Robot Learning Dual-Arm Manipulation Tasks by Trial-and-Error and Multiple Human Demonstrations

Sulabh Kumra

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Robot Learning Dual-Arm Manipulation Tasks by Trial-and-Error and Multiple Human Demonstrations

A Thesis

By

Sulabh Kumra

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering

Supervised by
Professor Dr. Ferat Sahin

Department of Electrical and Microelectronic Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
July 2015

Approved by:

Dr. Ferat Sahin, Professor
Thesis Advisor, Department of Electrical and Microelectronic Engineering

Dr. Gill R. Tsouri, Associate Professor
Committee Member, Department of Electrical and Microelectronic Engineering

Dr. Sildomar T. Monteiro, Assistant Professor
Committee Member, Department of Electrical and Microelectronic Engineering

Dr. Sohail A. Dianat, Professor
Department Head, Department of Electrical and Microelectronic Engineering
To my beloved parents Dr Krishan and Dr Sangeeta, and my precious sister Jigyasa.
ACKNOWLEDGMENTS

Along this research road, I have met brilliant people who played a vital role in accomplishing this goal. First, I would like to thank my advisor, Dr. Ferat Sahin for continuous guidance, supervision and support. His words of inspiration and wisdom have found themselves contributing to the basis of my success.

I would like to thank Rochester Institute of Technology for providing the resources needed and providing me the opportunity to work as a Graduate Teaching Assistant and Graduate Research Assistant during my research period.

Shitij, thank you for being an excellent mentor and a great friend. You have been my inspiration since my first Principles of Robotics lab under you.

I am grateful to all my peers and fellow researchers at the Multi-Agent Biorobotics Laboratory. Each of you has contributed your unique roles in the completion of this work.

For the rest not mentioned I thank you all.
ABSTRACT

In robotics, there is a need of an interactive and expedite learning method as experience is expensive. In this research, we propose two different methods to make a humanoid robot learn manipulation tasks: Learning by trial-and-error, and Learning from demonstrations. Just like the way a child learns a new task assigned to him by trying all possible alternatives and further learning from his mistakes, the robot learns in the same manner in learning by trial-and error. We used Q-learning algorithm, in which the robot tries all the possible ways to do a task and creates a matrix that consists of Q-values based on the rewards it received for the actions performed. Using this method, the robot was made to learn dance moves based on a music track.

Robot Learning from Demonstrations (RLfD) enable a human user to add new capabilities to a robot in an intuitive manner without explicitly reprogramming it. In this method, the robot learns skill from demonstrations performed by a human teacher. The robot extracts features from each demonstration called as key-points and learns a model of the demonstrated task or trajectory using Hidden Markov Model (HMM). The learned model is further used to produce a generalized trajectory. In the end, we discuss the differences between two developed systems and make conclusions based on the experiments performed.
TABLE OF CONTENTS

Chapter                                                                 Pages

ACKNOWLEDGMENTS ........................................................................................................ iii

ABSTRACT ....................................................................................................................... iv

TABLE OF CONTENTS .................................................................................................... v

LIST OF TABLES ........................................................................................................... viii

LIST OF FIGURES ........................................................................................................... ix

LIST OF ABBREVIATIONS ............................................................................................ xi

CHAPTER 1: INTRODUCTION ....................................................................................... 1
  1.1 Motivation ................................................................................................................. 2
  1.2 Contributions .......................................................................................................... 3
  1.3 Publication .............................................................................................................. 4

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW ........................................ 5
  2.1 Robot Learning to Dance .......................................................................................... 5
  2.2 Robot Learning from Demonstration ........................................................................ 8
    2.2.1 What to learn? .................................................................................................... 9
    2.2.2 How to learn? ................................................................................................... 10
    2.2.3 HMIs for Demonstration .................................................................................. 11

CHAPTER 3: LEARNING BY TRIAL AND ERROR ..................................................... 15
  3.1 System Configuration ............................................................................................. 16
  3.2 Beat Tracking System ........................................................................................... 18
  3.3 Robot Motion Learning System .............................................................................. 20
    3.3.1 Q-learning Algorithm ....................................................................................... 20
3.3.2 State and Action ............................................................................................... 23
3.3.3 Selection Function ........................................................................................... 23
3.3.4 Update Function ............................................................................................... 24
3.4 Experiments ............................................................................................................ 24
3.5 Discussion ............................................................................................................... 26

CHAPTER 4: LEARNING FROM DEMONSTRATIONS ............................................. 28
4.1 Problem Statement .................................................................................................. 29
4.2 Proposed Approach ................................................................................................. 30
4.3 Data Collection ....................................................................................................... 32
  4.3.1 Demonstrations ................................................................................................ 32
  4.3.2 Forward Kinematics ......................................................................................... 34
  4.3.3 Data storage ..................................................................................................... 37
4.4 Task Learning ......................................................................................................... 37
  4.4.1 Key-point Extraction ........................................................................................ 37
  4.4.2 Clustering ......................................................................................................... 41
  4.4.3 Model Learning using HMM ........................................................................... 42
  4.4.4 Dynamic Time Warping .................................................................................. 46
  4.4.5 Generalized Trajectory ..................................................................................... 47
4.5 Robot Execution...................................................................................................... 48
4.6 BLInK ..................................................................................................................... 48
4.7 Experiments ............................................................................................................ 49
  4.7.1 Stacking cup task ............................................................................................. 52
  4.7.2 Pick and place task ........................................................................................... 56
4.8 Discussion ............................................................................................................... 61

CHAPTER 5: CONCLUSION ......................................................................................... 63

CHAPTER 6: FUTURE WORK ...................................................................................... 65

BIBLIOGRAPHY ............................................................................................................. 66

Appendix A: Baxter Arm Specifications ................................................................. 69

A1. Joint Names............................................................................................................ 69

A2. Link Lengths ........................................................................................................ 69

A3. Range of Motion - Bend Joints ........................................................................... 70

A4. Range of Motion - Twist Joints .......................................................................... 71

A5. Joint Sensor Resolution ....................................................................................... 73

A6. Miscellaneous ..................................................................................................... 73
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1: Detected beat rate</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>Table 2: DH table</td>
<td></td>
<td>37</td>
</tr>
<tr>
<td>Table 3: Data statistics for stacking cup task</td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>Table 4: Data statistics for pick and place task</td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>Table 5: Comparison of two proposed methods</td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>Table 6: Link Length Table (Bend Joints)</td>
<td></td>
<td>71</td>
</tr>
<tr>
<td>Table 7: Link Length Table (Twist Joints)</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Table 8: Flexure Stiffness (K)</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Table 9: S1 Spring Specs</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Table 10: Peak Torque Per Joint</td>
<td></td>
<td>73</td>
</tr>
<tr>
<td>Table 11: Camera Specifications</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>Table 12: On Board CPU</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>Table 13: Component Weights</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>Table 14: Electrical Power</td>
<td></td>
<td>75</td>
</tr>
<tr>
<td>Table 15: Miscellaneous Specifications</td>
<td></td>
<td>76</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1: Robot playmates Jimmy and Jenny in the playground [25]</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2: Full Body Imitation with Balance Control [28]</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3: Control of humanoid robot using master exoskeleton [30]</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4: Simon interacting with a teacher [31]</td>
<td>13</td>
</tr>
<tr>
<td>Figure 5: Tele-operation using haptic device [32]</td>
<td>14</td>
</tr>
<tr>
<td>Figure 6: Overall configuration of system</td>
<td>17</td>
</tr>
<tr>
<td>Figure 7: Typical beat tracking steps. (a) The input musical audio signal. (b) Amplitude spectrum of STFT of the signal. (c) Complex domain onset detection function. (d) A demonstration of beat tracking with dotted vertical lines representing beat locations</td>
<td>19</td>
</tr>
<tr>
<td>Figure 8: Q learning flow chart</td>
<td>23</td>
</tr>
<tr>
<td>Figure 9: Demonstration of Baxter dancing on ‘We Will Rock You’ after learning the moves</td>
<td>26</td>
</tr>
<tr>
<td>Figure 10: Control policy learning and execution</td>
<td>30</td>
</tr>
<tr>
<td>Figure 11: Block diagram of the proposed RLfD system</td>
<td>31</td>
</tr>
<tr>
<td>Figure 12: Perform demonstration using kinesthetic teaching</td>
<td>33</td>
</tr>
<tr>
<td>Figure 13: Joint names of left limb of Baxter [40]</td>
<td>35</td>
</tr>
<tr>
<td>Figure 14: Approach to mathematically modeling Baxter’s arm</td>
<td>36</td>
</tr>
<tr>
<td>Figure 15: Key-point selection criteria</td>
<td>38</td>
</tr>
<tr>
<td>Figure 16: Detected key-points from a trajectory</td>
<td>40</td>
</tr>
<tr>
<td>Figure 17: Extracted key-points from multiple demonstrations</td>
<td>41</td>
</tr>
<tr>
<td>Figure 18: Centroids of the k clusters</td>
<td>42</td>
</tr>
</tbody>
</table>
Figure 19: Graphical representation of HMM [43] ........................................................... 43
Figure 20: BLInK home screen ......................................................................................... 49
Figure 21: Centroids extracted from pick and place task ................................................. 50
Figure 22: Centroids extracted from multiple demonstrations of painting task .......... 51
Figure 23: Demonstration of stacking cup task ................................................................. 53
Figure 24: Robot performing learned stacking cup task .................................................. 54
Figure 25: Demonstrated and learned trajectory for stacking cup task ......................... 55
Figure 26: Demonstration of pick and place task ............................................................... 58
Figure 27: Robot performing learned pick and place task ............................................... 59
Figure 28: Demonstrated and learned trajectory for pick and place task ....................... 60
Figure 29: Joint names .................................................................................................... 69
Figure 30: Link lengths .................................................................................................... 70
Figure 31: Range of Motion (Bend Joints) .................................................................... 70
Figure 32: Range of Motion (Twist Joints) ..................................................................... 71
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF</td>
<td>Autocorrelation Function</td>
</tr>
<tr>
<td>BLInK</td>
<td>Baxter Learning Interface by Kumra</td>
</tr>
<tr>
<td>BPM</td>
<td>Beats per Minute</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>FK</td>
<td>Forward Kinematics</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>IK</td>
<td>Inverse Kinematics</td>
</tr>
<tr>
<td>JTAS</td>
<td>Joint Trajectory Action Server</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>RLfD</td>
<td>Robot Learning from Demonstrations</td>
</tr>
<tr>
<td>SARSA</td>
<td>State Action Reward State Action</td>
</tr>
<tr>
<td>SoS</td>
<td>System of Systems</td>
</tr>
<tr>
<td>STFT</td>
<td>Short-term Fourier Transform</td>
</tr>
</tbody>
</table>
CHAPTER 1: INTRODUCTION

This chapter provides an overview of the thesis and its contributions to increase the learning and interactive capabilities of robots through teaching them tasks by demonstrations. It discusses the main challenges involved in this process and the assumptions over which the proposed solution is based. It also outlines the approach used to facilitate the transfer of task knowledge from demonstrations.

