to representing an assembly skill by a hybrid dynamic system where a discrete event controller models the skill. The discrete controller is represented as an Hidden Markov Model (HMM) [41]. The deficiency for this approach is that as the task becomes complex, the HMM will demand a tremendous training data set. Dantam *et al.* [66] developed a method to transfer human demonstration to robots. The demonstration is interpreted into a sequence of object connection symbols which can be further transformed into the task language. The assembly task they handled is simple and only the location of the part is required. Takamatsu *et al.* [67] aimed to recognize assembly tasks through human demonstration. Two rigid polyhedral objects are recognized from noisy data obtained by a conventional 6 degree-of-freedom (DOF) object-tracking system. Assembly tasks can be basically expressed as chains of two-object relationships. However, they ignored the problem of occlusion between the parts. Aleotti *et al.* [68] demonstrated the assembly task through simulation, which does not address the action recognition and object recognition task. A task planner is designed to analyze the demonstration and segment it into a sequence of action primitives.

1.5 Organization of the Dissertation

In the following chapters, each part of the learning framework will be introduced. In Chapter 2, human robot collaborative manipulation task learning through PbD and reinforcement learning is described, which is a type of low-level skill learning of basic actions. In Chapter 3, an action recognition system with a Kinect sensor is developed which considers the correlation between objects and actions, as well as the sequential constraints of actions. In Chapter 4, The action recognition framework is extended to handle more challenging tasks, such as assembly skill acquisition, using multiple sensing modalities. Based on the previous work, Chapter 5 presents a Portable Assembly Demonstration (PAD) system which can generate a complete robot skill script for complex part assembly from human demonstration. Chapter 6 describes how to implement the skill scripts generated by the PAD system on a real robot. Chapter 7 presents the conclusions and future work is discussed.

Chapter 2

Human Robot Table Lifting through PbD and Reinforcement Learning

In this chapter, the robot imitation learning and reinforcement learning framework is presented as a solution to low-level skill learning. Section 2.1 introduces the motivation of the proposed framework. In Section 2.2, the problem to be solved is described. In Section 2.3, the experiment setup is introduced. Section 2.4 gives details of the framework. In Section 2.5, experiment results are given and discussed.

2.1 Motivation

Before demonstrating a complex task, the low-level skills (primitive actions) should be learned first. For co-robots to be useful, one of the fundamental abilities they should possess, is to work collaboratively with humans. A common example where collaboration would be required, is in a cooperative manipulation task, where a human-robot team has to manipulate an object of interest. Human-robot collaboration (HRC) is a research field with a wide range of applications and high economic impact [69]. Technology that enables robots to work collaboratively with humans can be widely applied in the industry as well as in common day-to-day scenarios. This field has seen a renewed interest in recent years because of the possibility of humanoid robots residing along with us in the near future. In this chapter, we are going to explore how to enable a robot to learn such basic manipulative skills in an interactive way. These basic skill can be learned at the low level and they can be later used in high-level skill learning.

2.2 Problem Statement

The task we consider is how a robot learns to collaborate with a human to lift a dummy table. The reason of using a dummy table is because the robot can not handle heavy objects. Two skills need to be learned: reaching the table and lifting the table up with the human. A two-phase framework which combines PbD and reinforcement learning is developed to enable the robot to learn dynamic tasks in a reasonable amount of time. In the first phase, PbD is used to teach the robot how to reach out to the table. In the second phase, reinforcement learning is used for keeping the table horizontal.

2.3 Experimental Setup

This section presents the experimental platform developed for human-robot table lifting. The hardware includes a motion capture system, a humanoid robot and a dummy table. Fig. 2.1 illustrates the experimental setup.

The goal is to extract the task constraints from the human demonstrations, map the constraints to the robot's internal frame and let the robot learn to approach the table and keep the table horizontal while lifting it with the human. Multiple demonstrations of human's hand approaching the table are given to the robot. Markers are attached and rigid-bodies are created to collect information of interest. The position vector and the rotation matrix of each rigid-body are defined as follows:

- The position vector of the human's left wrist (${}^{HLW}P_W$).
- The position vector of the robot's left wrist (${}^{RLW}P_W$) and its torso for reference (${}^{RT}P_W$).
- Four markers on the table's corners to create the table object. Its position vector is represented as (${}^{TAB}P_W$). Mid-points of these markers are calculated for each end and they denote

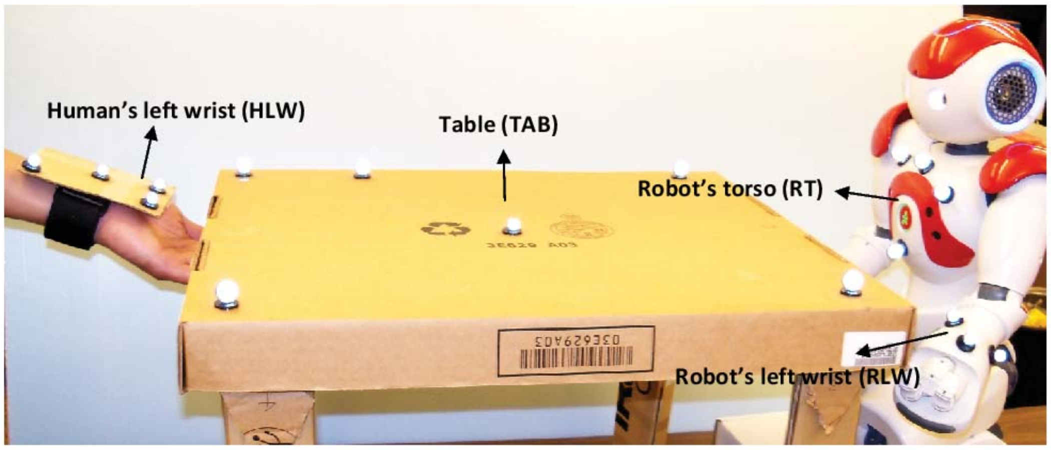

Figure 2.1: Experimental setup.

the position of the human robot-ends ($Z1$ and $Z2$).

- ${}^{TAB}R_W$ and ${}^{RT}R_W$ denote the rotation matrices of the table and the robot's torso respectively.

Each variable has three parts, the upper left superscript denotes the name of the variable, the subscript denotes the which frame the variable is in, where P indicates a position vector and R indicates a rotation matrix.

There are totally four frames used which are shown in Fig. 2.2: the world frame W, the table's frame T, the robot's internal frame RI and external frame RE. The world frame is the frame of the Vicon system. All the position vectors and rotation matrices are initially in the world frame. The table's frame is used to observe the human's hand motion with respect to the table during the demonstration. The robot's external frame is used for calibration, and the robot's internal frame is used for controlling the robot.

2.3.1 Motion Capture System

The motion capture system used for our experiments is the Vicon MX motion capture system [70]. The system consists of 12 Vicon T-40 cameras. Each camera can capture a 10 bit grayscale image at a resolution of 4 megapixels. It can capture data at speeds upto 100 frames per second. However, in our experiments, we only need the speed of 10 frames per second. The system is

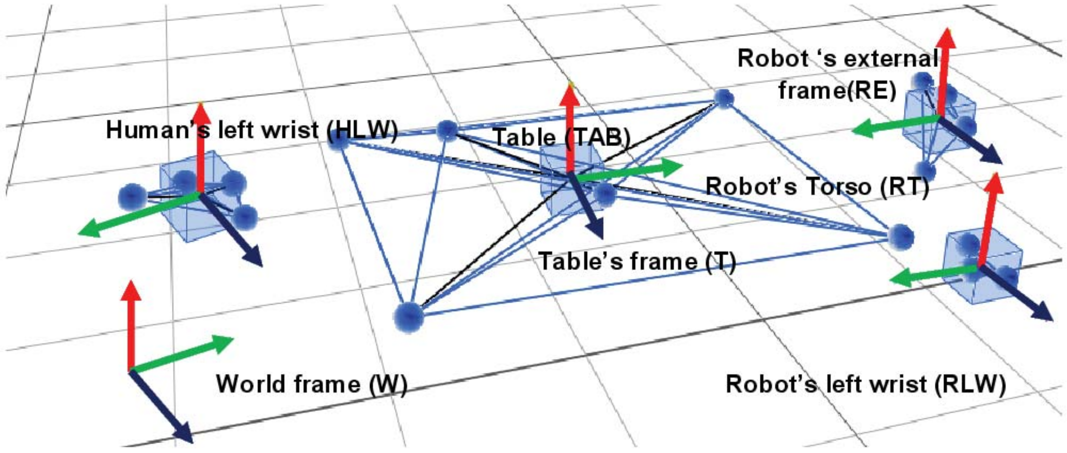

Figure 2.2: The frames used in the experimental setup.

equipped with sophisticated dynamic reconstruction algorithms for real time tracking. A Gigabit Ethernet port is provided for connecting the cameras to the system. With the given system, we can track any optical marker within a tolerance of 0.7 mm. We can create rigid bodies which are three markers attached to a solid body in a specific pattern. The Vicon Tracker software is used for capturing the rigid-body data. The algorithms used in Tracker are optimized for tracking rigid bodies.

The Nao humanoid robot [71] is used for the experiment. It is an autonomous, programmable and medium-sized humanoid robot which has 21 degrees of freedom (DOF), developed by Aldebaran Robotics. The robot can be controlled remotely using telnet-like commands on a wireless network. The SDK provided by the company includes an inverse kinematics procedure to control end-effector positions with respect to a frame of reference located in its torso.

2.3.2 Calibration

The robot's end-effector has to be controlled with respect to its internal frame of reference. But the data obtained from motion capture, is for the markers placed on the robot's body. Hence, a correspondence between the externally attached markers and the controllable points on the robot has to be found out. The robot's SDK can provide the position of the robot's end effector with respect to its internal frame of reference. Hence we can obtain a model of transformation given the

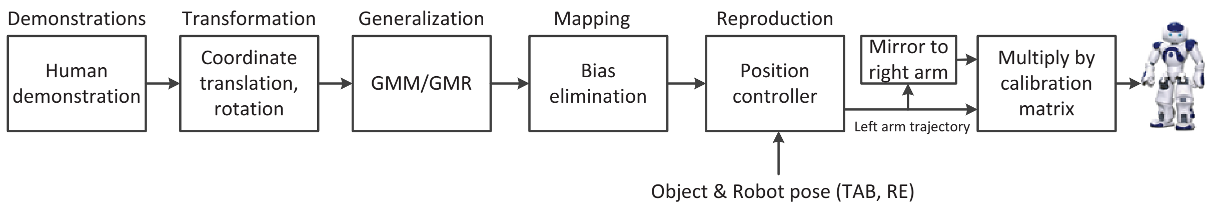

Figure 2.3: The block diagram of the imitation learning phase.

motion capture data and the corresponding robot's data. We model this relation as a homogeneous transformation which includes scaling, translation and rotation. The homogeneous transformation takes the form of a calibration matrix.

For calibration, the robot waves its hand in random trajectories trying to cover all the possible joint configurations of its arms. While it is doing so, positions are collected simultaneously from the motion capture system (denoted by A) and forward kinematics is applied to robot's internal joint encoders (denoted by B). The linear least squares formula used to calculate this homogeneous transformation (H) can be expressed as

$$H = (A^T A)^{-1} A^T B \tag{2.1}$$

2.3.3 Imitation Learning Phase

Firstly, the trajectories of interest have to be derived from the human demonstrations. Then we have to extract critical constraints from the trajectories. Gaussian Mixture Model (GMM) is used to encode the set of demonstrated trajectories (representation phase). Gaussian Mixture Regression (GMR) [72] is then applied to retrieve a smooth generalized version of these trajectories and associated variances (generalization phase). After mapping the constraints to the robot's perspective, the robot can generate its own trajectories based on the constraints. In the reproduction phase a position controller is derived from the generalized trajectories for the new position and orientation of the robot and the table. The block diagram of the proposed learning algorithm is shown in Fig. 2.3. The details of the block diagram are described next.

Coordinate transformation

This section explains the required coordinate transformations. ${}^{HLW}P_W$, ${}^{RT}P_W$ and ${}^{TAB}P_W$ are all in the Vicon's world frame. The human's wrist trajectory with respect to the table (denoted by ${}^{HLW}P_T$) is of interest for the learning purpose. So we need to transform the trajectory from the world frame to the table's frame. This transformation includes translation and rotation given by

$$ {}^{HLW}P_T = {}^{TAB}R_W({}^{HLW}P_W - {}^{TAB}P_W) \tag{2.2} $$

- Generalization

For imitation learning we adopt the probabilistic learning framework proposed by Calinon *et al.* [20]. Let $\left\{\varepsilon_j\right\}_{j=1}^{N}$ denote the N demonstrations. Each demonstration is normalized to 100 time steps. Each datapoint $\varepsilon_j = \left\{t_j, \varepsilon_j^S\right\}$ consists of a time step t_j and a coordinate of position ε_j^S which is a point in the trajectory of the human's left wrist with respect to the table, ${}^{HLW}P_T$. The dataset is first modeled by a Gaussian Mixture Model(GMM) of K components [73]. Each data point is defined by its probability density function

$$ p(\epsilon_j) = \sum_{k=1}^{K} \pi_k N(\epsilon_j; \mu_k, \Sigma_k) \tag{2.3} $$

where, π_k are prior probabilities and $N(\epsilon_j; \mu_k, \Sigma_k)$ are Gaussian distributions defined by centers μ_k and covariance matrices Σ_k, whose temporal and spatial components can be represented separately as

$$ \mu_k = (\mu_k^T, \mu_k^S), \quad \Sigma_k = \begin{pmatrix} \Sigma_j^{TT} & \Sigma_j^{TS} \\ \Sigma_j^{ST} & \Sigma_j^{SS} \end{pmatrix} \tag{2.4} $$

Based on the GMM, a generalized version of the trajectories is computed by applying Gaussian Mixture Regression (GMR). The procedure is as follows. For each component k, the expected distribution of likelihood of ε_j^S given a time step t_j and Gaussian mixture component k is defined by

$$ p(\epsilon_j^S | t_j, k) = N(\epsilon_j^S;\ \hat{\epsilon}_k^S, \hat{\Sigma}_k^{SS}) \tag{2.5} $$

$$\hat{\epsilon}_k^S = \mu_k^S + \Sigma_k^{ST}(\Sigma_k^{TT})^{-1}(t_j - \mu_k^T) \tag{2.6}$$

$$\hat{\Sigma}_k^{SS} = \Sigma_k^{SS} - \Sigma_k^{ST}(\Sigma_k^{TT})^{-1}\Sigma_k^{TS} \tag{2.7}$$

By taking the complete GMM into account, the expected distribution is defined by

$$p(\epsilon_j^S|t_j) = \sum_{k=1}^{K} \beta_{k,j} N(\epsilon_j^S;\ \hat{\epsilon}_k^S, \hat{\Sigma}_k^{SS}) \tag{2.8}$$

where $\beta_{k,j}$ is the probability of the component k responsible for t_j. By using the linear transformation property of Gaussian distribution, and estimation of the conditional expectation of ϵ_j^S given t_j is thus defined by $p(\epsilon_j^S\ |t_j)\ \propto\ N(\hat{\epsilon}_j^S, \hat{\Sigma}_j^{SS})$, where the parameters of the Gaussian distribution are defined by

$$\hat{\epsilon}_j^S = \sum_{k=1}^{K} \beta_{k,j}\hat{\epsilon}_k^S,\ \ \hat{\Sigma}_j^{SS} = \sum_{k=1}^{K} \beta_{k,j}^2 \hat{\Sigma}_k^{SS} \tag{2.9}$$

By evaluating $\left\{\hat{\epsilon}_j^S, \hat{\Sigma}_j^{SS}\right\}$ at different time steps t_j, a generalized form of the trajectories $\hat{\epsilon} = \left\{t_j, \hat{\epsilon}_j^S\right\}$ and associated covariance matrices $\hat{\Sigma} = \left\{\hat{\Sigma}_j^{SS}\right\}$ representing the constraints along the task can be computed [74].

Correspondence problem

In the next step, the constraints derived from ${}^{HLW}P_T$ will be applied to the robot. Since the human's dimension is different from the robot, the constraints derived for ${}^{HLW}P_T$ have to be mapped to the robot's end effector with respect to the table ${}^{RLW}P_T$. This problem can be simplified if we consider only the position of the human's wrist with respect to the table. Then, it only needs to compensate for the dimension difference between the human's wrist and the robot's end effector. The dimension difference is nothing but a constant bias. A simple method is proposed to calculate this dimension difference. We put markers on a fixed object, then we let the human with markers on the wrist and robot with markers on the end effector touch the same point on the box respectively. The coordinates with respect to the fixed object are obtained. The difference of these two coordinates is the dimension difference.

Reproduction

In the reproduction phase, a new trajectory for the robot's end effector ${}^{RLW}P_{RE}$ has to be produced based on the generalized version of ${}^{RLW}P_T$. Given the ${}^{TAB}P_W$ during reproduction phase, ${}^{RLW}P_{RE}$ can be derived as follows:

We have ${}^{RLW}P_T$ which is

$$ {}^{RLW}P_T = {}^{TAB}R_W({}^{RLW}P_W - {}^{TAB}P_W) \tag{2.10} $$

${}^{RLW}P_W$ can be obtained as

$$ {}^{RLW}P_W = {}^{TAB}R_W^{-1}\,{}^{RLW}P_T + {}^{TAB}P_W \tag{2.11} $$

Finally we can derive ${}^{RLW}P_{RE}$ as

$$ {}^{RLW}P_{RE} = {}^{RT}R_W({}^{RLW}P_W - {}^{RT}P_W) \tag{2.12} $$

${}^{RLW}P_{RE}$ is the trajectory of the robot's left end effector with respect to its torso, which will be enacted by the robot after converting it into the trajectory with respect to the robot internal torso frame.

Mirroring the trajectory

In the imitation learning phase, only left hand demonstrations are provided to the robot. This trajectory is mirrored to obtain the corresponding trajectory of robot's right end effector. The mirroring process keeps all the X and Z coordinates in the trajectory of the left arm as they are and reverses all the Y coordinates.

2.3.4 Reinforcement Learning

Once the robot holds the table successfully, it switches to learn the second task, table lifting with human. The robot has to learn a reactive controller which generates a robot behavior based on the human motion in order to keep the table horizontal. We use a reinforcement learning algorithm to fulfill this task. For reinforcement learning, the state, action space and the rewards have to

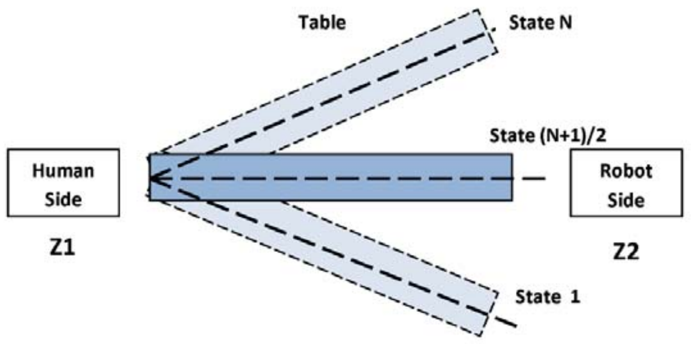

Figure 2.4: State definition for the reinforcement learning.

be defined. Instead of having a complex contact environment, the environment in our task is simplified. The inclination of the table object determines the state. The action space consists of a predetermined discrete set of commands which move the robot's hand-tip up or down by specified distances. The objective of the task is to keep the table horizontal during the task. Accordingly, the reward structure has been designed to give a positive reward if the robot decreases the slope of the table or a negative reward if the robot increases the incline of the table. The reward r is calculated as

$$r = (|Z2 - Z1|)_t - (|Z2 - Z1)|)_{t+1} \tag{2.13}$$

where $Z1$ and $Z2$ represent the position of the human-end and the robot-end of the table respectively.

The state definition for N states is shown in Fig. 2.4. In our task, we chose $N = 5$.

The update for the Q-learning algorithm is given by

$$\Delta Q(s_t, a_t) = \alpha[r + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{2.14}$$

where α is the learning rate, γ is the discount factor.

In order to speed up the reinforcement learning phase, a guided-exploration method is used. The guided learning algorithm is given below.

$Visit(s_i, a_i)$ is a counter which counts the number of visits to the state-action tuple (s_i, a_i). Given a state, the action selection for exploration is done on the basis of number of visits to the particular state-action pair. The state-action pair explored least is given more priority.

Algorithm 1 Guided Q Learning

```
1: Initialize $Visit(s_i, a_i) = 0 \ \forall i \in N$
2: Initialize Q-table $Q(s_i, a_i) = 0 \ \forall i \in N$
3: while Learning phase do
4:     $t = timestep$
5:     $s_t = getState()$
6:     Select $a_t \leftarrow \text{argmin}(Visit(s_t, a))$
7:     Take action $a_t$
8:     $Visit(s_t, a_t) \leftarrow Visit(s_t, a_t) + 1$
9:     $r = getReward()$
10:    Update $Q(s_t, a_t)$ using Eq. (2.14), based on reward r.
11: end while
```

2.4 Experimental Results

In this section, first the imitation learning results are presented, followed by the reinforcement learning results.

2.4.1 Imitation Learning Results

In the imitation learning phase, multiple demonstrations are given by the human. In each demonstration, the human tried to approach the same position of the table with his or her left hand from an arbitrary initial position. An open source code from `http://www.calinon.ch/` has been used to run the GMM/GMR. The GMM/GMR results are shown in Fig. 2.5. Generalized trajectories and constraints are thus obtained. From the results, it can be seen that the constraint on the human-hand's initial position is very loose. In contrast, the constraint on the human hand's final position is very strict, which indicates the final position of the robot's end effector with respect to the table is constant. After compensating for the size difference between human's hand and robot's hand, the robot can generate its own trajectory given the constraints extracted. Each time the table is moved to a new position, a new trajectory is reproduced by the position controller so that the

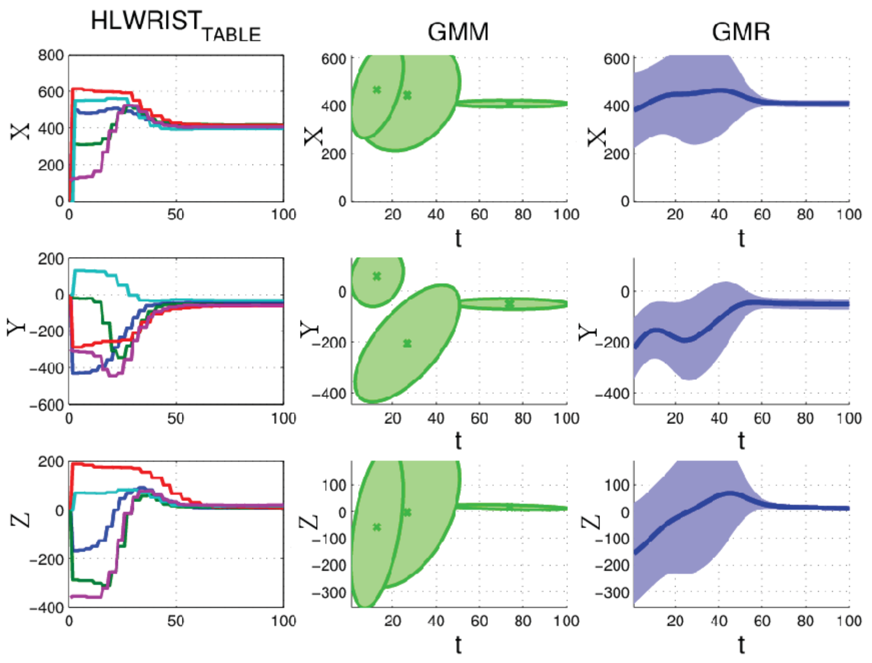

Figure 2.5: Trajectory encoding and generalization.

robot can successfully approach the table. Finally the calibration matrix is used to convert the trajectories from the robot's external frame to the robot's internal frame. In the imitation learning phase, only left hand demonstrations are provided to the robot. This trajectory is mirrored to obtain the corresponding trajectory of robot's right end effector. The results are shown in Fig. 2.6. The images (a)-(d) show the robot replaying the generalized trajectories extracted from the demonstrations and the images (e)-(f) show the robot reproducing the trajectories in a new situation (different pose of the table). It is observed that the trajectories generated are smooth, with which the robot successfully approaches the table.

