ABSTRACT

Title of dissertation:	ROBOTS LEARNING MANIPULATION TASKS FROM DEMONSTRATIONS AND PRACTICE
	Ren Mao, Doctor of Philosophy, 2017
Dissertation directed by:	Professor John S. Baras Department of Electrical and Computer Engineering

Developing personalized cognitive robots that help with everyday tasks is one of the on-going topics in robotics research. Such robots should have the capability to learn skills and perform tasks in new situations. In this thesis, we study three research problems to explore the learning methods of robots in the setting of manipulation tasks. In the first problem, we investigate hand movement learning from human demonstrations. For practical purposes, we propose a system for learning hand actions from markerless demonstrations, which are captured using the Kinect sensor. The algorithm autonomously segments an example trajectory into multiple action units, each described by a movement primitive, and forms a task-specific model. With that, similar movements for different scenarios can be generated, and performed on Baxter Robots.

The second problem aims to address learning robot movement adaptation under various environmental constraints. A common approach is to adopt motion primitives to generate target motions from demonstrations. However, their generalization capability is weak for novel environments. Additionally, traditional mo-

tion generation methods do not consider versatile constraints from different users, tasks, and environments. In this work, we propose a co-active learning framework for learning to adapt the movement of robot end-effectors for manipulation tasks. It is designed to adapt the original imitation trajectories, which are learned from demonstrations, to novel situations with different constraints. The framework also considers user feedback towards the adapted trajectories, and it learns to adapt movement through human-in-the-loop interactions. Experiments on a humanoid platform validate the effectiveness of our approach.

In order to further adapt robots to perform more complex manipulation tasks, as the third problem, we are investigating a framework that the robot could not only plan and execute the sequential task in a new environment, but also refine its actions by learning subgoals through re-planning/re-execution during the practice. A sequential task is naturally considered as a sequence of pre-learned action primitives, each action primitive has its own goal parameters corresponding to the subgoal. We propose a system to learn the subgoals distribution of given task model using reinforcement learning by iteratively updating the parameters in the trials. As a result, by considering the learned subgoals distribution in sequential motion planning, the proposed framework could adaptively select better subgoals to generate movements for robot to execute the task successfully. We implement the framework for the task of "openning a microwave" involving a sequence of primitive actions and subgoals and validate it on Baxter platform.

Robots Learning Manipulation Tasks from Demonstrations and Practice

by

Ren Mao

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Acknowledgments

This thesis becomes possible with the kind support and help of many people. I would like to extend my sincere thanks to all of them who enrich my graduate experience that I will cherish forever.

Formost, I would like to express my sincere gratitude to my advisor, Professor John S. Baras for giving me an invaluable opportunity to broaden my knowledge base and supporting me diving deep into some real-world problems. He continually and convincingly conveyed a spirit of adventure in regard to amazingly broad areas of research, and an excitement in regard to chanlleging problems. Without his deep insights and guidance this dissertation would not have been possible. I would also like to express my special thanks to Dr. Yiannis Aloimonos and Dr. Cornelia Fermüller for their persistent help and imparting their knowledge and expertise in the study. I would like to thank Dr. Gang Qu and Dr. Behtash Babadi for their efforts to serve on my thesis committee and their invaluable discussions and insights.

In addition, thank you to my girlfriend Lulin Jiang, for all her love and support. Many thanks to Dr. Yezhou Yang, Dr. Xiangyang Liu, Dr. Xiangnan Weng, Dr. Yuchen Zhou and Wentao Luan, who have enriched my graduate life in many ways, with whom I had many fruitful discussions that gave me inspiration for many ideas. Also I would like to thank Mrs. Kim Edwards for her great administrative support.

Lastly, I would like to acknowledge the support offered by DARPA (through ARO) grant W911NF1410384, by NSF grants CNS-1035655, CNS-1544787, SMA-1540917 and SMA-1248056, and by NIST grant 70NANB11H148.

Table of Contents

List of Figures

Chapter 1: Introduction

Robotics are becoming more and more popular in recent years thanks to various applications for robots to perform in different environments. In some factories the robots are already used to perform simple tasks like picking and placing. As a further step, people are investigating personalized robots to do daily tasks like cooking in kitchen or assisting the elderly. These personalized robots could also be used in industries such as serving food, customized manufacturing and even health care. However, there are still many open research problems in developing such personalized cognitive robots.

Most of daily tasks that the robots need to perform are manipulation tasks. In order to have robots perform those tasks, they will need to be capable of adapting to new scenarios as they may be facing to different objects and tasks. Hence, it is impractical to preprogram all the skills into these robots. Instead, such robots should have the capability to learn the skills for manipulating objects autonomously.

In this thesis, we study three research problems to explore the learning methods of robots in the setting of manipulation tasks. There are many challenges for a robot to learn a versatile set of manipulation skills.

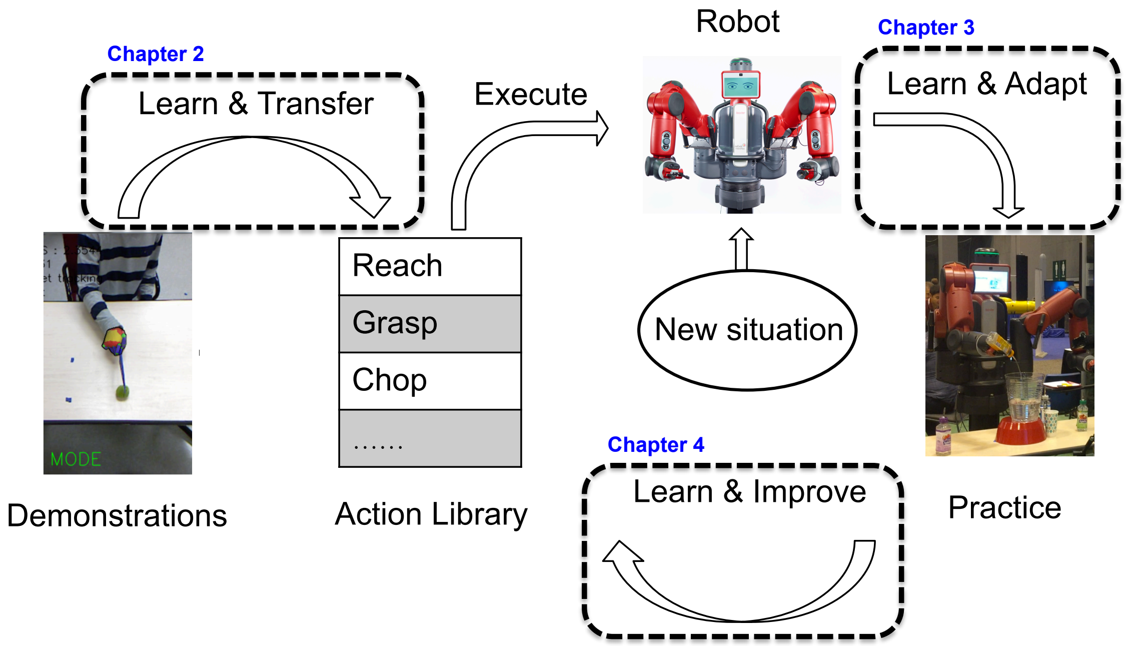


Figure 1.1: Overview of robots learning for manipulation tasks from demonstrations and practice.

1.1 Motivations

Let us consider an example personalized robot where the robot serves as a butler and prepares meal in a kitchen. To begin with, the robot needs to learn basic skills so that it can perform the task in new situations. However, even simple tasks, like cutting a cucumber, may be realized in thousands of different ways. Therefore, it is impractical to teach robots by enumerating every possible skill. An intuitive solution is to decompose the task into smaller primitive actions so that the robot can learn the skills from observing a human. Therefore, as Fig. 1.1 illustrated, in the Chapter 2, we firstly study hand movement learning from human demonstrations. For practical purposes, we propose a system for learning hand actions from markerless demonstrations, which are captured using the Kinect sensor.

The algorithm autonomously segments an example trajectory into multiple action units, each described by a movement primitive, and constructs a task-specific model. With that, the robots could learn and transfer the demonstrations to a library of skills and therefore generate similar movements for different scenarios to perform the task.

Then when the robot is asked to perform a task in a new environment, it could select appropriate skills from the pre-learnt library given the task model. The next challenge is to generate motions adapting to new scenarios. Traditional motion generation methods do not consider versatile constraints from different users, tasks, and environments. For example, the robot may face to an environment where there are obstacles newly perceived while standard movement imitation learning only mimicks the motion and may collide with the obstacles. To tackle this problem, we propose a co-active learning framework for learning to adapt the movement of robot end-effectors in Chapter 3 as illustrated in Fig. 1.1. It is designed to adapt the original imitation trajectories, which are learned from demonstrations, to novel situations with different constraints. The framework also considers user feedback towards the adapted trajectories, and it learns to adapt movement through human-in-the-loop interactions.

In order to further adapt robots to perform more complex manipulation tasks, another challenge is to let robot learn and improve itself by practice. A sequential task is naturally considered as a sequence of pre-learned action primitives, each action primitive has its own goal parameters corresponding to the subgoal. For example, openning a microwave needs a pulling movement after reaching grasping the handle, where the part of handle to be grasped is the subgoal of reaching grasping action. As the third problem in Chapter 4, as illustrated in Fig. 1.1, we are investigating a framework that the robot could not only plan and execute the sequential task in a new environment, but also refine its actions by learning subgoals through re-planning/re-execution during the practice. The proposed system learns the subgoals distribution of given task model using reinforcement learning by iteratively updating the parameters in the trials. As a result, by considering the learned subgoals distribution in sequential motion planning, the proposed framework could adaptively select better subgoals to generate movements for robot to execute the task successfully.

1.2 Main contributions

In this thesis, we aim to address the following questions: (1) How to learn from users' demonstrations of performing manipulation tasks and produce generative action models that the robot can use for execution in different scenario? (2) How to adapt the pre-learnt skill to new environment where there are newly perceived obstacles and other constraints? And how to learn user preferences for such adaptation? (3) Given a sequential task involving a sequence of primitive actions, how to learn and improve the subgoals of the primitive actions through practice so that robot could plan and execute the sequential task successfully? We assume that manipulation tasks can be decomposed into elemental movements in temporal domain according to their common structure while each primitive could be specific to the task with different subgoals. In the first question, we are trying to learn and transfer demonstrations to a library of primitive skills. As a further step, the second question is trying to learn and adapt primitive action to new environment with different constraints. Moreover, the third question is to investigate subgoals learning framework to allow the robot improve itself during the practice.

1.2.1 Learning Hand Movements from Markerless Demonstrations

To answer the first question, we propose an approach for learning the hand movement from markerless demonstrations for humanoid robot tasks. In this work, we demonstrate a markerless system for learning hand actions from movement demonstrations, which are captured using the Kinect sensor. Our approach autonomously segments an example trajectory into multiple action units, each described by a movement primitive, and forms a task-specific model with Dynamic Movement Primitives (DMPs). Using proposed method, we learn a generative model of a human's hand task such as cutting from observations. Similar movements for different scenarios can be generated, and performed on Baxter Robots. The proposed method provides a potentially fully automatic way to learn hand movements for humanoid robots from demonstrations, and it does not require special hand motion capturing devices.

1.2.2 Learning to Adapt Hand Movement in Manipulation Tasks

To address the second question, we propose an framework for interactive learning of movement adaptation for manipulation tasks. Our system could generalize robots' movements learned from demonstrations to fulfill constraints perceived in a new environment. It is able to adapt trajectories according to user preferences. We present an approach for robot learning preferences to adapt trajectories by updating reward weights based on users' feedback. The user thus can co-actively train the robot in-the-loop by demonstrating desired trajectories. We also implement the optimization schema for the skill of "transferring objects" which considers obstacles and different user preferences for the movements. The implementation is validated on a humanoid platform (Baxter). The proposed method generalizes offline learned movement skills to novel situations considering obstacle avoidance and other task-dependent constraints, and it also provides a way to learn how to adapt the movement in on-line interactions with user's feedback.

1.2.3 Dynamic Motion Planning for Sequential Tasks with Subgoals Learning

We answer the last question by proposing an approach of dynamical motion planning for sequential manipulation tasks with subgoals learning. We present a system to generate robots' movements according to sequential task specifcations with adapted subgoals for each primitive action in the task, considering perception constraints in a new environment and embodiment constraints of robot itself. The proposed approach allows the robot to improve sugoals of each primitive action by updating parameters of its distribution through re-planning and re-execution trials. We implement the planning schema for the task of "openning a microwave" involving a sequence of primitive actions and subgoals. And we validate the implementation on a humanoid platform (Baxter) to support our claims.