Nowadays, most of the robots used in the industry are pre-programmed and necessitate a well-defined and controlled environment. Reprogramming these robots is often an expensive process necessitating an expert. Enabling the robot to learn tasks by demonstrating them would streamline the robot installation and task reprogramming. In a long-term perspective, the vision is that robots will move out of factories into our homes and offices. Robots should be able to learn day-to-day tasks at homes like cleaning, cooking food, and filling dishwasher or washing machine. Apparently, robot-learning techniques are required to enable robots to adapt and work in a dynamic environment, in contrast to well-defined factory environment. That is why robot learning is one of the key research areas in the field of robotics. However, constructing a robot that is able to learn by observation is still a challenging problem. Although prototype platforms for robot learning by demonstration have been around for more than a decade, many complications have restrained the robots to operate only in restricted laboratory environments. Some of the key challenges are perception, task recognition, task generalization, motion planning, and object manipulation.
The work presented in this thesis aims to provide robots with task learning capabilities that address some of the problems stated above, and thus reduce the amount of time and expertise required for development of an intelligent autonomous robot system. A natural approach for this problem will be to have robots learn tasks from a teacher’s demonstration, consequently increasing the ability of robots to interact with people and relieving the user from writing the program by hand.

The motivation and objectives of this thesis are discussed in Sections 1.1 and 1.2, respectively.

1.1 Motivation
The possibilities of robotics applied in a wide range of fields are encouraging. Robots are expected to aid human beings in homes, medicinal services and numerous different fields. Although, until today robots are mainly utilized at production line floors, these robots are pre-programmed to perform well-defined tasks. In these tasks, robots effectively outflank human beings in both accuracy and speed, since they work in a well-defined environment [1] [2] [3] [4]. In such environments, robots have practically no vulnerability about the condition of their surroundings. The manner in which robots operate have to change in order to empower them to leave production line floors and become the adaptable robots of which humans have been dreaming for ages. The mechanical design and the control architecture of robots should be designed in such a way that it enable robots to learn from their surroundings and allow safe Human Robot interaction. Learning is an imperative perspective in making flexible robots. Pre-programming a robot to attain a wide-ranging skill set in a continually changing environment is unfeasible. It is not possible for
programmers to program a robot’s behavior for every possible situation. Robot learning algorithms will empower robots to learn new abilities and acclimate their abilities to the changing environment.

Human beings have the capability to learn novel activity representations regardless of the noisy sensory input by using previously learnt contextual knowledge, since various human activities often share the analogous underlying structures. For example, when we observe someone performing a task in which his hand is transferring an object to some another place, where the grasping action is not visible because of some obstructions, we can still infer that a grasping action must have occurred before the object was lifted. From this analogy, if a robot has information about a minimal set of actions, which are repeatedly used in human robot interaction environments; it can increase the performance of learning new tasks. The use of such knowledge enables a learner to incrementally acquire a new knowledge without the need of excessive verification processes, resulting in a more natural interaction.

1.2 Contributions

The main contributions for this research are:

- Development of a dance learning system for a robot that consists of two systems: Beat tracking system and Robot motion learning system. Implemented Q-learning algorithm.

- Design of a novel Robot Learning from Development (RLfD) approach to enable a robot to learn trajectories from human demonstrations. Implemented key-point
extracted, k-means clusters, Hidden Markov Model (HMM), Dynamic Time Warping (DTW) algorithms.

- Derivation of forward kinematics (FK) and inverse kinematics (IK) equations for a dual-arm 7DoF robot.

- Development of a terminal based Human Machine Interface for the developed RLfD system called BLInK (Baxter Learning Interface by Kumra).

- Comparison and analysis of results of the two learning methods developed.

1.3 Publication

CHAPTER 2: BACKGROUND AND LITERATURE REVIEW

This chapter presents a literature review of the Machine Learning and Artificial Intelligence algorithms used in our work. First, it presents examples of robot learning how to dance on its own by extracting key features from music tracks. Next, it discusses current methods and work done in the field of robot learning from demonstration (RLfD). The chapter also discusses existing techniques for robot learning from multiple demonstrations.

For human beings, one of the rudiments of social behavior is to understand each other’s intentions, and learning from observation and interaction. Skill transfer and learning have been well studied and the most common forms are demonstration, observation, verbal instruction and physical guidance [5]. When learning a complex task, a significant number of features have to be extracted and processed. Other way of learning skills is to learn skills by trials and exploration. In the next section, work done related to robot learning to dance on a music track has been discussed.

2.1 Robot Learning to Dance

As the public has begun to have more interest in the robotic field, we can easily see that many companies and researchers are trying to prepare and provide events with robotics system. For example, 20 Nao Robots, Aldebaran Robotics’ humanoid robot, demonstrated a synchronized dance on France Pavilion Day [6]. Because a dancing robot can be an important part in such events, there have been many related studies for dancing robots. There are several different approaches that are used in the generation of dance movements
in humanoid robots [7] [8] [9]. Worldwide, robotics and artificial intelligence researchers are attempting to make robots dance to the sound of music tracks [10], and make them participate in collaborative musical performances with humans [11]. One method that was used before was the one in which the user sets different joint positions of the robot for different time intervals and using a smoothing algorithm these frames are joint together and a dance motion is generated. Another approach and a more popular one is to make the robot dance through imitation learning in which the robot observes the dance motions of the human and extracts these motions and transfers them to their robotic joints [12] [13]. The method being used in this paper is based on non-IEC approach in which a fitness function is developed and keyframe concept is used in which values will be chosen by a genetic algorithm [14] [15].

Several methods have been used until date for making a robot dance as mentioned before but making it to learn how to dance with both IEC and non-IEC methods, comparing both methods and making it to learn how to dance by synchronization with the beats of the music has not been implemented. The first advantage of the proposed algorithms is to try to reduce human inputs and make the robot learn dance-like behaviors using less or no human interactions. Secondly, by using reinforcement learning [16] we make the robot dance in synchronization with the music and improve its dancing to music. Three diverse Q-learning algorithms have been used to learn complex behaviors using layered Reinforcement Learning [17]. In [18], as a preliminary, they developed a general experience replay framework which can be combined with essentially any incremental reinforcement learning technique and instantiate this framework for approximate Q-learning and SARSA algorithms. In another paper, we can see an adaptation mechanism
based on reinforcement learning that can read subconscious body signals from a human companion, and can then use this information to adjust gaze meeting, interaction distances and motion speed and timing in human-robot interactions [19]. Reinforcement Learning is used with Decision Trees, where it uses decision trees to learn the model by generalizing the relative effect of actions across the states [20]. In this, the combination of the learning approach with the targeted exploration policy enables fast learning of the model. Kamio et al integrated the technique of genetic programming and reinforcement learning to enable the robot to adapt its actions to the real world environment [21]. To modulate the mixture of dynamical systems initialized from user demonstration, Kormushev et al. implemented Expectation Maximization (EM) based Reinforcement Learning. They conducted two experiments: one was a reaching task, where the robot has to reach a position while avoiding obstacles and other was a dynamic pancake-flipping task [22]. Jens et al present a survey on state of the art reinforcement learning in robotics. In their work, they highlighted the key challenges faced in robot reinforcement learning, and compared various successful applications in the field of robotics. Our work is similar to the reward based learning applications described in the survey [23].

Marek P. Michalowski et al. have pursued research in the same manner [24]. They developed a robotic system that can interact with users. They used a small robot, Keepon, which is able to synchronize its movement with those of a child on a pressure-sensor board. The child on the board dances to the given music and the pressure-sensor sends data to the robotic system. Then, Keepon can synchronize its movement with those of the child’s rhythmic dancing. With the system, the researchers showed the importance of rhythmic synchrony in social interaction. Guy Hoffman and Keinan Vanunu [25] performed research
in a similar way as well. They studied various effects of robotic companionship on agent perception and music enjoyment.

In the next section, concept of robot learning from demonstration and recent work in this field is discussed.

### 2.2 Robot Learning from Demonstration

The presence of robots in society has become even more prevalent. Whether it is a highly sophisticated exploration rover in space, or an entertaining robot at home, a successful autonomous operation of the robot requires a robust control algorithm. To develop such control algorithm, a policy development process is required, which is restricted to experts in the field. Non-robotics experts might know how to interact with robots, and have an idea about what a robot should do. However, they are restricted to do anything as they lack the extensive knowledge required to develop the desired control algorithm.

Robot Learning from Demonstration (RLfD) [26] is a technique to enable a robot to perform new tasks autonomously. Instead of necessitating users to logically decompose and manually program a robot for a desired behavior, RLfD enables to derive robot controller by observing human's own performance. The goal is to easily extend and adapt robot capabilities to novel situations, even by users who lag programming ability. Robot learns a model of a task based on the demonstrations performed by the teacher.

Work in the field of Robot Learning from Demonstration started in 1980s. However, until date most of the robots are tediously hand programmed for each of the task they have to perform. RLfD tries to minimize this challenging step by allowing users to
teach their robot to fit in their specific needs. The belief is that the techniques of RLfD are user friendly and enable robots to be employed more widely in daily activities with no need of specialist humans. Moreover, by using expert knowledge of human teacher, in form of demonstrations, the overall learning process would be faster as compared to learning by trial-and-error, mainly in high-dimensional spaces.