2.4.2 Reinforcement Learning Results

For state-action space consisting of 5 states and 5 actions, the performance of random exploration is compared with that of guided exploration. For training, in each experiment, the number of iterations is fixed to 100. To test the speed of convergence, the experiment was performed 100 times. Fig. 2.7 shows the speed of convergence for these two algorithms for a single experiment

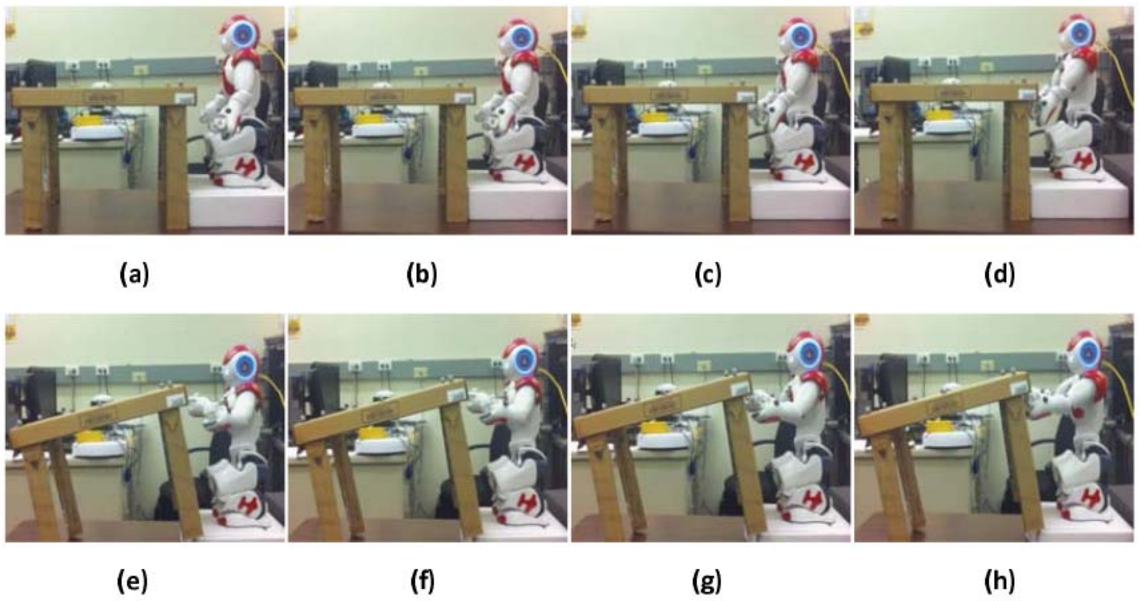

Figure 2.6: (a)-(d) Replaying the generalized trajectory. (e)-(h) Reproducing the generalized trajectory in an unknown table pose.

respectively. It is observed that the guided exploration policy converges much faster and is more stable than the random exploration policy. On average the random exploration took more than 100 trials to reach an optimal policy whereas the guided learning algorithm could reach the optimal policy within 60 trials.

After learning the optimal policy, we apply it to the robot. Fig. 2.8 shows the positions of the ends of the table for human and robot side. The moving range of the table is within 20 cm. Fig. 2.9 shows the whole process of the table lifting task. From these figures, we can see that the robot can follow the human's action and perform the table lifting task successfully. However we could also observe some jerks during the task which are due to the imperfections in the position controlled end effector of the robot. Also the movement of the robot's end effector has some delays, since it takes some time for the robot to realize the human's action.

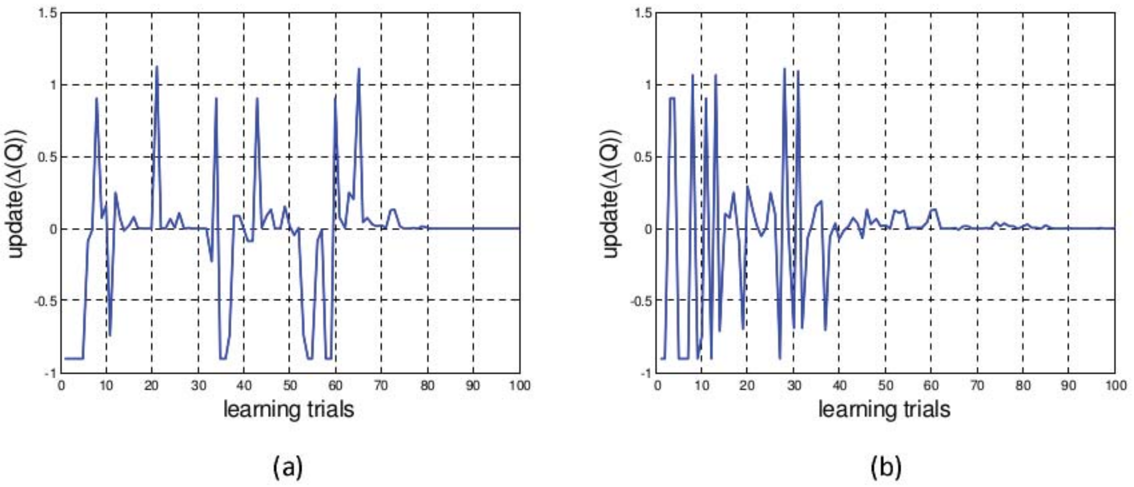

Figure 2.7: (a) Random exploration learning performance. (b) Guided exploration learning performance.

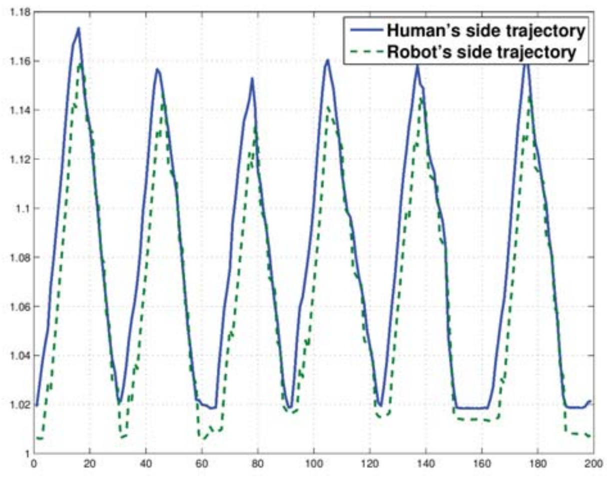

Figure 2.8: Trajectory of the robot's movement and human's movement during lifting the table.

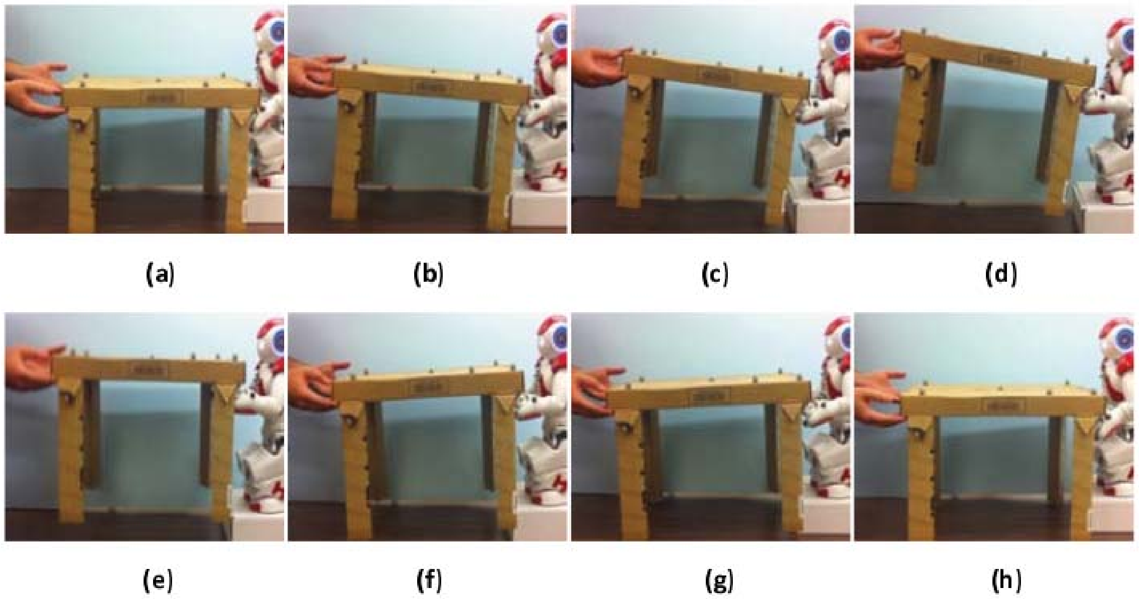

Figure 2.9: Snapshots of human robot performing the table lifting task. (a)-(d) the robot lifting the table up. (e)-(h) the robot putting the table down.

2.5 Summary

This chapter proposed a low-level skill learning strategy for robot skill acquisition. As shown in the overall framework, the low-level skills (primitive actions) learned become part of the knowledge base. It is a two-phase learning framework which combines imitation learning and reinforcement learning. In the first phase, using imitation learning the robot can reach out and hold the end of the table. A Programming by Demonstration (PbD) algorithm is used to accomplish this. From the demonstration, the human left wrist position with respect to the table is recorded. Through compensating the size difference between the human wrist and robot's end effector, the trajectory of the robot left end effector with respect to the table is obtained. Then the relationship between the robot's external frame and internal frame is calibrated. After that, the trajectory which can be implemented by the robot is derived. GMM/GMR is applied to the multiple demonstrations for generalization. The generalized trajectory is reproduced according to the new table position.

In the second phase, through reinforcement learning, the robot can learn to collaborate with a human for the table lifting task. With the guided exploration strategy for Q-learning, the learning speed is improved. Using the entire framework, the robot can learn to perform the collaborative table-lifting task quickly and successfully. Such low-level skills learned can be treated as primitive actions in high-level skill learning. In our task, the robot only behaves as a follower and simply reacts to the human's action. For future works, if the robot can predict human's motion, the performance can be improved. Also, in the reinforcement learning phase, the states and actions are discrete. Using continuous state-action representation can make the robot's action smoother.

Chapter 3

RGB-D Sensor based Manipulative Action Recognition

To obtain high-level skills, the action used in the demonstration is an important clue because it can tell the robot how to implement a task. In this chapter, by adopting an Xtion sensor [75], we aim to recognize manipulative actions. Section 3.1 introduces the motivation of the proposed framework. In Section 3.2, the problem to be solved is formulated. Section 3.3 presents the proposed methodology. Section 3.4 describes the experiment procedures and gives the experimental results.

3.1 Motivation

In recent years, human action recognition has found many applications, such as human-computer interaction, surveillance, video games, etc. [32]. The use of human action recognition in robotics is a new research frontier. As robots start entering human environments, there is a great need for robots to acquire new skills in an efficient way. To fulfill this task, the robot has to understand the actions conducted by the demonstrator.

Traditional action recognition methods merely consider motion related features captured by either motion sensors or cameras. However, the capability to differentiate very similar actions mainly depends on the distinction of features. Therefore, more and more attention has been focused on how to utilize other information sources to help improve action recognition, which is inspired by psychological theories. Previously action and perception are treated as two separate human information processes [76]. Recent studies in experimental psychology have confirmed the role of object recognition in action understanding and vice-versa [77, 78]. In this chapter, we aim to build an accurate action recognition system by considering the contextual constraints such as object/action correlation and sequential constraint between actions. Such an action recognition system can be used in robot skill learning through human demonstration.

3.2 Problem Statement

In this chapter, by adopting an Xtion sensor we aim to solve the manipulative action recognition problem. When equipped with this capability, a robot can understand human's skills and learn from human demonstration. In this work, we choose "preparing and having breakfast" as an example, which is shown in Fig. 3.1. The proposed approach should be able to recognize the sequence of actions involved in preparing and having breakfast. Since many manipulative actions in human's daily life are not very distinctive to each other, an efficient algorithm is needed to distinguish similar actions. In many vision-based action recognition systems, the computational process is often divided into three steps, namely human detection, human tracking and human action recognition. Since the Xtion sensor already realizes human detection and tracking, we only focus on the third step, action recognition.

3.2.1 System Setup

The system setup includes both hardware and software. The hardware consists of an Xtion sensor and a DELL desktop computer with a 2.4GHz Intel Core(TM)2 CPU. The Xtion sensor is used for both action and object recognition. It is a low-cost RGB-D camera that can provide both color images and depth information. Its function is almost the same as Kinect [79]. However, it is lighter than the Kinect sensor and is powered using a USB cable. This sensor has been

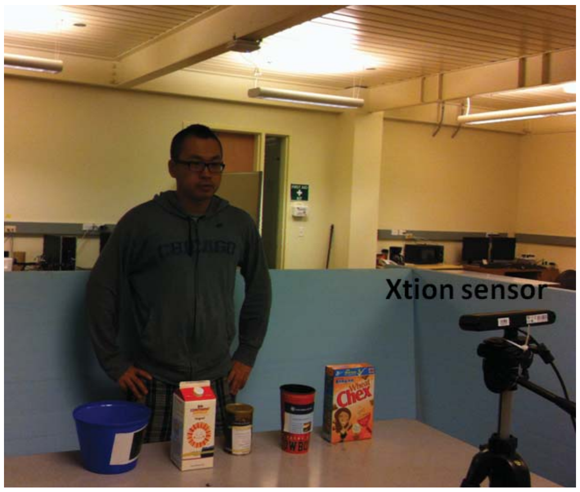

Figure 3.1: Breakfast scenario. The objects in the scene, from left to right are bowl, milk box, coffee can, cup, cereal box.

widely used in many robotic applications for object recognition, manipulation and human robot interaction [1, 80, 40]. In this work, by appropriately setting the Xtion sensor as shown in Fig. 3.1, it can observe the human motion and the objects he manipulates at the same time.

For the software, all the programs run under Robot Operating System (ROS) [81] in Linux. ROS provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. One of the advantages of the ROS framework is that a system built using ROS consists of a number of processes, potentially on a number of different hosts, connected at runtime in a peer-to-peer topology [81]. So instead of writing a whole program for all the functions, we can write each function as a node, and then run these nodes simultaneously. This parallel processing structure can enhance the efficiency of the program.

3.2.2 Formulation of the Problem

Given the above setup, the input to the recognition system has two parts: the observation of the object and the observation of the human action. The observation of the object ψ_o includes 3D point cloud along with associated features $\{P_c, f_1, f_2 ... f_k\}$.

The observation of the action ψ_a is a sequence of skeleton joints angles of the human subject $\{\Gamma_1, \Gamma_2 ... \Gamma_t\}$, where at each time point t, $\Gamma_t = \{\Theta_1, \Theta_2 ... \Theta_p\}$, Θ_i is the *ith* joint angle. Each joint could have up to three joint angles. p is the total number of the joint angles. The reason of choosing joint angles instead of joint positions is because joint angles are view and scale-invariance. On the other hand, according to [82], instead of directly using the original joint angle time series data, one can also extract various types of features such as the mean or variance of joint angle time series, or the maximum angular velocity of each joint. However, in our scenario, we are trying to recognize breakfast preparation actions which usually have relatively short durations and smooth velocities. Therefore, instead of apply a function that maps the joint angle sequence to a single scalar value, we use the joint angle sequence as our feature.

We assume the object set $O = \{O_1, O_2 ... O_m\}$, and the action set $A = \{A_1, A_2 ... A_n\}$. The problem is to develop an algorithm to decide on $A_j \in A$ given the above observation. In other words, the goal of the action recognition is to find a mapping function f,

$$A_j = f(\psi_o, \psi_a) \tag{3.1}$$

Each element in ψ_a is a temporal sequence.

3.3 Methodology

A multi-level probabilistic framework is proposed to recognize the human manipulative actions. The whole framework is shown in Fig. 3.2. There are totally three levels in this framework. In the low level, HMMs are employed to model the dynamics of the actions. In the middle level, a Bayesian model is used to capture object/action dependencies. In the high level, the contextual constraints are modeled with an HMM to refine the mid-level recognition results.

3.3.1 Object Recognition

We adopt the ROS object recognition package developed by the RoboEarth team [83]. The package introduces a way to build up and use an extensive sensor-independent object model database. There are three steps for object recognition in RoboEarth.

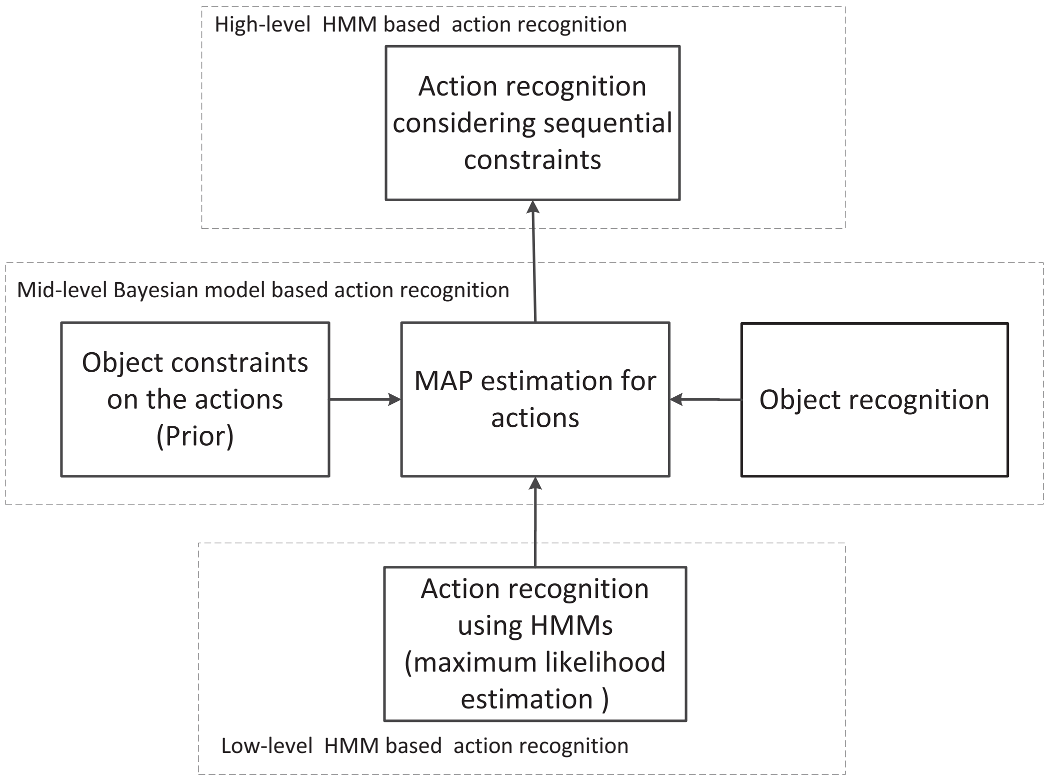
High-level HMM based action recognition
Action recognition considering sequential constraints
Mid-level Bayesian model based action recognition
Object constraints on the actions (Prior)
MAP estimation for actions
Object recognition
Action recognition using HMMs (maximum likelihood estimation)
Low-level HMM based action recognition

Figure 3.2: The multi-level action recognition framework.

3D Model Creation

To build a 3D object model, an Xtion sensor is used along with a marker pattern. At the beginning, the object of interest is placed at the center of the predefined marker pattern on a rotating table such as a Lazy Susan as shown in Fig. 3.3. Subsequently, the Lazy Susan is rotated smoothly by the user so that the Xtion sensor records the object from different views. The marker pattern puts a limit on the size of the object. Typically, the object should have a diameter less than 13 cm so that it will not cover the maker pattern. Fig. 3.4 shows the 3D model of a coffee can created using this method.

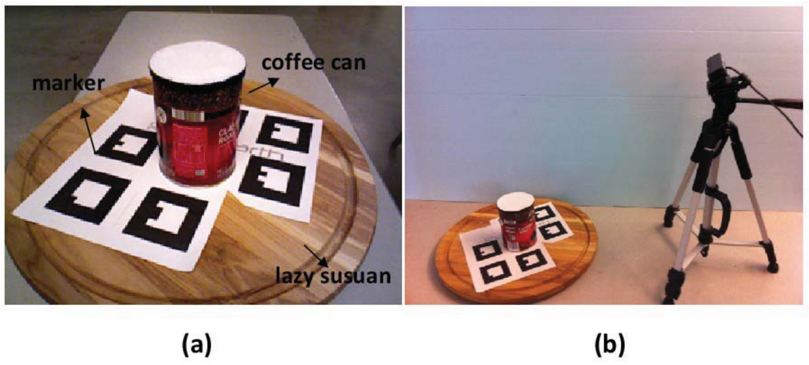

Figure 3.3: Setup for 3D object model modeling. (a) the camera view. (b) the object recording setup.

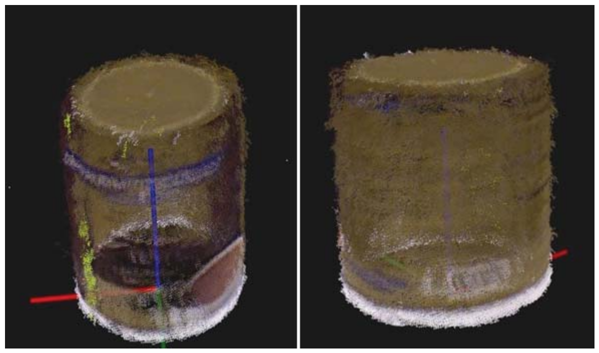

Figure 3.4: Two different views of the 3D model of a coffee can.

Feature Extraction

Each object model consists of several recordings from around 100 viewpoints. For each recording, a 3D point cloud along with the SURF features [84] associated with some of those points are stored. There are currently two different recognition algorithms implemented that make use of the objects stored in the database: one for using common RGB cameras and the other for using the Xtion. We use the recognition algorithm using the Xtion.

Recognition

In the recognition phase, the depth information for the feature points in the camera image is used to compare the distances between given feature points with the distances in the respective

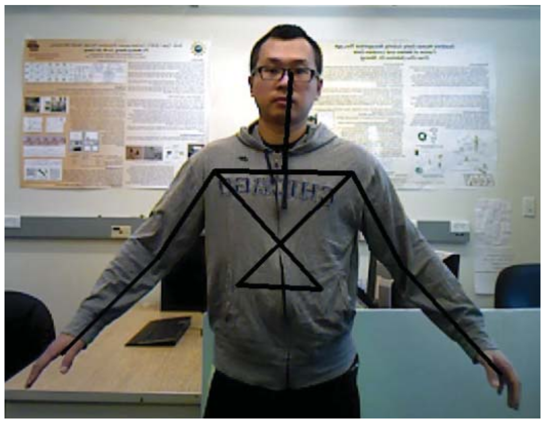

Figure 3.5: Human skeleton tracking.

object model feature points. This check is used to discard the less likely correspondences. Additionally, the pose of the recognized object is estimated, which is a rigid transformation between the point cloud models. To obtain a good recognition result, the position of the object should be in certain range (0.7-1.5 meters) with respect to the Xtion sensor.

3.3.2 HMMs for Low-level Action Recognition

Modeling the low-level dynamics of human motion is important for human motion recognition. It serves as a quantitative representation of simple movements so that they can be recognized in a reduced space by using the motion trajectories [85]. We propose an HMM based approach that does not require explicit motion segmentation to perform real-time action spotting and classification from continuous user motion. The HMM approach to action recognition is motivated by the successful application of HMM techniques to speech recognition problems. Each action is characterized by an HMM.

Each model is trained with 20 sets of training data at a sampling rate of 20 Hz. The training data is a sequence of right arm joint angles which can be accessed through the Openni [86] driver and the skeleton tracker function as shown in Fig. 3.5. All the actions are conducted using right arm. Therefore, four joint angles from two right arm joints are considered: the roll and yaw angles of the right elbow, the roll and pitch angles of the right shoulder. The number of state in each HMM is 10. The number of observation symbol is 8. They are set by heuristic method.

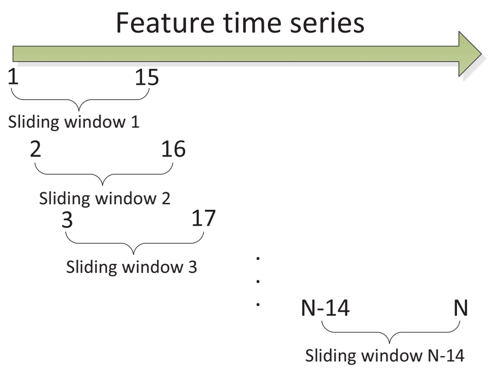

Figure 3.6: Data segmentation in recognition phase.

In the training phase, there are four steps:

- Step 1: Segment the training set.
- Step 2: Quantify the vectors into observation symbols using the K-means clustering [87].
- Step 3: Set up the initial HMM parameters.
- Step 4: Parameter estimation using the EM method [88].

In the recognition phase, as shown in Fig. 3.6, the testing data is segmented using a fixed sliding window of size 15. The step size for the sliding window is 1, therefore, no data is missed. The probability of having the observation sequence given the model $P(\psi_a|\lambda_j)$ is computed for each sliding window. ψ_a is a sequence of feature vector. λ_j is the HMM model for action A_j. Solving this problem allows us to choose the model which best matches the observations. For further details about the low-level HMM please see our previous work [89].