1.3 Outline

The chapters of this thesis were written such that they can be read independently. Chapter 2 presents the method for learning individual manipulation skills from demonstrations. Chapter 3 describes the approach for learning to adapt movement skills in new environment with different constraints. In Chapter 4, we discuss the framework for learning subgoals of sequential task through practice. Fig. 1.1 presents an overview of robots learning framework and the topics covered in the thesis and their corresponding chapters. In the last sections of Chapter 2 to 4, we also give a summary of our work and provide additional insights into the future extensions.

Chapter 2 presents the framework to learn skills from markerless demonstrations, which are captured through the Kinect sensor. The chapter also introduces an overview of dynamic movement primitives, and how they are used for hand movement learning. This chapter is based on [1].

Chapter 3 describes a movement adaptation approach using model predictive control, which generalizes the imitation trajectories to new environment with different constraints such as obstacle avoidance and safety margin. The chapter also presents a co-active learning framework for learning preferences of movement adaptation using users' feedback. This chapter is based on [2].

Chapter 4 explains methods for learning subgoals distribution for given sequential task models through iterative trials. The learned subgoals distribution is used to connect primitive actions, where each of them is represented as a subgraph for connectivity of robot states, therefore to construct spatiotemporal graph for sequential motion planning.

Chapter 5 presents the main conclusions of the thesis. It also summarizes discussions and potential extensions on related problems for learning manipulation skills.

Chapter 2: Learning Hand Movements from Markerless Demonstrations

We present a framework for generating trajectories of the hand movement during manipulation actions from demonstrations so the robot can perform similar actions in new situations. In this work: 1) we extract and transform hand movement trajectories using a state-of-the-art markerless full hand model tracker from Kinect sensor data; 2) we develop a new bio-inspired trajectory segmentation method that automatically segments complex movements into action units, and 3) we develop a generative method to learn task specific control using Dynamic Movement Primitives (DMPs). Experiments conducted both on synthetic data and real data using the Baxter research robot platform validate our approach.

2.1 Motivation

Developing personalized cognitive robots that help with everyday tasks is one of the on-going topics in robotics research. Such robots should have the capability to learn how to perform new tasks from human demonstrations. However, even simple tasks, like making a peanut jelly sandwich, may be realized in thousands of different ways. Therefore, it is impractical to teach robots by enumerating every possible task. An intuitive solution is to have a generative model to enable the robot to perform the task learned from observing a human. Since the essence of human actions can be captured by skeletal hand trajectories, and most of the daily tasks we are concerned with are performed by the hands, learning new tasks from observing the motion of the human hands becomes crucial.

There are several previous approaches for learning and generating hand movements for a robot, but they either use external markers or special equipments, such as DataGloves, to capture the example trajectories [3–5]. Such approaches are not practical for the kind of actions of daily living, which we consider here. In this work, our system makes use of a state-of-the-art markerless hand tracker [6], which is able to reliably track a 26 degree of freedom skeletal hand model. Its good performance is largely due to reliable 3D sensing using the Kinect sensor and a GPU based optimization. Building on this tool, we propose to develop a user-friendly system for learning hand movements.

The generation of trajectories from example movements using data gloves has been a hot topic in the field of humanoids recently. Krug and Dimitrov [7] addressed the problem of generalizing the learned model. They showed that with proper parameter estimation, the robot can automatically adapt the learned models to new situations. Stulp and Schaal [8] explored the problem of learning grasp trajectories under uncertainty. They showed that an adaptation to the direction of approach and the maximum grip aperture could improve the force-closure performance.

Following the idea that human hand movements are composed of primitives [9, 10], the framework of Dynamic Movement Primitives (DMPs) has become very

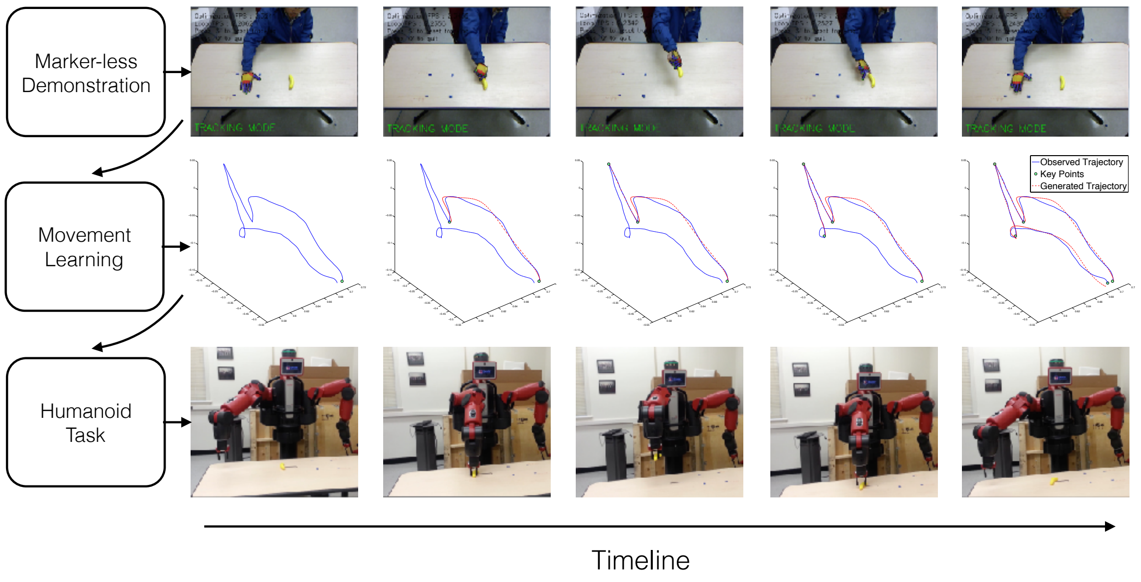


Figure 2.1: Overview of learning hand movements for humanoid tasks. Placing task is an example shown here, and the drawing on hand in demonstration video is indicating hand tracker.

popular for encoding robot trajectories recently. This representation is robust to perturbations and can generate continuous robot movements. Pastor et al. [11] further extended the DMPs model to include capabilities such as obstacle avoidance and joint limits avoidance. The ability to segment complex movements into simple action units plays an important role for the description. With proper segmentation, each action unit can be well fit into one DMP [12, 13].

We propose an approach for learning the hand movement from markerless demonstrations for humanoid robot tasks. Fig. 2.1 gives an overview of our framework. The main contributions of this work are: 1) We demonstrate a markerless system for learning hand actions from movement demonstrations. The demonstrations are captured using the Kinect sensor; 2) Our approach autonomously segments an example trajectory into multiple action units, each described by a movement primitive, and forms a task-specific model with DMPs; 3) We learn a generative model of a human's hand task from observations. Similar movements for different scenarios can be generated, and performed on Baxter Robots.

2.2 Related Work

A variety of methods [14] have been proposed to visually capture human motion. For full pose estimation both appearance-based and model-based methods have been proposed. Appearance-based methods [15] are better suited for the recognition problem, while model-based methods [6] are preferred for problems requiring an accurate estimation pose. To capture hand movement, Oikonomidis et al. [6] provide a method to recover and track the real world 3D data from Kinect sensor data using a model-based approach by minimizing the discrepancy between the 3D structure and the appearance of hypothesized 3D model instances.

The problem of real-time, goal-directed trajectory generation from a database of demonstration movements has been studied in many works [16]- [17]. Ude et al. [16] have shown that utilizing the action targets as a query point in an example database could generate the learned movement to new situations. Asfour et al. [18] use Hidden Markov Models to generalize movements demonstrated to a robot multiple times. Forte et al. [17] further address the problem of generalization from robots' learned knowledge to new situations. They use Gaussian process regression based on multiple example trajectories to learn task-specific parameters.

Ijspeert et al. [9, 19] have proposed the DMP framework. They start with a simple dynamical system described by multiple linear differential equations and transform it into a weakly nonlinear system. It has many advantages in generating motion: It can easily stop the execution of movement without tracking time indices as it doesn't directly rely on time, and it can generate smooth motion trajectories under perturbations. In [8], Stulp et al. present an approach to generate motion under state estimation uncertainties. They use DMP and a reinforcement learning algorithm for reaching and reshaping. Rather than grasping an object at a specific pose, the robot will estimate the possibility of grasping based on the distribution of state estimation uncertainty. The system is tested with new object positions and other state estimation uncertainty distributions.

Krug and Dimitrov [7] propose a method to model grasping movement using movement primitives learned from several demonstrations. They focused on parameter estimation of dynamical systems and formulated the problem as a constrained nonlinear least squares problem and matched the demonstrated trajectories by solving a quadratic program. An implicit dynamical system is maintained to resemble multiple demonstrated trajectories. Experimental results show that the implicit system learned from combined Dynamic Systems (DS) can ensure predictable behavior over the state space.

The segmentation of complex movements into a series of action units has recently received attention due to its importance to many applications in the field of robotics. Meier et al. [13, 20] develop an expectation maximization method to estimate partially observed trajectories. They reduce the movement segmentation problem to a sequential movement recognition problem. Patel et al. [21] use a Hierachical Hidden Markov Model to represent and learn complex tasks by decomposing them into simple action primitives.

Our method utilizes Kinect sensors to capture real word 3D skeleton data of human hand from markerless demonstration; then it automatically segments observed trajectories into multiple action units by the motion characteristics; then, it further represents the hand movement by a generative model based on multiple DMPs and applies the learned model to generate similar movements for new situations. The experiments show that the generated trajectory could be used to drive humanoids with arms and effectors to perform similar tasks. Finally, the generative model could also be used to induce action grammars for describing the hand task.

2.3 Movement Learning

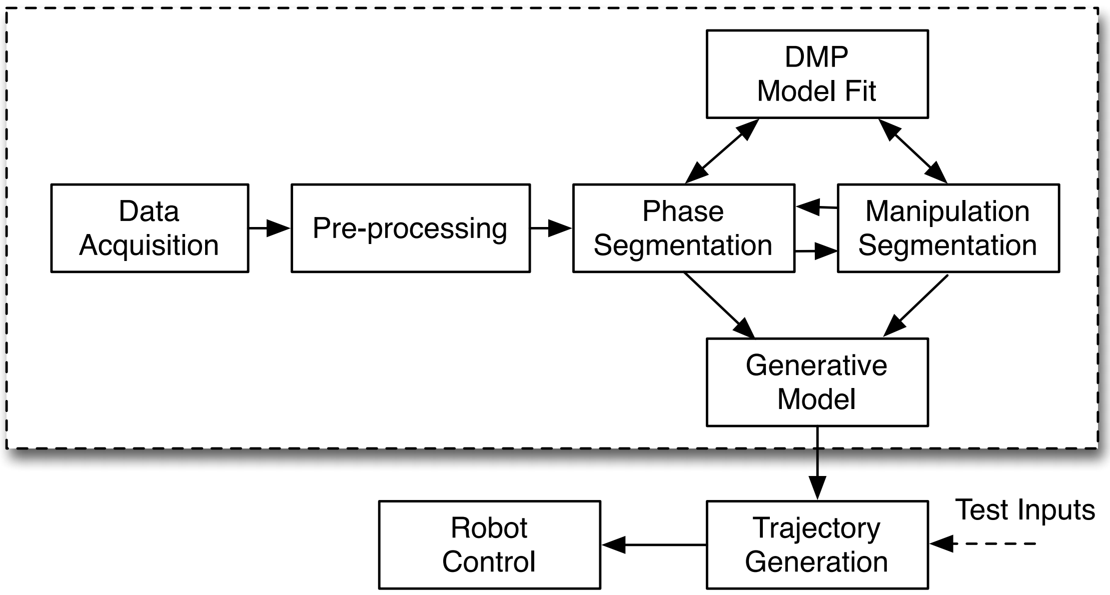


Figure 2.2: Block diagram of hand movement learning

Our hand movement learning method has three steps: 1) acquire trajectories in Cartesian space from demonstration; 2) segment the trajectories using key points and 3) represent each segment with a generative model. Fig. 2.2 presents an overview of our approach. Firstly, the data collected from observed trajectories of the movements of the palm and the fingertips using the markerless hand tracker [6] are pre-processed by applying moving average smoothing to reduce the noise. Next a trajectory segmentation method is applied to find in a bio-inspired way the GRASP and RELEASE points that reflect the phases of movement [22]. Then, because of the complexity of the hand's movement when manipulating objects, a second round of segmentation is applied to the trajectories between the GRASP and RELEASE points to decompose the real movement into periodical sub-movements. Finally, we train the model of Dynamical Movement Primitives (DMPs) [4] to generatively model each sequential movement.

In the following sections, we present each component in Fig. 2.2 in detail.