2.2.1 What to learn?

What to learn from demonstration is essentially the problem of finding which features of the demonstration need to be learned and modelled. For a specific task, some observable features of the environment may be unrelated and can be ignored. The way to figure out what is important to learn from demonstration is by understanding the metric used to estimate robot's behavior. If the objective is to have a robot learn a certain sequence of movements (e.g., waving hand), then learning to reproduce the trajectory demonstrated by the teacher is adequate. This is based on the assumption that the environment remains the same during the robot’s performance and thus it cannot influence robot’s behavior.

![Figure 1: Robot playmates Jimmy and Jenny in the playground](image)
In Figure 1, the robots are trained to arrange boxes by size, and not by their color. This means that the metric used to define whether the robots have successfully performed the task is based only on the size of the boxes, and not their color [27]. Teaching what is important and what is not can be done in several ways. One of the ways is to take a statistical perspective and consider the relevant parts (i.e. region of input space, dimension, etc) of the recorded data, which are consistent across all demonstrations [28]. In case the dimension of the recorded data is too high, large number of demonstrations is required to gather sufficient statistics. Another way is to have the teacher aid the robot decide what is relevant by pointing out parts of the demonstrated task that are most significant [29].

To make a robot learn in dynamic environments, the task that the robot learn should depend on and influenced by the state of the environment. For example, to learn grasping capability, the robot should not only record the exact trajectory of a specific instance of a demonstration (say, grasping a coffee mug from the table), otherwise it might fail to grasp the object in a dynamic environment. This happens because of the fact that the robot has no exemplification of the high-level goals of the task. In this thesis, we focus on learning by creating a model of the task, which is balance between precise trajectories demonstrated by the teacher and high-level task representations.

2.2.2 How to learn?
How to learn is the problem of defining how the robot should really perform the learned task to maximize the metric created while solving the ‘What to learn’ problem. Frequently, because of physical differences, a robot cannot perform a task in the exact same way as a
human did. For example, if the teacher uses a foot to push an object, it might not be possible for a wheeled robot to bump it.

Although, humans and robots inhabit in same space, interact with same objects, and superficially analogous, they still interact and perceive the world in dissimilar ways. To assess the similarities between robot and human behaviors, we should understand that humans and robots might occupy different state spaces, of possibly different dimensions. States of human (demonstrator) and robot (imitator) can correspond in two different ways: Perceptual equivalence and Physical equivalence [26].

2.2.3 HMIs for Demonstration

The Human Machine Interface (HMI) used to perform demonstrations plays a crucial role in the manner the data is collected and transmitted. In this section, we discuss three main techniques.

The simplest method is directly recording human motions. This method is used when key interest is in kinematics of the demonstrated motions. Any of the existing motion tracking systems, like exoskeleton, vision based system, or similar wearable motion
sensors can be used. Figure 2 shows full body imitation being used for learning balance control of the robot [30]. Motion tracking sensor is used to capture operator arms motion and then robot imitates operator arm motion for Programming by Demonstration for a Dual Arm Robot [31].

Figure 3 shows an example of controlling a humanoid robot Dexto:Eka: using a master exoskeleton [32]. The robot mimics the motion of the operator using the exoskeleton worn by a tele-operator. Joint values of the two arms of the tele-operator are measured using the exoskeleton and mapped to the joints of the robotic arm. These external means of motion tracking are advantageous as they provide accurate measurements of the angular displacement of most of the human joints. Although this method also let the human operator to freely move all the joints, it needs a good solution to the mapping problem.

Another method of transferring the skill from the teacher to the robot is using ‘kinesthetic teaching’. In this method, the teacher physically guides the robot through the
task. One of the advantage of this approach is that the physical mapping is not required, as the teacher uses robot's body to demonstrate the task. Moreover, it provides a natural training interface for a robot to learn a task from human teacher. For these reasons, in this thesis, we are using kinesthetic teaching to teach the robot various tasks.

Two methods of interaction for kinesthetic teaching in RLfD has been compared by Baris et al [33]. They studied the effects of different demonstrations and concluded that keyframe and trajectory demonstrations have their relative advantages. Based on these conclusions, they introduced a hybrid mode of interaction in which the user can chain together trajectory and keyframe segments.

Going further than kinesthetic teaching, immersive tele-operation can be used, which not only limits the human teacher to use the robot's end-effectors and sensors to perform a task, but also limits to the human teacher’s perception to those of the robot. Figure 5 shows HRP-2 robot being taught by a human teacher how to lift a beam of wood.
in collaboration with another human [34]. As this task requires transfer of force, the robot is being tele-operated by another human using a haptic device. This device enables the human teacher to sense the forces applied at the end-effector of the robot and then adapt the motion accordingly. The robot learns the task from recording the motion induced by the human and from recording the forces perceived at the robot’s end-effector.

Figure 5: Tele-operation using haptic device [34]
CHAPTER 3: LEARNING BY TRIAL AND ERROR

This chapter presents the first learning method that was designed to make a dual flexible 7DoF arm robot learn to dance as a child does. We discuss the two systems that were developed to extract features from a music track and then learn dance moves based on the features of the music track. In the end, we discuss the advantages and disadvantages of such a system.

Dance has always played an important role in human development and behavior. From a complex point of view, dance can be seen as a mean of nonverbal communication, although people are usually interested only in its artistic and playful side. Most of the times dance is strongly associated with music and its movements depend on well-defined music properties. Rhythm is the music element that most influences the dancing performance, and it includes several aspects such as the beat, the meter and the tempo. Although dance is something commonly done by humans and some animals, it seems appropriate to extend it to robotics. This extension will for sure provide new form of entertainment.

From a researcher’s point of view, robot dance is one way to show the developments of robotic technology. Especially, a robot’s synchronized motions with certain music can make some people think that the robot is intelligent, as it seems to understand the music. However, most of the robots are pre-programmed for a specific music. All the movements are made manually and the synchronization process is done manually as well. This process has one advantage that the robot’s dance is at a choreography level but it also has a disadvantage that programmers have to decide what kind of motions will be shown and when those motions should be shown. Even if they
accomplish these tasks, the process is highly inflexible and cannot be applied to delayed music input or any other (not pre-programmed) music input. Other methods employed involve direct interaction with, or imitation of, human participant.

For these reasons, we thought that two disadvantages of current robot dance technology could be solved by giving the robot’s ability to dance with simple rhythmic motions just like those of humans. Thus, we proposed a system that can recognize the beats of the music input and then the robotic system learns on its own to decide which motions should be selected for which beat. This way, the robot will learn to dance very much like how a human child learns a task on its own by trying all possibilities of a task. This type of approach is usually referred as trial and error method.

In the next section, the system configuration of the proposed robotic system is introduced, which consists of a beat tracking system and a robot motion learning system.

3.1 System Configuration

The proposed system consists mainly of two systems: the beat tracking system and the robot motion learning system. Considering the musical input of the system, how and how deeply to analyze the music is one important point to consider. For this, we divided the types of dance into two. The first one dances with simple rhythmic motions, like arm and hand movements; the second one requires the dancer’s own interpretation of the music and own knowhow to express their interpretation. As can be guessed, anyone can dance to the first type of music but expertise is needed for the second type. Since we are using Baxter to perform the dance moves, it can only perform arm movements; we concluded that we should aim for the first type of dance. In the first type of dance, humans do not think much
and just track the basic beat. Therefore, we concluded that the most important feature is the basic beat and that the effects of the other features are not significant.

![Figure 6: Overall configuration of system](image)

The overall system configuration is shown in Fig. 1. This system consists of two parts, which includes real-time beat extraction system and a dance system to make robot learn dance moves. The first system takes a music track in .mp3 or .wav format, applies Short-term Fourier Transform (STFT), and finds the beats per minute (BPM) using the peak-to-peak duration. This is then fed to the robotic system, where the robot learns which move is possible for which beat and creates a Q matrix, consisting the probabilities (Q values) for going from one pose to another. These values are later used to perform the
dance, i.e. complete sequence of movements. The two systems were developed on ROS based platform and coded in Python. [35]

3.2 Beat Tracking System

The method that is explained in this part will extract the beat rate of real-time music. Audio beat tracking algorithms commonly begin with a transformation of the input signal into an intermediary signal, often referred to as onset detection function, between the audio data and output beats. For onset detection function, we adopt a more utilitarian method, proposed by Duxbury [36], which takes account of both energy and phase information of the signal.

At first, we calculate the STFT of a mono musical audio signal with $f_s = 44100$ kHz sampling rate, which is an absolutely indispensable time-frequency analysis in traditional beat-tracking algorithm. Given spectrum information, onset detection function $\Gamma(m)$ can be computed. For a full derivation, see [36]. Figure 7(b) shows an example of the amplitude spectrum of STFT while Figure 7(c) shows the onset detection function of a piece of strongly rhythmic rock music. Real-time beat tracking aims at extracting beat period (the interval between successive beats) and beat alignment (the offset from the beginning of the frame to the first predicted beat within this frame) in order to locate beats within next DF frame. Inspired by Davies’ research [37] on two-state switching model, we apply the general state model to our real-time module. The main idea of this stage is to use the analysis of the immediate past 6 seconds audio data for the prediction of beat occurrences within the upcoming 1.5 seconds.
To infer beat period, an unbiased autocorrelation $ACF(l)$ of a modified detection function is calculated using:

$$ACF(l) = \frac{\sum_{m=0}^{B_r-1} \hat{\Gamma}(m) \hat{\Gamma}(m-l)}{B_r - l}$$

(3.1)

where $l = 0, 1, \ldots, B_r - 1$ corresponds to the lag of $ACF(l)$ and $\hat{\Gamma}(m)$ represents the modified $\Gamma(m)$. Refer to [37] for the derivative process of $\hat{\Gamma}(m)$.

Given the lag, the music tempo, in beats per minute (BPM), can be estimated by:
Then, the autocorrelation function is passed through a shift invariant comb filter bank. Next, the beat period $\tau_r$ is identified as the index of the maximum value of the output sequence of comb filter bank. Note that $\tau_r \leq \tau_{\text{max}} = B_h$ is constrained; that means there is at least one beat for a duration of 1.5 seconds.