3.3.3 Bayesian Network for Modeling Object/action Dependency

The object/action dependency is modeled by a Bayesian network shown in the lower part of Fig. 3.7. The object which is selected for manipulation is denoted by $O_s \in O$. The manipulating action is denoted by A_j. ψ_o and ψ_a are the observation of the selected object and the observation of the action, respectively, where $\psi_o \in \Psi_o$ and $\psi_a \in \Psi_a$. At this level, the goal is to find A_m which

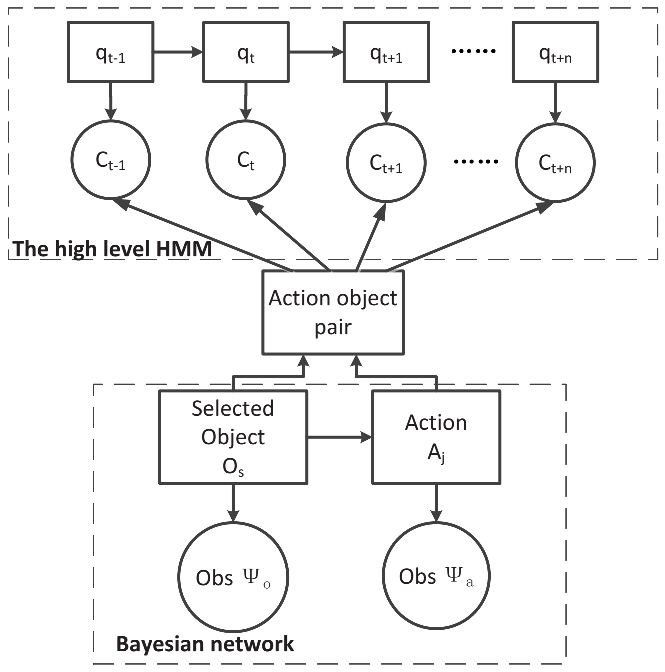

Figure 3.7: The hierarchical probabilistic model.

generates the maximum posterior likelihood (MAP) estimation,

$$A_m = arg\ \max_{A_j} P(A_j|\psi_o, \psi_a) \tag{3.2}$$

According to Bayesian rule,

$$P(A_j|\psi_o, \psi_a) \propto P(\psi_a|A_j, \psi_o) \cdot P(A_j|\psi_o) \tag{3.3}$$

As shown in the Bayesian model, we made the assumption that ψ_a is independent of ψ_o given A_j. Therefore, $P(\psi_a|A_j, \psi_o) = P(\psi_a|A_j)$. $P(\psi_a|A_j)$ is interpreted as $P(\psi_a|\lambda_j)$ which is the output the low-level HMMs. Applying the total probability theorem, we have

$$P(A_j|\psi_o) = \sum_i P(A_j|O_i, \psi_o) \cdot P(O_i|\psi_o) \tag{3.4}$$

$P(A_j|O_i, \psi_o) = P(A_j|O_i)$, since A_j and ψ_o are independent given O_i. $P(A_j|O_i)$ is the prior probability characterizing the object/action dependency, which can be calculated based on the occurrence of A_j given O_i in the training set. On the other hand, $P(O_i|\psi_o)$ is the object classification results in terms of probability.

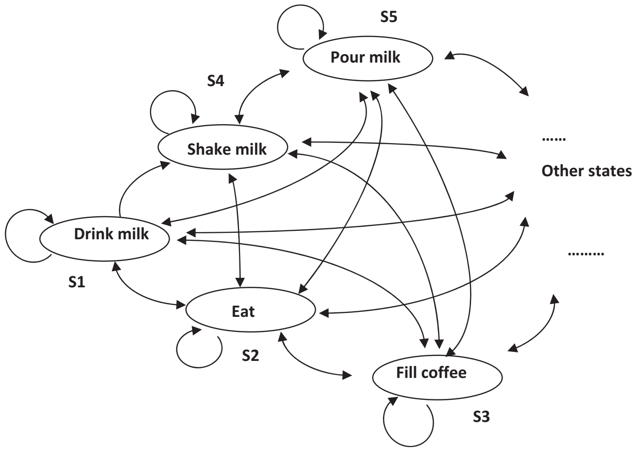

Figure 3.8: The constraints in the upper level HMM.

3.3.4 HMM Modeling of Sequential Constraints

In most human daily activities, actions usually follow certain patterns. For example, “drink milk” is more likely to happen after “pour milk” than “shake milk”. Such sequential constraints can be modeled by a high-level HMM and will be used to improve the action recognition accuracy.

The structure of the proposed high-level HMM is shown in the upper block of Fig. 3.7. In the model, the state is defined as the combination of action and object as $Action(Object)$. For example, “drink(cup)” and “drink(milk)” are two different states in the upper level HMM. The observations are the outputs from the Bayesian network. The parameters of the model $\lambda_h(A_h, B_h, \pi_h)$ are obtained from the observed patterns of the experimenter, which may be different from person to person. The transitional matrix A_h is estimated from the observed action sequence. The observation symbol probability distribution matrix B_h is the accuracy matrix of each action/object pair given the decision from the Bayesian network. The sequential constraint is shown in Fig. 3.8. The goal is to find the optimal state sequence associated with the given observation sequence.

Table 3.1: Object and action dependencies. Darker color represents lower probability.

Object	Action type					
	drink	pour	stir	eat	shake	fill
milk bottle	□	□	░	■	░	■
cup	□	░	□	■	▒	■
bowl	▒	▒	░	□	▓	■
coffee can	■	■	■	■	▓	□
cereal box	■	■	■	▓	▓	□

3.4 Experiments and Results

In this section, we first explain the experiment procedures. Then, we analyze the experimental results.

3.4.1 Experiment Procedure

Scenario

A breakfast scenario as shown in Fig. 3.1 is used to validate our theoretical framework. The procedure consists of two parts: preparing breakfast and having breakfast. Five objects are used, O ={ “coffee can”, “milk box”, “cup” , “bowl”, “cereal box”}. Six different actions are defined, A ={“drink”, “pour”, “stir”, “eat”, “shake”, “fill”}. Currently we only focus on right arm movement. However, it is very convenient to extend to any joints of interest. At any time, we assume only one object is manipulated. The training data is collected from one subject while the model is tested by 4 subjects including the trainer.

Training

Each level of the probabilistic model needs training. For the low-level HMMs, each model is trained with fifteen sets of training data at a sampling rate of 20 Hz. A rule-based method is adopted for data segmentation. The starting pose is defined, and each training data set consists of twenty data points after the starting point. For the Bayesian model, each object has certain probabilities regarding the actions that can be applied on it. The probability is estimated using the ratio of the number of each action on the object over the number of all the actions on the object respectively. Table 3.1 presents the probability of actions applied on each object. The color from light to dark indicates the probability from high to low. For the high-level HMM, the transition matrix is obtained from the training action sequence.

Testing

The trained models are tested in real-time. The object recognition is affected by occlusion and shadowing caused by the human hand. However, the actions are not affected by the objects, since we are focusing on the arm movement and the human skeleton tracker is robust to minor occlusion. The action recognition follows the procedures shown in Fig. 3.9. The objects on the table are recognized first. Based on the relative position between each object and human's hand, the object to be manipulated can be determined. Then, the low-level HMM based action recognition will be triggered. It will be combined with the object recognition to generate the decision from the Bayesian model. Finally, the Bayesian filtering generates the output based on the previous and the current Bayesian model outputs.

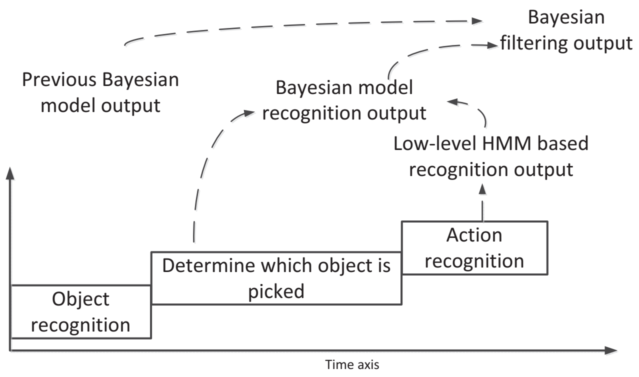
Bayesian filtering output
Previous Bayesian model output
Bayesian model recognition output
Low-level HMM based recognition output
Action recognition
Determine which object is picked
Object recognition
Time axis

Figure 3.9: Real time recognition procedure.

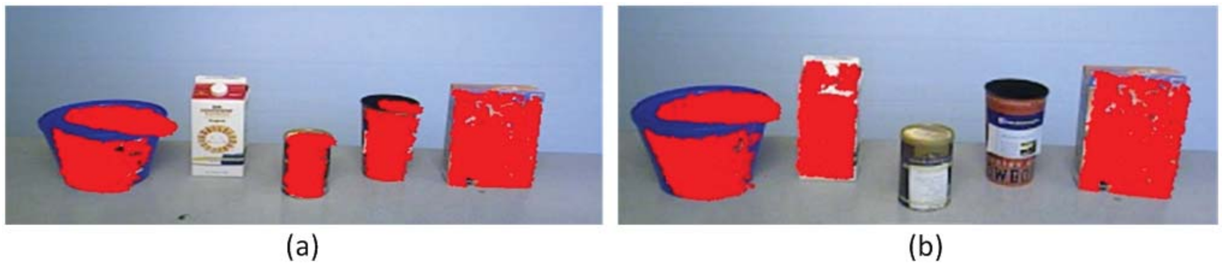

Figure 3.10: Object recognition results 1. The objects from left to right are bowl, milk box, coffee can, cup, cereal box. (a) bowl, coffee can, cup and cereal box have been recognized. (b) bowl, milk box, cereal box have been recognized.

3.4.2 Results

We first present the object recognition results. Figure 3.10 shows two snapshots of object recognition, in which different objects are recognized. As long as the object in the camera view is detected once, we can obtain the object type and pose. Only the object type is used as the object context. When the object moves fast, the recognition is not stable. Therefore, the objects are recognized when they are put on the table. On the other hand, the human's hand position can be calculated based on the human skeleton. According to the relative position of the human's hand and the object, the object to be manipulated can be determined. To verify the robustness of the object recognition algorithm, we put two objects with similar shape in the field of camera view, the coffee can 1 (in the middle) and coffee can 2 (on the right). The goal is to recognize coffee can 1. Fig .3.11 (b), Fig .3.11 (c) and Fig .3.11 (d) show that the algorithm can recognize the coffee can 1 even with occlusion and similar objects in the scene. The object recognition accuracy is shown in Table 3.2.

For action recognition, two offline experiments are designed to verify the proposed framework. To make the performance comparison of each level consistent, we only show the action type and do not show the object involved, since the low-level HMM does not consider the object information. In the first scenario, the subject manipulates each object multiple times. The purpose is to evaluate the accuracy of the Bayesian network decision. Parts of the testing results are shown in Fig. 3.12. It indicates that the low-level HMMs cannot distinguish action “fill”, “stir” and “pour”

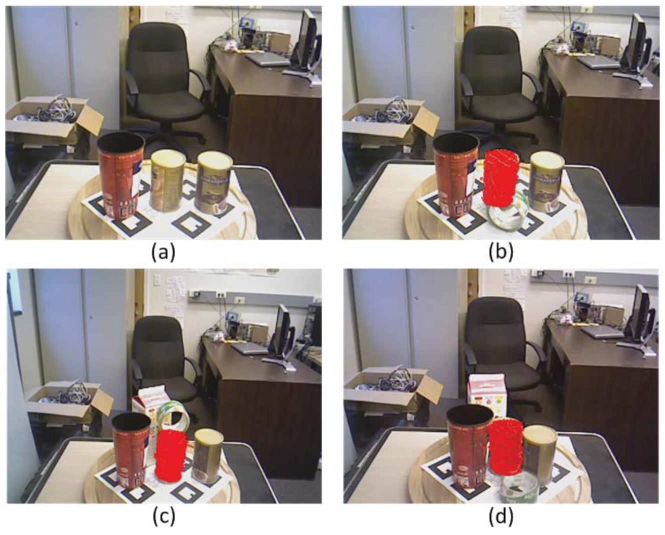

Figure 3.11: Object recognition results 2. (a) Camera view. The goal is to recognize the coffee can in the middle of the three objects. (b), (c) Recognition result with occlusion and similar object in the scene. (d) Recognition result in cluttered scene.

from each other, since they have similar motion features. Therefore, with the motion features alone, the recognition performance is poor. Table 3.3 shows the accuracy of the low-level HMMs. However, most of these similar actions can be differentiated from each other effectively by considering the object involved. This is because the conditional probability $P(action|object)$ can be used to distinguish these actions. For example, $P(fill|milk\,box) = 0, P(stir|milk\,box) = 0.05$, while $P(pour|milk\ box) = 0.6$. The comparison between the low-level HMM output and Bayesian network output are shown in Fig. 3.12. Table 3.4 shows the recognition results when the object types are considered.

However, there are still misclassification between action “stir” and “pour”. The reason is that according to Table 3.1 which presents the occurrence of action given the object. Action “stir” and action “pour” are all possible for object “cup”. The conditional probability $P(stir|cup)$ and $P(pour|cup)$ are not distinctive enough from each other. Therefore, the Bayesian model is unable to distinguish these two similar actions effectively. The Bayesian filtering approach can solve this problem because for the object “cup”, the action “pour” is more likely to happen after “drink” than “stir”. Fig. 3.13 shows that, the first “pour” action is recognized as “stir” falsely by the

Table 3.2: The confusion matrix of object recognition

test type	decision type					accuracy
	milk box	cup	bowl	coffee can	cereal box	
milk box	**0.95**	0.05	0	0	0	0.95
cup	0.02	**0.98**	0	0	0	0.98
bowl	0	0	**1.00**	0	0	1.00
coffee can	0	0.02	0	**0.98**	0	0.98
cereal box	0	0	0	0	**1.00**	1.00

Table 3.3: Recognition accuracy of the low-level HMMs

test type	decision type							accuracy
	drink	pour	stir	eat	shake	fill	not detected	
drink	**0.85**	0.04	0	0.06	0	0	0.05	**0.85**
pour	0	**0.56**	0.22	0	0	0.12	0.10	**0.56**
stir	0	0.28	**0.52**	0	0	0.12	0.08	**0.52**
eat	0.04	0	0.10	**0.76**	0	0	0.10	**0.76**
shake	0	0	0	0	**0.90**	0	0.10	**0.90**
fill	0	0.44	0.12	0	0	**0.32**	0.12	**0.32**

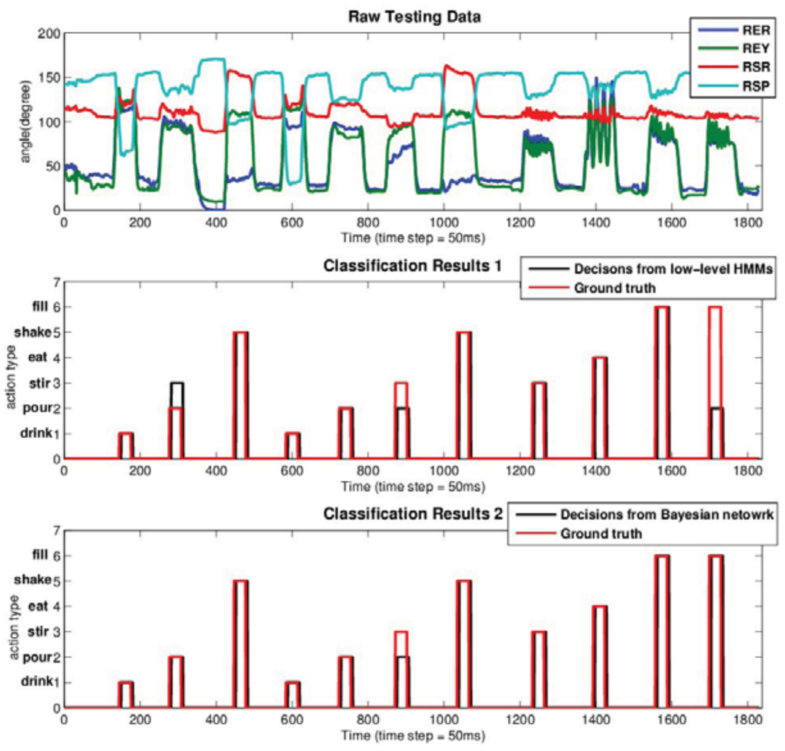

Figure 3.12: Action recognition results with object constraints. Results of 11 actions are shown here. The Bayesian network output is action object pair. The object type is not shown in the figure. The legend RER,REY,RSR,RSP represent the angle of right elbow roll, right elbow yaw, right shoulder roll and right shoulder pitch respectively.

Table 3.4: Recognition accuracy of the Bayesian model

test type	decision type							accuracy
	drink	pour	stir	eat	shake	fill	not detected	
drink	**0.86**	0.04	0	0.06	0	0	0.04	**0.86**
pour	0	**0.54**	0.33	0	0	0.05	0.08	**0.54**
stir	0	0.26	**0.62**	0.04	0	0.02	0.06	**0.62**
eat	0.04	0	0.08	**0.78**	0	0	0.10	**0.78**
shake	0	0	0	0	**0.90**	0	0.10	**0.90**
fill	0	0.04	0.02	0	0	**0.84**	0.10	**0.84**

Bayesian model. However, it is not corrected, because its previous action is “shake”. The sequential constraint between these two actions is not tight. The second wrong decision is corrected

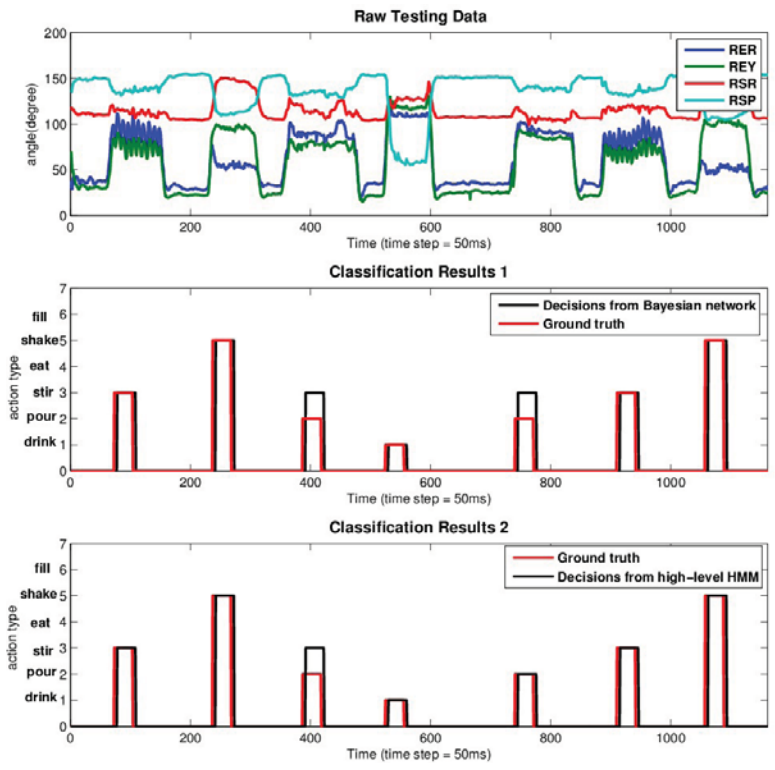

Figure 3.13: Action recognition results with object constraints and sequential constraints. Results of 7 actions are shown here. The high-level output is action object pair. The object is cup. The legend RER,REY,RSR,RSP represent the angle of right elbow roll, right elbow yaw, right shoulder roll and right shoulder pitch respectively.

since it happened after action “pour”, which has strong sequential constraints with action “pour”. Table 3.5 shows the accuracy of the high-level HMM. We also conducted real time experiments. In this experiment, the subject does a series of actions during breakfast. The result indicates that the low-level HMM decision is poor when the action is very similar to some other actions. On the other hand, the high-level HMM can correct wrong decisions from the Bayesian models effectively. Therefore, the results of the real-time experiment are consistent with the offline results.

The action recognition performance is highly dependent on the skeleton tracking. Therefore, three subjects with different heights and weights were asked to verify the robustness of the system. Each of them was asked to do breakfast routine several times until enough actions were collected for each action type. The profiles of the subjects are show in Table. 3.6. Subject 1 is the trainer.

Table 3.5: Recognition accuracy of the high-level HMM

test type	decision type							accuracy
	drink	pour	stir	eat	shake	fill	not detected	
drink	**0.86**	0.02	0	0.06	0	0	0.04	**0.86**
pour	0	**0.85**	0.07	0	0	0	0.08	**0.85**
stir	0	0.04	**0.88**	0.04	0	0	0.04	**0.88**
eat	0.04	0	0.05	**0.85**	0	0	0.06	**0.85**
shake	0	0	0	0	**0.92**	0	0.08	**0.92**
fill	0	0.04	0.02	0	0	**0.86**	0.08	**0.86**

Table 3.6: The profiles of the subjects

subject No.	sex	height(cm)	weight(pounds)
1	male	177	180
2	female	155	100
3	male	185	190
4	male	170	160

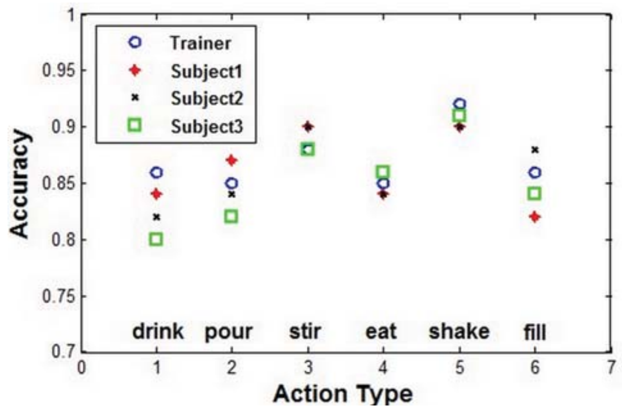

Figure 3.14: Recognition performance on different subjects.

The comparison results are shown in Fig. 3.14. It shows that the performance is stable when the algorithm is tested on different subjects.

We also test the action recognition algorithm on the MSRDailyActivity 3D dataset [93]. The

Table 3.7: Recognition Accuracy Comparison for MSRDailyActivity 3D dataset

Method	Accuracy
Dynamic Temporal Warping [90]	0.54
Actionlet Ensemble [91]	0.857
DCSF + joint positions [92]	0.882
Proposed Method	0.87

dataset was captured using a Kinect. There are totally 16 activities. Each action was performed by ten subjects twice. We extract the joint angles of both arms from the skeleton joint data. There are totally four joint angles: elbow roll and yaw of both arms, shoulder roll and pitch of both arms. Most of the actions involve object. However, the dataset does not include object information. Therefore, object context and sequential constraints are not considered here. We use half of the data for training and the other half for testing. The average accuracy of our algorithm is compared with that of some existing approaches, as shown in Table. 3.7. The performance of our algorithm is among the best.

3.5 Summary

Recognition of human actions using a vision based approach is a challenging task if the actions have similar motion features. This chapter investigates how to improve the performance of manipulative action recognition through a RGB-D sensor. A human body can be represented as an articulated system of body segments connected by joints. Based on that human motion can be treated as a temporal evolution of the spatial configuration of these body segments. Therefore, if we can reliably extract and track the human skeleton, action recognition can be performed by classifying the temporal evolution of human skeleton. With the emergence of low-cost RGB-D sensors, a skeleton of a person can be easily obtained. These sensors provide 3D depth data of the scene, which is robust to illumination changes and offers more useful information to recover 3D human skeletons.

A multi-level probabilistic model is proposed as a solution to the action recognition problem. In the low-level, human motion is modeled using an HMM. The joint angle sequence is used as the motion feature. In the training phase, the training data is segmented into vectors. Before they are fed to HMMs, these vectors are quantified into discrete observation symbols. Then HMM parameters are estimated using the EM method iteratively. In the recognition phase, the testing data is input to each HMM. The HMM that outputs the maximum probability is chosen as the decision. In the mid-level, the dependencies between object and action are modeled in a Bayesian model, actions which are either too subtle to perceive or too similar to discriminate can be recognized with high accuracy. The idea is that the object type can be used as a prior knowledge for action recognition, for example, if a cup is held in a hand, actions like "pouring" or "drinking" are more likely to happen than "eating". In the high-level, with the action sequential constraints, similar actions on the same object can be differentiated effectively. Because in most human daily activities, actions usually follow certain patterns.

Both online and offline experimental results verify that our framework outperforms the approaches that use motion feature only. The experiment on subjects with different body shapes show the robustness of the system. The accuracy test using the MSRDailyActivity 3D dataset proves that the performance of our algorithm is comparable to that of the existing work. However, we use much simpler features. In this work, the object recognition helps action recognition while the future work is to exploit the mutual dependency so that object recognition and action recognition can help each other.

Chapter 4

Fine Manipulative Action Recognition through Sensor Fusion

In this chapter, we extend our previous work to recognize fine manipulative actions. Multiple sensors are adopted to completely capture the related features. Information fusion methods are designed to generate more accurate decisions. Section 4.1 introduces the motivation of the proposed framework. In Section 4.2, the problem to be solved is formulated. Section 4.3 presents the proposed methodology. Section 4.4 describes the experiment procedures and gives the experimental results.

4.1 Motivation

Nowadays, the demand for intelligent robots is not limited to repetitive tasks. Teaching a robot delicate assembly skills through human demonstration can avoid lengthy robot programming, while little technical expertise is required for the operator. To fulfill this goal, one of the fundamental problems is to understand complex human assembly actions. Assembly actions may involve arm and finger motions and interactions with the manipulated objects. For example, hammering involves obvious arm and wrist movement while minor finger motion. Screwdriving is mainly about finger rolls. Drilling only involves finger pressure on the buttons. To capture manip-

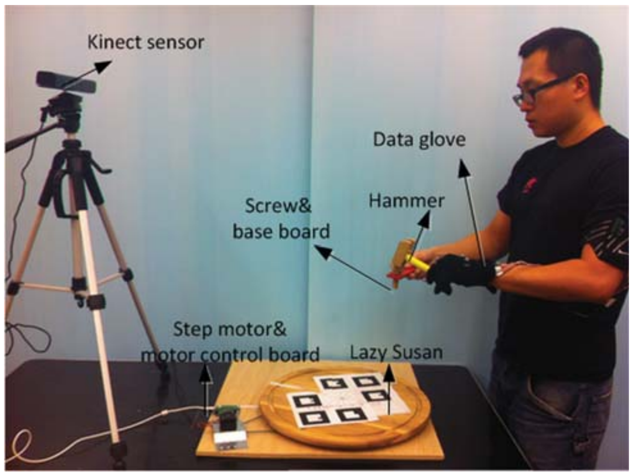

Figure 4.1: The setup of the Portable Assembly Demonstration (PAD) system.

ulative actions in assembly tasks, traditionally vision or motion sensors are used [94, 95]. However, recognizing manipulative actions using vision-based methods has several difficulties. First, the interaction between the human hand and the object, such as the force applied can not be captured. Second, tracking human hands is difficult since occlusions between the objects and hands occur frequently. On the other hand, wearable sensors can offer stable information about human arm and finger motion. However, they cannot capture the information regarding the object. Therefore, it is desired that sensors with multiple modalities are used to recognize assembly actions. In this work, three kinds of sensors are used to fully capture the fine manipulative actions during assembly tasks. The captured information includes human motion, object geometry and visual appearance, and human/object force interaction.