2.3.1 Data Acquisition from Markerless Demonstrations

The Kinect FORTH Tracking system [6] has been widely used as a state-of-the-art markerless hand tracking method for manipulation actions [23]. The FORTH system takes as input RGB + depth data from a Kinect sensor. It models the geometry of the hand and its dynamics using a 26 DOF model, and treats the problem of tracking the skeletal model of the human hand as an optimization problem using Particle Swarm Optimization. The hand model parameters are estimated continuously by minimizing the discrepancy between the synthesized appearances from the model and the actual observations.

Unlike most other hand data capturing approaches such as those using Data-Gloves, the FORTH system is a fully markerless approach, which makes it possible to achieve a natural human-robot interaction in daily life activities, such as teaching humanoids kitchen actions with bare hands.

In this work, our humanoid is equipped with the FORTH system to track the hand. The observed 3D movement trajectories of the hand, palm, and finger joints are stored as training data.

Since our goal is to generate human-like hand movement on humanoids, we first convert the collected data from Kinect space into Robot space. The robot space is the base frame which takes the robot body center as origin. Then we transform the data from absolute trajectories to relative trajectories with respect to the demonstrator's body center, which is fixed during the demonstration. Then we perform a moving average smoothing on the transformed data to reduce the noise.

In order to learn movements down to the finger level, we also compute the distance between the index finger and the thumb to the DMPs. Without loss of generality, we assumed the robot gripper would have a fixed orientation which is set the same as the demonstrator's.

2.3.2 Dynamic Movement Primitives (DMPs) Model

DMPs [17] are widely used for encoding stereotypical movements. A DMP consists of a set of differential equations that compactly represents high dimensional control policies. As an autonomous representation, they are goal directed and do not directly depend on time, thus they allow the generation of similar movements under new situations.

In this work we use one DMP to describe one segment of the robot trajectory. The discrete trajectory of each variable, y, of the robot hand's Cartesian dimensions, is represented by the following nonlinear differential equations:

$$\tau\dot{v} = \alpha_v(\beta_v(g-y)-v)+f(x) \quad (2.1)$$

$$\tau\dot{y} = v \quad (2.2)$$

$$\tau\dot{x} = -\alpha_x x, \quad (2.3)$$

where (1) and (2) include a transformation system and a forcing function f, which consists of a set of radial basis functions, $\Psi(x)$, (equations (4) and (5)), to enable the robot to follow a given smooth discrete demonstration from the initial position y_0 to the final configuration g. Equation (3) gives a canonical system to remove explicit time dependency and x is the phase variable to constrain the multi-dimensional movement in a set of equations. v is a velocity variable. α_x, α_v, β_v and τ are specified parameters to make the system converge to the unique equilibrium point $(v,y,x)=(0,g,0)$. $f(x)$ and $\Psi(x)$ are defined as:

$$f(x)=\frac{\sum_{k=1}^{N}\omega_k\Psi_k(x)}{\Sigma_{k=1}^{N}\Psi_k(x)}x \quad (2.4)$$

$$\Psi_k(x)=exp(-h_k(x-c_k)^2), h_k>0, \quad (2.5)$$

where c_k and h_k are the intrinsic parameters of the radial basis functions distributed along the training trajectory.

Trajectory Learning: The parameters ω_k in (2.4) are adapted through a learning process such that the nonlinear function $f(x)$ forces the transformation system to follow the observed trajectory $y(t)$. To update the parameters, the derivatives $v(t)$ and $\dot{v}(t)$ are computed for each time step. Based on that, the phase variable $x(t)$ is evaluated by integrating the canonical system in (3). Then, $f_{target}(x)$ is computed according to (2.1), where y_0 is the initial point and g is the end point of the training trajectory. Finally, the parameters ω_k are computed by linear regression as a minimization problem with error criterion $J = \sum_x (f_{target}(x) - f(x))^2$.

Trajectory generation: To generate a desired trajectory, we set up the system at the beginning. The unique equilibrium point condition $(v, y, x) = (0, g, 0)$ is not appropriate here since it won't be reached until the system converges to a final state. The start position is set to be the current position y'_0, the goal is set to be the target position g_{target}, and the canonical system is reset by assigning the phase variable $x = 1$. By substituting the learned parameters ω_k and adapting the desired movement duration τ, the desired trajectory is obtained via evaluating $x(t)$, computing $f(x)$, and integrating the transformation system (2.1).

2.3.3 Movement Segmentation

In human movement learning, a complex action is commonly segmented into simple action units. This is realistic since demonstrations performed by humans can be decomposed into multiple different movement primitives. Specifically for most common human hand movements, it is reasonable to assume that the observed trajectory generally has three subaction units: 1) A reach phase, during which the hand moves from a start location till it comes in contact with the object, just before the grasp action; 2) A manipulation phase, during which the hand conducts the manipulation movement on the object; 3) A withdraw phase, which is the movement after the point of releasing the object.

In both the reach and the withdraw phases, the movements usually can be modelled well by one DMP. However, the manipulation movement could be too complicated to model it with only one or two DMPs. Therefore, our approach is to run a second round of segmentation on the manipulation phase. In this phase we segment it at detected key points and model each segment with a different DMP. The generated trajectory from these DMPs would best fit the training one. Next we describe our segmentation algorithm in detail:

Grasp & Release Candidates: The first step of our algorithm is to identify the GRASP and RELEASE points in the observed trajectories. Given the observed trajectory y, the velocity v and acceleration $\dot{v}$ can be computed by deriving first and second order derivatives followed by a moving average smoothing. Following the studies on human movement [12], the possible GRASP and RELEASE points are derived as the minima points in the motion of the palm. We selected the palm since humans intentionally grasp/release the objects stably by slowing the hand movement. The GRASP point occurs after the human closes the hand, and we find it as the local maxima in the motion of the finger-gap trajectory. The RELEASE point happens before the human opens the hand, and it can be found in a similar way. In this work, we compute a reference trajectory $s(t)$ for each Cartesian dimension representing the motion characteristics as a combination of v and $\dot{v}$ as equation (2.6). Here λ is used to adjust the weight between velocity and acceleration. We also compute $s(t)_{gap}$ for the finger-gap trajectory.

$$s(t) = \lambda \cdot v(t)^2 + (1-\lambda) \cdot \dot{v}(t)^2 \tag{2.6}$$

$$s(t)_{gap} = \lambda_{gap} \cdot v(t)_{gap}^2 + (1-\lambda_{gap}) \cdot \dot{v}_{gap}(t)^2 \tag{2.7}$$

Therefore, for each dimension, the first local minima of $s(t)$ follows the first maxima of $s(t)_{gap}$, and is considered a possible GRASP point candidate. The last local minima of $s(t)$ succeeds the last maxima of $s(t)_{gap}$, and is considered a possible RELEASE point candidate. We take up to three extrema for grasping and three for releasing, and put them into the candidate set C_{grasp} and $C_{release}$ as described in equation (2.8).

$$\begin{aligned}
S &= \{t \mid t = \arg\min_t \; s(t)\} \\
S_{gap} &= \{t \mid t = \arg\max_t \; s(t)_{gap}\} \\
S_c &= \{(t_{grasp}, t_{release}) \mid t_{grasp} < t_{release}; t_{grasp}, t_{release} \in S; \\
&\quad \exists t \in S_{gap}, t < t_{grasp}; \exists t \in S_{gap}, t > t_{release}\}
\end{aligned} \tag{2.8}$$

Manipulation Segmentation: Given the pair of GRASP and RELEASE points, we can get the manipulation phase trajectories. We then attempt to segment the manipulation phase trajectories into subactions. Following the same assumption that hand movements may change at the local minima of the velocity and acceleration, we extract the candidates of the first key points by selecting the first local minima, which follows the first maxima of $s(t)$ during the manipulation phase for each Cartesian dimensional trajectory. If there is no such candidate, our algorithm directly models the current trajectory's segment by one DMP and returns the error between the model-generated and observed trajectories. If there is one possible key point candidate, we use one DMP to model the former part of the trajectory segmented by it and compute the error. Then we recursively apply the same algorithm for the rest of the trajectories to compute key points as well as errors. By summing up the errors, we select the key point with minimal error among all candidates. The selected key point is added to the key point set as described in equation (2.9).

Please refer to Algorithm 1 for details.

$$\begin{aligned}
(t_0, t_n) &:= (t_{grasp}, t_{release}), \forall (t_{grasp}, t_{release}) \in S_c \\
S_{key} &= \{(t_1, \cdots, t_{n-1}) | \forall i = 0, \cdots, n-1 : t_i < t_{i+1}; t_i \in S\} \\
J_{mani} &= \min_{(t_1, \cdots, t_{n-1}) \in S_{key}} \textstyle\sum_{i=1}^{n-1} \sum_{t_i}^{t_{i+1}} (y(t) - y(t)_{generated})^2
\end{aligned} \tag{2.9}$$

Evaluation: We consider the movement segmentation as a minimization problem with error criterion $J(t) = \sum_{i=1,2,3}(y(t)^i - y(t)^i_{generated})^2$. It sums up the errors over all dimensions of the trajectories. For each possible pair of GRASP and RELEASE points ($t_{grasp} \in C_{grasp}$, $t_{release} \in C_{release}$), we first use two separate DMPs to model the reach and withdraw phase trajectories and compute their errors as $J_{reach} = \sum_{t=1}^{t_{grasp}} J(t)$ and $J_{withdraw} = \sum_{t=t_{release}}^{end} J(t)$. Given the manipulation phase trajectory, we segment it further as described above in order to model complex movement, for example chopping. The error for the manipulation phase trajectory (J_{mani}) is then computed. The total error ($J_{whole} = J_{reach} + J_{withdraw} + J_{manipulation}$) is used as the target function. The final GRASP and RELEASE points are obtained by solving $(t^*_{grasp}, t^*_{release}) = \arg\min J_{whole}$ as described in equation (2.10).

$$\begin{aligned}
J_{reach} &= \textstyle\sum_{t=1}^{t_{grasp}} (y(t) - y(t)_{generated})^2 \\
J_{withdraw} &= \textstyle\sum_{t=t_{release}}^{end} (y(t) - y(t)_{generated})^2 \\
(t^*_{grasp}, t^*_{release}) &= \arg\min_{(t_{grasp}, t_{release}) \in S_c} J_{reach} + J_{mani} + J_{withdraw}
\end{aligned} \tag{2.10}$$

Algorithm 1 Manipulation Phase Segmentation

```
Input: $t_{start}, t_{end}$
Output: $Keys$,$J_{error}$
  procedure SEGMENT
      $Keys_c = \emptyset, Keys = \emptyset$
      for all Cartesian dimension $i \in (1, 2, 3)$ do
          $Set_{min} \leftarrow$ FINDMINS$(s(t)^i, t_{start}, t_{end})$
          $Set_{max} \leftarrow$ FINDMAXS$(s(t)^i, t_{start}, t_{end})$
          if $\exists t_c \in Set_{min} > Set_{max}(0)$ then
              $Keys_c \leftarrow Keys_c +$ smallest $t_c$
          end if
      end for
      $J_{error} \leftarrow$ FITDMP$(y(t), t_{start}, t_{end})$
      if $Keys_c = \emptyset$ then return $Keys, J_{error}$
      end if
      for all $t_c \in Keys_c$ do
          $J_{former} \leftarrow$ FITDMP$(y(t), t_{start}, t_c)$
          $Keys_{latter}, J_{latter} \leftarrow$ SEGMENT$(t_c, t_{end})$
          if $J_{error} > J_{former} + J_{latter}$ then
              $J_{error} \leftarrow J_{former} + J_{latter}$
              $Keys \leftarrow t_c + Keys_{latter}$
          end if
      end for
      return $Keys, J_{error}$
  end procedure
```

2.3.4 Movement Generation

After we have found the best GRASP and RELEASE points along with the key points set $(t_1, t_2, \cdots, t_n)$ during the manipulation phase, our system now is able to model the hand movement by:

- **DMPs:** Including two DMPs for the reach and withdraw phases and a set of DMPs for each segment in the manipulation phase, yielding $n + 3$ DMPs.
- **Key Points Set:** A series of best key points $(t_0, t_1, t_2, \cdots, t_{n+1})$ for movement segmentation and their corresponding relative motion vectors $(\vec{MV}_1, \vec{MV}_2, \cdots, \vec{MV}_n)$. The relative motion vectors are computed as $\vec{MV}_i = \vec{y}(t_i) - \vec{y}(t_{i-1}), i = 1, \cdots, n$, where $t_0 = t_{grasp}, t_{n+1} = t_{release}$. Note that the relative motion vectors from t_n to t_{n+1} are abundant for our model.
- **Grasping Finger-gap:** Given the best GRASP and RELEASE points, we compute the average of the finger-gaps during the manipulation phase for representing the distance a parallel gripper should generate for the same object.