The process of beat alignment induction is similar to the way in which we extract beat period. The beat alignment is identified by cross correlating the onset detection function with an impulse train equally spaced by $\tau_r$. Figure 7(d) gives a demonstration of real-time beat tracking.

### 3.3 Robot Motion Learning System

#### 3.3.1 Q-learning Algorithm

The proposed learning algorithm is Q-Learning, which was first proposed by Watkins in 1989 [38]. It is a learning approach that estimates the value functions of the status and action. Q-Learning studies the mapping from environment status to strengthen the signal value function. In the process of interacting with the surroundings, the robot attempts to discover which actions can generate the most reward. The selected action affects all subsequent rewards, in addition to the current reward and the next status.

Q-Learning problem is defined as follows. At each time $t = 0, 1, 2 \ldots$, a robot achieving the current state $s_t \in S$ by observing the surroundings, $S$ is the set of the feasible surrounding status. The current state $s_t$ choosing an action $a_t \in A(s_t)$, $A(s_t)$ is a set of feasible actions under the current state $s_t$. Executing the action $a_t$ and receiving a reward $r_{t+1} \in R$ at

\[
\text{tempo} = \frac{60}{l \times \Delta t}
\]
time \( t+1 \), then reaching the new state \( s_{t+1} \). The objective of the learning is to have an optimal strategy \( \pi : S \times A \rightarrow [0, 1] \). The strategy \( A=\bigcup_{s \in S} A(s) \) can make the total sum of the reward \( V^\pi (S_t) = \sum r_t \), \( 0 \leq \gamma \leq 1 \) is maximum or minimum, and \( \gamma \) is a discount factor in the range of 0 to 1.

Each agent's goal is set to maximize the expected sum of positive rewards and negative rewards (punishments) from a random initial state by learning. The goal is to learn a strategy \( \pi : S \rightarrow A \), that is able to map the current state, \( s \), to an expected action \( a \) and execute the action in the state, \( s \). Learning process of \( \pi \) is a Markova decision process that can be accomplished by the interaction of the learning agent and the surroundings, and at least one optimal decision \( \pi \) is determined. When the model of the surrounding is unknown, an agent can find the expected sum of positive rewards and negative rewards (punishments) \( Q(s, a) \) when action \( a \) is taken in the current state, \( s \), through learning the evaluation function \( Q: S \times A \rightarrow R \), and then updating the value of \( A \) through iterations. The Q-learning update rule is as follows:

\[
Q(s, a) \leftarrow Q(s, a) + \alpha (r + \beta \max_{a'} Q(s', a') - Q(s, a))
\]  

(3.3)

where,

- \( s \) is the current state,
- \( s' \) is the next state,
- \( r \) is the reinforcement signal that is obtained after executing \( a \) of the state \( s \),
- \( \beta \) (0 ≤ \( \beta \) ≤ 1) is the discounted factor, and
- \( \alpha \) (\( \alpha > 0 \)) is the learning rate.
The algorithm 1 shows the learning steps for each move.

### Algorithm 1: Q-learning algorithm for learning moves

**Robot Q-learning** $(S, A, \alpha, \beta)$

1. **Parameters**
   2. $S$ is the set of poses
   3. $A$ is the set of actions for each state
   4. $\alpha$ is the learning rate
   5. $\beta$ is the discount

6. **Variables**
   7. array $Q[S, A]$
   8. previous state $s$
   9. previous action $a$
   10. initialize the array $Q[S, A]$ with positive values
   11. observe the current state $s$
   12. **repeat**
   13. select action $a$ using epsilon greedy policy
   14. perform action $a$
   15. observe reward $r$ for this action and the next state $s'$
   16. $Q[s, a] \leftarrow Q[s, a] + \alpha(r + \beta \max_{a'} Q[s', a'] - Q[s, a])$
   17. $s \leftarrow s'$ **until** termination

where,

- $Q[s, a]$ is a matrix containing the component of Q matrix (state and action pairs),
- State $s$ is a pre-defined pose,
- Next state $s'$ is the next pre-defined pose,
- Action $a$ is moving from one pose to another,
- Next action $a'$ is moving from next pose to another pose,
- $r$ is the reward received,
- $\alpha$ is the learning rate, and
- $\beta$ is the discount factor.
3.3.2 State and Action

In our case, we have described state as a pose of robot, i.e. a set of 14 joint angles for the two arms of Baxter. We have defined 21 unique pose for the robot, which act as 21 states for the algorithm. The sensors on robot gives the current joint angles, which lets algorithm know about the current state. An action is defined as the task of moving from one state or pose to another. At every state, the robot has a set of 20 possible actions, i.e. it can go to any one of the other 20 pose.

3.3.3 Selection Function

The selection function selects the action that has the maximum Q-value for a given pose with a probability of $\beta$. Initially, the value of $\beta$ was set up low (as 0.09), to let the robot explore so that most of the time it will select a random action. However, a low value of $\beta$ will make the robot learning slow, but it will allow the robot to explore all actions. Therefore, the value of $\beta$ was set high (as 0.75) at later stage. For higher values of $\beta$, each new step will dominate the previously learned Q-value and make the robot look as if it has
no memory of the previous conditions it learnt. This change was done only in the advanced robot learning process.

### 3.3.4 Update Function

The Q-values of all state-action pairs are stored in a 21×21 matrix, which was randomly initialized at the beginning. We initialized all the values as positive 10 (being optimistic). ‘Q-learning algorithm require an update function that evaluates the result of an action performed in a given state in order to reward the correct behavior by a scalar’ [39]. If the performed action is correct and is suitable for the robot, a positive reward is given, which increases the Q-value, and if the performed action is incorrect or unsuitable for the robot, a negative reward (punishment) decreases the Q-value. A set of rules governed by the update function, helps to produce certain dance move on the two robotic arms. Therefore, the coefficients, update function, rewards and penalty should be carefully selected in order to remove any undesired actions and get the best suitable actions. The rewards are given in our algorithm is as follows. If both the arms of the robot reach the next pose from the current pose in four beats, it gets a reward of +2. If only one of the arms reach the next pose from the current pose, it gets a reward of +1. In the case when both the arms do not reach the next pose in four beats, a negative reward of -1 is given.

### 3.4 Experiments

The developed algorithm was tested on a dual flexible 7-DoF arm robotic platform called Baxter [40]. It is a new collaborative industrial robot that is designed to work with people without the need for safety cages. It provides a robust, safe and affordable platform for
innovation. It has been specifically designed with keeping in mind the needs of academic labs and corporate research departments.

Baxter was made to learn to dance on two songs with different beat rate. The beat detection algorithm extracted the BPM and length of the music track. Table 1 shows the comparison of original beat rate with the obtained beat rate for each track.

<table>
<thead>
<tr>
<th>Track</th>
<th>Original BPM</th>
<th>Detected BPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>We Will Rock You</td>
<td>83</td>
<td>80</td>
</tr>
<tr>
<td>Dangerous - Michael Jackson</td>
<td>111.89</td>
<td>113</td>
</tr>
</tbody>
</table>

The extracted BPM and length of the music track was fed to the robotic system. Robot then tried all possible moves and received the reward based on the policy. After more than 1000 moves, we decided to stop the learning as Baxter has learned the moves according to the given music track. The learned moves are in the form of Q-values which are stored in the Q[s, a] matrix. To perform the dance, Baxter selects the next move, using the Q[s, a] matrix, which will lead to maximum reward for current and future moves. The trajectory for moving from one pose to another was produced by a ROS based package for Baxter called Joint Trajectory Action Server (JTAS). Figure 9 demonstrates Baxter dancing on ‘We Will Rock You’ after learning the dance moves.
Using the same approach, Baxter was also made to learn to dance on ‘Dangerous’ by Michael Jackson. It was observed that Baxter was making faster moves while dancing on ‘Dangerous’ as compared to while it was dancing on ‘We Will Rock You’. While dancing on ‘Dangerous’, Baxter was selecting moves which can be completed in 2.1 seconds and while dancing on ‘We Will Rock You’, Baxter was selecting the moves which can be completed in 3 seconds. In addition, it was observed that this was actually proportional to the beat rate of the music tracks, which verifies correctness of the algorithm.

### 3.5 Discussion

In this chapter, we present a SoS that makes a robot learn to dance similar to how a child learns to dance by exploring all the possibilities. With our beat extraction system, we were
able to get the beat rate of the given music track. Q-learning system was developed to make the robot learn dance moves based on the beat rate and duration of the music track. With this system of systems (SoS), we don’t need to pre-program the dance moves and synchronize the robot’s motion for dancing. In addition, this SoS has plasticity that it can adapt to various types of music tracks. However, there is a limitation in the beat extraction algorithm. The algorithm works well only with the music tracks with constant beat rate. A system can be developed to detect in real time, the change in the beat rate and the same learning algorithm will be able to learn moves according to the continuously changing beat rate. As judging whether the robot is dancing good or bad is a very subjective, we validated the correctness of the developed method by verifying that the robot adapts to a new music track with different beat rates and selects the dance moves which as suitable for the beat rate of the new music track.

As this learning technique is slow, we implemented another technique based on robot learning from demonstration. In the next chapter, we discuss the proposed RLfD approach in detail.
CHAPTER 4: LEARNING FROM DEMONSTRATIONS

This chapter introduces the second method that was developed for learning and reproduction of trajectories and tasks for a 7DoF robotic arm. First, the initial steps of the proposed approach are addressed, which include the key-point extraction technique and clustering algorithm. Then, details of implementation of HMM are discussed. Producing the generalized trajectory from the centroids and aligning them using DTW is also discussed. In the end, experiments done for teaching and reproducing two tasks are reported.

Robot learning from demonstrations (RLfD) refers to the technique of teaching skills to a robot by giving examples of the desired behavior through human demonstrations. It is similar to the way human beings learn a new skill from demonstrations performed by the teacher. One of the great advantage of this technique is that it eliminates the need of expert level technical knowledge to program a robot. Moreover, it captures key features from the demonstration provided by a teacher who is expert in the specific task, which a robotic programmer might not be able to program. Thus, this technique has great potential in industrial and home robotic applications.