4.2 Problem Statement

In this chapter, by adopting multimodal sensors, we aim to solve the fine manipulative action recognition problem. When equipped with this capability, a robot can better understand human skills. Assembly tasks are chosen to evaluate our proposed algorithms.

4.2.1 System Setup

The hardware platform as shown in Fig. 4.1 consists of a Kinect sensor on a tripod, a motor-driven Lazy Susan, a data glove that senses finger joint angles and forces, and a computer for data processing (not in the picture). The Kinect sensor is used to capture the information about the objects and the human skeleton motion. As a low cost RGB-D camera, the Kinect has been widely used in many robotic applications for object recognition and human robot interaction [40]. The Kinect has three operation modes: 1) template extraction mode, 2) object recognition mode and 3) human tracking mode. To extract object templates, the object is put at the center of the Lazy Susan. As the motor rotates, object templates (the points above the Lazy Suan) from different angles are extracted. They will be used later for object recognition. To scan and recognize objects, the Kinect faces downward 45 degree at the objects on the Lazy Susan. To track the human skeleton, the Kinect looks horizontally at the demonstrator. The pose of the Kinect can be controlled through the motor embedded in it.

The data glove is used to measure the finger joint angles and finger forces. Due to the lack of a commercial data glove that measures finger joint angles such as the CyberGlove [96], we use the Vicon motion capture system to track markers attached to the glove fingers. The motion capture system has an accuracy of 0.7 mm. To capture finger motion, the hand model needs to be created first. Instead of building a full hand model, we only model the thumb and index fingers, which is sufficient to characterize the hand motion in our assembly task. The finger model created is shown in Fig. 4.2 which captures wrist angles, index finger angles and thumb finger angles.

Assembly tasks always involve forces applied by hands. Therefore, force is also an important clue for action recognition. To measure the finger forces, the FlexiForce sensors [97] are used. It can detect and measure a relative change in force or applied load. The range of this sensor is from 0 to 1 lb. An XBee module [98] samples 4 channels and sends the result to the receiver. We attach a FlexiForce sensor to each finger (except the pinkie finger) through a rubber inner glove as shown in Fig. 4.3.

For the software, all the programs run in Robot Operating System (ROS) [81] in Linux. The overall flowchart is shown in Fig. 4.4. There are totally three kinds of node: three information

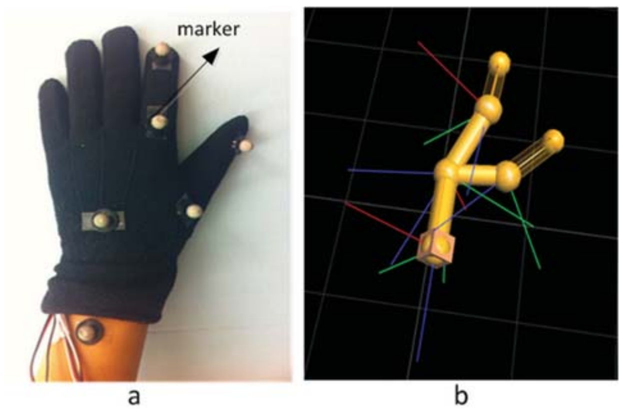

Figure 4.2: Finger model created. (a) the placement of the markers. (b) finger model created.

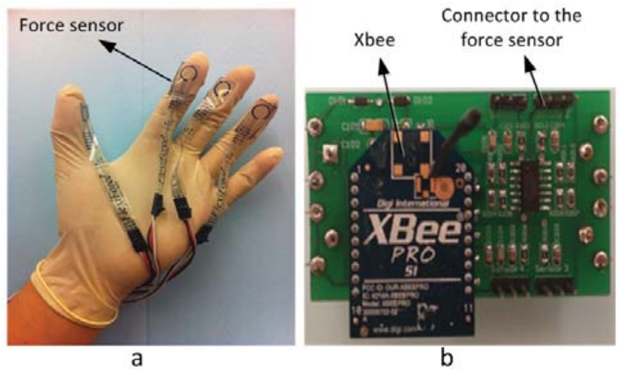

Figure 4.3: Force sensor setup. (a) the force sensors on the finger tips. (b) the Xbee module for wireless communication.

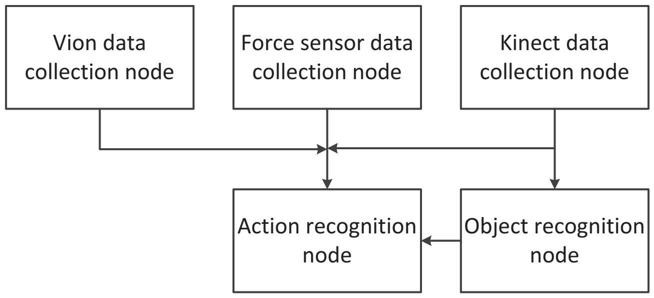
Vion data collection node
Force sensor data collection node
Kinect data collection node
Action recognition node
Object recognition node

Figure 4.4: The flowchart of the action recognition system.

collection nodes, object recognition node and action recognition node. The three information collection nodes receive the data from the force sensor, Vicon system and Kinect sensor respectively. The message filter in ROS synchronizes the multiple data sources. The object recognition node is for recognizing the parts and tools involved. The action recognition node considers both the multiple sensor data and the object recognition node output for improved accuracy.

4.2.2 Formulation of the Problem

Given the above setup, the input to the recognition system consists of two parts: the observation of the object and the observation of the manipulative action. The observation of the object ψ_o is the decision of the object recognition algorithm. The observation of the action ψ_a has three parts. The first part comes from the human skeleton tracked by the Kinect sensor which is shown in Fig. 3.5. $\{\Gamma_1, \Gamma_2\}$ are the left shoulder and left elbow joints. Four joint angles are extracted from these two joints. The second part is from the Vicon system, which is a sequence of finger and wrist joints angle data. The joints are index finger joint, thumb finger joint and the wrist. They are represented as $\{\alpha_1, \alpha_2, \alpha_3\}$ respectively. For each joint α_i, the roll, pitch and yaw angles are captured. The third part is the force data from the four fingers $\{\beta_1, \beta_2, \beta_3, \beta_4\}$. We assume the object set $O = \{O_1, O_2 ... O_m\}$, and the action set $A = \{A_1, A_2 ... A_n\}$. The problem is to develop an algorithm to decide on $A_i \in A$ given the above observation, or to find a mapping function f,

$$A_i = f(\psi_o, \psi_a) \tag{4.1}$$

where

$$\psi_a = \{\Gamma_1, \Gamma_2, \alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \beta_4\} \tag{4.2}$$

Each element in ψ_a is a temporal sequence.

4.3 Methodology

A probabilistic framework is proposed to recognize the fine manipulative actions. The framework is shown in Fig. 4.5. There are two levels in this framework. In the low level, hidden Markov models (HMMs) are employed to model the temporal variations of the actions. In the high level, a Bayesian model is used to capture object/action dependencies. The details about the HMMs are introduced in 3.3.2.

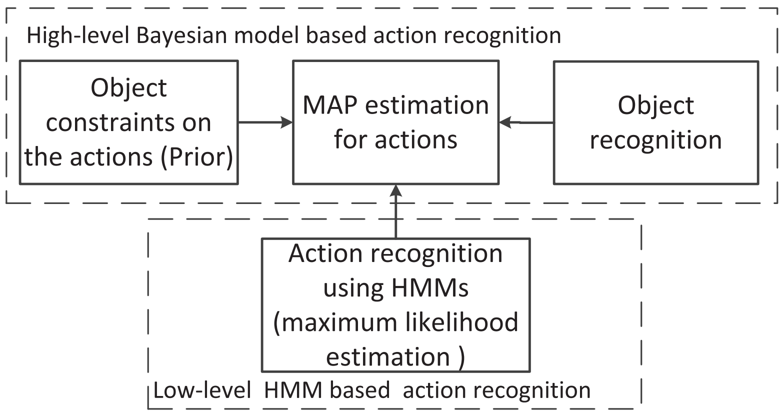
High-level Bayesian model based action recognition
Object constraints on the actions (Prior)
MAP estimation for actions
Object recognition
Action recognition using HMMs (maximum likelihood estimation)
Low-level HMM based action recognition

Figure 4.5: The two-level action recognition framework.

4.3.1 Object Recognition

With the emergence of RGB-D cameras, color and depth based object recognition has been receiving more attention recently. With RGB-D cameras, not only the object type but also its pose can be obtained efficiently, which is useful for object manipulation. By recognizing the parts to be assembled and the tools to be used, we can better recognize the assembly action. The challenge is that the size of the parts and tools are usually small and traditional 2D feature based recognition algorithms are not effective due to the limited number of features. Therefore we adopt a 3D feature based recognition algorithm.

To recognize objects, the 3D templates for each object will be created beforehand. For each object, we will capture multiple templates (2 for symmetric object and 4 for asymmetric object) from different camera views. To extract the point cloud of the object on the Lazy Susan, all the distant points will be filtered out first. Then the top surface of the Lazy Susan is extracted and all the points above that surface are considered as the points of the object.

3D object recognition is performed based on the recognition module in Point Cloud Library (PCL) [99]. The recognition process is as follows. First, from the point cloud of the whole scene, only points above the Lazy Susan are extracted which greatly decreases the number of points to be processed. It can also help improve the recognition accuracy since the false alarms can be prevented effectively. Second, the normals of each point in both the model point cloud and the scene point cloud are computed using the 10 nearest neighbors. Third, each point cloud is downsampled to find a small number of keypoints, which will then be associated to a 3D descriptor in order to perform keypoint matching and determine point-to-point correspondences. The density of the sampled point clouds is adjusted according to the size of the object. To extract enough features, higher sampling densities are required for small objects. According to [100], the SHOTCOLOR [101] descriptor, which is a color version of SHOT (Unique Signatures of Histograms) [102], has competitive performance among the 3D descriptors. Therefore we choose SHOTCOLOR as the descriptor. Fourth, point-to-point correspondences between model descriptors and scene descriptors need to be determined. The last step is to cluster the previously found correspondences. One correspondence grouping algorithm developed in [103] are used. The object recognition results gives both identity and pose of the recognized object. Only identity information is used in our algorithm.

4.3.2 Fusion Method

The feature-level fusion method is shown in Fig. 4.6-(a) which consists of feature selection, HMM-based classification and Bayesian filtering. In general, different sets of features characterize different manipulative actions. For example, finger forces vary during actions such as "screwing" and "fixing". The skeleton joint angles capture the motion of the actions like "hammering" and "wrenching". However, some of the features may be irrelevant and have to be removed, since the presence of irrelevant features requires more computational cost. In addition, it may reduce the classification accuracy. Algorithms used for selecting features fall into two categories: the wrapper methods and the filter methods [104]. The wrapper method uses cross-validation to predict the benefits of adding or removing a feature from the feature subset used. While the filter method does not rely on any knowledge of the algorithm to be used [105]. The wrapper method is computationally intensive. Since each new subset is used to train a model, which is tested on a hold-out set. Counting the number of mistakes made on that hold-out set gives the score for that subset. Therefore, we choose the filter method which uses a proxy measure instead of the error rate to score a feature subset. This measure is chosen to be fast to compute, whilst still capturing the usefulness of the feature set. Our feature selection approach is inspired by the variance preservation

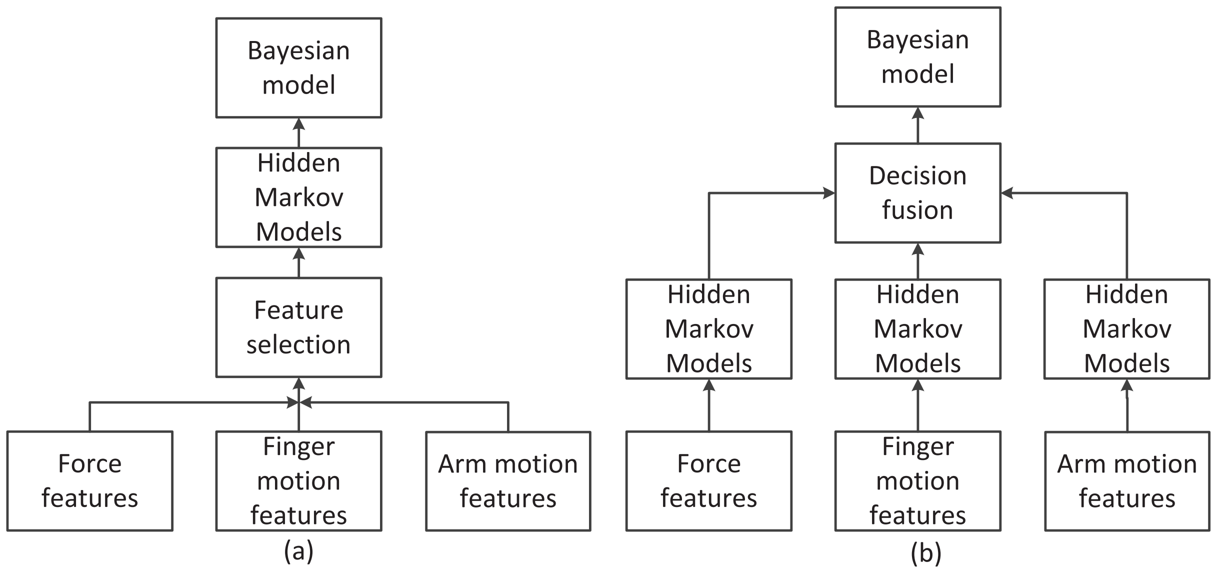

Figure 4.6: Fusion approaches. (a) feature-level fusion method. (b) decision-level fusion method.

algorithm [106], which selects a subset of features that preserve the variance contained in the data. Since the assembly action are all non-stationary actions. Therefore, the feature sequence which can characterize the action should show a decent amount of variance. The variance of each feature in the training set for each action is calculated. The range of the feature data is normalized to the same scale. The variance ratio ρ is used as the selection measurement

$$\rho_{ij} = v_{ij} / \sum_i (v_{ij}) \tag{4.3}$$

where v_{ij} is the variance of the *ith* feature in action *j*'s training data set. As long as ρ_{ij} is over a predefined threshold, which indicates it is a significant feature for the *jth* action, the *ith* feature will be kept. The recognition models will be trained using the features selected.

On the other hand, for the decision-level fusion method as shown in Fig. 4.6-(b), features are divided into different groups. Multiple classifiers are created using the different groups of features. To fuse the decision from each classifier, we design a measure which indicates the confidence of each decision. The confidence index is defined as

$$\zeta_j = maxP(\psi_a|\lambda_{ij}) / \sum_k (P(\psi_a|\lambda_{kj})) \tag{4.4}$$

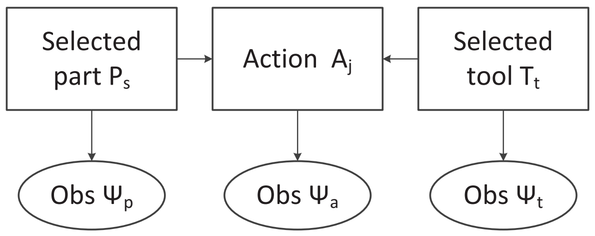

Figure 4.7: The Bayesian Model for action recognition.

where λ_{ij} is the HMM for action i using feature group j. ζ_j is the confidence index of the decision from classifier j. The decision of the classifier with the strongest confidence index ζ_s will be chosen.

$$arg\ \underset{i}{max} P(\psi_a|\lambda_{is}) \tag{4.5}$$

where λ_{is} denotes the type s HMM (with the strongest confidence) for action i.

4.3.3 Bayesian Network for Modeling Object/action Dependency

There are two kinds of objects involved in the assembly task: tools and parts. They both have correlation with the action, which can be modeled using a Bayesian model. Here A_j denotes a manipulative action. The Bayesian model is shown in Fig. 4.7. ψ_p is the decision of the part recognition. ψ_t is the decision of the tool recognition. where $\psi_p \in \Psi_p$, $\psi_t \in \Psi_t$ and $\psi_a \in \Psi_a$.Ψ_p and Ψ_t are the set of part and tool type respectively.

At this level, the goal is to find the maximum posterior likelihood (MAP) estimation, $arg\ \underset{A_j}{max} P(A_j|\psi_a, \psi_p, \psi_t)$. According to the Bayesian rule,

$$P(A_j|\psi_a, \psi_p, \psi_t) \propto P(\psi_a|A_j, \psi_p, \psi_t) \cdot P(A_j|\psi_p, \psi_t) \tag{4.6}$$

As shown in the Bayesian model, we made an assumption that ψ_a is independent of ψ_p and ψ_t given A_j. Therefore, $P(\psi_a|A_j, \psi_p, \psi_t) = P(\psi_a|A_j)$. $P(\psi_a|A_j)$ can be interpreted as $P(\psi_a|\lambda_j)$ which is the output of the low-level HMMs. λ_j is the HMM model for action A_j. Applying the total probability theorem, we have

$$\begin{aligned} &P(A_j|\psi_p, \psi_t) = \\ &\sum_m \sum_n P(A_j|P_m, T_n, \psi_p, \psi_t) \cdot P(P_m, T_n|\psi_p, \psi_t) \end{aligned} \tag{4.7}$$

where P_m, T_n is part m and tool n. A_j is independent of ψ_p, ψ_t given P_m, T_n. Therefore,

$$P(A_j|P_m, T_n, \psi_p, \psi_t) = P(A_j|P_m, T_n) \tag{4.8}$$

$P(A_j|P_m, T_n)$ is the prior probability characterizing the action that occurs on part P_m using tool T_n, which can be calculated based on the occurrence of A_j given P_m and T_n in the training set. On the other hand, we have

$$P(P_m, T_n|\psi_p, \psi_t) = P(P_m|\psi_p) \cdot P(T_n|\psi_t) \tag{4.9}$$

$P(P_m|\psi_p)$ is the part classification accuracy and $P(T_n|\psi_t)$ is the tool classification accuracy.

4.4 Experiments and Results

In this section, we first introduce the experiment setup. Then, we analyze the experimental results.

4.4.1 Experiment Setup

An assembly scenario as shown in Fig. 4.1 is used to validate our theoretical framework. Fig. 4.8 captures all the parts and tools. The action, tool and part sets are shown in Table. 4.1. Currently the force sensors and markers are attached to left arm and hand. However, it is straightforward to extend them to both arms. At each time, we assume only one part is manipulated. The training data is collected from one subject while the model is tested on 5 subjects including the trainer.

Table 4.1: The action, tool and part sets.

No.	1	2	3	4
Action	hammering	screwing	wrenching	sawing
Tool	hammer	screw driver	wrench	saw
Part	screw	red base board	blue base board	bolt
No.	5	6	7	
Action	drilling	rasping	fixing	
Tool	drill	wood planer	G clamp	
Part	N/A	N/A	N/A	

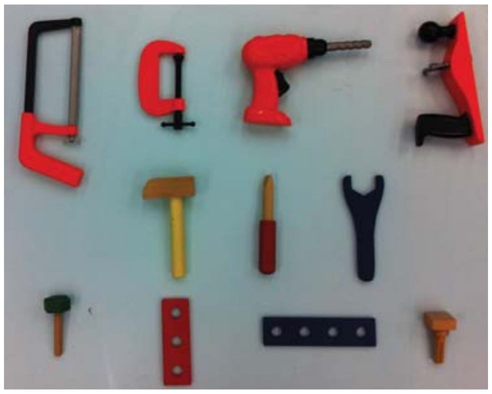
Figure 4.8: The parts and tools. From left to right, the top row shows saw, G clamp, drill, and wood planner. The middle row shows hammer, screw driver, and wrench. The bottom row shows screw, red base board, long base board, and bolt.

4.4.2 System Evaluation

Object Recognition

We evaluate the robustness of our approach under occlusion and in cluttered scenes. Fig. 4.9 (a) shows that with minor occlusions, the red base board can still be recognized. Fig. 4.9 (b) shows that our algorithm works correctly in cluttered scenes. The two red blocks in the cluttered scene which are similar to the red base board are not misclassified as the red base board. The recognition accuracy for each part and tool is shown in Table. 4.2.

Table 4.2: The object recognition accuracy.

Object	saw	G clamp	drill	wood planner
Accuracy	0.92	0.93	0.95	0.92
Object	hammer	screw driver	wrench	screw
Accuracy	0.97	0.87	0.88	0.85
Object	red base board	blue based board	bolt	
Accuracy	0.86	0.87	0.89	

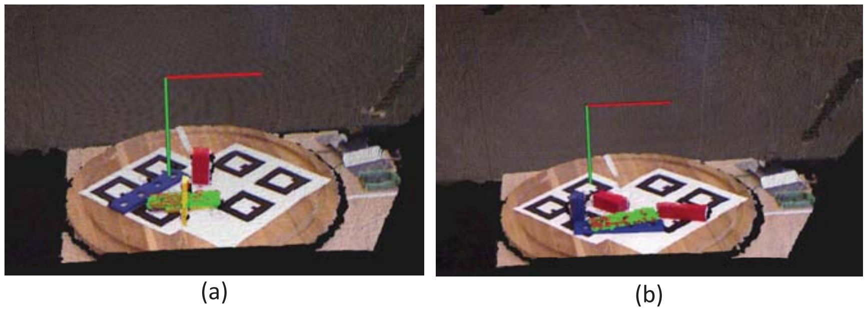

Figure 4.9: Results of board recognition, where the recognized objects are highlighted in green. (a) recognition of the red base board with occlusion from a bolt. (b) recognition of the red base board in a cluttered scene.

Table 4.3: The feature selection output

Feature / Action	relevant features			
hammering	elbow roll	elbow yaw	-	-
screwing	index finger roll	thumb finger roll	thumb finger force	index finger force
wrenching	shoulder roll	elbow roll	elbow yaw	-
sawing	elbow roll	elbow yaw	shoulder pitch	-
drilling	index finger force	index finger yaw	-	-
rasping	elbow roll	elbow yaw	shoulder pitch	-
fixing	index finger roll	thumb yaw	-	-

Action Recognition

The action recognition algorithm is evaluated using real-time experiments. The subject is asked to do each action 100 times. For the feature-level fusion method, the selected features for each

Table 4.4: The list of recognition approaches.

Type No.	**Classification approach**
1	arm motion feature based HMMs
2	finger motion feature based HMMs
3	force feature based HMMs
4	selected feature based HMMs
5	Decision-level fusion method + Bayesian model
6	Feature-level fusion method + Bayesian model

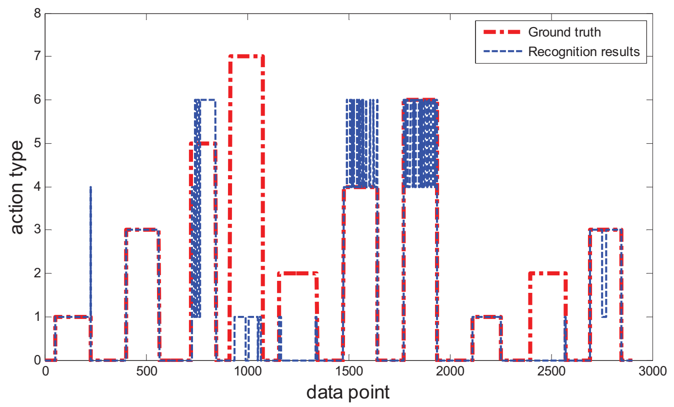

Figure 4.10: Recognition results of the HMMs using left arm motion features.

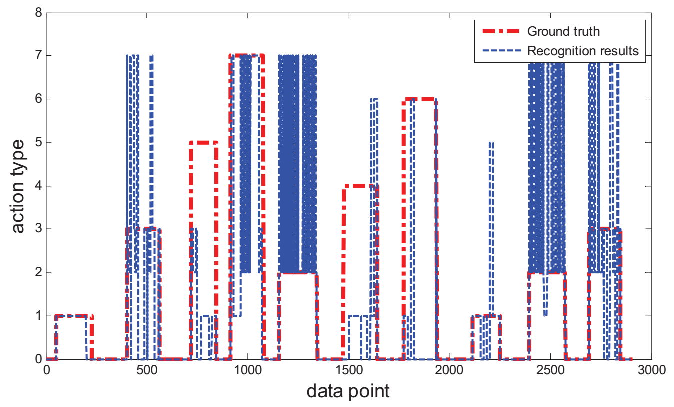

Figure 4.11: Recognition results of the HMMs using left finger motion features.

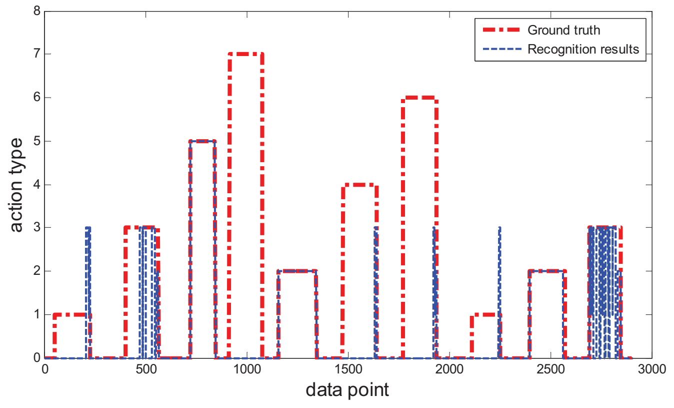

Figure 4.12: Recognition results of the HMMs using left finger force features.