Given the testing inputs: the initial locations of the robots palm, the new locations of the object to grasp and release, and the expected movement time, our generative model generates the motion trajectories using the following 3 steps:

Step 1) Generate new key points' locations during the movement $(\vec{y}(t_i)', i = 0, 1, \cdots, n + 1)$. Taking the learned relative motion vectors, we compute locations of new key points as $\vec{y}(t_i)' = \vec{y}(t_{i-1})' + \vec{MV}_i, i = 1, \cdots, n$, where $\vec{y}(t_0)'$ is the new grasping location and $\vec{y}(t_{n+1})'$ is the new releasing location for different scenarios.

Step 2) Scale the duration time of each segment based on the new total time/speed. Since we have a key points set in the learned model, the new duration time for each segment in the manipulation phase $\tau_i, i = 0, \cdots, n$ can be computed.

Step 3) Use learned DMPs to generate each of the segments accordingly. The reach and withdraw phases are generated directly with the test inputs, while the segments in the manipulation phase are generated according to inputs computed from the above steps. For example, $(\vec{y}(t_{i-1})', \vec{y}(t_i)', \tau_i')$ would be used as input to the ith DMP for generating the ith segment trajectory in the manipulation phase.

We then concatenate the generated trajectories into the new movement trajectory $\vec{y}(t)'$, which is then used to control the robot effector. At the same time, we also enforce the learned grasping finger-gap on the robot's parallel gripper during the manipulation phase.

2.4 Experiments

This section describes experiments conducted to demonstrate that our system can learn from markerless demonstrations and generate similar actions in new situations. We first had our robot observe demonstrations. The object was placed on a table, and a human was asked to move his right hand to grasp the object, manipulate it, then release it and withdraw his hand. Three typical tasks are considered: Place, Chop and Saw. In order to validate our method, for each task we collected two sequences. One was used for learning and the other was used for testing. The movement was tracked by the FORTH system [6] at 30 fps and the raw data was

Table 2.1: Hand movement learning for different tasks with different weights λ in motion reference trajectories $s(t)$

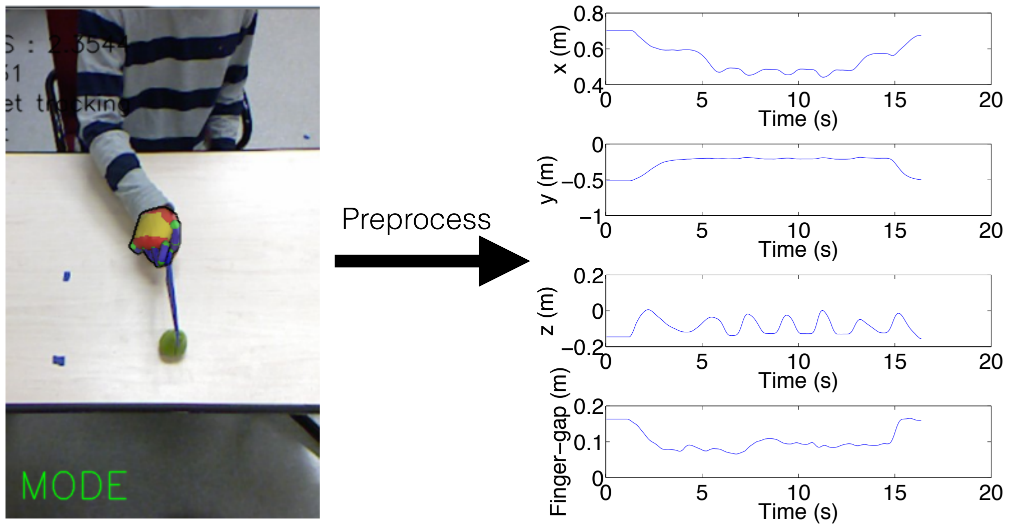
Motion weight	Place (m^2)	Chop (m^2)	Saw (m^2)
$\lambda = 0$	0.1302	0.3614	1.5793
$\lambda = 0.1$	0.1238	0.3697	0.6058
$\lambda = 0.2$	0.1238	0.3539	0.5805
$\lambda = 0.3$	0.1238	0.3539	0.5816
$\lambda = 0.4$	0.1238	0.3537	0.5826
$\lambda = 0.5$	0.1238	0.3537	0.5837
$\lambda = 0.6$	0.1238	0.3532	0.5837
$\lambda = 0.7$	0.1238	0.3532	0.5848
$\lambda = 0.8$	0.1238	0.3532	0.5848
$\lambda = 0.9$	0.1238	0.3532	0.5848
$\lambda = 1$	0.1238	0.3532	0.5848

transformed into robot space, as shown in Fig. 2.3(a).

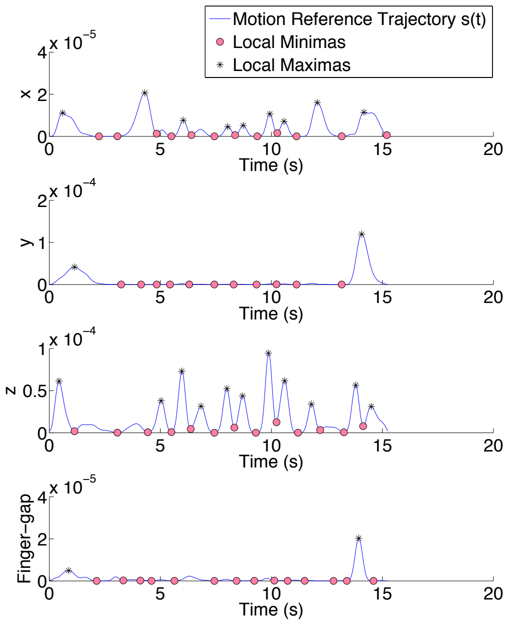
2.4.1 DMPs Model training

We calculated the motion reference trajectories $s(t)$ with motion weight $\lambda = 0.5$, found the local minima and maxima (Sec. 2.3), as shown in Fig. 2.3(b). Applying the learning algorithms by fixing the number of basis functions to 30 in each DMP model, our system generated trajectories (Fig. 2.4).

In order to investigate how motion reference trajectory of palm would be affected by the weight between its velocity and acceleration, we applied our learning algorithms for different tasks with different weights λ. The learned error for the whole trajectories in different tasks and different weights are reported in Table 2.1.

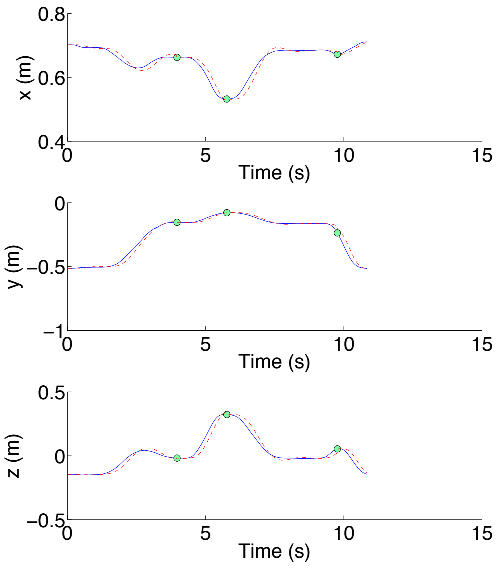


(a)

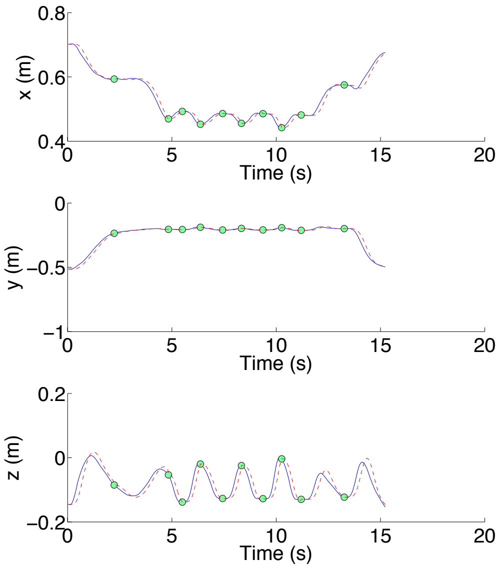


(b)

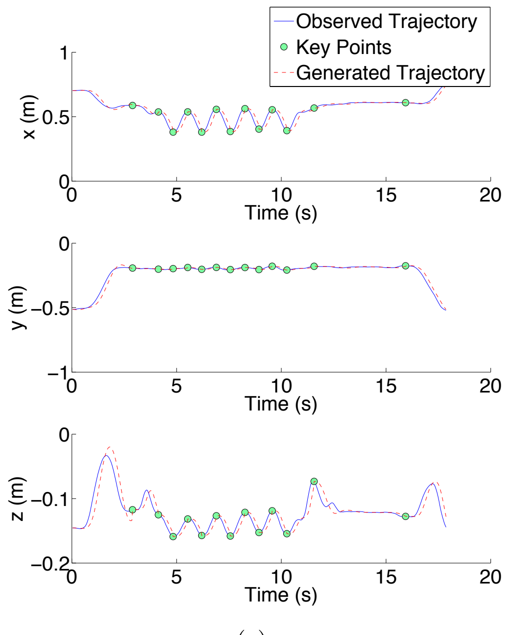
Figure 2.3: Data acquisition and preprocessing for chopping task: (a) Transform action of Kinect sensor data to trajectories in robot space; (b) Computed motion reference trajectories $s(t)$ for Cartesian dimensions and finger-gap with $\lambda = 0.5$.



(a)



(b)



(c)

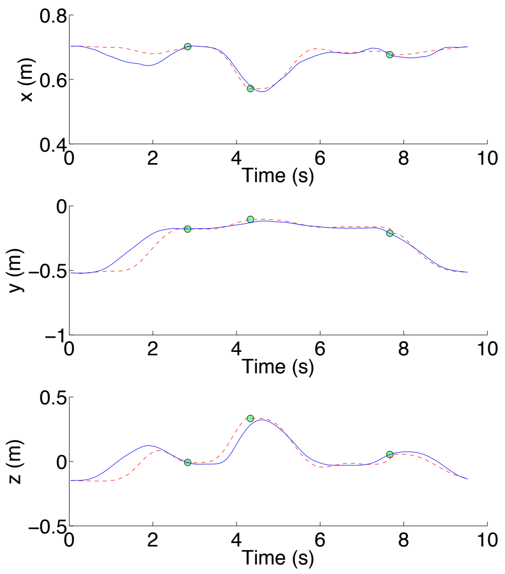
Figure 2.4: Generated trajectories for learning different hand tasks: a) Place, b) Chop, c) Saw.

2.4.2 Experiments in Simulation

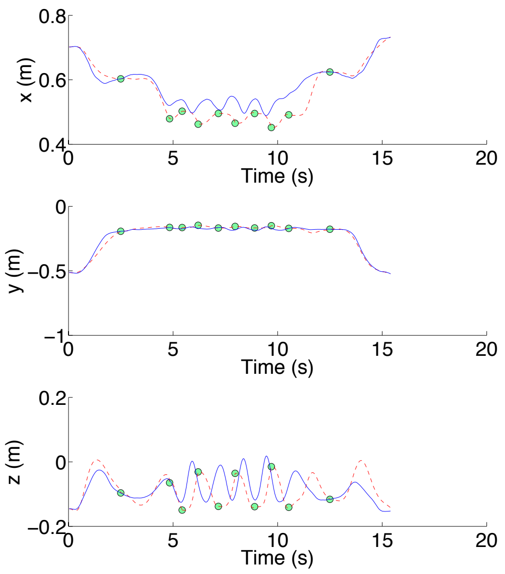
We show how well our approach is able to generalize movement trajectories for different actions by comparing with the testing sequences. For the testing sequence, we applied the same pre-processing to transform it into robot space. We also extracted the grasping and releasing locations, as well as their duration times. We passed them as parameters to the trained model. The trajectories generated are shown in Fig. 2.5.

The motion patterns generated by different humans for the same action largely differ from each other. After comparing the generated trajectories with the observed trajectories of the testing sequences of different tasks, we found that in general their motion patterns are quite similar. Even for relatively complex tasks for example chop, our generated trajectories are similar to the observed human trajectories. This shows that our proposed model is good for learning and generating hand movements for manipulation tasks.