Learning the mapping between world state and actions is called as a policy [41]. This allows a robot to choose an action based on its current state. In RLfD, a robot learns a new policy from demonstrations provided by the human teacher. A demonstration is defined as sequence of state–action pairs that are recorded throughout the teacher’s performance of the required robot behavior. RLfD uses the dataset of recorded
demonstrations to form policy to reproduce the demonstrated behavior. This approach is in contrast to the other approach discussed in the last chapter, where policy is learned based on experience. We have segmented the RLfD problem into three fundamental phases: *data collection* from demonstrations, *learning behavior* by deriving a policy, and *robot execution* or reproduction of the demonstrated behavior.

### 4.1 Problem Statement

In robotics, there is a need of an interactive and expedite learning method as experience is expensive. Robot Learning from Demonstration enables a robot to learn a policy from demonstrations performed by teacher. An interactive method can be developed, where a robot can learn behavior or tasks from human demonstration.

RLfD can be considered as a subset of *Supervised Learning*. As in case of supervised learning, the agent takes the labelled training data and the algorithm learns an approximate model to fit this data. In RLfD, the training data is collected from demonstrations provided by the teacher. Figure 10 (top) shows training data D being acquired from demonstrations by teacher to derive a policy.

The robot world comprises of states $S$ and actions $A$. Mapping between the states, is defined by a probabilistic transition function called state transition matrix given by:

$$T(s' | s, a): S \times A \times S \rightarrow [0, 1].$$

(4.1)

The set $A$ contains high-level behaviors as well as low-level motions. We make an assumption that the states are not fully-observable or hidden. However, the learning algorithm has the access to observed state $Z$, with the mapping $M: S \rightarrow Z$. The
policy \( \pi: Z \rightarrow A \) selects the next action based on observation of the robot state. Figure 10 (bottom) shows a one cycle of policy execution.

![Policy Learning Diagram](image1.png)

**Figure 10: Control policy learning and execution**

### 4.2 Proposed Approach

The proposed RLfD system consists of three fundamental phases. Figure 11 shows a block diagram of the proposed approach. First phase is the *data collection* phase, in which the data is acquired from the demonstrations performed by the teacher. Real-time joint angle values and gripper state are received by the system, which applies Forward Kinematics to find the position and orientation of the manipulator. This process is repeated for all the demonstration and the time series of position and orientation are stored in a file.
Figure 11: Block diagram of the proposed RLfD system
The second phase is the *task learning* phase, in which the stored data from the first phase is used to learn a model of the task or the behavior. From each demonstration, *key-points* of trajectories are extracted and then k-means clustering is used to cluster the *key-points* from all demonstration. Centers of each cluster is calculated and mapped to a unique symbol representing a state. Baum-Welch algorithm is used to learn a policy from the demonstrations to find the probability matrix of state-action pairs. Then, Viterbi algorithm is used to find the most probable sequence of states. DTW is used to align the time vector of the learned trajectory.

Third phase is the *robot execution* phase, in which the saved model is used to produce the learned behavior. Cubic smoothing-spline regression is used to determine a generalized trajectory from the most probable sequence of states. Inverse kinematics is applied to obtain a generalized trajectory suitable for the robot.

### 4.3 Data Collection

Data collection is the first phase of the system, where the robot collects data from the physical demonstrations performed by human teacher. In the next three sub-sections, we discuss the main steps of this phase.

#### 4.3.1 Demonstrations

In our work, we have used *kinesthetic teaching* to demonstrate the task or trajectories to the robot. Kinesthetic teaching is a way of teaching in which, the teacher physically holds the robot arm or manipulator, and demonstrates the task by moving the robotic arm. The
The main advantage of this technique is that there is no need to map the real world space to robot world space, as the performed demonstrations are with respect to robot world.

One major roadblock for this type of teaching is that it requires a robot that has a feedback system to record the joint position in real-time. However, we do not face this problem as we are using one of the most advanced research robots called Baxter, which has an inbuilt feedback system to measure the joint angles from all fourteen joints and manipulator state of the dual-arm robot. This allows us to read these joint values from joint trajectory server in real time.

To get more precision for fast movements, we have increased the default rate (100Hz) of reading current joint angles from the robot to 1000 Hz. This means that the system reads joint values and manipulator states from the robot 1000 times in a second. These values are stored with a timestamp and sent to the forward kinematics function.

**Figure 12: Perform demonstration using kinesthetic teaching**
Figure 12 shows an example demonstration being performed using kinesthetic teaching to demonstrate a pick and place task to Baxter. In this, the teacher holds the force sensor at the manipulator of each robotic arm and guides the robot through the task. To change the state of the gripper, physical button near the manipulator are used. One button closes the gripper and the other opens the gripper. The system records the gripper states along with the joint angles throughout the demonstration, and for each demonstration.

4.3.2 Forward Kinematics

‘Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters’ [42]. Kinematics equations are series of transformation matrices to characterize the relative movement at each joint with respect to the previous joint. Instead of going through the complex matrix multiplications for calculating the relative movements at each joint, we use Denavit-Hartenberg parameters or DH parameters. Jacques Denavit and Richard Hartenberg introduced them in 1955, to standardize the coordinate frame for spatial linkage.

Each individual homogeneous transformation $A_i$ is calculated as:

$$A_i = \text{Rot}_{z, \theta_i} \text{Trans}_{z, d_i} \text{Trans}_{x, a_i} \text{Rot}_{x, a_i}$$

$$= \begin{bmatrix}
-\sin \theta_i & -\cos \theta_i & 0 & 0 \\
\cos \theta_i & \sin \theta_i & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
1 & 0 & 0 & a_i \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & 0 & 0 \\
\sin \theta_i & \cos \theta_i & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

$$= \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & 0 & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i & 0 & a_i \sin \theta_i \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
\cos \theta_i & 0 & a_i & 0 \\
0 & \cos \theta_i \cos \alpha_i - \sin \theta_i \sin \alpha_i & \sin \theta_i \cos \alpha_i + \cos \theta_i \sin \alpha_i & 0 \\
0 & \sin \theta_i \cos \alpha_i + \cos \theta_i \sin \alpha_i & \cos \theta_i \cos \alpha_i - \sin \theta_i \sin \alpha_i & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

(4.2)

where $\theta_i$, $d_i$, $a_{i,i+1}$ and $a_{i,i+1}$ are known as DH parameters associated with link $i$ and joint $i$. 
We use Baxter as our robot for learning, thus we calculated the DH parameters for both the limbs of the robot to perform the forward kinematics. Figure 13 shows the joint names of left limb of the robot.

![Image of Baxter robot showing joint names](image)

**Figure 13: Joint names of left limb of Baxter [43]**

We follow the standard convention of placing the frames at each joint to calculate the DH table. Figure 14 shows an example of placing the frames on first two revolute joints and the physical parameters used to mathematically modelling Baxter’s arm. Table 2 shows the derived DH parameters for all the joints and links for Baxter. These parameters are used to calculate the individual transformation matrices for each of the link.
Now, we use the DH parameters to calculate $A_1$, $A_2$, …. $A_7$. The homogeneous transformation matrix that expresses the position and orientation of the manipulator is called a transformation matrix $[T]$, and is calculated as:

$$T = A_1 * A_2 * A_3 * A_4 * A_5 * A_6 * A_7$$  \hspace{1cm} (4.3)
Table 2: DH table

<table>
<thead>
<tr>
<th>Joint</th>
<th>$\theta$</th>
<th>$d$</th>
<th>$a$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\Theta_1$</td>
<td>0.2703</td>
<td>0.069</td>
<td>-1.571</td>
</tr>
<tr>
<td>2</td>
<td>$\Theta_2$</td>
<td>0</td>
<td>0</td>
<td>1.571</td>
</tr>
<tr>
<td>3</td>
<td>$\Theta_3$</td>
<td>0.3644</td>
<td>0.069</td>
<td>-1.571</td>
</tr>
<tr>
<td>4</td>
<td>$\Theta_4$</td>
<td>0</td>
<td>0</td>
<td>1.571</td>
</tr>
<tr>
<td>5</td>
<td>$\Theta_5$</td>
<td>0.3743</td>
<td>0.01</td>
<td>-1.571</td>
</tr>
<tr>
<td>6</td>
<td>$\Theta_6$</td>
<td>0</td>
<td>0</td>
<td>1.571</td>
</tr>
<tr>
<td>7</td>
<td>$\Theta_7$</td>
<td>0.2295</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4.3.3 Data storage

For each measurement, the calculated position and orientation are stored into a data file, along with the timestamp and the gripper state. This marks the end of the first phase.

4.4 Task Learning

Task learning is the second phase of the system. In this phase, the data stored during the first phase is used to learn a model of the task.

4.4.1 Key-point Extraction

We detect the characteristic features of each demonstration, and we call it key-points. By doing this, we avoid having large number of hidden states to be trained by HMM [44]. We represent every recorded movement during the demonstration by a set of time-discrete sequence. For each arm we have a sequence $P_{d,1}$, $P_{d,2}$, …, $P_{d,len(d)}$ that defines the positions of manipulator, and a sequence $O_{d,1}$, $O_{d,2}$, …, $O_{d,len(d)}$ that defines the
orientations of manipulator over time, where \( l(d) \) is defined as the length of demonstration \( d \). \( P_i \) is a three dimensional vector, whereas \( O_i \) is a four dimensional vector.

A point \( P_{d,i} \) is selected as a key-point \( K_{d,j} \) in the sequence of points \( P_{d,1}, P_{d,2}, \ldots, P_{d,\text{len}(d)} \) if:

\[
\angle(P_{d,i} - P_{d,i-1}, P_{d,i} - P_{d,i+1}) < 2\pi - \epsilon_1
\]

\[
\|P_{d,i} - P_{d,i-1}\| < \epsilon_2, i - \tau_{d,j-1} > \epsilon_3,
\]

\[
\|P_{d,i} - P_{d,\tau_{d,j-1}}\| > \epsilon_4
\]

\[
\|P_{d,i} - P_{d,\tau_{d,j-1}}\| \geq \epsilon_5, i - \tau_{d,j-1} > \epsilon_6,
\]

\[
\|P_{d,n} - P_{d,\tau_{d,j-1}}\| < \epsilon_7 \forall n \in [\tau_{d,j-1}, i), \quad (4.4)
\]

where:

\( \tau_{d,j} \) denotes the time-stamp of demonstration \( d \) and \( j^{th} \) key-point.