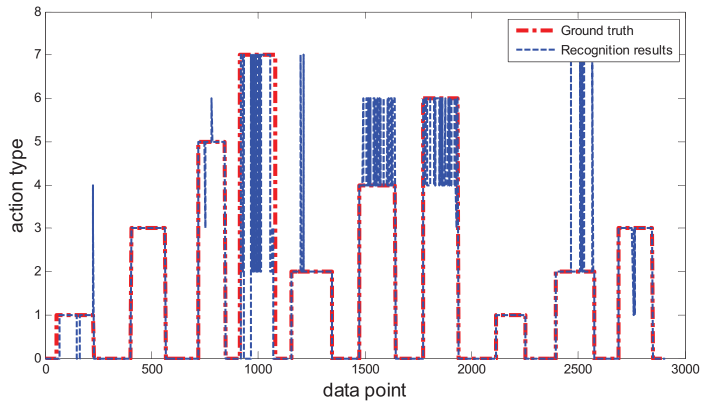

Figure 4.13: Recognition results of the decision-level fusion method.

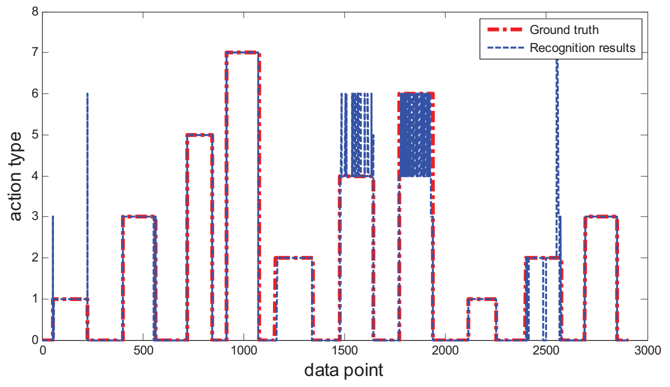

Figure 4.14: Recognition results of the feature-level fusion method.

action are listed in Table. 4.3. They are all considered for each HMM model. In addition to the

HMMs used by the feature-level and decision-level fusion methods, three types of HMMs that use a subgroup of features are also implemented for performance comparison purpose. They are force-feature based, finger-motion-feature based, and arm-motion-feature based HMMs. All the algorithms are listed in Table. 4.4.

Fig. 4.10, 4.11, 4.12 show the partial (10 out of 100 actions) recognition results of the HMMs using three kinds of features. The Type 1 HMMs can recognize the manipulative actions "hammering" (1) and "wrenching" (3) accurately. However, it fails to distinguish between the action "sawing" (4) and "rasping" (6). The reason is that the arm movement of these two actions are very similar. For the other actions which do not involve major arm movement, the Type 1 HMMs does not perform well. Similarly, Type 2 HMMs can recognize action "screwing" (2) and"fixing" (7) which involve obvious finger movement. However, the finger motion pattern of these two actions is similar which causes misclassification as shown in Fig. 4.11. However, the force applied during "screwing" and "fixing" is different. The Type 3 HMMs can recognize "screwing" (2) quite well. The action "drilling" (5) does not have obvious arm or finger movement, but the index finger tip force variation during pushing and releasing the button can be recognized by Type 3 HMMs. Overall, different types of HMMs have their own advantages on recognizing different actions. Therefore, the combination of all these HMMs should improve the recognition performance.

Fig. 4.13 shows that the output of the decision-level fusion outperforms each individual HMM. Despite minor errors, it can recognize most of the actions accurately. However, confusion between similar actions still exists. Action "sawing" (4) and "rasping" (6) can not be distinguished since none of the classifier can recognize them correctly. Neither does action "fixing" (7). In contrast, the feature-level fusion method performs well on recognizing action "fixing" (7) as shown in Fig. 4.14. The reason is that the combination of the selected features makes the action "fixing" (7) distinguishable. However, misclassification between similar actions still exists. Action "sawing" (4) and "rasping" (6) can not be distinguished since none of the classifiers can recognize them correctly. Compared to the decision-level fusion method, the feature-level fusion method does not involve feature grouping (by hand). Also, one HMM is built for each action instead of three. Therefore, computationally, it is more efficient than the decision-level method.

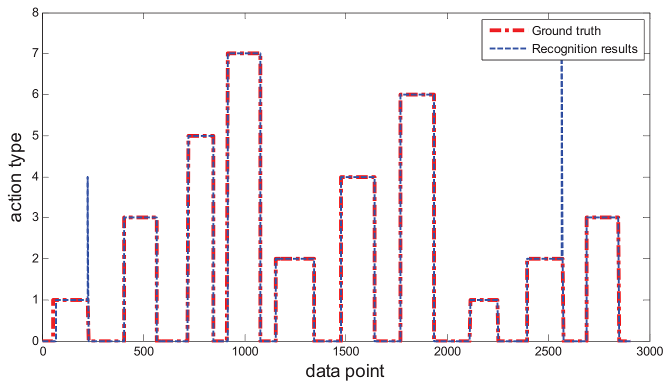

Figure 4.15: Recognition results of the decision-level fusion method + Bayesian model.

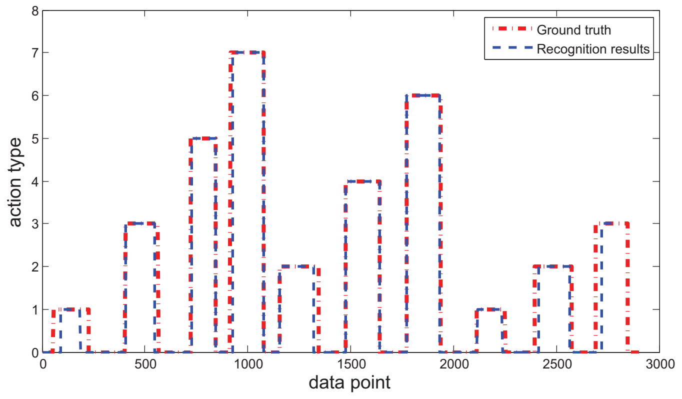

Figure 4.16: Recognition results of the feature-level fusion method + Bayesian model.

Finally, we show the results of the Bayesian model outputs, which adds the object/action dependencies on top of the two fusion methods. Fig. 4.15 and 4.16 shows that the Bayesian model can eliminate the confusion between action "sawing" (4) and "rasping" (6). In addition, it can further improve the accuracy. The recognition accuracy of the decision-level, feature-level fusion methods and the Bayesian model are shown in Table. 4.5, 4.6, 4.7 respectively.

Four additional subjects with different heights and weights are asked to verify the robustness of the system. Each of them was asked to repeat 80 times for each action type. The profiles of the subjects are shown in Table. 4.8. The training data is collected from Subject 1. The Bayesian model outputs are shown in Fig. 4.17. It shows that the performance is stable when the algorithm is tested on different subjects.

Table 4.5: Recognition accuracy of the decision-level fusion method

test type	decision type								accuracy
	hammer	screw	wrench	saw	drill	rasp	fix	missed	
hammer	**0.88**	-	-	0.05	-	0.02	-	0.05	**0.88**
screw	-	**0.80**	-	-	-	-	0.15	0.05	**0.80**
wrench	0.02	-	**0.94**	-	-	-	-	0.04	**0.94**
saw	-	-	-	**0.53**	-	0.45	-	0.02	**0.56**
drill	-	-	0.02	-	**0.90**	0.03	-	0.05	**0.90**
rasp	-	-	-	0.51	-	**0.48**	-	0.01	**0.48**
fix	-	0.46	-	-	-	-	**0.50**	0.04	**0.50**

Table 4.6: Recognition accuracy of the feature-level model

Test type	Decision type								Accuracy
	hammer	screw	wrench	saw	drill	rasp	fix	missed	
hammer	**0.85**	-	0.05	0.06	-	0.02	-	0.02	**0.85**
screw	-	**0.90**	0.01	-	-	-	0.07	0.02	**0.90**
wrench	-	-	**0.95**	-	-	-	-	0.05	**0.95**
saw	-	-	-	**0.56**	-	0.40	-	0.04	**0.56**
drill	-	-	-	-	**0.92**	-	-	0.08	**0.92**
rasp	-	-	-	0.47	-	**0.51**	-	0.02	**0.51**
fix	-	0.06	-	-	-	-	**0.88**	0.06	**0.88**

Table 4.7: Recognition accuracy of the Bayesian model

Test type	Decision type								Accuracy
	hammer	screw	wrench	saw	drill	rasp	fix	missed	
hammer	**0.94**	-	-	0.02	-	0.02	-	0.02	**0.94**
screw	-	**0.95**	-	-	-	-	0.02	0.03	**0.95**
wrench	-	-	**0.98**	-	-	-	-	0.02	**0.98**
saw	-	-	-	**0.96**	-	0.02	-	0.02	**0.96**
drill	-	-	-	-	**0.95**	-	-	0.05	**0.95**
rasp	-	-	-	0.04	-	**0.92**	-	0.04	**0.94**
fix	-	0.02	-	-	-	-	**0.92**	0.06	**0.92**

Table 4.8: The profiles of the subjects

Subject No.	**Sex**	**Height(cm)**	**Weight(pounds)**
1	male	177	180
2	female	155	100
3	male	185	190
4	male	168	140
5	male	177	170

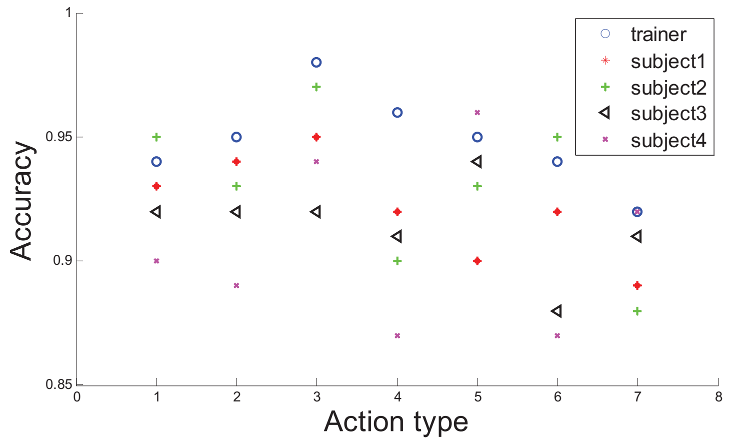

Figure 4.17: Recognition performance on different subjects.

4.5 Summary

To understand human demonstration, manipulative actions need to be recognized. This chapter investigates how to improve the performance of fine manipulative action recognition using multi-modal sensors. Two information fusion methods (decision-level and feature-level) are proposed to combine the multiple data source. In the feature-level fusion method, feature selection is applied to select related features first. The feature selection approach is inspired by the variance preservation algorithm, which selects a subset of features that preserve the variance contained in the data, since the assembly actions are all non-stationary actions. Then, HMM models are trained with the selected features for each action. In the decision-level fusion method, features are divided into three groups: arm joint angle group, finger angle group and force feedback group. An HMM-based classifier is built for each feature group. A confidence index is designed. It is calculated using the maximum output (probability) divided by the sum of of all the outputs. Then, the classifier which has the maximum confidence index is chosen to make the global decision. Moreover, with the dependencies between object and action modeled in a Bayesian network, actions which are too similar to discriminate can be recognized with high accuracy. Online experimental results indicate that the feature-level fusion method outperforms the decision-level fusion method in terms of both computational efficiency and accuracy. The experiment on subjects with different body shapes shows the robustness of the system.

In this work, the force sensor we adopt has a very limited range (0 N to 1 N), we need to replace them with wide range ones. In addition, instead of just tracking the thumb and index finger motions, a full hand model should be created to fully capture all kinds of fine manipulative motions.

Chapter 5

Automated Complex Assembly Skill Acquisition through Human Demonstration

Based on the work introduced in the previous chapters, in this chapter we build a complex skill acquisition system. The used tools, the manipulated parts and the manipulative actions are simultaneously recognized. The assembly state which indicates the relative part pose is estimated using 3D models of the parts. Finally, skill scripts are automatically generated. Section 5.1 introduces the motivation of this chapter. In Section 5.2, the problem to be solved is formulated. Section 5.3 presents the proposed methodology. Section 5.4 describes the experiment procedures and gives the experimental results.

5.1 Motivation

A complex assembly task usually involves many parts and tools while being conducted in many steps, which introduces several challenges. First, it is still a nontrivial task to recognize small parts and tools using computer vision based methods [107]. Second, it is hard to recognize fine assembly actions purely based on human motion features [108]. Third, to obtain the relative position and orientation between the parts using 2D vision is challenging, since occlusion frequently occurs between assembled parts [109]. Finally, to capture the information of both the objects (parts and tools) and human movement, traditional approaches use multiple sophisticated sensors, which is costly and not easy to use.

Existing human skill demonstration systems focus on extracting either human motion information or object information from the demonstration. In [67], object information is extracted from the demonstration to create chains of two-object relationships. In [68], the demonstration is segmented into a sequence of primitives that describe the user actions. In [110], humans are tracked by multiple calibrated stereo cameras for human motion imitation. However, for assembly skill learning, both the objects and human motion during the demonstration are important clues. The human motion can tell how to accomplish a task while the object information can tell what parts are being manipulated or what tools are being used. Dillmann [59] built a human skill demonstration platform using data gloves to capture human motion while using multiple cameras for object recognition.

In this chapter, in contrast to previous works, we adopt a single RGB-D camera to develop a Portable Assembly Demonstration (PAD) system. This PAD system can generate skill scripts by capturing the information about human motion, the tools being used, and the parts being manipulated during human demonstration of a complex assembly task. The PAD system has the following features. First, it can recognize small assembly parts and tools used based on both color and depth information. Second, with the parts and tools as the contextual information, the assembly action can be recognized more accurately and effectively. Third, the final state (post-condition) of the assembled parts can be reliably estimated, which represents the effect of the assembly action. The state describes the relative position and orientation between the parts in the final assembly. With this capability, for example, it is possible to estimate how deep a bolt is hammered into a hole, or what is the angle a screw is rotated in the demonstration, which is important for the robotic assembly process.

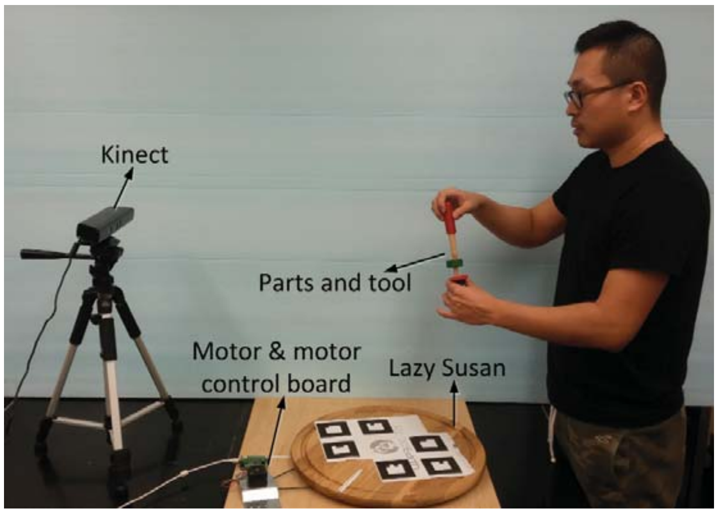

Figure 5.1: The hardware setup of the PAD system.

5.2 System Overview

5.2.1 PAD System Setup

The proposed Portable Assembly Demonstration (PAD) system is shown in Fig. 5.1, which consists of a Kinect sensor on a tripod, a motor-driven Lazy Susan, and a computer for data processing (not in the picture). The functionality of the PAD system is introduced in the previous chapter, except that we did not use the data glove.

5.2.2 Formulation of the Problem

Generally a complex assembly task which involves *n* parts can be represented as a sequence of subtasks as shown in Equation (5.1) and Equation (5.2):

$$\zeta = \underbrace{\underbrace{\underbrace{p_1}_{M_1{}^1} \oplus \underbrace{p_2}_{M_2{}^1}}_{M_1{}^2} \oplus \underbrace{p_3}_{M_2{}^2} \oplus \ldots}_{M_1{}^{n-1}} \oplus \underbrace{p_n}_{M_2^{n-1}} \tag{5.1}$$

$$\zeta = \underbrace{\underbrace{\underbrace{p_1}_{M_1{}^1} \oplus \underbrace{p_2}_{M_2{}^1}}_{M_1{}^3} \oplus \underbrace{\underbrace{p_3}_{M_1{}^2} \oplus \underbrace{p_4}_{M_2{}^2}}_{M_2{}^3} \oplus \ldots}_{M_1{}^{n-1}} \oplus \underbrace{p_n}_{M_2^{n-1}} \tag{5.2}$$

where ζ is the final product, p_n is the *nth* part, $\oplus$ is the assembly action on the two parts. The complex assembly task can be treated as a sequence of two-part assembly subtasks. As shown in Equation (5.1), in a two-part assembly task, the models for each part are defined as $M_1{}^1$ and $M_2{}^1$. First, take p_1's model as $M_1{}^1$, p_2's model as $M_2{}^1$, The superscript denotes the index of the subtask. Then, $M_1{}^2$ and $M_2{}^2$ will be used to denote $p_1 \oplus p_2$ and p_3 respectively. This process continues until the last part is assembled. On the other hand, the implementation of a complex task can also follow the sequence show in Equation (5.2) using the same idea.

For the case when there are multiple actions applied to the parts p_1 and p_2, we have

$$\begin{aligned} M_1{}^{m+1} &= p_1 \oplus_1 \oplus_2 \ldots \oplus_m p_2 \\ &= \underbrace{p_1}_{M_1{}^1} \oplus_1 \underbrace{p_2}_{M_2{}^1} => \ldots => \underbrace{\underbrace{p_1}_{M_1{}^m} \oplus_m \underbrace{p_2}_{M_2{}^m}}_{M_1{}^{m+1}} \end{aligned} \tag{5.3}$$

$M_1{}^{m+1}$ is the model of $p_1 \oplus_m p_2$. $\oplus_m$ is the *mth* action on part p_1 and p_2. It will be used as the M_1 for the $(m+1)th$ subtask. => denotes the action sequence flow.

Given the above setup, two problems need to be solved. The first is to recognize the action and all the parts/tools involved. The second is to estimate the assembly state which represents the effect of the action.

The input to the action recognition system has two parts: the observation of the objects and the observation of the human action. The observation of the objects has two parts, ψ_p is the decision of the part recognition which is one of the parts in the assembly sets. ψ_t is the decision of the tool recognition system which is one of the tools in the tool set. If the action does not involve any tool then $\psi_t = null$. The observation of the action ψ_a is a sequence of skeleton data of the human subject $\{\Gamma_1, \Gamma_2 ... \Gamma_t\}$, where $\Gamma_t = \{\Theta_1, \Theta_2 ... \Theta_k\}$, Θ_i is the angle of joint i and k is the total number of joints. We assume the part set $P = \{P_1, P_2 ... P_m\}$, the tool set $T = \{T_1, T_2 ... T_n\}$, and the action set

$A = \{A_1, A_2...A_q\}$. The action recognition is to develop an algorithm to decide on $A_i \in A$ given the above observation. In other words, the goal of the action recognition is to find a mapping function f_a,

$$A_i = f_a(\psi_a, \psi_p, \psi_t) \tag{5.4}$$

To obtain the assembly state, the input includes the 3D models of each individual part and the 3D model of the assembled part after each action. The goal of assembly state estimation is to find a mapping function f_s,

$$S = f_s(M_1, M_2, M_1 \oplus M_2) \tag{5.5}$$

Here, S is the assembly state. M_1 and M_2 are the models of the two parts involved. $M_1 \oplus M_2$ is the 3D model of the assembled part which is a joint of part M_1 and M_2. To estimate the assembly state, M_2 is treated as the reference part. Then the state S is the pose of the non-reference part with respect to the reference part. Based on the solutions to the two problems mentioned, we can generate a skill script Σ as follows

$$\Sigma = (\Sigma_1, \Sigma_2, \Sigma_3...\Sigma_n) \tag{5.6}$$

where each assembly skill Σ_i is a 5-tuple symbolic description defined as

$$\Sigma_i =< A_i, T_i, P_{1i}, P_{2i}, S_i > \tag{5.7}$$

where P_{1i} and P_{2i} are the two parts involved in the ith step.

5.3 Methodology

In this section, the approaches to object recognition, action recognition and assembly state estimation are described. We adopt the same approaches used in the previous chapter for object and action recognition. Here, we just introduce the assembly state estimation approach.

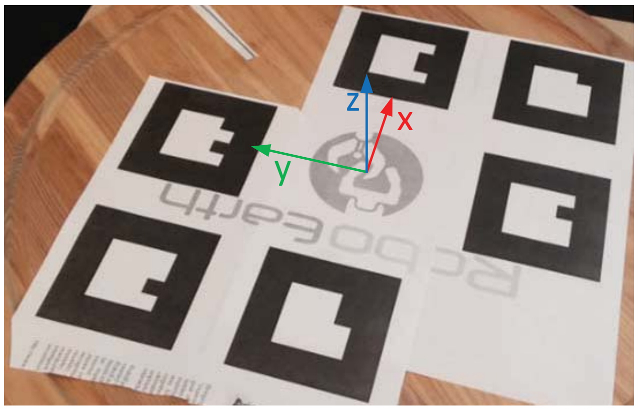

Figure 5.2: Coordinate system defined by the 6 markers in [1].

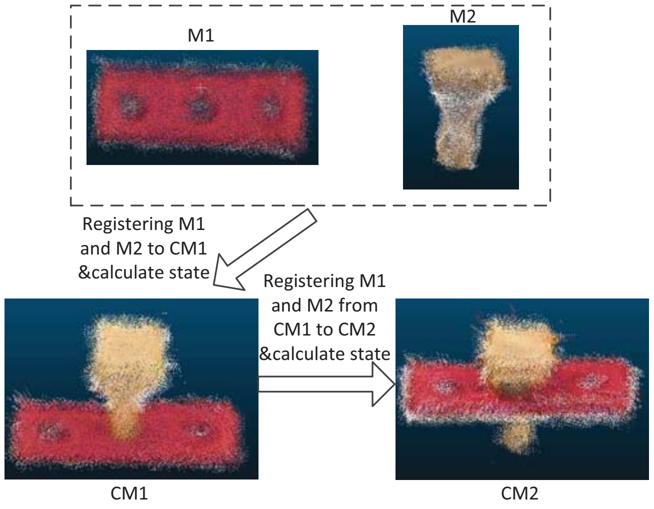
M1
M2
Registering M1 and M2 to CM1 &calculate state
Registering M1 and M2 from CM1 to CM2 &calculate state
CM1
CM2

Figure 5.3: State estimation for a two-step assembly task. M_1 and M_2 are the 3D models of the base board and the bolt. CM_1 is the 3D model of the assembled part after the first action. CM_2 is the 3D model of the assembled part after the second action.

5.3.1 Assembly State Estimation

To estimate the assembly state, we first define a common coordinate system based on 6 markers, as shown in Fig. 5.2. One of the two parts will be treated as the reference part. Therefore, the assembly state is the pose of the non-reference part with respect to the reference part.

Fig. 5.3 shows a two-step assembly task that inserts and hammers a square bolt into a hole on the base board. With the 3D scanner, first we create the 3D models for these two parts. Since they are in the same coordinate frame, the initial state S_0 can be represented by a 4×4 identity matrix. The 3D model of the assembled part is created after an action is applied to these two parts. The 3D model of each part is registered to the 3D model of the assembled part.

We adopt a two-step point cloud registration approach. In the first step, Fast Point Feature Histograms (FPFH) [111] are extracted from both object and scene models. It is an informative pose-invariant local feature which represents the underlying surface model properties at a point. Based on the correspondences between features, Sample Consensus Initial Alignment (SAC-IA) [112] is used to find the initial alignment. In the second step, the ICP (Iterative Closest Point) [113] algorithm is adopted to calculate the transformation based on the initial alignment. With this approach, the position and orientation of the individual models can be arbitrary with respect to the model of the assembled part. The transformation matrix obtained by our approach is shown in Equation (5.8).

$$T = \begin{pmatrix} R_{11} & R_{12} & R_{13} & D_x \\ R_{21} & R_{22} & R_{23} & D_y \\ R_{31} & R_{32} & R_{33} & D_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.8}$$

The translation along each axis is given by D_x, D_y, D_z. To calculate the Euler angles about each axis, let ψ, θ, ϕ be the Euler angles about the X, Y, Z axis, respectively, then we have the following equations [114].

$$\begin{cases} \psi = atan2(R_{32}/R_{33}) \\ \theta = -sin^{-1}R_{31} \\ \phi = atan2(R_{21}/R_{11}) \end{cases} \tag{5.9}$$

Here we define two transformation matrices. T_{ri} represents the pose change of the reference part after action i; while T_{pi} represents the pose change of the non-reference part after action i. The assembly state S_i after action i can be calculated as

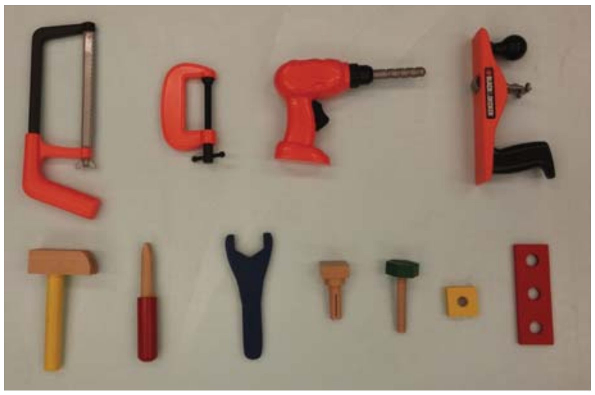

Figure 5.4: The parts and tools. From left to right, the top row are saw, G clamp, drill, and wood planner. The bottom row are hammer, screw driver, wrench, bolt, screw, nut and red board.