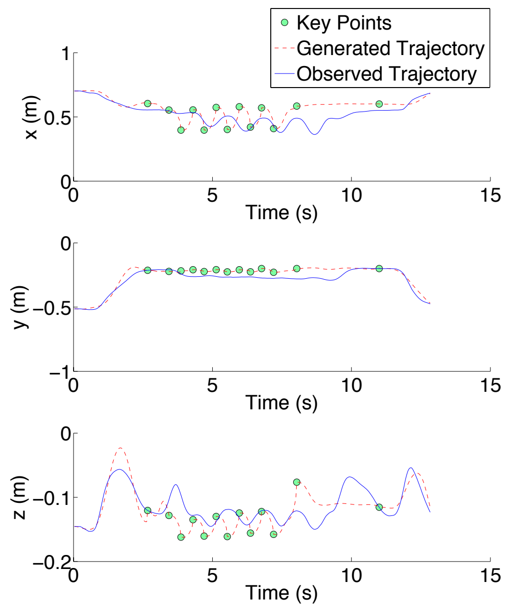
We further tested our trained model by generating trajectories for different grasping and releasing locations. We offset the grasping and releasing locations by 5, 10 and 20 cm on the table away from the location of the demonstration. The generated trajectories for the Chop task are shown in Fig. 2.6. The figure shows that the motion trajectories are still consistent and the generated movements are still quite similar to the ones from the demonstrations. Our approach achieves a certain level of spatial generality while maintaining human-like trajectories.



(a)

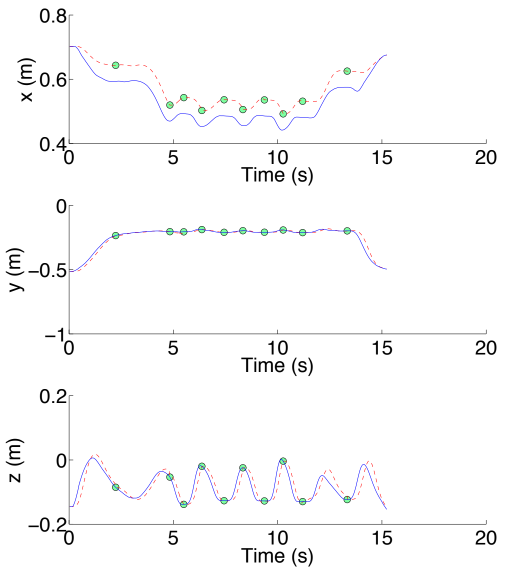


(b)

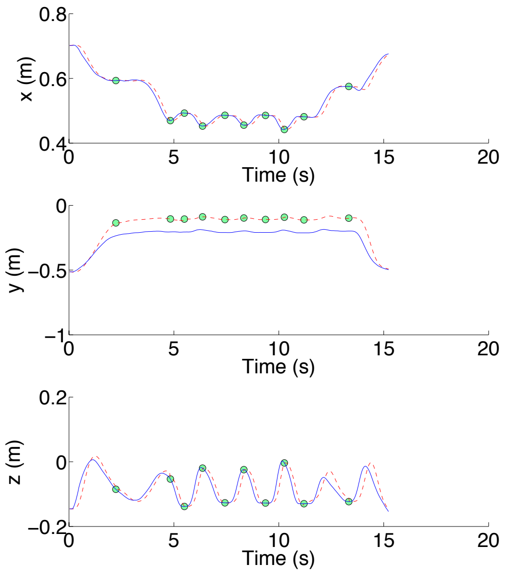


(c)

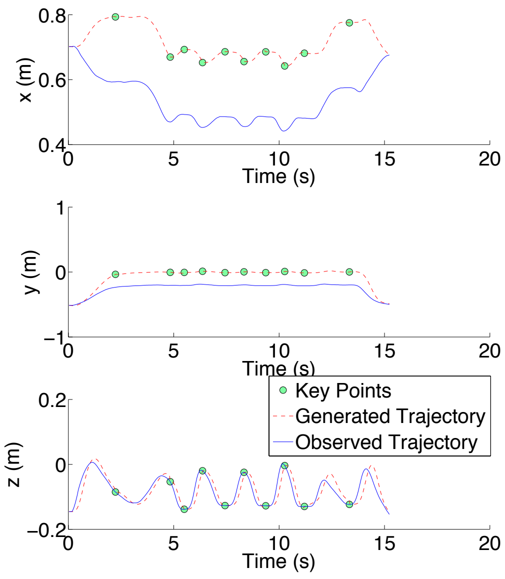
Figure 2.5: Comparison between generated trajectories and observed testing trajectories for different hand tasks: a) Place, b) Chop, c) Saw.



(a)



(b)



(c)

Figure 2.6: Generated trajectories with different scenarios for chopping task: a) object is shifted 5 cm in x direction, b) object is shifted 10 cm in y direction, c) object is shifted 20 cm in both x and y directions.

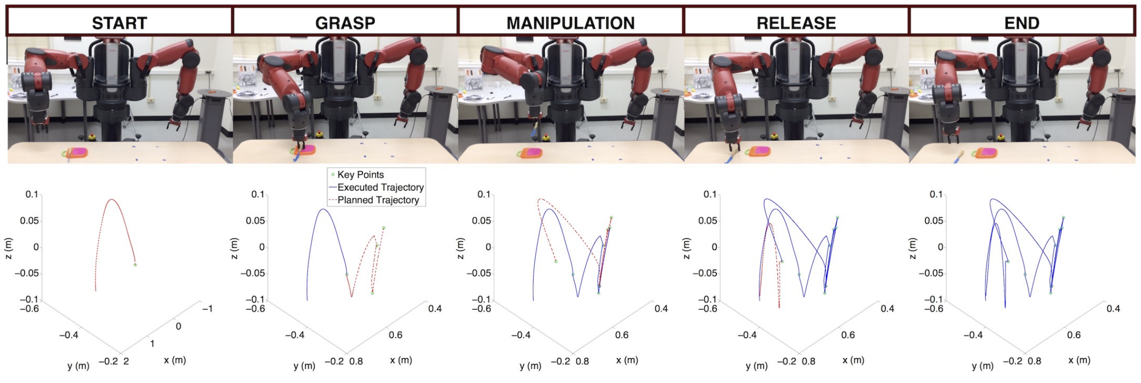


Figure 2.7: Baxter Experiment: Front view and 3D trajectory of generated movement for chopping task.

2.4.3 Test on the Robot

In this experiment, we showed that our approach can be used to teach the Baxter robot to perform a similar task from demonstrations using the FORTH hand tracking data. We mounted a Kinect sensor on our Baxter. Given the object location, using our method, we could generate the hand movement trajectories and use them to control Baxter's gripper movement. Fig. 2.7 shows the front view and 3D trajectory of the generated movement for the chopping task running on Baxter.

2.4.4 Grammar Induction for Hand Task

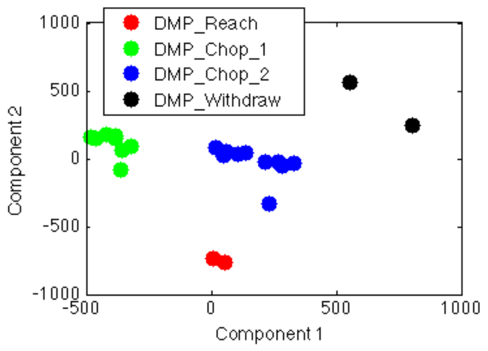
A study by [24] suggested that a minimalist generative grammar, similar to the one in human language, also exists for action understanding and execution. In this experiment, we demonstrated the applicability of our generative model in grammar induction for hand tasks.

With learned DMPs as primitives, we induced a context-free action grammar for the task as follows. Firstly, we concatenated the learned parameters from the different dimensions of the DMPs into feature vectors and applied PCA to transform these vectors into a lower dimensional space. Then we applied K-means clustering with multiple repetitions to cluster DMPs into groups. Besides the two groups of DMPs for Reach and Withdraw phases, we considered two other groups of DMPs for stretching and contraction in the Manipulation phase.

The labelled data from two trails of the chopping task in PCA space are shown in Fig. 2.8(a). Based on clustering labels, we could label each DMP and generate the primitive labels for the observed task. For example, the Chop task in Fig. 2.8(a) can be represented by the sequence of primitives: "Reach Chop1 Chop2 Chop1 Chop2 Chop1 Chop2 Chop1 Chop2 Withdraw". Similar sequences can be found in other Chop trails. After applying the grammar induction technique [25] on the sequences of the primitives, we induced a set of context-free grammar rules, as shown in table. 2.8(b). S is the starting non-terminal. This action grammar enables us to produce generatively new Chop actions and it shows that our generative model is well suited as a basis for further research on learning hand actions guided by semantic principles.

2.5 Summary

We presented a framework for learning hand movement from demonstrations for humanoids. The proposed method provides a potentially fully automatic way to learn hand movements for humanoid robots from demonstrations, and it does not require special hand motion capturing devices. Possible extensions include:



(a)

$$\begin{array}{lcll} S & \rightarrow & Reach\ A\ Withdraw & (1) \\ A & \rightarrow & A\ Chop1\ Chop2 & \\ & & \mid Chop1\ Chop2 & (2) \end{array}$$

(b)

Figure 2.8: Grammar induction for chopping task: (a) DMPs clusters in 2D PCA space; (b) Grammar rules induced from observed movement.

1) Due to the limitation of Baxter's effector, we can only map finger-level movements onto a parallel gripper by transfering the orientation and distance between the thumb and the index finger. One potential extension is to further investigate the eligibility of using our current model to map finger-level movements onto robot hands with fingers.

2) Recent studies on human manipulation methods [22, 26] show that they generally follow a grammatical, recursive structure. It would be very interesting to further investigate the possibility of combining bottom-up (the trajectory segmentation algorithms presented here) with top-down processing (action semantics) and develop a method to learn action grammars based on action units segmented by the presented framework.

3) In this work, in order to focus on the trajectory generation problem, we assumed the object location as input from perception. Currently, we investigate how to integrate additional information about objects, such as their affordances. The modules evaluating object affordances detect the graspable parts of daily kitchen

and workshop tools using different learning mechanisms [27]. This will enable our humanoid to know not only how but also where to grasp.

Chapter 3: Learning to Adapt Hand Movement in Manipulation Tasks

In this work we address the problem of interactive robot movement adaptation under various environmental constraints. A common approach is to adopt motion primitives to generate target motions from demonstrations. However, their generalization capability is weak for novel environments. Additionally, traditional motion generation methods do not consider versatile constraints from different users, tasks, and environments. In this work, we propose a co-active learning framework for learning to adapt the movement of robot end-effectors for manipulation tasks. It is designed to adapt the original imitation trajectories, which are learned from demonstrations, to novel situations with different constraints. The framework also considers user feedback towards the adapted trajectories, and it learns to adapt movement through human-in-the-loop interactions. Experiments on a humanoid platform validate the effectiveness of our approach.

3.1 Motivation

Trajectory learning from human demonstrations has been studied in the field of Robotics for decades because of to its wide range of applications in both industrial and domestic environments. A popular approach uses so-called Motion Primitives

(MPs) to parameterize the observed human motion and reproduce similar motions with different initial and target states. However, it is widely known that general MPs methods, such as Dynamic Movement Primitives (DMPs) [4], exhibit limited capability for generalizing to new environments involving other constraints. Moreover, the learning used in standard MPs does not allow incorporating user preferences, such as preferred movements under geometric constraints. However, humanoid applications in real world environments would greatly benefit from a practical robot movement learning framework that take user preferences and environmental constraints into consideration.