![Diagram showing key-point selection criteria](image)

**Figure 15: Key-point selection criteria**
This means that if the angle between the vector that goes from point $P_i$ to its successor $P_{i+1}$ and the vector that goes from point $P_i$ to its predecessor $P_{i-1}$ is less than $2\pi - \epsilon_1$, then point $P_i$ is selected as a key-point. Figure 15 shows an example of finding angles between the two vectors in a two dimensional plane. To detect only sharp corners in the manipulator trajectory as key-points $\epsilon_1$ should be high.

To detect the key-points $O_{d,j}$ of the orientation angles of the manipulator, the same selection criterion for key-points is used. A point $O_{d,j}$ is selected as a key-point $K_{d,j}$ in the sequence of points $O_{d,1}, O_{d,2}, \ldots, O_{d,len(d)}$ if:

$$\angle(O_{d,i} - O_{d,i-1}, O_{d,i}, O_{d,i} - O_{d,i+1}) < 2\pi - \epsilon_8$$

$$\lor \quad \|O_{d,i} - O_{d,i-1}\| < \epsilon_9, i - \tau_{d,j-1} > \epsilon_{10},$$

$$\|O_{d,i} - O_{\tau_{d,j-1}}\| > \epsilon_{11}$$

$$\lor \quad \|O_{d,i} - O_{\tau_{d,j-1}}\| > \epsilon_{12}, i - \tau_{d,j-1} > \epsilon_{13},$$

$$\|O_{d,n} - O_{\tau_{d,j-1}}\| < \epsilon_{14} \forall n \in [\tau_{d,j-1}, i)$$

(4.5)

A major challenge in detection of the key-points using this approach is to tune the values of thresholds $\epsilon_1 - \epsilon_{14}$, which will decide the number of key-points extracted from the recorded task or trajectories. If the number of detected key-points is low, some relevant characteristics of the task or trajectories can get missing from the generalization step. Whereas, if the detection of key-points is high, over fitting of the recorded task or trajectories can take place.
Keeping in mind the tradeoff between generalization and over-fitting, reasonable values for $\epsilon_1 - \epsilon_{14}$ were experimentally determined. Figure 16 shows the result of the detected key-points for the position of the manipulator. This result consists of all the detected key-points, for both position and orientation. Therefore, some of the key-points in the plot might not be a key-point in three-dimensional plan plotted for position, but is a key-point in four-dimensional plan of the orientation. Figure 17 shows the result of key-point extraction algorithm on multiple demonstrations. It shows the key-points extracted from three demonstration. The key-points are marked with circles on each original trajectory of the demonstration.
4.4.2 Clustering

In contrast to the common key point approach used by Asfour [44], we cluster the key-points derived from all the demonstration and calculated the centroids of each cluster. To solve this problem, we use k-means clustering algorithm [45]. The k-means clustering algorithm takes the number of clusters as an input to generate k clusters, from a set of observation vectors. It returns the centroids for each of the k clusters formed. For our system, we set the number of clusters k equal to average number of key-point in each demonstration.

A key-point vector is classified with a cluster if the centroid of the cluster is closest to it, i.e. closer to centroid than any other centroids. K-means clustering algorithm attempts to minimize the distortion, which is the sum of the squared distances between each key-point vector and its dominating centroid. At each step, k-means refines the choice of
centroids and tries to reduce the distortion. When this distortion change gets below a threshold, the algorithm stops.

Figure 18: Centroids of the k clusters

Figure 18 shows the result of the implement k-means clustering algorithm to cluster the key-points and find the centroids of the clusters. In the plot, each centroid is marked with big red circles. In the next section, we discuss how these centroids are used to learn a model of the task or trajectories using a HMM.

4.4.3 Model Learning using HMM

A HMM (Hidden Markov Model) is a directed graphical model that is a statistical Markov chain of a sequence of unobserved or hidden states and a corresponding sequence of observation variables. It has been widely applied in the field of handwriting recognition, speech recognition, DNA sequence analysis, etc.
Figure 19 shows a graphical representation of HMM, where $S_t$ denotes the hidden states and $O_t$ denotes the observed variables at time instants $t \in \{1, 2, 3, \ldots, t-1, t, t+1, \ldots\}$. The probability of going to $j^{th}$ state at time $t+1$, given that the current is state $i$ at time $t$, is denoted by:

$$a_{ij} = P(S_{t+1} = j | S_t = i). \quad (4.6)$$

Using these probabilities, we form the state transition matrix, given by:

$$A = \{a_{ij}\}, \quad \forall i, j = \{1, 2, 3, \ldots, N_s\}, \quad (4.7)$$

where:

$N_s$ denotes the number of hidden states in the HMM.

In this research, the number of hidden states $N_s$ was set equal to the number of centroids derived from the demonstrations.

The initial state probabilities is defined as the probability of model being in state $i$ at time $t = 1$, and is given by:

$$\pi = \{\pi_i = P(S_1 = i)\}, \quad \forall i = \{1, 2, 3, \ldots, N_s\}. \quad (4.8)$$

The observation probability is defined as the probability of observing a symbol $q_k$
at time $t$ given the model is in state $i$, and is denoted as:

$$b_t(k) = P(q_k | S_t = i). \quad (4.9)$$

Using these probabilities, we form the observation probability matrix, given by:

$$B = \{b_t(k)\}, \forall \ i = \{1, 2, 3, \ldots, N_s\} \text{ and } k = \{1, 2, 3, \ldots, Q\}, \quad (4.10)$$

where:

- $Q$ is the number of observation symbols.

A complete HMM is described as:

$$\lambda = \{\pi, A, B\}. \quad (4.11)$$

In our research, we used a discrete-HMM for learning a model of the demonstrated trajectories, which required the recorded continuous trajectories to be mapped to discrete values. The key-point extraction and clustering technique applied in section 4.4.1 Key-point Extraction and 4.4.2 Clustering were for vector quantization. As it is required to use most of the $k$ clusters in order to preserve most of the features in the continuous trajectory, we mapped each cluster into a discrete symbol $o_{n,m}$ to form a codebook of symbols $\{q_1, q_2, q_3, \ldots, q_k\}$, where $k$ is the number of clusters formed. The sequences of these observations form the set, and is given by:

$$O = \{O_m = (o_{1,m}, o_{2,m}, o_{3,m}, \ldots, o_{N,m,m})\}, \forall \ m = \{1, 2, 3, \ldots, M\}, \quad (4.12)$$

where:

- $M$ denotes the total number of demonstrations.
The efficiency of learning with HMM depends on the number of available observations. As in RLfD, it is preferred to keep the number of observations or demonstrations low, an appropriate initialization of the model parameters is imperative. Next, we discuss the initialization process of the HMM parameters.

We implemented the Bakis left–right topology [47] to model the demonstrated task. The forward-transition probabilities \( a_{i,i+1}, a_{i,i+2}, \ldots \) and self-transition probabilities \( a_{i,i} \) were initialized in the state transition matrix \( A_{i,j} \) as:

\[
\begin{align*}
    a_{i,i+1} &= (1/\tau_{i,\delta})(1/Z), \\
    a_{i,i+2} &= (1/4\tau_{i,\delta})(1/Z), \\
    \text{and } a_{i,i} &= (1 - 1/\tau_{i,\delta})(1/Z),
\end{align*}
\]

where:

\( \tau_{i,\delta} \) is the amount of time spent in state \( i \), and

\( Z \) is a normalizing constant to make sure \( \sum_j a_{i,j} = 1 \).

All other state transition probabilities were set as zero, to make sure that transition to those states is impossible. Output probabilities \( b_i(k) \) were initialized as:

\[
b_i(k) = n_{i,\delta}(q_k)/\tau_{i,\delta},
\]

where:

\( n_{i,\delta}(q_k) \) denotes the number of times \( q_k \) is observed in state \( i \) of \( O_\delta \).

The state probabilities \( \pi \), were initialized as \( \pi = [1 \ 0 \ldots \ 0] \). This ensures that the learning always starts from the starting point.
Once the model parameters were initialized, Baum–Welch algorithm [47] was used to train on all demonstrations or observations $O_1, O_2, \ldots, O_M$. Later, Viterbi algorithm [47] was used to determine the most probable sequence of hidden state. Due to the nature of the implemented Bakis left-right topology, some hidden states or centroid points were not present in all the observed trajectories. We called them as *zero-points* and were ignored in the generalized trajectory.

**4.4.4 Dynamic Time Warping**

Since the length and velocities of the demonstrated trajectories differ, the time frames of extracted key-points of each demonstration are different. To produce the learned trajectory, we need to align the set of key-points along a common time vector. To solve this problem, we use DTW (Dynamic Time Wrapping). In DTW, the temporal alignment of the clusters made by key-points is done by aligning the entire trajectory with respect to a reference trajectory.

DTW sequence alignment technique forms a matrix that consists of distances between two time-series, which is then used to find an optimal path that minimizes the distance between the two time-series. For a given test sequence $t_1, t_2, \ldots, t_T$ of length $T$, and a reference sequence $r_1, r_2, \ldots, r_R$ of length $R$, the distance matrix is calculated as:

$$
H(x, y) = \|r_x - t_y\|^2, \quad \forall \ x = 1, 2, \ldots, R, \ \text{and} \ y = 1, 2, \ldots, T. \quad (4.15)
$$

In our research, to calculate the distance between two time-series, Euclidean $l_2$-norm is used. The optimal alignment path $g(x, y)$ is calculated as:

$$
g(x, y) = H(x, y) + \min \{g(x - 1, y), g(x - 1, y - 1), g(x, y - 1)\} \quad (4.16)
$$
For selection of the reference sequence, forward algorithm was used to find the demonstration that has the highest probability for the learned model. As our data received from demonstration was eight dimensional data, we implemented multidimensional DTW, and the distance matrix is calculated as:

\[
H(x, y) = \sum_{d=1}^{D} \left\| r_x^d - t_y^d \right\|^2, \quad \forall \ x = 1, \ldots, R, \text{ and } y = 1, \ldots, T. \quad (4.17)
\]

where:

D is the number of dimensions, and in our case, D = 8.