Case 1: S_i and S_{i-1} are calculated using the same models.

$$S_i = (T_{ri})^{-1} \cdot S_{i-1} \cdot T_{pi} \tag{5.10}$$

Case 2: S_i and S_{i-1} are calculated using different models.

$$S_i = (T_{ri})^{-1} \cdot T_{pi} \tag{5.11}$$

5.4 Experiment and Results

Experiments are conducted to verify the proposed method. An assembly scenario is used to validate our theoretical framework. Fig. 5.4 shows the parts and tools used in our experiment.

5.4.1 Object Recognition

Template Extraction

The point clouds of the parts on top of the Lazy Susan are extracted. They will be used for part recognition. The point clouds of the base board are shown in Fig. 5.5. It shows that our approach can effectively extract the points that belong to the part from the scene point cloud. It is worth mentioning that the plane function of the Lazy Susan corresponds to the camera pose. If the

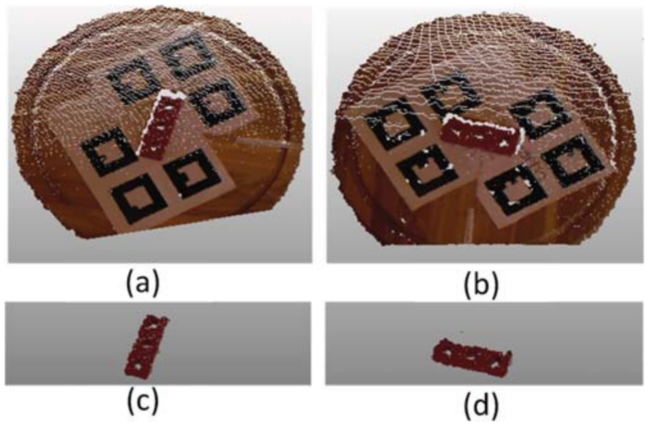

Figure 5.5: Extraction of the based board's points from the scene point cloud. (a) and (b) show the scene point cloud with two different rotation angles of the Lazy Susan. (c) and (d) show the extracted points of the base board.

camera pose is changed, the plane function should be recalculated. To reliably recognize objects, two templates are needed for symmetrical objects and four for asymmetrical objects.

Object Recognition

The top row in Fig. 5.6 shows the recognition results of the screw driver and the hammer respectively. The bottom row shows the recognition of the base board and the screw. The recognized objects are highlighted in red. In order to recognize small objects like the screw, the down sampling radius for the scene has to be small enough so that enough keypoints can be extracted. In addition, we evaluate the robustness of our approach under occlusion and in cluttered scenes. Fig. 5.7 (a) and (b) show that with minor occlusions, the board can still be recognized. Fig. 5.7 (c) and (d) show that our algorithm works correctly in cluttered scenes. The two red blocks in the cluttered scene which are similar to the base board are not misclassified as the base board. The recognition accuracy is shown in Table. 5.1.

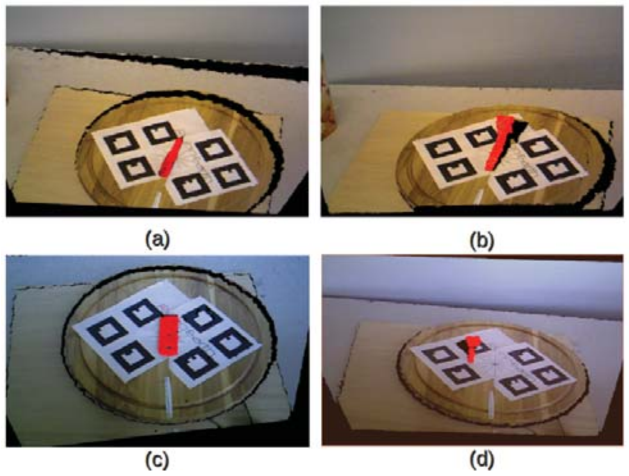

Figure 5.6: Results of tool and part recognition, where the recognized objects are highlighted in red. (a) recognition of the screw driver. (b) recognition of the hammer. (c) recognition of the base board. (d) recognition of the screw.

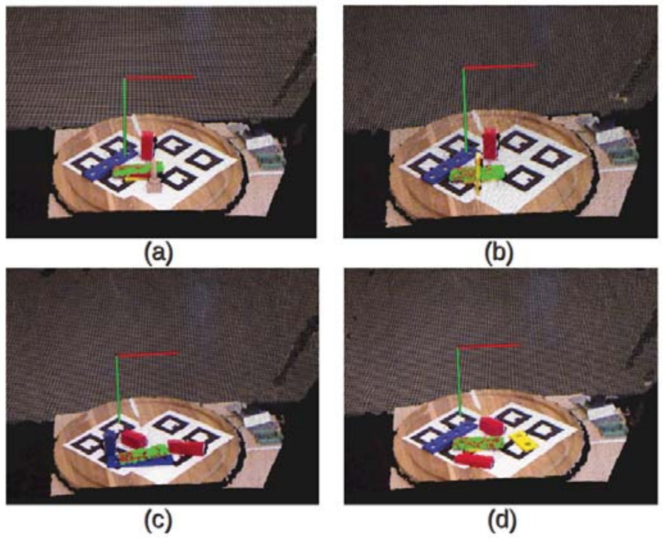

Figure 5.7: Results of board recognition, where the recognized objects are highlighted in green. (a) recognition of the board with occlusion from a bolt. (b) recognition of the board with occlusion from another board. (c) recognition of the board in cluttered scene 1. (d) recognition of the board in cluttered scene 2.

5.4.2 Action Recognition

For action recognition, segmentation is implemented implicitly by the HMMs. If the outputs of all the HMMs are very small, it indicates that there is no action or the action has already finished.

Table 5.1: The object recognition accuracy.

Object	saw	G clamp	drill	wood planner
Accuracy	0.92	0.93	0.95	0.92
Object	hammer	screw driver	wrench	bolt
Accuracy	0.97	0.87	0.88	0.85
Object	screw	nut	red board	
Accuracy	0.86	0.87	0.89	

Table 5.2: Recognition accuracy of the HMMs

test type	decision type										accuracy
	hammering	screwing	wrenching	sawing	drilling	rasping	fixing	inserting	aligning	missed	
hammering	**0.90**	-	-	0.02	-	0.04	-	-	-	0.04	**0.90**
screwing	-	**0.75**	-	-	-	-	0.22	-	-	0.03	**0.75**
wrenching	-	-	**0.93**	0.02	0.01	-	0.02	-	-	0.02	**0.93**
sawing	-	-	-	**0.66**	-	0.32	-	-	-	0.02	**0.66**
drilling	0.02	-	0.02	-	**0.89**	-	-	-	-	0.07	**0.89**
rasping	-	-	-	0.34	-	**0.62**	-	-	-	0.04	**0.62**
fixing	-	0.25	-	-	-	-	**0.72**	-	-	0.03	**0.72**
inserting	-	-	-	-	-	0.02	-	**0.86**	0.10	0.02	**0.86**
aligning	-	-	-	-	-	0.01	-	0.06	**0.87**	0.06	**0.87**

Two kinds of manipulative actions are tested. For the actions that involve tools, both tool/action and part/action dependencies are considered. The priority of the tool/action dependencies is higher than that of the part/action dependencies, since the tool used usually offers more clue about the action than the part does. While for the actions that do not involve tools, only part/action dependencies are considered. As the human hand (with or without tool in the hand) approaches the part, the Bayesian network takes effect. To evaluate the action recognition system, the training data set is collected from one subject while the HMMs are tested on four other subjects. Each action is conducted 50 times by four subjects. The accuracy of the HMMs and the Bayesian model are shown in Table 5.2 and 5.3 respectively. With the object context, the Bayesian model can differentiate actions more effectively.

Table 5.3: Recognition accuracy of the Bayesian model

test type	decision type										accuracy
	hammering	screwing	wrenching	sawing	drilling	rasping	fixing	inserting	aligning	missed	
hammering	**0.94**	-	-	0.02	-	0.02	-	-	-	0.02	**0.94**
screwing	-	**0.95**	-	-	-	-	0.02	-	-	0.03	**0.95**
wrenching	-	-	**0.98**	-	-	-	-	-	-	0.02	**0.98**
sawing	-	-	-	**0.96**	-	0.02	-	-	-	0.02	**0.96**
drilling	-	-	-	-	**0.95**	-	-	-	-	0.05	**0.95**
rasping	-	-	-	0.04	-	**0.92**	-	-	-	0.04	**0.94**
fixing	-	0.02	-	-	-	-	**0.92**	-	-	0.06	**0.92**
inserting	-	-	-	-	-	-	-	**0.91**	0.07	0.02	**0.91**
aligning	-	-	-	-	-	-	-	0.05	**0.90**	0.05	**0.90**

5.4.3 Assembly State Estimation

In this section, the point cloud registration method used is evaluated first. Then the assembly state estimation results are given.

Point Cloud Registration Method Evaluation

The computation cost of the point cloud registration is proportional to the size of the point cloud. However, reducing the size of point clouds may impact the accuracy of registration. Therefore, we design an experiment to evaluate how the size of the point cloud affects the accuracy of a pure rotation assembly. The accuracy of the calculated angle ϕ is evaluated against the portion of the points kept. Fig. 5.8 indicates that as long as more than half of the points are kept, the accuracy will be consistently high.

Another experiment is implemented to test the robustness of our registration approach. The model of a bolt is registered to its flipped model. Fig. 5.9 (a) and (b) are the input point cloud and the target point cloud respectively. Fig. 5.9 (c) is the registration result of the ICP algorithm without initial alignment. The registration failed since without initial alignment, the ICP algorithm falls into a local minimum. Fig. 5.9 (d) is the result of the Sample Consensus Initial Alignment algorithm. It roughly aligns the input cloud to the target cloud. However some obvious misalignment still exists, since the correspondences found are not always correct. The last figure is the

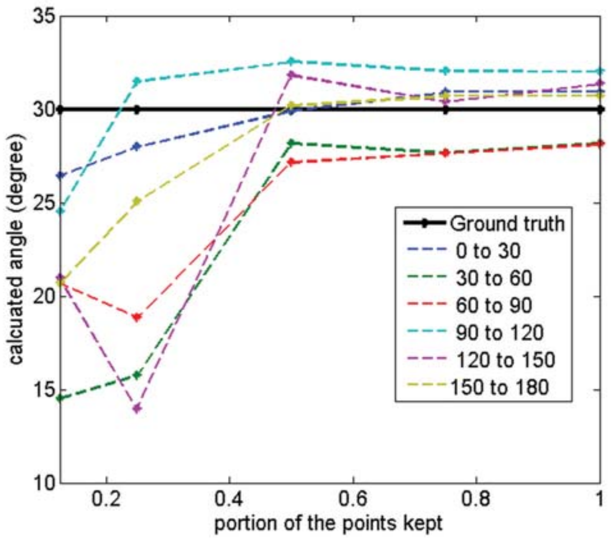

Figure 5.8: Accuracy of the rotation estimation with respect to the portion of point cloud kept. Different colors denote registration from the models with different intial angles. The angular difference between the input model and the target model of each registration is 30 degree.

result of the proposed approach. The ICP is applied after initial alignment which can overcome the limitation of each individual algorithm and the registration results are better.

Assembly State Estimation

The experimental procedure is shown in Table. 5.4, while Fig. 5.10 shows the six steps of the assembly task and Fig. 5.11 shows the parts and the corresponding models after each step. First, the bolt is inserted in one hole of the base board. Second, the bolt is hammered into the hole. Third, the bolt is wrenched by 30° with respect to the base board. Fourth, a nut is aligned to the other hole. Fifth, a screw is inserted into this hole. Finally, a screwdriver is used to screw the base board and the nut.

The initial state S_0 is

$$\begin{aligned} S_0 &= \{D_x, D_y, D_z, \psi, \theta, \phi\} \\ &= \{0.0\,mm, 0.0\,mm, 0.0\,mm, 0.00^\circ, 0.00^\circ, 0.00^\circ\} \end{aligned} \tag{5.12}$$

Fig. 5.12-(a) gives the results of registering ${M_1}^1$ and ${M_2}^1$ to CM_1 after the "inserting" action. The

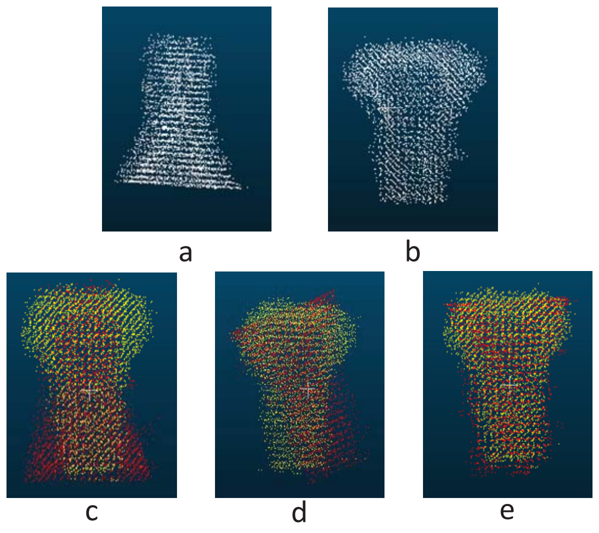
Figure 5.9: Registration results. All the figures show the downsampled 3D models without color information. (a) the input 3D model. (b) the target model. (c) registration result of ICP. (d) registration result of SAC-IA. (e) registration result of our approach. ((c) - (e): Red ones indicate the registration results. Yellow ones indicate the target model.)

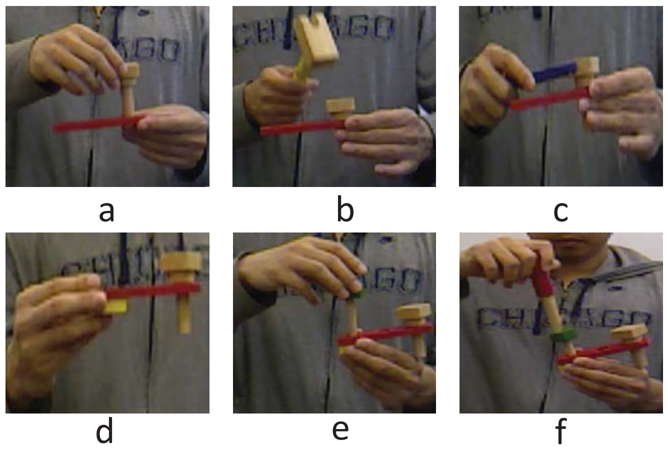
Figure 5.10: The actions used in the experiment. (a) inserting the bolt. (b) hammering the bolt. (c) wrenching the bolt. (d) aligning the nut to the hole on the base board. (e) inserting the screw. (f) screwing the screw.

state S_1 which represents the pose of M_2 with respect to M_1 is estimated as:

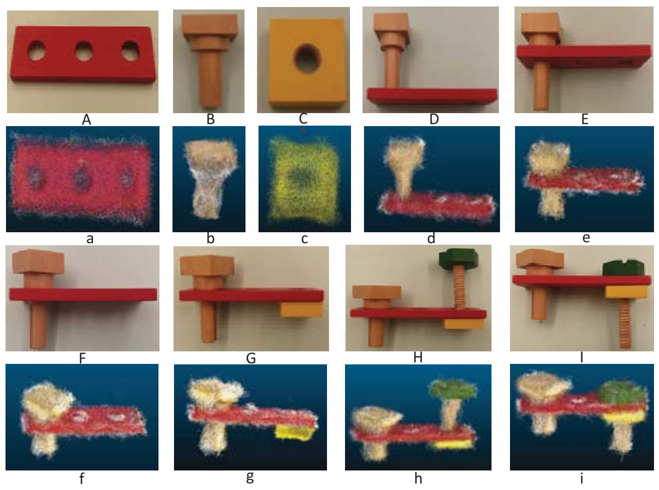

Figure 5.11: The parts and the associated model. (A) red board. (B) bolt (C) nut (D) parts pose (CP1) after action 1. (E) parts pose (CP2) after action 2. (F) parts pose (CP3) after action 3. (G) parts pose (CP4) after action 4. (H) parts pose (CP5) after action 5. (I) parts pose (CP6) after action 6. (a) - (i) are the models of (A)-(I).

Table 5.4: The procedure of the assembly task

step	**p1 (reference)**	**p2**	**action⊕**	**p1⊕ p2**
1	red board	bolt	inserting	CP1
2	red board	bolt	hammering	CP2
3	red board	bolt	wrenching	CP3
4	CP3	nut	aligning	CP4
5	CP4	screw	inserting	CP5
6	CP4	screw	screwing	CP6

$$\hat{S}_1 = \{43.2\,mm, 1.7\,mm, -2.8\,mm, 0.50°, 0.12°, 1.01°\} \tag{5.13}$$

The major movement is a translation along the positive direction of X axis by 43mm. Fig. 5.12-(b) shows the registration of ${M_1}^2$ and ${M_2}^2$ to CM_2. The major movement of ${M_1}^2$ is translation along the vertical axis (Z axis). The ground truth of S_2 is $\{43.0mm, 0.0mm, -34.0mm, 0.00°, 0.00°, 0.00°\}$.

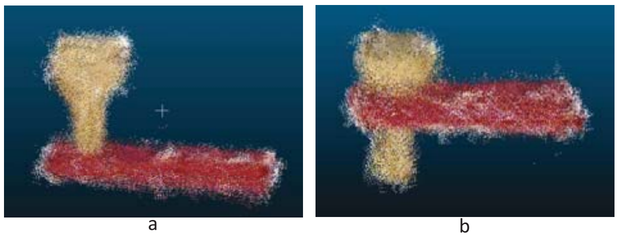

Figure 5.12: Registration results. (a) register ${M_1}^1$ and ${M_2}^1$ to CM_1 after action 1 “inserting”. (b) register ${M_1}^2$ and ${M_2}^2$ to CM_2 after action 2 “hammering”.

The estimated S_2 is

$$\hat{S}_2 = \{44.1\,mm, 3.5\,mm, -35.9\,mm, 0.95°, -0.74°, 1.41°\} \tag{5.14}$$

After wrenching, the bolt is rotated anticlockwise by 30° around the base board. The ground truth of S_3 is $\{43.0\,mm, 0.0\,mm, -34.0\,mm, 0.00°, 0.00°, 30.00°\}$. The estimated S_3 is

$$\hat{S}_3 = \{43.5\,mm, -4.1\,mm, -33.2\,mm, 2.21°, 1.36°, 31.47°\} \tag{5.15}$$

Similarly, this process goes on until the final part is assembled. The estimated assembly state $\hat{S}_4$, $\hat{S}_5$, $\hat{S}_6$ are as follows,

$$\hat{S}_4 = \{-42.7\,mm, 2.2\,mm, -2.8\,mm, 3.65°, 2.74°, 4.21°\} \tag{5.16}$$

$$\hat{S}_5 = \{-46.2\,mm, -1.0\,mm, 0.9\,mm, 4.39°, 4.26°, 5.12°\} \tag{5.17}$$

$$\hat{S}_6 = \{-45.2\,mm, 5.2\,mm, -51.2\,mm, 3.88°, 3.54°, 7.26°\} \tag{5.18}$$

The registration result of the last step is shown in Fig. 5.13. As the assembled part becomes more complicated, the registration result gets less accurate. The reason is because the chance of occlusion increases. The registration performance of each step is given in Table 5.5 in terms of root mean square (RMS in mm). It is worth noting that for some symmetrical parts, the rotation angle cannot be uniquely determined, which may require other source of information.

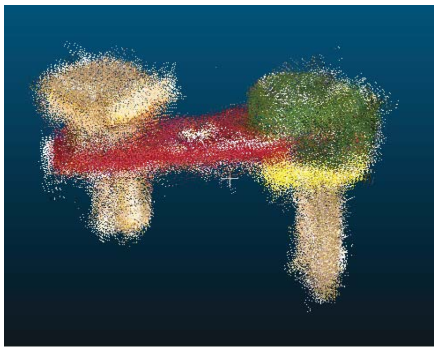

Figure 5.13: Registration results. Register ${M_1}^5$ and ${M_2}^5$ to CM_6 after action 6 "screwing".

Table 5.5: The registration performance

step	1	2	3
RMS error (mm)	1.280	1.662	1.862
step	4	5	6
RMS error (mm)	1.7014	1.9014	2.4624

The assembly process is repeated 10 times. The average error for translation and rotation is within 10 *mm* and 10° respectively. Finally, after the human demonstration, the following script can be generated automatically:

$$\begin{cases} \Sigma_1 =< inserting, N/A, bolt, baseboard, \hat{S}_1 > \\ \Sigma_2 =< hammering, hammer, bolt, baseboard, \hat{S}_2 > \\ \Sigma_3 =< wrenching, wrench, bolt, baseboard, \hat{S}_3 > \\ \Sigma_4 =< aligning, N/A, nut, CP3, \hat{S}_4 > \\ \Sigma_5 =< inserting, N/A, screw, CP4, \hat{S}_5 > \\ \Sigma_6 =< screwing, screwdriver, screw, CP4, \hat{S}_6 > \end{cases} \tag{5.19}$$

where N/A stands for "Not Applicable". In summary, with the action recognition and assembly state estimation methods developed in the paper, we can automatically obtain the assembly script from human demonstration.

5.5 Summary

To teach robots complex assembly skills, the robots should be able to recognize the objects (parts and tools) involved, the actions applied, and the effect of the actions on the parts. Recognizing the subtle assembly actions is a non-trivial task, and it is challenging to estimate the effect of the actions on the assembly part due to the small part sizes.

In this chapter, we propose a Portable Assembly Demonstration (PAD) system for robots to learn complex assembly skills from humans. A complex task can be divided into a sequence of two-part subtasks. Therefore, the human demonstration can be sequentially segmented. Based on a RGB-D camera, tools and parts used in the assembly are recognized based on the 3D point cloud, which helps to effectively recognize complex assembly actions. To recognize objects, the 3D templates for each object are created beforehand. Point-to-point correspondences between model descriptors and scene descriptors are then determined. Each correspondence generates a vote for the object pose. If an object pose gets enough votes, it indicates that the object is recognized. To recognize assembly actions, HMMs are trained for each assembly action to capture the temporary change. On top of that, a Bayesian model is built to characterize the object/action dependencies. In this chapter, the object includes both part and tool. To estimate the assembly state, 3D models of the individual parts and the assembled parts are created using the PAD system. A two-step registration approach is adopted to estimate the assembly state. First initial alignment is obtained between the source and target point clouds. Then ICP is applied to get the final result. In this case, the source and target point clouds can have random initial poses. This registration method is robust to minor occlusions and works well for small parts. Experiments have verified and evaluated the proposed PAD system and the associated theoretical framework. Our PAD system can be used for robots to acquire assembly skills, which can help automate the future factories. Currently the limited accuracy of the created 3D models restricts the current PAD system to only simple parts. In future work, we will explore the use of more accurate 3D sensors and 3D modeling methods and also try on more complicated parts.

Chapter 6

Implementing Assembly Skills on the Baxter Robot

In this chapter, we teach a dual-arm robot how to implement the assembly scripts. Section 6.1 introduces the hardware platform. In Section 6.2, the method to track the assembly parts is presented. Section 6.3 presents the basic action learning approach. Section 6.4 describes the experiment procedures and gives the experimental results.

6.1 System Overview

The implementation procedure as shown in Fig. 6.1 includes three steps: basic action learning, complex skill demonstration and skill implementation. The complex skill demonstration is introduced in the previous chapter. In basic level learning, the basic skills are learned through human scaffolding. The trajectory of the robot left gripper with respect to the reference part (the part that is fixed during the demonstration) is recorded. The reference part is tracked using an Xtion sensor. Therefore, the relationship between the Xtion frame and the robot frame has to be calibrated. The trajectories extracted from multiple demonstrations are generalized. The generalized trajectory will be reproduced according to the pose of the reference part.

After the basic skill learning, a complex assembly task that consists of multiple steps is demon-

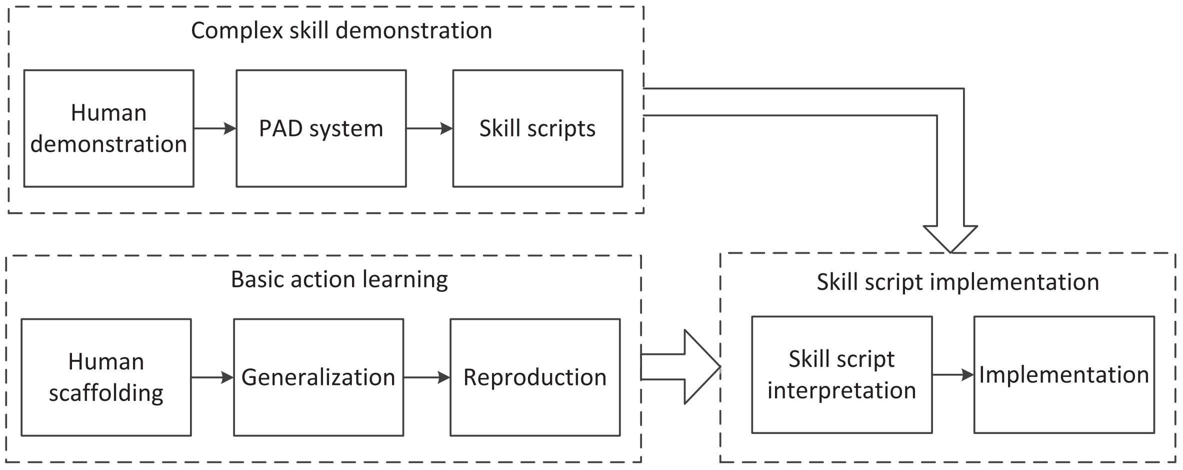

Figure 6.1: The assembly skill implementation procedure.

strated to the Baxter robot using the PAD system. The skill scripts generated by the PAD system are interpreted. The interpretation involves obtaining part and tool information, confirming the assembly action, and assembly state mapping. After that, the robot implements the skill script.