Let's start with a common example. A human user teaches a humanoid how to transfer a bottle from different start and end states. Using an off-the-shelf approach, the robot can learn the motion by acquiring MPs from demonstrated trajectories and applying them to generate new trajectories given different initial and target states. However, solely following the generated trajectories may fail in a slightly altered environment, such as when a bowl is blocking the trajectory as illustrated in Fig. 3.2(a). Therefore, while being able to imitate the movement pattern learned from human demonstrations, at the same time the robot should be able to adapt the learned movement to a novel environment with newly introduced constraints for a successful execution. Here we assume that these constraints are presented to the robot only during the task execution phase (testing phase), and not during the training phase. In this work, we propose an optimization based framework for adapting trained movements to novel environments. The first goal of our system is to generate adapted trajectories, as shown in Fig. 3.2(b), that can: 1) follow

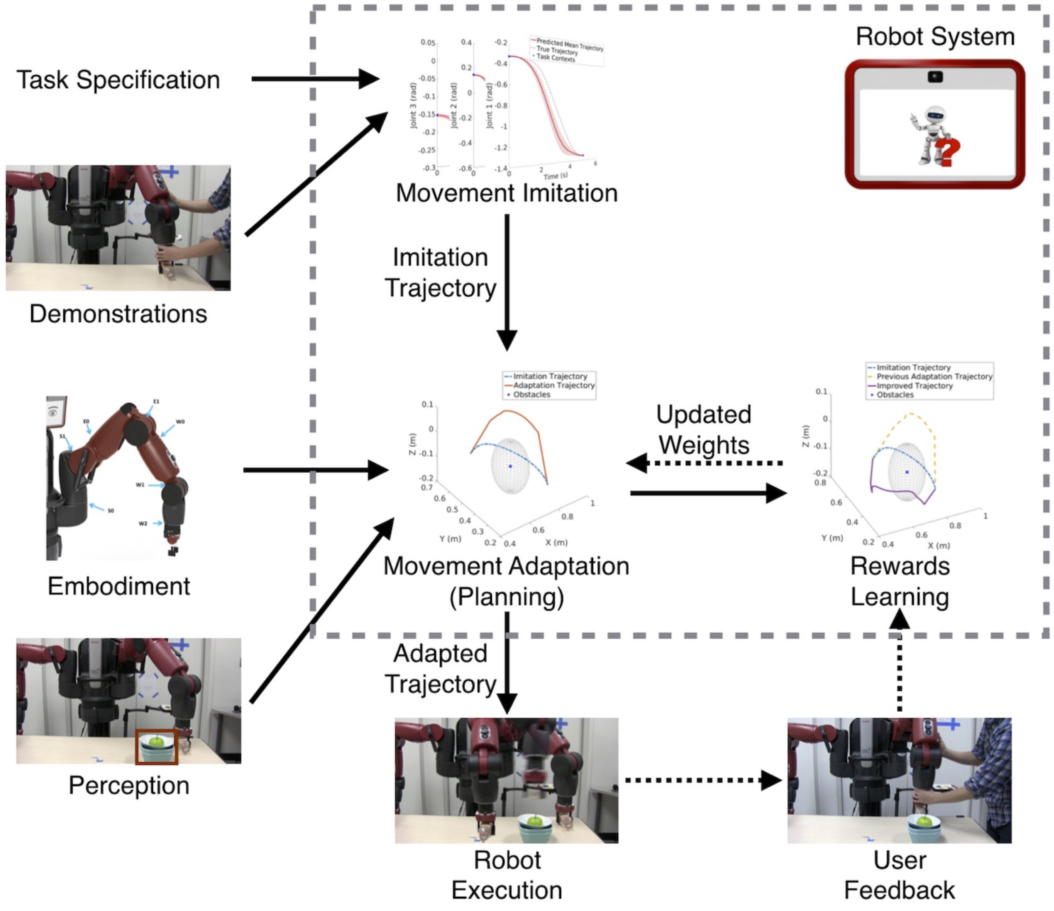
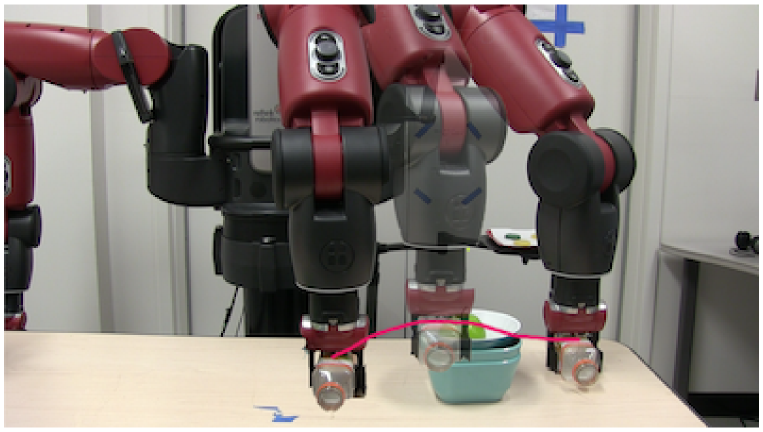


Figure 3.1: System for learning movement adaptation for manipulation tasks. Dashed lines indicate feedback.

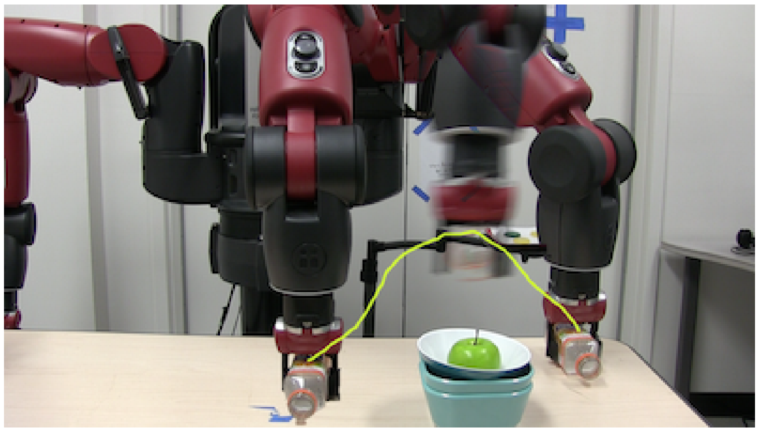
demonstrated trajectories for the purpose of preserving movement patterns, and 2) fulfill novel constraints perceived from the environment during the testing phase.

Moreover, new environmental constraints perceived during the testing phase could be more complex than simply encountering an obstacle. Building on the last example, this time, let's consider the situation where the target bottle is leaking. Ideally an intelligent robot that understands the situation should avoid moving the bottle over the bowl, but follow the movement path *around* it. We could simply adjust in the optimization the objective function for movement adaptation. But what if in another scenario the robot is asked to transfer a knife while avoiding obstacles *above* them to prevent potential scratches? Constraints of this nature are not only associated with the context of the task, i.e, leaking bottle or knife as the manipulated object, but also with the user's preference, i.e, avoiding the bowl in a certain manner. To account for these preferences, a human-in-the-loop on-line adaptation system is necessary. In the optimization framework for generating manipulation trajectories presented in this work, we first treat the reward weights as adjustable parameters that adapt the quality of the trajectory. Then based on user feedback, the framework learns the preferred behavior, that fulfills constraints, by updating the the reward weights. Therefore, the learned behavior can be generalized to different situations with similar constraints. As illustrated in Fig. 3.2(c), after a few iterations of on-line learning, the robot is able to generate a trajectory adapted in accordance with the learned preferences.

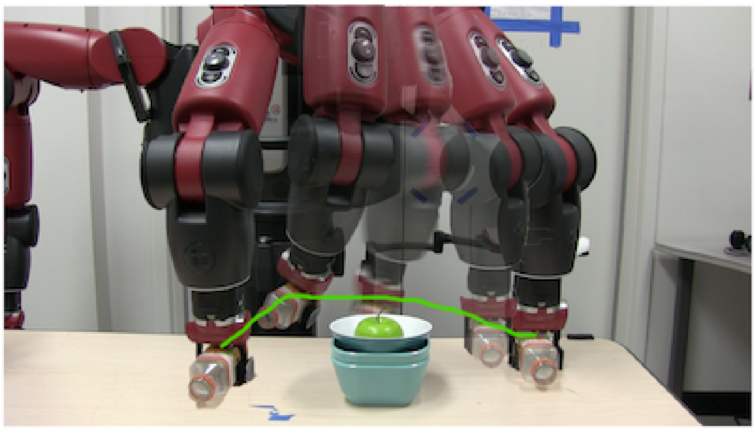
This work proposes an approach for interactive learning of movement adaptation for manipulation tasks. Fig. 3.1 illustrates the proposed system. The main



(a)



(b)



(c)

Figure 3.2: Baxter Transferring Leaking Bottle: (a) Movement imitation, failed to avoid the bowl; (b) Movement adaptation with initial weights, successfully avoided the bowl with a path above it but spilled water into the bowl; (c) Movement adaptation with weights learned for user preferences, successfully avoided the bowl with a path around it and avoided spilling water in the bowl.

contributions of this work are: 1) A system to generalize robots' movements learned from demonstrations to fulfill constraints perceived in a new environment. It is able to adapt trajectories according to user preferences; 2) An approach for robot learning to adapt trajectories by updating reward weights based on users' feedback. The user thus can co-actively train the robot in-the-loop by demonstrating desired trajectories; 3) An implementation of the optimization schema for the skill of "transferring objects", considering obstacles and different geometric user preferences for the movements. We validate the implementation on a humanoid platform (Baxter), and the experimental results support our claims.

3.2 Related Work

Various approaches have been proposed in robotics for learning manipulation movements. A well known approach is imitation learning [18], which focuses on mimicking human demonstrations, and this approach works well when learning from demonstration (LfD) techniques [11] are applicable. However, it only allows to reproduce learned movements in similar environments. To deal with novel environments, extended approaches [28] augment the trajectory generation with additional cost terms or different objective functions as criterion of the trajectories' quality. The criterion is based on human experts' prior knowledge about the task or environment before the execution phase. Then, the motion is generalized to similar situations using predefined constraints. These approaches do not consider user preferences. Here, we present another layer of exploration and learning to adapt the trained movement by considering novel environment constraints, such as observed obstacles and task preferences.

Approaches [29] for encoding the trajectory as motion primitives have been proposed for various forms of generalization and modulation, such as Gaussian mixture regression and Gaussian mixture models [5, 30]. In [31], a mixture model was used to estimate the entire movement skill from several sample trajectories. Another class of approaches employs Hidden Markov models [32].

One popular representation to encode motion from demonstrated trajectories, originally introduced in [4], is Dynamic Movement Primitives (DMPs). It consists of differential equations with well-defined attractor properties and a non-linear learnable component that allows modeling of almost arbitrarily complex motion. A number of methods have been developed to expand DMPs to cope with new environments. To avoid obstacles in new environments, Guenter et al. [30] developed a Gaussian mixture model to enable the robot adapting in a constrained environment by combining dynamical system with reinforcement learning. Park et al. [33] introduced the gradient of a dynamic potential field to the differential equation of the DMPs as an acceleration term, which depended on the relative distance and velocity between a robots end effector and an obstacle. Hoffmann et al. [34] also added an acceleration term in the DMP formulation inspired by biology to avoid colliding with a moving obstacle, relating the position of the end effector to the position of the obstacle. Calinon et al. [35] also proposed to safely avoid collision with a human by estimating a risk indicator that modulates repulsive force. Besides obstacle avoidance, other modulations for trajectory generated by DMPs are studied such as force feedback modulation [36, 37] and joint-angle limits [29].

Recently, Probabilistic Movement Primitives (ProMPs) [38] was proposed as an alternative representation. It learns a trajectory distribution from multiple demonstrations and modulates the movement by conditioning on desired target states. Incorporating the variance of demonstrations, the ProMPs approach handles noise from different demonstrations and provides increased flexibility for reproducing movement. However, all these approaches hardly deal with novel environments such as involving different obstacles. In our work, we first train the robot using ProMPs, and then generalize these trained motion primitives to newly introduced environmental constraints.

In order to enable MPs to adapt to novel environments with obstacles [33], Kober et al. [28] proposed an augmented version of DMPs which incorporates perceptual coupling to an external variable. They first learned the initial dynamic models by standard imitation learning and subsequently used a reinforcement learning method for self-improvement. Ghalamzan et al. [39] proposed a three-tiered approach for robot learning from demonstration that can generalize noisy task demonstrations to new target states and to environments with obstacles. They encoded the nominal path generated from a Gaussian Mixture Model with DMPs and generated a trajectory for a new target state. Then they adapted the DMP-generated trajectory to avoid obstacles by formulating an optimal control problem regarding the reward function learned from demonstrations by inverse optimal control. This approach allows a non-expert user to teach a robot the desired response to different objects but requires offline training in the environment containing the obstacles to learn the reward function. However, in real world scenarios, the human users often have different preferences for the generated trajectories according to various environments and tasks, while it is extremely challenging for them to provide the optimal trajectories in every situation. To account for this, in our approach, the human users can interactively provide sub-optimal suggestions on how to improve the trajectory and the robot learns the preference for different constraints, and incorporates it to generate more applicable trajectories.

User preferences for a robot's trajectories have been studied in the field of human robot interaction (HRI). Sisbot et al. [40] proposed to model user specified preferences as constraints on the distance of the robot from the user, the visibility of the robot and the users arm comfort. Then a path planner fulfilling such user preferences was provided. Ashesh Jain et al. [41] proposed a co-active learning method to learn user preferences over generated trajectories for manipulation tasks by iteratively taking user sub-optimal feedback, and the optimal trajectory was selected based on the learned reward function. In our work, we adopt the co-active learning paradigm and further propose a reward formulation to model user preferences over constraints for movement generation. Then we integrate it with movement adaptation through optimization based planning.

3.3 Co-active Learning for Movement Generalization

For the problem of robot learning from demonstrations [11], a common practice is to offline learn the skills by encoding the trajectories with movement patterns such as DMPs [1]. During the testing phase, they can then be used to generalize the movement to novel situations with slight alterations, such as different initial and target states. However, this generalization capability does not apply to novel environments with different obstacles or to a new task contexts with a variety of manipulated objects. In this work, we propose a complementary framework for generalizing off-line learned movement skills to novel situations, and in addition we incorporate on-line learning preferences of how to generalize from human's feedback co-actively.