### 4.4.5 Generalized Trajectory

After we have a common timeline for the model learned for the sequence of centroids, we need to connect these centroids to generate a generalized trajectory. To solve this problem, we used cubic smoothing-spline regression to determine a generalized trajectory from the most probable sequence of centroids. This method is widely applied for fitting a smooth curve to large set of scattered data. In our case, we want to generate a smooth trajectory from the set of scattered centroid points. The spline curve was interpolated at intervals equal to the period size of clusters, which produced a generalized trajectory suitable for Baxter.

This marks the end of the task learning phase. The learned model is stored in multiple files containing centroids, most probable sequence, and timeline of the task or trajectory. In the next section, we discuss how robot uses this to perform the learned task or trajectory.
4.5 **Robot Execution**

Once the model of the task is saved, we use the stored files to perform the task. The files contain the centroids, which are the position and orientation of the manipulator. For the robot to reproduce the learned task, we need to convert these position and orientation values into joint angles values that are suitable for the robot. To solve this problem, we use Inverse Kinematics. This technique is just the opposite of Forward Kinematics. It provides the set of joint angle values for a given the Cartesian pose of the manipulator.

We use the Baxter’s IK server to perform the inverse kinematics during the execution of the task. It is a build-in service from Baxter SDK. It takes the position and orientation values as the input and returns seven valid joint angles values to achieve it. These values are send to the robot and it moves to that state using JTAS.

4.6 **BLInK**

For the teacher to interact with our robot Baxter, we developed a terminal based interface called BLInK (Baxter Learning Interface by Kumra). Figure 20 shows a screen shot of the home screen of the interface. It enables the teacher to teach the task and save the demonstrations of each task with a task name. This name can then be used to learn a model of the task and even perform the task. As the model of the task is saved with the task name, user can easily reproduce the task by just entering the task name. The interface warns the user if the model of the task does not exist or the model is not valid.

BLInK also has a feature to plot the saved demonstrated trajectories along with the learned trajectories. In the next section, we discuss the experiments performed with the robot. BLInK was used as interface with the robot to perform these experiments and plot
the learned trajectories. Due to limitations of the interface, we can only plot the position of the manipulator in three dimensions.

![Figure 20: BLInK home screen](image)

4.7 Experiments

To evaluate the approach, we performed the experiments on the research robot by Rethink Robotics called Baxter. It is a dual-arm 7DoF robot. A modular approach was used to create the framework and all the algorithms were developed as separate packages in python. This made it easier to test and debug each part of the framework separately. In this section, we discuss experiments performed and the results of those experiments.

The key-point extraction algorithm and clustering algorithm plays a vital role in the efficiency of the model. The variables for tuning the key-point extraction algorithm were
experimentally determined. To make sure this algorithm works well on wide variety of task and complex trajectories, we validated them on several different tasks. Figure 21 shows the key-points and centroids of the clusters for a complex pick and place task. In this task, the robot picks up a mobile phone from a table and places at some other place on the table, then again picks up the mobile from the new location, and places it back to its original place. Three demonstrations were performed to teach this task. Figure 22 shows key-points and centroids of the clusters for a painting task. In this, the robot holds a paintbrush and is taught to paint a wall. This task involves a number of complex trajectories. Five demonstrations were performed to teach this task. As we observe in both the cases, the algorithm is able to extract key features of the demonstration and form clusters of the key-points of all the demonstrations. These centroids are then mapped to unique symbols and act as the states for the HMM.

Figure 21: Centroids extracted from pick and place task
In the next two sub-sections, we discuss the details of the two complex tasks that the robot learned through multiple demonstrations.
4.7.1 Stacking cup task

The first task is to place a cup on a stack of cups. In this task, the robot takes the cup from a human and puts it on a stack of cups. Figure 23 shows robot learning the stacking cup task from a human teacher through demonstration. Three demonstrations were performed to teach this task. Table 3 shows the data statistics for this task.

<table>
<thead>
<tr>
<th>Demonstration #</th>
<th># of recorded values</th>
<th># of key-points extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstration 1</td>
<td>18024</td>
<td>18</td>
</tr>
<tr>
<td>Demonstration 2</td>
<td>19759</td>
<td>19</td>
</tr>
<tr>
<td>Demonstration 3</td>
<td>18140</td>
<td>20</td>
</tr>
</tbody>
</table>

The learning variables for this task were as follows:

**Number of clusters formed:** 19

**Observations:**

Demo 1: [0, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 14, 16, 16, 18]

Demo 2: [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

Demo 3: [0, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 15, 16, 18, 16, 18]

**Learned Model:**

[0, 0, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 15, 16, 18]
Figure 23: Demonstration of stacking cup task
Figure 24: Robot performing learned stacking cup task
Figure 25: Demonstrated and learned trajectory for stacking cup task
As we can observe in Figure 24, the robot was able to reproduce the task successfully. This task involved learning of precise motion of the manipulator to precisely drop the cup on top of the stacked cups. Figure 25 shows the manipulator position for the demonstrated task and the learned trajectory. The small triangles on the trajectories marks the key-points extracted from that demonstration, and the red circles show the centroids of the clusters formed. In total, 19 clusters were formed and they acted as states for the HMM. Numbers from 0-18 named these centroids. The cyan line shows the generalized trajectory. This plot only shows the position of the manipulator only. They complete learning process involved the learning of the orientation of the manipulator as well.

4.7.2 Pick and place task

The second task is a pick and place task, in which the robot picks up a block of wood and places it at some other spot and then pick the block from that spot and places it back to the original spot. This is a much more complex task than the previous task. Figure 26 shows the robot learning the pick and place from a human teacher through demonstration. Table 4 shows the data statistics for this task.

<table>
<thead>
<tr>
<th>Demonstration #</th>
<th># of recorded values</th>
<th># of key-points extracted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demonstration 1</td>
<td>31510</td>
<td>24</td>
</tr>
<tr>
<td>Demonstration 2</td>
<td>28593</td>
<td>21</td>
</tr>
<tr>
<td>Demonstration 3</td>
<td>27025</td>
<td>21</td>
</tr>
</tbody>
</table>
The learning variables for this task were as follows:

**Number of clusters formed:** 21

**Observations:**

Demo 1: [0, 0, 2, 3, 4, 5, 6, 7, 9, 10, 11, 10, 9, 13, 14, 15, 16, 17, 19, 18, 19, 17, 20, 20]
Demo 2: [1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Demo 3: [1, 1, 2, 3, 4, 4, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 19, 18, 19, 20, 20]

**Learned Model:**

[0, 1, 1, 2, 3, 4, 6, 7, 9, 10, 11, 10, 9, 13, 14, 15, 16, 17, 19, 18, 19, 20, 20, 20]

It can be observed in Figure 28 that the three demonstrations were a lot different from each other, though they were performing the same task. The small triangles on the trajectories mark the key-points extracted from that trajectory, and the red circles show the centroids of the clusters formed. In total, 21 clusters were formed and they acted as states for the HMM. Numbers from 0-20 named these centroids. The cyan line shows the generalized trajectory. Please note that the trajectory in the graph is not the actual trajectory the robot went through. The derived trajectory in the plot is just by connecting the states with linear lines to illustrate the flow of states. The actual trajectory of the robot used JTAS spline interpolation and was much smoother than the one in the plot.

In both the experiments, the robot was able to learn the complete task and successfully reproduce the learned task. These tasks look simple for a human being, but they involve learning of complex eight-dimensional time-series data for the robot. With our approach, the robot was able to learn most of the task in a few seconds.
Figure 26: Demonstration of pick and place task
Figure 27: Robot performing learned pick and place task
Figure 28: Demonstrated and learned trajectory for pick and place task
4.8 Discussion

In this chapter, we presented a framework for RLfD. This framework has three phases: Data collection, Task Learning and Robot Execution. The data collection module collects and stores the eight dimensional time-series data from each demonstration. The Task Learning module then uses this data to learn a model of the task. As we can observe from the experiments, the key-points are extracted from each of the recorded demonstration. This was done to limit the number of observed states for the HMM. As we can observe from the data statistics of the two experiments, the number of extracted key-points is much less than the recorded values. This reduction in the number of states lead to a faster learning of the model. Initially, we tried to learn the model of the task based on the joint angles values, and it avoided the implementation of the forward and inverse kinematics. Then, we implemented FK and IK and used the position and orientation of the manipulator, which made it easier to visualize the trajectories of the task. In addition, the key-points extracted from both the methods were the same in most of the task. Thus, we decided to move forward with the second approach using the position and orientation, as it is easier to plot and visualize the position of the manipulator.

The model learning part of our approach is similar to [46], but the key point extraction technique we implemented allowed us to precisely extract key-points from the demonstrations. By adjusting the threshold values $\epsilon_1 - \epsilon_{14}$, we were able to find tune the key point extraction algorithm to extract only sharp corners from the demonstrated trajectory. The k means clustering technique that we used for clustering the key points allowed us to include all the key points from all the demonstrations performed. Thus, we were able to preserve the key features of the task that were missing from some
demonstrations. Thus, we got more key features of the task as compared to Asfour et al. [44], where they used the common key points concept, i.e. they just use the key points that are found in all demonstrations. Therefore, the generalized trajectory only includes the features that are present in all demonstrations.

It was experimentally determined that three or four demonstrations for most of the task, tends to produce a generalized trajectory closest to the most probable demonstrated trajectory. The learning of the orientation played a crucial role as well. For many tasks like picking up the wooden block, the orientation of the gripper has to be perfectly aligned with the object to grab the object from the right place. The model learning also involved learning of the gripper states. The model learned should not only learn whether the gripper should be closed or open, but also the width it should open or close to grasp or release an object. This ensures that the robot does not break delicate objects by learning the grasping width of the gripper during the task.
CHAPTER 5: CONCLUSION

This thesis presented two frameworks for robot learning: Robot Learning by trial-and-error and Robot Learning from Demonstrations. Both methods have their own advantages and can be used according to the task to be learned. Robot learning by trial-and-error is a slow learning process as the robot tries all possibilities to do the task and creates a Q-matrix containing the Q-values for every action. It then uses these values to get the most probable sequence of actions to perform the task. This approach has great application where it is tough for a human being to demonstrate the task. As in our first approach, we made the robot learn to dance on a music track. The robot was able to extract the beats from the music track and did dance moves based on the beat rate. It would be a tedious task for programmer to program a robot for all the moves and then synchronize with the music track. With the use of our approach, the robot was able to learn to dance on a music track just the way a child learns a new task by trying out various ways to do the task.