6.1.1 Hardware Setup

The hardware setup for the implementation mainly includes a Baxter robot [115] and an Xtion sensor. We use the same assembly parts and tools introduced in Chapter 5. The Baxter robot is shown in Fig. 6.2. It is a dual arm robot which is capable of performing a variety of simple yet critical production tasks while safely and intelligently working next to people. Its 360° sonar and front camera can be used for human presence detection, vision-guided movement, and object detection. Each arm has 7-degrees of freedom for maximum flexibility and range [116]. The Xtion sensor is used to track the assembly parts.

6.1.2 Calibration

The pose of each part is tracked using augmented reality (AR) markers by an Xtion sensor. Tracking the parts is very challenging because of the limited size and occlusion between the assembly parts. Therefore, AR markers attached to each part are tracked. By making the marker

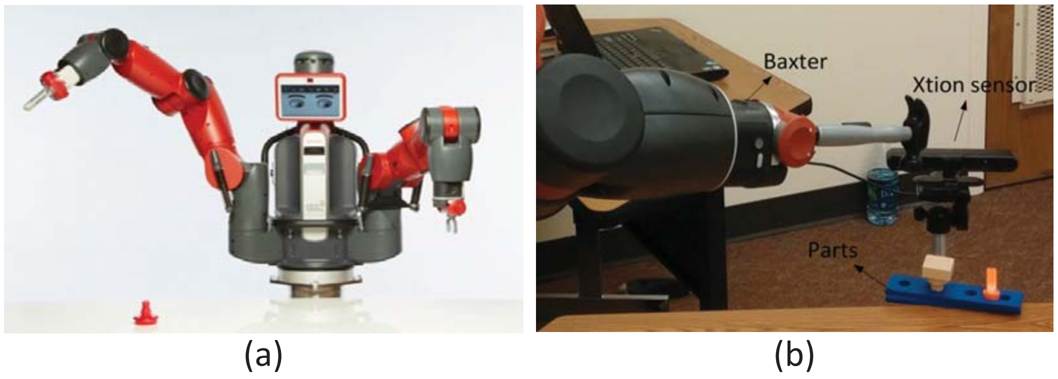

Figure 6.2: The robot implementation setup. (a) The Baxter robot [2]. (b) The workspace of the Baxter robot.

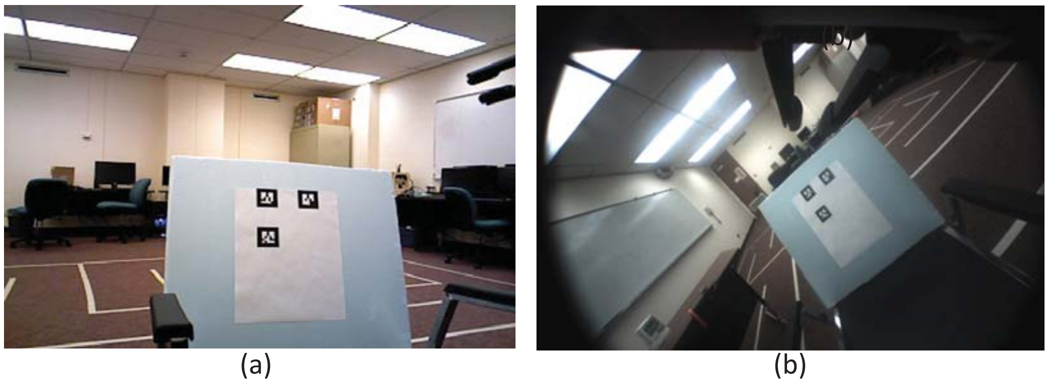

Figure 6.3: Camera views. (a) The Xtion color image view. (b) The left gripper camera view.

reasonably small (2cm by 2cm), we can reduce the chance of occlusion. In order to record the trajectory of the left gripper with respect to the reference part, calibration between the Baxter base frame (R) and the Xtion camera frame (X) is required. A 3D coordinate frame can be represented by a 4x4 matrix which is also called homogeneous transformation matrix containing information of both rotation R and translation T. The relationship between two different frames can be represented by a rigid transformation:

$$\begin{bmatrix} {}^{B}P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{B}_{A}R & {}^{B}_{A}T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}P \\ 1 \end{bmatrix} \tag{6.1}$$

The homogeneous transformation matrix G_{AB} is defined as

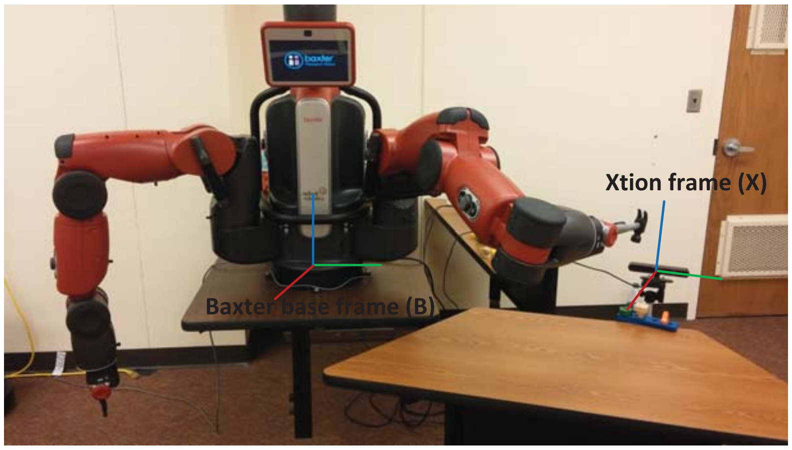

Figure 6.4: Camera calibration results. (a) Robot and camera positions. (b) Relationship between the camera frame and selected frames of the Baxter robot.

$$G_{AB} = \begin{bmatrix} {}^{B}_{A}R & {}^{B}_{A}T \\ 0 & 1 \end{bmatrix} \tag{6.2}$$

In our case, the goal is to find the homogeneous transformation matrix G_{XB} which represents the transformation relationship between Xtion frame (X) and Baxter base frame (B). However, it is impossible to calibrate these two frames directly. Therefore, the left gripper camera frame (L) and the Xtion frame (X) is calibrated. After which G_{XL} can be obtained. We have

$$G_{XB} = G_{XL} \cdot G_{LB} \tag{6.3}$$

G_{LB} is given by the robot SDK since it is determined by the robot internal joint configuration. To find G_{XL}, an AR marker pattern is used. The marker pattern is put in both camera views as shown in Fig. 6.3. After both cameras detect the AR marker pose, G_{XA} and G_{LA} are found through regular camera calibration [117] where A denotes denotes the marker pattern frame. Then we have

$$G_{XL} = G_{XA} \cdot G_{LA}^{-1} \tag{6.4}$$

The Xtion frame (X) and the Baxter base frame (B) are shown in Fig. 6.4 after calibration.

6.1.3 Object Tracking

We use the ALVAR tracking library [118] for marker tracking. It features an adaptive threshold to handle a variety of lighting conditions, optical flow based tracking for more stable pose estimation, and an improved tag identification method that does not significantly slow down as the number of tags increases. Different markers are encoded with different IDs. A single marker is attached to the part that does not have significant rotations. For the parts that involve significant rotations, multiple markers with different IDs are attached to the opposite sides of the part. The performance of the marker tracking is evaluated. The Vicon MX motion capture system is used to obtain the ground truth of the object tracking.

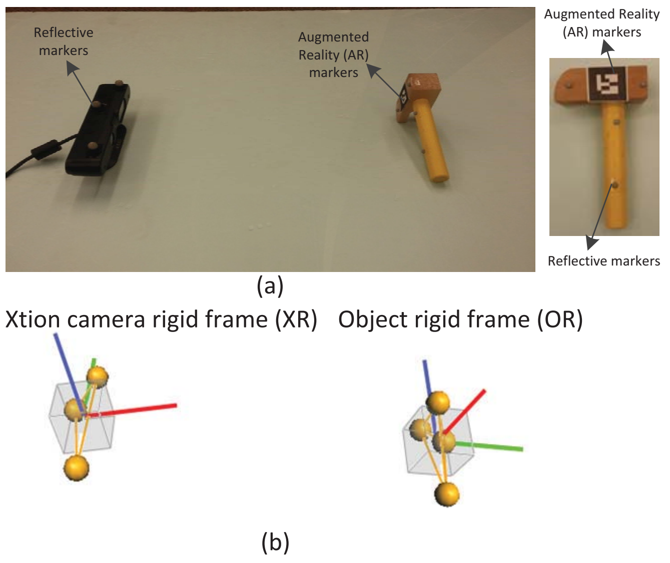
Reflective markers
Augmented Reality (AR) markers
Augmented Reality (AR) markers
Reflective markers
(a)
Xtion camera rigid frame (XR)
Object rigid frame (OR)
(b)

Figure 6.5: Marker placement. (a) reflective markers on camera and object (hammer). (b) frames created using reflective markers.

As shown in Fig. 6.5, reflective markers are put on the target for tracking using Vicon. Meanwhile, an AR marker is also attached to the target which is tracked by the Xtion camera. Therefore it is necessary to find the relationship between the object AR marker frame and the rigid body frame formed by the reflective markers on the object. To solve this problem, multiple frames are defined. They are Vicon frame (V), the Xtion camera rigid frame (XR), the Xtion camera frame (X), the object AR marker frame (O) and the object rigid frame (OR). The frame transform tree is

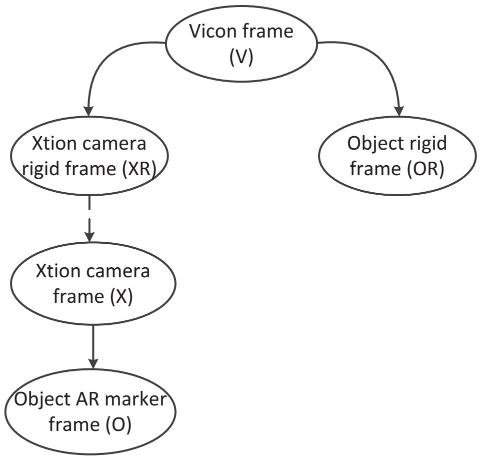

Figure 6.6: The frame transform tree. The dash line represents the unknown relationship.

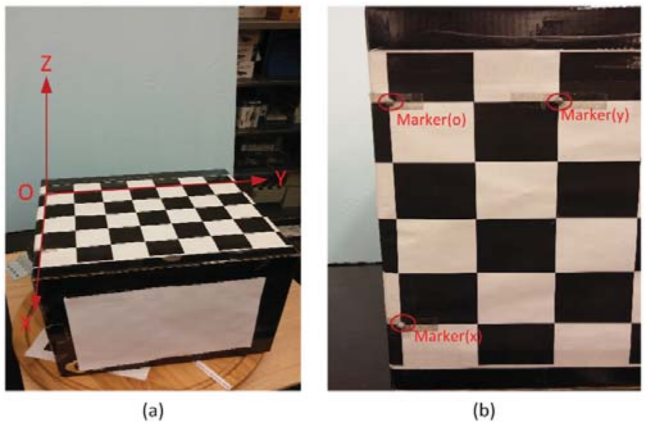

Figure 6.7: The chessboard frame (CH). (a) the chessboard frame defined in the calibration method. (b) the placement of three markers which are used to create the same frame.

shown in Fig. 6.6.

The Vicon frame is defined by the Vicon system. The relationship between the rigid bodies created and the Vicon system can be obtained using Vicon SDK. The relationship between the Xtion camera frame (X) and the object AR marker frame (O) is the output of the AR marker tracking algorithm. The relationship between the object AR marker frame (O) and the object rigid frame (OR) is fixed. Therefore, the key step is to find the relationship between the the Xtion camera rigid frame (XR) and the Xtion camera frame (X). To find this relationship, a chessboard is used to conduct the calibration and a chessboard frame (CH) is introduced. We apply a traditional calibration method [119] to obtain both extrinsic and intrinsic parameters of the Xtion color camera. Here, we are more interested in the extrinsic parameters which represent the relationship between the camera frame (X) and the chessboard frame (CH). On the other hand, by using three markers attached to the chessboard, we can manually create the chessboard frame. Fig. 6.7-(a) shows the chessboard frame (CH) defined by the calibration method. Fig. 6.7-(b) shows the placement of three markers to create the same frame in Vicon. Then, we can have the marker positions $\{O_{CH}, X_{CH}, Y_{CH}\}$ in the chessboard frame (CH) and $\{O_V, X_V, Y_V\}$ in the Vicon frame (V). The goal is to find a rigid transformation that optimally aligns the two sets in the least squares sense. The singular value decomposition method (SVD) [120] is used to fulfill this task. Then, the relationship between the camera frame (X) and the Vicon frame (V) is found. Also, the relationship between the Vicon frame (V) and the camera rigid frame (XR) is known from the Vicon tracking output. The relationship between the frame (X) and the frame (XR) is obtained. After bridging the gap shown in Fig. 6.6, the relationship between the object AR marker frame (O) and the object rigid frame (OR) is found.

Now, we can evaluate the AR marker tracking performance. We move the object (hammer) randomly within the marker's reliable tracking range (0.1m - 0.5m). One of the tracking evaluation results is given in Fig. 6.8. The peaks shown in the figure are caused by the sudden motion of the tracked target. The tracking algorithm can catch up with the target in a few seconds. The dynamic tracking error is not a problem since it is assumed that the object motions will be slow or mainly stationary. The average translational error is within 1.0 cm and the average rotational error is within 5 degrees.

6.2 Basic Action Learning

The basic actions are learned through human scaffolding. Fig. 6.9 shows one demonstration of the hammering actions. There are four actions taught in total: grasping, inserting, hammering, and screwdriving. Due to the limited capabilities of the Baxter robot, the following assumptions are made:

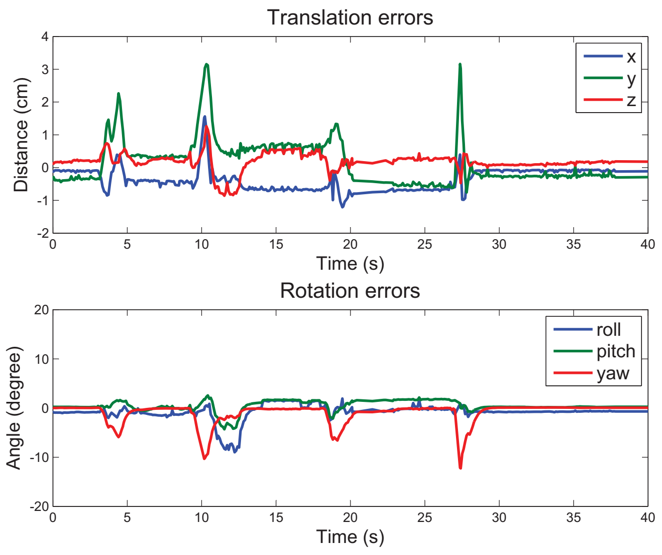

Figure 6.8: Tracking accuracy evaluation. The peaks are caused by the quick motion of the tracked target.

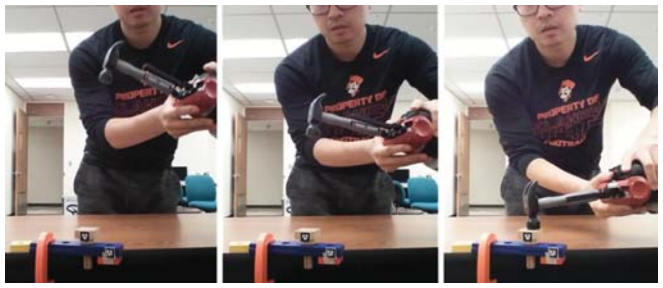

Figure 6.9: Human scaffolding for the hammering action.

- The robot always grasps the part from an overhead position.
- Once a part is grasped, there is no relative motion between the part and the gripper.
- The tool is fixed on the gripper manually.

For each action, the robot arm is guided to the desired pose multiple times from different initial poses. Let $\left\{\varepsilon_j\right\}_{j=1}^{M}$ denote the M demonstrations. Each demonstration is normalized to 40 time steps using interpolation. Each data point in ε_j, $\{t_i, \gamma_i\}$, consists of a time step t_i and a pose γ_i which is the pose of the left gripper with respect to the part, ${}^{lg}P_{pt}$. To extract the principle task and joint space constraints from multiple demonstrations, the Gaussian Mixture Model (GMM)

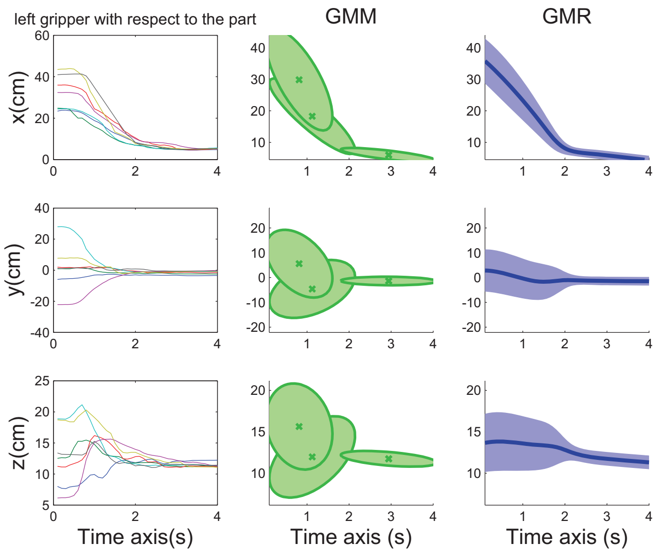

Figure 6.10: Trajectory encoding and generalization 1. The figures on the left column represent the trajectories of x,y, and z positions of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances.

and Gaussian Mixture Regression (GMR) methods [57] are adopted. The data is first normalized before applying GMM/GMR. The normalization is to make each demonstration have the same data length. The normalized dataset is first modeled by a Gaussian Mixture Model (GMM) which has three components and three states. These parameters are experimentally decided using a trial-and-error method. Based on the GMM, a generalized version of the trajectories is computed by applying Gaussian Mixture Regression (GMR).

Fig. 6.10 and 6.11 show the results of applying GMM/GMR to multiple "hammering" demonstrations which characterize the left gripper's pose with respect to the bolt. The results indicate that the positional constraints of the hammering action are tight at the end, since the hammer always needs to strike the small top surface of the bolt. On the other hand, the rotational constraints are loose, which also makes sense since the hammer can approach the bolt from multiple directions. This method is applied to all the other basic actions required.

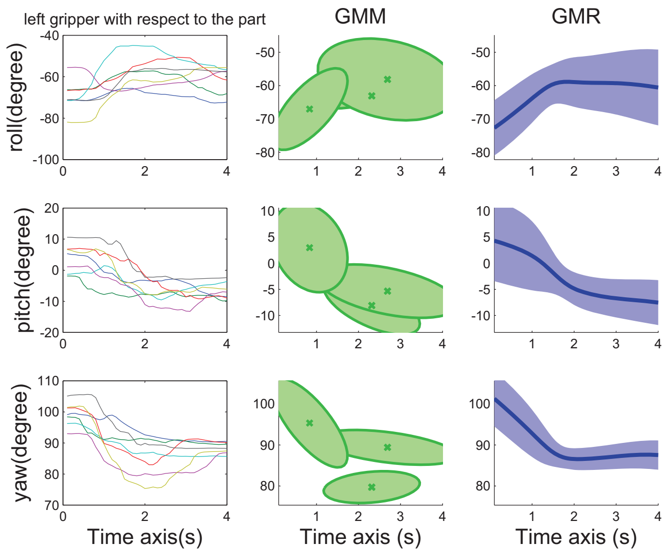

Figure 6.11: Trajectory encoding and generalization 2. The figures on the left column represent the roll, pitch, and yaw rotations of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances.

When a new pose of the part is obtained by the Xtion sensor, it can be transformed into the pose with respect to the Baxter base frame ${}^{\mathrm{pt}}P_{base}$. Then the reproduced pose at each time point can be derived as follows:

$$ {}^{\mathrm{lg}}P_{base} = {}^{\mathrm{lg}}P_{pt} \cdot {}^{\mathrm{pt}}P_{base} \tag{6.5} $$

Ultimately, ${}^{\mathrm{lg}}P_{base}$ must be converted from the task space to the joint space using the given Inverse Kinematic function, since the robot is controlled in the joint space. Fig. 6.12 shows the robot executing the hammering action.

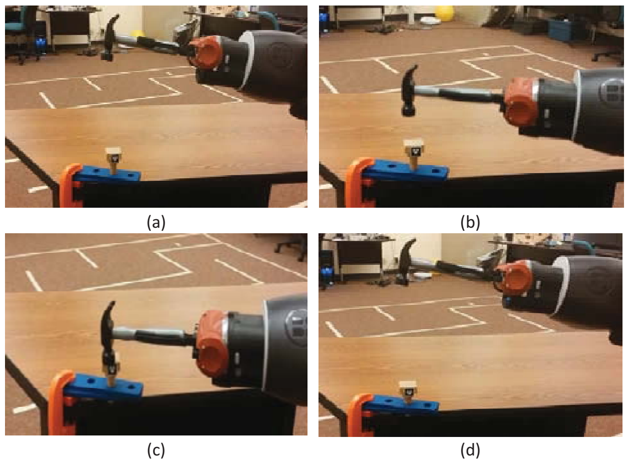

Figure 6.12: Reproducing the hammering action given the bolt pose.

6.3 Script Implementation

6.3.1 Experiment Procedure

After the basic actions are learned, a four-step assembly task is demonstrated to the robot using the PAD system. The detail of the task has been described in the previous chapter.

- Step 1: Inserting a bolt into a baseboard.
- Step 2: Hammering the bolt down.
- Step 3: Inserting a screw into the baseboard.
- Step 4: Screwdriving the screw down.

The four skill scripts generated are:

- Script 1: $\Sigma_1 =< inserting, N/A, bolt, baseboard, S_1 >$
- Script 2: $\Sigma_2 =< hammering, hammer, bolt, baseboard, S_2 >$.
- Script 3: $\Sigma_3 =< inserting, N/A, screw, baseboard, S_3 >$
- Script 4: $\Sigma_4 =< screwdriving, screwdriver, screw, baseboard, S_4 >$

The generated scripts for each step are interpreted as follows: The action A tells the robot which action to apply. The tool T (if used) identifies which tool is used. The robot needs to grasp it before applying the action. The part P_1 identifies the part to which the action is applied. The part P_2 is the fixed reference part. Both of them are tracked in real time using AR markers.

The assembly state S describes the relative pose of the two assembly parts after they are assembled together. The coordinate of the 3D model of each part is defined by the marker frame on the lazy Susan as shown in Fig. 5.2. On the other hand, the robot uses AR markers to track each part. Therefore, for each part, the transformation between the 3D model frame and the AR marker frame has to be calculated. The transformed assembly state is represented as $S^{'}$.

Case 1: $S^{'}_i$ and $S^{'}_{i-1}$ are calculated using the same models.

$$S^{'}_i = (T_{ri} \cdot T_{r2a})^{-1} \cdot S^{'}_{i-1} \cdot T_{pi} \cdot T_{p2a} \tag{6.6}$$

Case 2: $S^{'}_i$ and $S^{'}_{i-1}$ are calculated using different models.

$$S^{'}_i = (T_{ri} \cdot T_{r2a})^{-1} \cdot T_{pi} \cdot T_{p2a} \tag{6.7}$$

$S^{'}_i$ is the transformed assembly state after action A_i. T_{r2a} is the transformation from the reference part 3D model frame to its AR marker frame. T_{p2a} is the transformation from the non-reference part 3D model frame to its AR marker frame. The robot repeats the assembly action A_i until $S^{'}_i$ is reached.

6.3.2 Results

Fig. 6.13 shows the robot inserting the bolt into the base board. Fig. 6.14 shows the robot hammering the bolt down to the base board. It hammers the bolt multiple times (three times in this example) until it reaches the desired pose. Fig. 6.15 shows the screw being inserted into the slot. Fig. 6.16 shows the robot screwdriving the screw down to a hole in the base board. This action is repeated 8 times until the assembly state is reached. To track the screw, two identical markers are attached on the two opposite sides of the screw. The reason is twofold. First, the screw rotates 180 degrees after one screwdriving action, so one marker is not enough to track it. Secondly, due to the

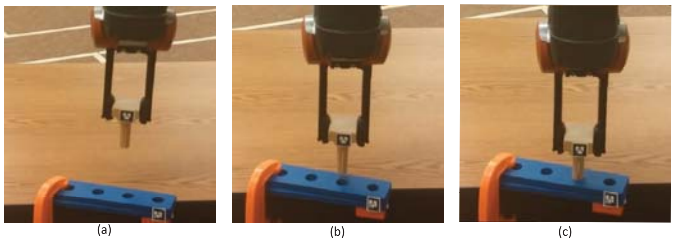

Figure 6.13: The robot implements script 1 (inserting bolt).