While facing a novel situation, the robot is given a manipulation task context $\boldsymbol{x}_c$ that describes the environment, the objects and any other task-related information. It could compute an imitation movement trajectory $\boldsymbol{y}_D$ by generalizing offline learned skills to new initial and target states. Such a trajectory can be executed if the new environment does not have obstacles and there are no other constraints inherited from the task.

To further generalize learned movement skills to more challenging situations, the robot has to generate an adapted trajectory $\boldsymbol{y}$ based on the task contexts $\boldsymbol{x}_c$ and the computed imitation trajectory $\boldsymbol{y}_D$. Here we use a reward function $f^*(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$ to reflect how much reward the adapted trajectory $\boldsymbol{y}$ can achieve for different contexts. This way, we can adapt the movement by solving an optimal control problem which outputs an adapted trajectory by maximizing the reward function f^*. The reward function consists of a Imitation Reward f_D describing the tendency to follow the imitation trajectory $\boldsymbol{y}_D$, a Control Reward f_C describing the smoothness of executing the adapted trajectory $\boldsymbol{y}$ and a Response Reward f_E describing the expected response given the environment. Although this reward function can be recovered from demonstrations by Inverse Optimal Control, as [39] suggests, it assumes that demonstrations are from experts, which bears an oracle reward function. In fact, it is common for non-expert users to provide non-optimal trajectories in practice. Also, [39] requires the manipulated objects or obstacles to exist during demonstration, and is hard to update the learned reward function online when the robot is facing situations that involve new objects. To learn the reward function which controls how the robot adapts trajectories under new contexts, we apply a co-active learning technique [41] in which the user only corrects the robot by providing an improved trajectory $\bar{\boldsymbol{y}}$ and then the robot updates the parameter $\boldsymbol{w}$ of $f(\cdot; \boldsymbol{w})$ based on the user's feedback. It is worth noting that this feedback only indicates $f^*(\bar{\boldsymbol{y}}, \boldsymbol{x}_c, \boldsymbol{y}_D) > f^*(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$, and $\bar{\boldsymbol{y}}$ may be non-optimal trajectories. With iterations of improvement, the robot could learn a function that approximates the oracle $f^*(\cdot)$ tightly.

3.4 Our System

Overall, after the robot has offline learned the movement skill from demonstrations, when facing a different task context $\boldsymbol{x}_c$ in a novel environment, the testing phase includes three stages: 1) Movement Imitation, which computes an imitation trajectory $\boldsymbol{y}_D$ by generalizing demonstrated movement to new initial and target states; 2) Movement Adaptation, which generates an adapted trajectory $\boldsymbol{y}$ under new task and environment contexts by maximizing the given reward function; 3)

Rewards Learning, which updates the parameters of estimated reward function according to the user's feedback through co-active learning. Fig. 3.1 demonstrates our proposed framework. In the following sections, we formulate each stage.

3.4.1 Movement Imitation

At the beginning, our system offline learns movement skills in an environment without obstacle or other constraints. In this work, we adopt the Probabilistic Movement Primitives (ProMPs) [38] for offline learning and movement imitation. It obtains a distribution over trajectories from multiple demonstrations, which captures the variations, and can be easily generalized to new initial and target states while imitating the movement.

To be specific, we consider that a robot's end-effector has d degrees of freedom (DOF) along with its arm, with its state denoted as $\boldsymbol{y}(t) = [y_1(t), \ldots, y_d(t)]^T$. The trajectory of the robot's end effector is represented as a sequence $\boldsymbol{\mathcal{T}} = \{\boldsymbol{y}(t)\}_{t=0,\ldots,T}$. We model each dimension i of $\boldsymbol{y}(t)$ using linear regression with n Gaussian time-dependent basis functions ψ and a n-dimensional weight vectors w_i as

$$y_i(t) = \psi(t)^T w_i + \epsilon_y, \tag{3.1}$$

where $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ denotes zero-mean i.i.d. Gaussian noise. With the underlying weight vectors $\boldsymbol{w} = [w_1^T, \ldots, w_d^T]^T$, the probability of observing a trajectory $\boldsymbol{\mathcal{T}}$ can be given by

$$p(\boldsymbol{\mathcal{T}}|\boldsymbol{w}) = \prod_t p(\boldsymbol{y}(t)|\boldsymbol{w}) = \prod_t \mathcal{N}(\boldsymbol{y}(t)|\boldsymbol{\Psi}(t)^T\boldsymbol{w}, \boldsymbol{\Sigma_y}), \tag{3.2}$$

where $\boldsymbol{\Psi}(t) = \text{diag}(\overbrace{\psi(t), \ldots, \psi(t)}^{d})$ and $\boldsymbol{\Sigma_y} = \sigma_y^2 \boldsymbol{I}_{d\times d}$.

3.4.1.1 Learning from Demonstrations

For each demonstration, the trajectory can be easily represented by a weight vector $\boldsymbol{w}$ which has fewer dimensions than the number of time steps. To capture trajectory variations from multiple demonstrations of the movement, a Gaussian distribution $p(\boldsymbol{w};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w})$ over the weights w is estimated. Therefore, the distribution of the trajectory $p(\boldsymbol{\mathcal{T}}|\boldsymbol{w})$ can be represented as

$$p(\boldsymbol{\mathcal{T}};\boldsymbol{\theta}) = \int p(\boldsymbol{\mathcal{T}}|\boldsymbol{w})p(\boldsymbol{w};\boldsymbol{\theta})d\boldsymbol{w} \tag{3.3}$$

$$= \prod_t \mathcal{N}(\boldsymbol{y}(t)|\boldsymbol{\Psi}(t)^T\boldsymbol{\mu_w}, \boldsymbol{\Psi}(t)^T\boldsymbol{\Sigma_w}\boldsymbol{\Psi}(t)^T + \boldsymbol{\Sigma_y}) \tag{3.4}$$

We can then estimate the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu_w}, \boldsymbol{\Sigma_w}\}$ by using maximum likelihood estimation as suggested in [38].

3.4.1.2 Trajectory Generation

In novel situations, the trajectory could be modulated by conditioning with different observed states. By adding an observation vector of $\boldsymbol{Y^*} = [\boldsymbol{y}_0^{*T}, \boldsymbol{y}_T^{*T}]^T$

indicating the desired initial state $\boldsymbol{y}_0^*$ and target state $\boldsymbol{y}_T^*$ with accuracy $\boldsymbol{\Sigma}_{\boldsymbol{y}}^*$, we apply Bayes theorem and represent conditional distribution for $\boldsymbol{w}$ as

$$
\begin{aligned}
p(\boldsymbol{w}|\boldsymbol{Y^*}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}'_{\boldsymbol{w}}, \boldsymbol{\Sigma}'_{\boldsymbol{w}}) \propto \mathcal{N}\left(\boldsymbol{Y}^*|\boldsymbol{\Psi}^{*T}\boldsymbol{w}, \boldsymbol{\Sigma}_{\boldsymbol{Y}}^*\right) p(\boldsymbol{w}) \\
\boldsymbol{\mu}'_{\boldsymbol{w}} &= \boldsymbol{\mu}_{\boldsymbol{w}} + \boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^*\left(\boldsymbol{\Sigma}_{\boldsymbol{Y}}^* + \boldsymbol{\Psi}^{*T}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^*\right)^{-1}\left(\boldsymbol{Y}^* - \boldsymbol{\Psi}^{*T}\boldsymbol{\mu}_{\boldsymbol{w}}\right) \\
\boldsymbol{\Sigma}'_{\boldsymbol{w}} &= \boldsymbol{\Sigma}_{\boldsymbol{w}} - \boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^*\left(\boldsymbol{\Sigma}_{\boldsymbol{Y}}^* + \boldsymbol{\Psi}^{*T}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{\Psi}^*\right)^{-1}\boldsymbol{\Psi}^{*T}\boldsymbol{\Sigma}_{\boldsymbol{w}}
\end{aligned}
\tag{3.5}
$$

where $\boldsymbol{\Psi}^* = [\boldsymbol{\Psi}(0), \boldsymbol{\Psi}(T)]$ and $\boldsymbol{\Sigma}_{\boldsymbol{Y}}^* = \mathrm{diag}(\boldsymbol{\Sigma}_{\boldsymbol{y}}^*, \boldsymbol{\Sigma}_{\boldsymbol{y}}^*)$ are augmented for observation vector $\boldsymbol{Y}^*$.

With a conditional distribution of $\boldsymbol{w}$, we can generate conditional trajectory distribution and easily evaluate the mean $\boldsymbol{y}_D$ and the variance $\boldsymbol{\Sigma}_D$ of the trajectory $\boldsymbol{\mathcal{T}}$ for any time point t according to Eq.(3.2) and Eq.(3.3). Therefore, the mean trajectory $\boldsymbol{y}_D(t)$ can be used as the imitation trajectory in movement adaptation and the variance $\boldsymbol{\Sigma}_D(t)$ can be used to indicate which parts or dimensions of the trajectory are more flexible to adapt. A larger variance reflects higher variations in demonstrations. It means more flexibility for modifying the corresponding part of the trajectory.

It is worth mentioning that, although we adopt ProMPs for movement imitation in this work, the proposed Movement Adaptation framework can be integrated into any other movement imitation learning technique.

3.4.2 Movement Adaptation

As mentioned before, if the environment of a new situation is exactly the same as the one during demonstration when ProMPs are learned, e.g, no obstacle, safety constraints or other new considerations, the robot can perform the movement optimally by directly following the imitation trajectory $\boldsymbol{y}_D \in \mathbb{R}^d$ generated by learned ProMPs in discrete time.

In this work, we want to have a system that can adapt to an environment with novel constraints. Thus, we model the movement adaptation as an optimal control problem with fixed time horizon T in discrete time. The output of the adaptation system is a new trajectory $\boldsymbol{y} \in \mathbb{R}^d$ in discrete time. The input consists of the task context $\boldsymbol{x}_c$ that describes the environment, the objects and any other task-related information which are obtained from the perception module, the imitation trajectory $\boldsymbol{y}_D$ which is generated from learned ProMPs, and the reward function $f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$ which represents the reward of the adapted trajectory $\boldsymbol{y}$ corresponding to the new situation.

3.4.2.1 Optimization with Constraints

Let's consider that the perception module detects N_{obj} objects in the environment, which may be obstacles during the manipulation. Each object is abstracted as a sphere in the space represented by its center location and semi-diameter $\{\boldsymbol{O}_k, d_k\}, k = 1, \ldots, N_{obj}$. We assume that the reward function can be modeled as accumulated sum of rewards from each state $\boldsymbol{y}(t)$ at time step t:

$$f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D) = \sum_{t=0}^{T} f_t(\boldsymbol{y}(t), \boldsymbol{x}_c, \boldsymbol{y}_D). \tag{3.6}$$

Because we are only modulating the trajectory, we can model the adaptation system as linear dynamics with the control signal $\boldsymbol{a} \in \mathbb{R}^m$, as it does not involve real physical dynamics. Considering the embodiment of robotic end-effectors, we can compute the end-effector's position in spatial space $\boldsymbol{E}(\boldsymbol{y})$ following the kinematics modeling [42]. Then, considering obstacles avoidance in spatial space, the target optimal policy $\pi^* = \{\boldsymbol{a}(t)^*\}_{t=0,\ldots,T-1}$ can be defined from Eq. (3.7) with constraints.

$$\max_{\boldsymbol{y}(t)} \sum_{t=0}^{T} f_t(\boldsymbol{y}(t), \boldsymbol{x}_c, \boldsymbol{y}_D) \tag{3.7}$$

$$\text{subj. to} \qquad \forall t = 0, \cdots, T-1 \tag{3.8}$$

$$\boldsymbol{U} \geq \boldsymbol{y}(t) \geq \boldsymbol{L} \tag{3.9}$$

$$\|\boldsymbol{E}(\boldsymbol{y}(t)) - \boldsymbol{O}_k\|^2 \geq d_k^2, \quad \forall k = 1, \cdots, N_{obj} \tag{3.10}$$

$$\boldsymbol{y}(T) = \boldsymbol{y}_D(T), \tag{3.11}$$

where $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are system matrices, Eq.(3.11) constrains the final position of the adapted trajectory, Eq.(3.9) constrains the trajectory within feasible limits, and Eq. 3.10 ensures the adapted trajectory can avoid obstacles safely by keeping a minimum distance d_k between the robot's end-effector and any object.