Learning by trial-and-error is an expensive learning process as it take a lot of time to create a model of the task by trying all the possible actions. For this reason, we developed the second framework, which involved learning tasks by demonstrations. In this approach, the robot learns a new task though multiple demonstrations by a teacher. The teachers physically demonstrate the task to the robot by guiding the manipulator through the desired trajectory. This demonstration process is possible because of the availability of the advanced robots like Baxter, which have feedback mechanism to provide the position and force feedback for every joint of the robot. This enables us to record the demonstrations and learn a model from the demonstrated trajectories. Our experiments shows that robot was able to learn a model of a complex task by 3-4 demonstrations, depending upon the
complexity of the task. Thus, the learning process gets faster and less expensive as the robot does not have to go through all the possible actions, and reduces the learning time. This approach works well for pick and place tasks, where a teacher can easily demonstrate the task to the robot.

Table 5: Comparison of two proposed methods

<table>
<thead>
<tr>
<th>Learning by Trail-and-Error</th>
<th>Learning from Demonstration</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Explore all possible trajectories.</td>
<td>- Limited to exploring only demonstrated trajectories.</td>
</tr>
<tr>
<td>- Slow learning process.</td>
<td>- Faster learning process.</td>
</tr>
<tr>
<td>- No human machine interface required.</td>
<td>- Human machine interface is required.</td>
</tr>
<tr>
<td>- Hard to design and implement for complex tasks.</td>
<td>- It can be easily implemented for complex tasks.</td>
</tr>
<tr>
<td>- Good for tasks that are hard to demonstrate.</td>
<td>- Good for task that a human teacher can demonstrate.</td>
</tr>
</tbody>
</table>

The results of the experiments performed using both the methods shows that the learning from demonstration produces the learned model in a lot lesser time as compared to learning by trial-and-error. However, the nature of the task learned by the two methods were completely different. Thus, we should select our learning approach based on the nature of the task. We should use learning from demonstration for the task where a teacher can demonstrate the task to the robot. However, in case when demonstrating the task in not possible, we can use the learning by trial-and-error approach to enable the robot to learn the task by trying all possibilities.
CHAPTER 6: FUTURE WORK

It is possible to apply both the proposed methods in this thesis to a wide variety of applications. A real time beat extraction algorithm can be developed to extract beat rate of a music track in real time, so that the Q-learning algorithm can be active while the robot is performing the dance and enabling the robot to continue to learn dance moves while performing the dance. This type of trial-and-error algorithm can be applied to solve other problems like teaching a robot to play games, or cook food.

Baxter can be trained to learn more complex task using the developed RLfD framework. With the advancements in the field of image processing and visual servoing, RLfD can be made more robust by allowing the robot to learn the position and orientation of the task relevant objects in addition to the position and orientation of the manipulator. This way, the robot will be able to adapt to the changing environment, as it will learn the environment parameters like distance from task relevant objects. For example, in case of the pick and place task, if we manually move the object to the some other location, the robot will fail to pick up the object. However, if we will be able to learn the position of the object with respect to the manipulator, we might be able to adjust to the changing position of the object. This will enable the robot to learn a task and then reproduce it in a dynamic environment.
BIBLIOGRAPHY


Appendix A: Baxter Arm Specifications

A1. Joint Names
Baxter's joint names are mentioned a number of times in this documentation. A labeled diagram can be found below (S=Shoulder, E=Elbow, W=Wrist)

![Joint names diagram](image)

Figure 29: Joint names

A2. Link Lengths
Link lengths for Baxter's joints are measured in mm, from the center of one joint to the center of the next. They can be found in the image below.
A3. Range of Motion - Bend Joints

The range of motion for each bend joint is presented below. The table shows the measurements in degrees and radians (the values are in the following format - +limit, -limit: total movement)
### Table 6: Link Length Table (Bend Joints)

<table>
<thead>
<tr>
<th>Joint</th>
<th>Range (Degrees)</th>
<th>Range (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>+60, -123:183</td>
<td>+1.047, -2.147:3.194</td>
</tr>
<tr>
<td>E1</td>
<td>+150, -3:153</td>
<td>+2.618, -0.052:2.67</td>
</tr>
<tr>
<td>W1</td>
<td>+120, -90:210</td>
<td>+2.094, -1.571:3.665</td>
</tr>
</tbody>
</table>

### A4. Range of Motion - Twist Joints

The range of motion for each twist joint is presented below. The table shows the measurements in degrees and radians (the values are in the following format - +limit, -limit: total movement)

![Figure 32: Range of Motion (Twist Joints)](image)
### Table 7: Link Length Table (Twist Joints)

<table>
<thead>
<tr>
<th>Joint</th>
<th>Range (Degrees)</th>
<th>Range (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>+173.5, -173.5:347</td>
<td>+3.028, -3.028:6.056</td>
</tr>
<tr>
<td>S0</td>
<td>+51, -141:192</td>
<td>+0.890, -2.461:3.351</td>
</tr>
<tr>
<td>W0</td>
<td>+175.25, -175.25:350.5</td>
<td>+3.059, -3.059:6.117</td>
</tr>
<tr>
<td>W2</td>
<td>+175.25, -175.25:350.5</td>
<td>+3.059, -3.059:6.117</td>
</tr>
</tbody>
</table>

### Table 8: Flexure Stiffness (K)

<table>
<thead>
<tr>
<th>Joint</th>
<th>Stiffness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Flexures (W0, W1, W2)</td>
<td>3.4deg @ 15Nm (~250Nm/rad)</td>
</tr>
<tr>
<td>Large Flexures (S0, S1, E0, E1)</td>
<td>3.4deg @ 50Nm (~843Nm/rad)</td>
</tr>
</tbody>
</table>

### Table 9: S1 Spring Specs

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring Type</td>
<td>JIS standard die spring: ASF 35 X 200</td>
</tr>
<tr>
<td>Free Length</td>
<td>200 mm</td>
</tr>
<tr>
<td>Stiffness (K)</td>
<td>9.6 N/mm</td>
</tr>
<tr>
<td>Operating length</td>
<td>101 mm - 154 mm</td>
</tr>
</tbody>
</table>
A5. Joint Sensor Resolution

- The resolution for the joint sensors is 14 bits (over 360 degrees); so $360/(2^{14}) = 0.021972656$ degrees per tick resolution.

- All of the joints have a sinusoidal non-linearity, giving a typical accuracy on the order of +/-0.10 degrees, worst case +/-0.25 degrees accuracy when approaching joint limits. In addition, there may be an absolute zero-offset of up to +/-0.10 degrees when the arm is not calibrated properly. Be sure to tare and calibrate the arms if you're trying to minimize accuracy errors in the joint sensors.

A6. Miscellaneous

The peak torque specification refers to the maximum amount of torque that can be applied to each joint.

<table>
<thead>
<tr>
<th>Joint</th>
<th>Peak Torque</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0, S_1, E_0, E_1$</td>
<td>50Nm</td>
</tr>
<tr>
<td>$W_0, W_1, W_2$</td>
<td>15Nm</td>
</tr>
</tbody>
</table>
### Table 11: Camera Specifications

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Resolution</td>
<td>1280 x 800 pixels</td>
</tr>
<tr>
<td>Effective Resolution</td>
<td>640 x 400 pixels</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>30 frames per second</td>
</tr>
<tr>
<td>Focal Length</td>
<td>1.2mm</td>
</tr>
</tbody>
</table>

### Table 12: On Board CPU

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>3rd Gen Intel Core i7-3770 Processor (8MB, 3.4GHz) w/HD4000 Graphics</td>
</tr>
<tr>
<td>Memory</td>
<td>4GB, NON-ECC, 1600MHZ DDR3</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>128GB Solid State Drive</td>
</tr>
</tbody>
</table>
### Table 13: Component Weights

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total weight (with pedestal)</td>
<td>298 lbs / 135.2 kg</td>
</tr>
<tr>
<td>One Arm</td>
<td>47 lbs / 21.3 kg</td>
</tr>
<tr>
<td>Torso</td>
<td>70 lbs / 31.8 kg</td>
</tr>
<tr>
<td>Pedestal</td>
<td>134 lbs / 60.8 kg</td>
</tr>
</tbody>
</table>

### Table 14: Electrical Power

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>DC-to-120V AC Inverter (Note: the Baxter robot has an internal PC, which cannot be powered directly off of 24V DC)</td>
</tr>
<tr>
<td>Operation</td>
<td>Standard 120VAC power. Robot power bus and internal PC both have “universal” power supplies and support 90 - 264V AC (47 - 63Hz)</td>
</tr>
<tr>
<td>Max Consumption</td>
<td>6A at 120V AC, 720W max per unit</td>
</tr>
</tbody>
</table>
Electrical Efficiency
87% to 92%

Power Supply
Uses medical-grade DC switching power supply for robot power bus

Tolerance to sags
Sags tolerated to 90V. Sustained interruption will require manual power-up

Voltage Flicker
Holdup time 20mS

Voltage Unbalance
Single phase operation only

<table>
<thead>
<tr>
<th>Description</th>
<th>Spec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen Resolution</td>
<td>1024 x 600 pixels</td>
</tr>
<tr>
<td>Positional Accuracy</td>
<td>+/- 5 mm</td>
</tr>
<tr>
<td>Max Payload (including end-effector)</td>
<td>5 lb / 2.2 kg</td>
</tr>
<tr>
<td>Gripping Torque (max)</td>
<td>10 lb / 4.4 kg</td>
</tr>
<tr>
<td>Infrared Sensor Range</td>
<td>1.5 – 15 in / 4 – 40 cm</td>
</tr>
</tbody>
</table>