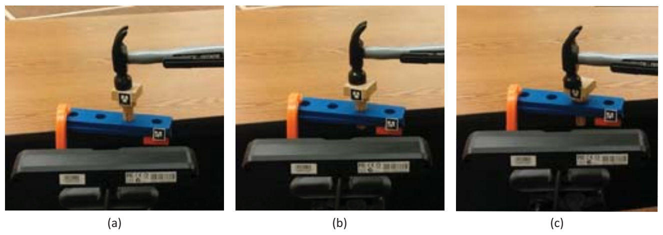

Figure 6.14: The robot implements script 2 (hammering bolt).

symmetric geometry of the screw, the use of two identical markers enables the robot to apply the same learned constraint in either configuration.

The successful rate of implementing each script is also evaluated and is shown in Table. 6.1. Each script is implemented 20 times. The script containing action "hammering" is the simplest for the robot, since the hammer only needs to hit the top of the bolt. On the other hand, the most challenging task for the robot is script 4 which involves "screwdriving", since the slot on the screw is very small. Due to the errors of the robot and the augmented marker tracking algorithm, the robot sometimes fails to insert the screwdriver onto the slot.

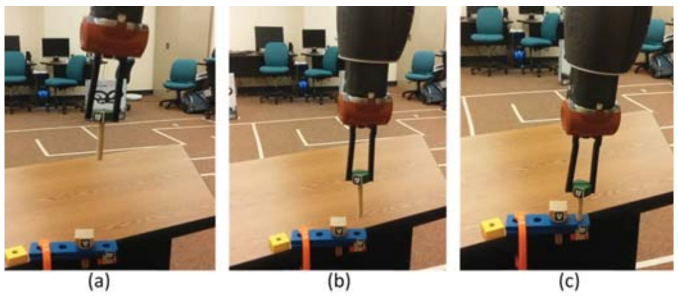

Figure 6.15: The robot implements script 3 (inserting screw).

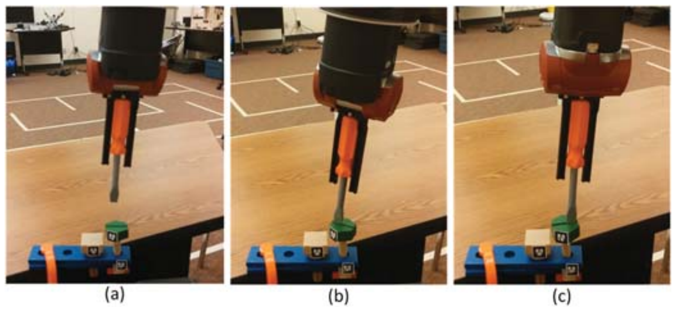

Figure 6.16: The robot implements script 4 (screwdriving).

Table 6.1: Successful rate of script implementation.

script	1	2	3	4
successful rate	0.7	0.9	0.8	0.6

6.4 Summary

This chapter verifies the PAD system with skill implementation on a real robot. The Baxter robot learns the basic actions through scaffolding first. Then, the skill scripts are implemented. The basic actions are learned with the following procedures. First of all, the robot base frame and the Xtion camera frame is calibrated using AR markers. Second, the relative pose between the left gripper and the part is recorded during the demonstration. Augmented markers are used to track parts. Third, GMM/GMR is applied to generalize multiple trajectories from the demonstrations. In each demonstration, both position and orientation information is captured. The generalized trajectory will be reproduced according to the tracked part pose. After learning the basic actions, the robot implements each skill script learned from human demonstration. For assembly action such as hammering, which has big tolerance on sensor errors, the successful rate of implementation is high. However, when it comes to very delicate actions like inserting a bolt into a hole or screwdriving, failures occur more frequently because this type of action is more sensitive to uncertainties caused by the sensor errors.

The grippers on the Baxter robot are only capable of open and close. Therefore, the robot grasps all the parts in the same way. In the future work, a more sophisticated robotic gripper should be used, so that the robot can learn how to grasp objects with different shapes using different approaches. On the other hand, in the implementation part, only positional feedback is considered. However, robotic assembly is a task that frequently requires physical contact between the robot and its environment. When the task contains uncertainties, additional sensing is needed to accomplish the assembly. Using force control is one way to handle these difficulties.

Chapter 7

Conclusions and Future works

7.1 Conclusions

In this dissertation, a robot skill learning framework is proposed which contains low-level skill learning and high-level skill learning. The learned low-level skills are treated as primitive actions in high-level learning. The acquired skills are then implemented on a real robot. The contributions of this dissertation are summarized as follow:

- Chapter 2 proposed a two-phase low level skill learning framework which combines imitation learning and reinforcement learning. In the first phase, using imitation learning the robot can reach out and hold the end of the table. A Programming by Demonstration (PbD) algorithm is used to accomplish this. In the second phase, through reinforcement learning, the robot can learn to collaborate with a human for the table lifting task. With the guided exploration strategy for Q-learning, the learning speed is improved. Using the entire framework, the robot can learn to perform the collaborative table-lifting task quickly and successfully. The learned low-level skills are treated as primitive actionsin high-level skill learning.

- Chapter 3 investigates how to improve the performance of manipulative action recognition through a RGB-D sensor. A multi-level probabilistic model is proposed as a solution. In the low-level, human motion is modeled using an HMM. Joint angle sequence is used as the motion feature. In the mid-level, the dependencies between object and action are modeled in a Bayesian model, actions which are either too subtle to perceive or too similar to discriminate can be recognized with high accuracy. In the high-level, with the action sequential constraints, similar actions on the same object can be differentiated effectively. Because in most human daily activities, actions usually follow certain patterns. Both online and offline experimental results verify that our framework outperforms the approaches that use motion feature only. The experiment on subjects with different body shapes show the robustness of the system.

- Chapter 4 extends Chapter 3's work by using multimodal sensors for fine manipulative action recognition. Two information fusion methods (decision-level and feature-level) are proposed to combine the multiple data sources. In the feature-level fusion method, feature selection is applied to select related features first. Then, HMM models are trained with the selected features for each action. In the decision-level fusion method, features are divided into three groups: arm joint angle group, finger angle group and force feedback group. An HMM-based classifier is built for each feature group. A confidence index is designed. It is calculated using the maximum output (probability) divided by the sum of of all the output. Then, the classifier which has maximum confidence index is chosen to make the global decision. Moreover, with the dependencies between object and action modeled in a Bayesian network, actions which are too similar to discriminate can be recognized with high accuracy.

- Based on the above work, in Chapter 5, a Portable Assembly Demonstration (PAD) system is developed for robots to learn complex assembly skills from humans. With a RGB-D camera, the PAD system can recognize the part/tool used, the action applied and the assembly state characterizing the spatial relationship between the parts. The experiment results proved that this PAD system can generate assembly scripts with good accuracy in object and action recognition as well as assembly state estimation. The skill script includes both symbolic information like the action type and trajectory-level clue as given by the assembly state.

- Chapter 6 verifies the PAD system with skill implementation on a real robot. The Baxter robot learns the basic actions through scaffolding first. Then, the skill scripts are implemented. The basic actions are learned with the following procedures. First of all, the robot base frame and the Xtion camera frame is calibrated using AR markers. Second, the relative pose between the left gripper and the part is recorded during the demonstration. Augmented markers are used to track parts. Third, multiple trajectories from the demonstrations are generalized. The generalized trajectory is reproduced according to the tracked part pose.

7.2 Future Works

The robot skill learning framework we proposed still needs to be further improved, some of the future works are listed below:

- Occlusions frequently happen to the assembled parts during the human demonstration. Algorithms which can handle significant occlusion need be designed and implemented.
- In our action recognition algorithm, only the object type is considered as context. However, it is more informative to know how an object is being used (associated affordances like movable, pourable, drinkable, etc) rather than only knowing what the object type is. Therefore, affordance information should be considered in the action recognition algorithm.
- Human assembly actions sometimes have significant variations. For the same action, different people have their own customized way to perform the action. Sometimes, the action depends on the pose of the manipulated part. Therefore, the action recognition algorithms should be robust to these variations.
- Low-level assembly actions can be represented by the relative motion between subparts and the associated force interactions. An accurate description of such motion and force interaction can help robots quickly learn these assembly actions. How to capture force interactions between subparts is a still an open question which can be further studied.
- In the robot implementation, only positional feedback is used. The sensor errors and uncertainties are not modeled. However, in robotic assembly, the uncertainties or errors introduced

by the part variance may cause implementation failure. When the task contains uncertainties, additional sensing is needed to accomplish the assembly. Using force control is one way to handle these difficulties.

- Currently, the robot always passively observes the demonstration. However, the robot should be able to give social feedback to the human through verbal communication, gestures or expressions if possible.

Bibliography

[1] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "Roboearth," *Robotics Automation Magazine, IEEE*, vol. 18, no. 2, pp. 69–82, 2011.

[2] Baxter robot. [Online]. Available: http://www.rethinkrobotics.com/

[3] H. Wang and X. Liu, "Adaptive shared control for a novel mobile assistive robot," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1725–1736, Dec 2014.

[4] J. Pile and N. Simaan, "Modeling, design, and evaluation of a parallel robot for cochlear implant surgery," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 6, pp. 1746–1755, Dec 2014.

[5] Robohub. [Online]. Available: http://robohub.org/foxbots-being-deployed-in-china/

[6] "Nextage robot. http://nextage.kawada.jp/en/," 2014.

[7] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[8] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *RSJ Advanced Robotics*, pp. 1521–1544, 2007.

[9] C. Atkeson and S. Schaal, "Learning tasks from a single demonstration," in *IEEE International Conference on Robotics and Automation*, vol. 2, apr 1997, pp. 1706–1712.

[10] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *International Symposium on Robotics Research*, 2004.

[11] S. Calinon, *Robot programming by demonstration*. EPFL Press, 2009.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Handbook of robotics chapter 59: Robot programming by demonstration," 2007.

[13] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactions through multiple observations of human demonstrations," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 4, pp. 667–675, 2003.

[14] M. N. Nicolescu and M. J. Mataric, "Natural methods for robot task learning: Instructive demonstrations, generalization and practice," in *In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2003, pp. 241–248.

[15] M. E. Harmon and S. S. Harmon, "Reinforcement learning: A tutorial," *WL/AAFC, WPAFB Ohio*, vol. 45433, 1996.

[16] J.-S. R. Jang and C.-T. Sun, *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, Inc., 1996.

[17] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, 1998.

[18] M. Ito, K. Noda, Y. Hoshino, and J. Tani, "Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model," *Neural Networks*, vol. 19, no. 3, pp. 323–337, 2006.

[19] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, vol. 37, no. 2, pp. 286–298, 2007.

[20] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, sept. 2008, pp. 367–372.

[21] Katana robot. [Online]. Available: http://openrave.org/en/main/ikfast/neuronics-katana.html

[22] S. Calinon, "Incremental learning of gestures by imitation in a humanoid robot," in *In Proceedings of the 2007 ACM/IEEE International Conference on Human-Robot Interaction*, 2007, pp. 255–262.

[23] A. Ude, "Trajectory generation from noisy positions of object features for teaching robot paths," *Robotics and Autonomous Systems*, vol. 11, no. 2, pp. 113–127, 1993.

[24] N. Delson and H. West, "Robot programming by human demonstration: adaptation and inconsistency in constrained motion," in *IEEE International Conference on Robotics and Automation*, vol. 1, April 1996, pp. 30–36.

[25] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactions through multiple observations of human demonstrations," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 4, pp. 667–675, 2003.

[26] M. Pardowitz, S. Knoop, R. Dillmann, and R. Zollner, "Incremental learning of tasks from user demonstrations, past experiences, and vocal comments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, pp. 322–332, 2007.

[27] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109 – 116, 2004.

[28] S. Ekvall and D. Kragic, "Learning task models from multiple human demonstrations," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, sept. 2006, pp. 358–363.

[29] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, “Teaching robots by moulding behavior and scaffolding the environment,” in *IN HUMAN-ROBOT INTERACTION*. ACM Press, 2006, pp. 118–125.

[30] C. Tung and A. Kak, “Automatic learning of assembly tasks using a dataglove system,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, aug 1995, pp. 1–8.

[31] K. Ikeuchi and T. Suchiro, “Towards an assembly plan from observation. i. assembly task recognition using face-contact relations (polyhedral objects),” in *IEEE International Conference on Robotics and Automation*, may 1992, pp. 2171–2177.

[32] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. Legrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, P. Klasnja, K. Koscher, J. Landay, J. Lester, D. Wyatt, and D. Haehnel, “The mobile sensing platform: An embedded activity recognition system,” *IEEE Pervasive Computing,*, vol. 7, no. 2, pp. 32–41, 2008.

[33] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[34] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *Proceedings of the British Machine Vision Conference*, 2009, pp. 1–11.

[35] A. Klser, M. Marszaek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *In BMVC08*, 2008.

[36] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 65–72.

[37] S. Marcel, O. Bernier, J.-E. Viallet, and D. Collobert, “Hand gesture recognition using input-output hidden markov models,” in *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2000, pp. 456–456.

[38] E. Sánchez-Nielsen, L. Antón-Canalís, and M. Hernández-Tejera, “Hand gesture recognition for human-machine interaction,” in *WSCG*, 2004, pp. 395–402.

[39] P. Breuer, C. Eckes, and S. Müller, “Hand gesture recognition with a novel ir time-of-flight range camera–a pilot study,” in *Computer Vision/Computer Graphics Collaboration Techniques*, 2007, pp. 247–260.

[40] J. Sung, C. Ponce, B. Selman, and A. Saxena, “Human activity detection from rgbd images.” *plan, activity, and intent recognition*, vol. 64, 2011.

[41] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” in *Proceedings of the IEEE*, 1989, pp. 257–286.

[42] O. Rashid, A. Al-Hamadi, and B. Michaelis, “Integration of gesture and posture recognition systems for interpreting dynamic meanings using particle filter,” in *International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, dec. 2010, pp. 47–50.

[43] Z. He, “A new feature fusion method for gesture recognition based on 3d accelerometer,” in *Chinese Conference on Pattern Recognition (CCPR)*, oct. 2010, pp. 1–5.

[44] K. Tran, I. A. Kakadiaris, and S. K. Shah, “Fusion of human posture features for continuous action recognition,” in *Proceedings of the 11th European Conference on Trends and Topics in Computer Vision - Volume Part I*, ser. ECCV’10, 2012, pp. 244–257.

[45] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowicz, and G. Troster, “Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario,” in *10th IEEE International Symposium on Wearable Computers*, Oct 2006, pp. 97–104.

[46] P. Lukowicz, J. A. Ward, H. Junker, M. Stger, G. Trster, A. Atrash, and T. Starner, “Recognizing workshop activity using body worn microphones and accelerometers,” in *In Pervasive Computing*, 2004, pp. 18–32.

[47] J. Yang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, “A framework for hand gesture recognition based on accelerometer and emg sensors,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. PP, pp. 1–13, 03 2011.

[48] Y.-T. Chen and K.-T. Tseng, “Multiple-angle hand gesture recognition by fusing svm classifiers,” in *IEEE International Conference on Automation Science and Engineering*, sept. 2007, pp. 527–530.

[49] Q. Wu, Z. Wang, F. Deng, Z. Chi, and D. Feng, “Realistic human action recognition with multimodal feature selection and fusion,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, pp. 875–885, July 2013.

[50] M. Bahrepour, N. Meratnia, Z. Taghikhaki, and P. J. M. Havinga, “Sensor fusion-based activity recognition for parkinson patients,” in *Sensor Fusion - Foundation and Applications*. InTech, June 2011, pp. 171–190.

[51] A. Kojima, M. Takaya, S. Aoki, T. Miyamoto, and K. Fukunaga, “Recognition and textual description of human activities by mobile robot,” in *International Conference on Innovative Computing Information and Control*, june 2008, p. 53.

[52] A. Gupta and L. Davis, “Objects in action: An approach for combining action understanding and object perception,” in *IEEE Conference on Computer Vision and Pattern Recognition*, june 2007, pp. 1–8.

[53] A. D. Wilson, S. Member, I. C. Society, A. F. Bobick, and I. C. Society, “Parametric hidden markov models for gesture recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 884–900, 1999.

[54] B. Yao and L. Fei-Fei, "Recognizing human-object interactions in still images by modeling the mutual context of objects and human poses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1691–1703, 2012.

[55] H. Kjellstrom, J. Romero, and D. Kragic, "Visual object-action recognition: Inferring object affordances from human demonstration," *Computer Vision and Image Understanding*, vol. 115, no. 1, pp. 81 – 90, 2011.

[56] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin, "Robot programming by demonstration (rpd): Supporting the induction by human interaction," *Machine Learning*, vol. 23, no. 2-3, pp. 163–189, 1996.

[57] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 367–372.

[58] C. Zhu and W. Sheng, "Realtime human daily activity recognition through fusion of motion and location data," in *2010 IEEE International Conference on Information and Automation*, june 2010, pp. 846–851.

[59] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[60] R. Zollner, O. Rogalla, and R. Dillmann, "Integration of tactile sensors in a programming by demonstration system," in *IEEE International Conference on Robotics and Automation*, vol. 3, 2001, pp. 2578–2583.

[61] S. Lee and H. Suh, "Skill learning and inference framework for skilligent robot," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 108–115.

[62] L. Kunze, A. Haidu, and M. Beetz, "Acquiring task models for imitation learning through games with a purpose," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 102–107.

[63] S. Ahmadzadeh, P. Kormushev, and D. Caldwell, "Visuospatial skill learning for object reconfiguration tasks," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 685–691.

[64] Y. C. Song and H. A. Kautz, "A testbed for learning by demonstration from natural language and rgb-depth video," in *AAAI*, 2012.

[65] G. Hovland, P. Sikka, and B. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," in *IEEE International Conference on Robotics and Automation*, vol. 3, 1996, pp. 2706–2711 vol.3.

[66] N. Dantam, I. Essa, and M. Stilman, "Linguistic transfer of human assembly tasks to robots," in *International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 237–242.

[67] J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Recognizing assembly tasks through human demonstration," *Int. J. Rob. Res.*, vol. 26, no. 7, pp. 641–659, Jul. 2007.

[68] J. Aleotti, S. Caselli, and M. Reggiani, "Toward programming of assembly tasks by demonstration in virtual environments," in *IEEE International Workshop on Robot and Human Interactive Communication*, 2003, pp. 309–314.

[69] M. B. Andrea Bauer, Dirk Wollherr, "Human-robot collaboration: A survey," *International Journal of Humanoid Robotics (IJHR)*, vol. 5, pp. 47 – 66, 2008.

[70] Vicon mx motion capture system. [Online]. Available: http://www.vicon.com/products/

[71] Aldebaran humanoid robot. [Online]. Available: http://www.aldebaran-robotics.com/en

[72] D. Cohn, Z. Ghahramani, and M. Jordan, "Active learning with statistical models." *Artificial Intelligence Research*, no. 4, pp. 129 – 148, 1996.

[73] G. Schwarz, "Estimating the dimension of a model." *Annals of Statistics*, vol. 6, pp. 461 – 464, 1978.

[74] S. Calinon, *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL/CRC Press, 2009.

[75] H. Gonzalez-Jorge, B. Riveiro, E. Vazquez-Fernandez, J. Martínez-Sánchez, and P. Arias, “Metrological evaluation of microsoft kinect and asus xtion sensors,” *Measurement*, vol. 46, no. 6, pp. 1800–1806, 2013.

[76] A. D. Milner and M. A. Goodale, *The Visual Brain in Action*, ser. Oxford Psychology Series. Oxford University Press, 1995, no. 27.

[77] D. N. Bub and M. E. J. Masson, “Gestural knowledge evoked by objects as part of conceptual representations,” *Aphasiology*, vol. 20, pp. 1112–1124, 2006.

[78] H. B. Helbig, M. Graf, and M. Kiefer, “The role of action representations in visual object recognition,” *Experimental Brain Research*, vol. 174, pp. 221–228, 2006.

[79] Z. Zhang, “Microsoft kinect sensor and its effect,” *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, 2012.

[80] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view rgb-d object dataset,” in *IEEE International Conference on Robotics and Automation (ICRA)*, may 2011, pp. 1817–1824.

[81] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.

[82] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, “Sequence of the most informative joints (smij): A new representation for human skeletal action recognition,” *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 24–38, 2014.

[83] D. Di Marco, A. Koch, O. Zweigle, K. Haussermann, B. Schiessle, P. Levi, D. Galvez-Lopez, L. Riazuelo, J. Civera, J. Montiel, M. Tenorth, A. Perzylo, M. Waibel, and R. van de

Molengraft, “Creating and using roboearth object models,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 3549–3550.

[84] H. Bay, T. Tuytelaars, and L. V. Gool, “Surf: Speeded up robust features,” in *In ECCV*, 2006, pp. 404–417.

[85] Y. Wu, T. S. Huang, and N. Mathews, “Vision-based gesture recognition: A review,” in *Lecture Notes in Computer Science*, 1999, pp. 103–115.

[86] Openni. [Online]. Available: http://www.openni.org/Documentation/

[87] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: Analysis and implementation,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, 2000, pp. 456–461.

[88] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society Series B Methodological*, vol. 39, no. 1, pp. 1–38, 1977.

[89] Y. Gu, H. Do, J. Evert, and W. Sheng, “Human gesture recognition through a kinect sensor,” in *IEEE International Conference on Robotics and Biomimetics*, Dec. 2012.

[90] M. Müller and T. Röder, “Motion templates for automatic classification and retrieval of motion capture data,” in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2006, pp. 137–146.

[91] J. Wang, Z. Liu, Y. Wu, and J. Yuan, “Mining actionlet ensemble for action recognition with depth cameras,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1290–1297.

[92] L. Xia and J. Aggarwal, “Spatio-temporal depth cuboid similarity feature for activity recognition using depth camera,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2834–2841.

[93] Z. Liu. [Online]. Available: http://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/

[94] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 16:1–16:43, Apr 2011.

[95] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowicz, and G. Troster, "Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario," in *2006 10th IEEE International Symposium on Wearable Computers*, 2006, pp. 97–104.

[96] Cybergloves. [Online]. Available: http://www.cyberglovesystems.com/

[97] Flexiforce sensors. [Online]. Available: http://www.tekscan.com/flexiforce.html

[98] J. A. Titus, *The Hands-on XBEE Lab Manual: Experiments That Teach You XBEE Wirelesss Communications*. Elsevier, 2012.

[99] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[100] L. A. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012.

[101] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *IEEE International Conference on Image Processing (ICIP)*, 2011, pp. 809–812.

[102] ——, "Unique signature of histograms for local surface description," in *Proceedings of the 11th European Conference on Computer Vision*, 2010, pp. 356–369.

[103] F. Tombari and L. Di Stefano, "Object recognition in 3d scenes with occlusions and clutter by hough voting," in *Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, 2010, pp. 349–355.

[104] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[105] S. Das, “Filters, wrappers and a boosting-based hybrid for feature selection,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 74–81.

[106] Z. Zhao, J. Cox, D. Duling, and W. Sarle, “Massively parallel feature selection: An approach based on variance preservation,” pp. 195–220, 2013.

[107] K. Lai, L. Bo, X. Ren, and D. Fox, “Sparse distance learning for object recognition combining rgb and depth information,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 4007–4013.

[108] J. Lei, X. Ren, and D. Fox, “Fine-grained kitchen activity recognition using rgb-d,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, 2012, pp. 208–211.

[109] N. Payet and S. Todorovic, “From contours to 3d object detection and pose estimation,” in *IEEE International Conference on Computer Vision*, Nov 2011, pp. 983–990.

[110] P. Azad, T. Asfour, and R. Dillmann, “Toward an unified representation for imitation of human motion on humanoids,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2558–2563.

[111] R. B. Rusu, “Semantic 3d object maps for everyday manipulation in human living environments,” Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[112] R. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3212–3217.

[113] P. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[114] K. Shoemake, “Animating rotation with quaternion curves,” in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

[115] E. Guizzo and E. Ackerman, "How rethink robotics built its new baxter robot worker," *IEEE Spectrum*, 2012.

[116] ——, "The rise of the robot worker," *Spectrum, IEEE*, vol. 49, no. 10, pp. 34–41, October 2012.

[117] Baxter kinect calibration. [Online]. Available: https://github.com/h2r/baxter_h2r_packages

[118] (2014) Alvar tracking library. [Online]. Available: http://graphics.cs.columbia.edu/wiki/doku.php?id=alvar

[119] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[120] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl*, vol. 21, pp. 1253–1278, 2000.

VITA

Ye Gu

Candidate for the Degree of

Doctor of Philosophy

Thesis: ROBOT SKILL LEARNING THROUGH HUMAN DEMONSTRATION AND INTERACTION

Major Field: ELECTRICAL ENGINEERING

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in your major at Oklahoma State University, Stillwater, Oklahoma in May, 2015.

Completed the requirements for the Master of Science in Engineering Management at University of Minnesota Duluth, Duluth, Minnesota/United States in 2010.

Completed the requirements for the Bachelor of Engineering in Electrical Engineering at Harbin Institute of Technology, Harbin, Heilongjiang/China in 2007.

Experience:

4+ years experiences in image processing for pattern recognition.
4+ years experiences in supervised and unsupervised robot skill learning.
Extensive experience in intelligent visual and wearable sensing, human robot interaction.