3.4.2.2 Model Predictive Control

In order to find an optimal solution of such a system with continuous state and action spaces, we adopt Model Predictive Control which computes the optimal actions in a finite prediction horizon. Therefore, by considering a prediction time horizon T_p, the optimal action $\boldsymbol{a}(i)^*$, at time step $i = 0, \dots, T-1$, can be solved by:

$$
\begin{aligned}
\max_{(\boldsymbol{a}(i),\cdots,\boldsymbol{a}(i+T_p-1))} & \textstyle\sum_{t=i+1}^{i+T_p} f_t(\boldsymbol{y}(t), \boldsymbol{x}_c, \boldsymbol{y}_D) \\
\text{subj. to} \quad & \forall t = i, \cdots, i+T_p-1 \\
& \boldsymbol{z}(t+1) = \boldsymbol{A}\boldsymbol{z}(t) + \boldsymbol{B}\boldsymbol{a}(t) \\
& \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{z}(t) \\
& \boldsymbol{U} \geq \boldsymbol{y}(t) \geq \boldsymbol{L} \\
& \|\boldsymbol{E}(\boldsymbol{y}(t)) - \boldsymbol{O}_k\|^2 \geq d_k^2, \quad \forall k = 1, \cdots, N_{obj} \\
& \boldsymbol{y}(T) = \boldsymbol{y}_D(T).
\end{aligned}
\tag{3.12}
$$

At each step i, the optimal actions $\{\boldsymbol{a}(i)^*, \cdots, \boldsymbol{a}(i+T_p-1)^*\}$ for T_p decision steps in the future are computed but only the action for the current step $\boldsymbol{a}(i)^*$ is performed. Therefore, it can deal with changing environments as these changes could be considered in the next decision steps.

3.4.2.3 Reward Function

In order to adapt robot movements to perform well in novel situations, considering only hard constraints such as obstacle avoidance, Eq.(3.10), does not suffice.

Thus, our framework further models a reward function $f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$ that reflects the amount of rewards that an adapted trajectory $\boldsymbol{y}$ can gain within the context $\boldsymbol{x}_c$ and $\boldsymbol{y}_D$. As the reward function $f(\boldsymbol{y})$ is assumed temporally discrete in Eq.(3.6), we model the reward function $f_t(\boldsymbol{y}(t))$ at t by three parts:

$$f_t(\boldsymbol{y}(t); \boldsymbol{w}) = f_{D,t}(\boldsymbol{y}(t); \boldsymbol{w}_D) + f_{C,t}(\boldsymbol{y}(t); \boldsymbol{w}_C) + f_{E,t}(\boldsymbol{y}(t); \boldsymbol{w}_E), \tag{3.13}$$

where the Imitation Reward f_D models the tendency to follow the imitation trajectory $\boldsymbol{y}_D$, the Control Reward f_C models the smoothness of executing the adapted trajectory $\boldsymbol{y}$ and the Response Reward f_E characterizes the expected response to the environment. Meanwhile, $\boldsymbol{w} = [\boldsymbol{w}_D^T, \boldsymbol{w}_C^T, \boldsymbol{w}_E^T]^T$ are parameters that affect the behavior of the movement adaptation. Next we describe each reward function in detail.

Imitation Reward: The Imitation Reward characterizes how well the adapted trajectory can imitate the demonstrations by the distance between points on $\boldsymbol{y}$ and $\boldsymbol{y}_D$. Recall that we have the variance $\boldsymbol{\Sigma}_D(t)$ of the imitation trajectory $\boldsymbol{y}_D$ by Movement Imitation 3.4.1.2, which indicates how flexible we can adapt the trajectory. Considering $\boldsymbol{\Sigma}_D(t) = \mathrm{diag}(\sigma_1^2(t), \ldots, \sigma_d^2(t))$ to be diagonal for the sake of simplicity, we model the Imitation Reward by the weighted distance:

$$f_{D,t}(\boldsymbol{y}(t); \boldsymbol{w}_D) = -(\boldsymbol{y}(t) - \boldsymbol{y}_D(t))^T \boldsymbol{V}(t)(\boldsymbol{y}(t) - \boldsymbol{y}_D(t)) \tag{3.14}$$

$$\boldsymbol{V}(t) = \mathrm{diag}(\boldsymbol{w}_D)\mathrm{diag}(e^{-\sigma_1^2(t)}, \ldots, e^{-\sigma_d^2(t)}), \tag{3.15}$$

where $\boldsymbol{V}(t)$ is a weight matrix consisting of parameters $\boldsymbol{w}_D$ and $\{e^{-\sigma_i^2(t)}\}$ in which the variances learned from demonstrations $\boldsymbol{\Sigma}_D(t)$ are modeled to affect adaptation rewards.

Control Reward: The Control Reward f_C characterizes the smoothness of executing the adapted trajectory $\boldsymbol{y}$ using the following formulation:

$$f_{C,t}(\boldsymbol{y}(t); \boldsymbol{w}_C) = -\boldsymbol{w}_C \|(\boldsymbol{y}(t) - \boldsymbol{y}(t-1))\|^2, \tag{3.16}$$

where $\boldsymbol{w}_C$ is the parameter to weigh this reward.

Response Reward: The Response reward f_E describes the expected response to the environment, such as safety considerations for obstacles and objects under manipulation. Here we give an intuitive examples for the Response Reward. Although we can ensure minimum distance to avoid obstacles using Eq.(3.12), as human users we still expect the robot to transfer a cup full of water *around* a laptop instead of *above* it, to avoid potential spills. Another example is that the user would prefer that the robot when manipulating sharp objects, such as knives, keeps a relatively larger distance from the human for safety consideration. As another example, for safety consideration, we prefer the robot to transfer fragile objects while staying close to the table top to maintain a safety margin. All the above preferences are specific to objects under manipulation and the exact environment. Thus, we set the Response Reward such that the better the adapted trajectory fulfills the preferences, the higher the reward is.

To formally represent the Response Reward, let us consider a scenario with N_{obj} obstacles on the table. The leftmost and rightmost locations of the table are $\boldsymbol{B}_1, \boldsymbol{B}_2$ and the table surface is $\boldsymbol{S}$, we then can formulate the Response Rewards as follows:

$$f_{E,t}(\boldsymbol{y}(t); \boldsymbol{w}_E) = -\left(\sum_{k=1}^{N_{obj}} \boldsymbol{w}_{O,k}^T \boldsymbol{\phi}_{O,k} + w_B \phi_B + w_S \phi_S\right) \tag{3.17}$$

$$\begin{aligned} \boldsymbol{\phi}_{O,k}^T &= \left[-\|\boldsymbol{E}(\boldsymbol{y}(t)) - \boldsymbol{O}_k\|, (\boldsymbol{E}(\boldsymbol{y}_D(t)) - \boldsymbol{E}(\boldsymbol{y}(t)))^T\right] \\ &\cdot \quad \exp\left(-\tfrac{\|\boldsymbol{E}(\boldsymbol{y}(t)) - \boldsymbol{O}_k\|^2}{d_k}\right) \end{aligned} \tag{3.18}$$

$$\phi_B = \sum_{i=1}^{2} \exp\left(-\frac{\|\boldsymbol{E}(\boldsymbol{y}(t)) - \boldsymbol{B}_i\|^2}{d_{min}}\right) \tag{3.19}$$

$$\phi_S = \|\boldsymbol{E}(\boldsymbol{y}(t)) - S\|^2, \tag{3.20}$$

where $\boldsymbol{\phi}_{O,k}$ represents the feature vector for preferences in avoiding obstacle $\boldsymbol{O}_k$, of which the first element denotes avoiding distance and the second element denotes the deviation vector as shown in Fig. 3.3. The preferred deviation vector is given as reward weights and the inner product between two vectors indicates the rewards of deviation considering the given preference. The exponential decay function is applied so that the features are only effective when the robot's end-effector is close to the obstacles. ϕ_B and ϕ_S are features related to safety by considering boarders and surface of the table. $\boldsymbol{w}_E = [\boldsymbol{w}_{O,1}^T, \ldots, \boldsymbol{w}_{O,N_{obj}}^T, w_B, w_S]^T$ are weights corresponding to the features respectively.

Given a set of parameters $\boldsymbol{w} = [\boldsymbol{w}_D^T, \boldsymbol{w}_C^T, \boldsymbol{w}_E^T]^T$, the MPC module generates an adapted trajectory by maximizing $f(\cdot; \boldsymbol{w})$. The robot could follow the adapted trajectory and execute the task facing the novel situation. However, the generated trajectory may not be sufficiently satisfying from a user's perspective, since the given or initialized parameters may not be accurate for modeling the rewards. To accommodate this issue, after the movement execution, our system allows the user to provide a better trajectory as feedback to update the parameters during the following Rewards Learning section.

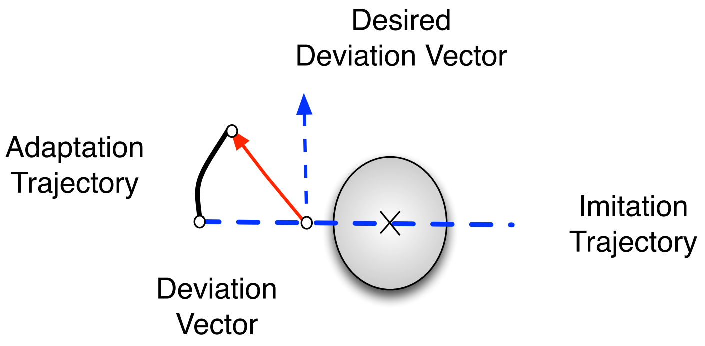


Figure 3.3: Illustration of deviation vector feature: vector from original imitation trajectory to an adapted one.

3.4.3 Rewards Learning

In this section, we describe how our system learns the reward function. Let us assume there is an oracle reward function $f^*(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$ that reflects exactly how much reward the adapted trajectory $\boldsymbol{y}$ can gain for each context. The goal of this module is to estimate such a reward function $f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D; \boldsymbol{w})$, where $\boldsymbol{w}$ are the parameters to be learned, that approximate the oracle reward $f^*(\cdot)$ tightly.

By rewriting Eq.(3.6) and Eq.(3.13) for the entire trajectory, we can have the reward function in a linear form represented by features and weights:

$$f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D; \boldsymbol{w}) = \boldsymbol{w}_D^T \boldsymbol{\phi}_D + \boldsymbol{w}_C^T \boldsymbol{\phi}_C + \boldsymbol{w}_E^T \boldsymbol{\phi}_E \tag{3.21}$$

$$\boldsymbol{\phi}_D = \left[\phi_{D,1}, \ldots, \phi_{D,d}\right]^T, \phi_{D,i} = -\sum_{t=0}^{T} \left(y_i(t) - y_{D,i}(t)\right)^2 e^{-\sigma_i^2(t)} \tag{3.22}$$

$$\boldsymbol{\phi}_C = -\sum_{t=1}^{T} \|(\boldsymbol{y}(t) - \boldsymbol{y}(t-1))\|^2 \tag{3.23}$$

$$\boldsymbol{\phi}_E = -\sum_{t=0}^{T} \left[\boldsymbol{\phi}_{O,1}^T(\boldsymbol{y}(t)), \ldots, \boldsymbol{\phi}_{O,N_{obj}}^T(\boldsymbol{y}(t)), \phi_B(\boldsymbol{y}(t)), \phi_S(\boldsymbol{y}(t))\right]^T \tag{3.24}$$

where $\boldsymbol{\phi}_D, \boldsymbol{\phi}_C, \boldsymbol{\phi}_E$ represent features of the entire trajectory corresponding to Imitation, Control and Response Rewards.

Since the user only provides a feedback trajectory $\bar{\boldsymbol{y}}$ and the system can not directly observe the reward function, we apply the co-active learning technique [41] in which the robot iteratively updates the parameter $\boldsymbol{w}$ of $f(\cdot; \boldsymbol{w})$ based on user's feedback. Note that this feedback only needs to indicate $f^*(\bar{\boldsymbol{y}}, \boldsymbol{x}_c, \boldsymbol{y}_D) > f^*(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D)$ and $\bar{\boldsymbol{y}}$ could be non-optimal trajectories. Algorithm 2 gives our learning algorithm for movement adaptation.

Note that α is a learning rate, which decays along iterations, and $\boldsymbol{C}$ in the weights projection part is a bounded set to ensure that the updated parameters $\boldsymbol{w}$ are in a feasible space. After iterations of improvements, the robot can learn an estimated reward function $f(\cdot; \boldsymbol{w}^*)$ that approximates the oracle reward function $f^*(\cdot)$ as proven in [43]. By maximizing the estimated reward function $f(\boldsymbol{y}, \boldsymbol{x}_c, \boldsymbol{y}_D; \boldsymbol{w}^*)$, the robot can generate an adapted trajectory $\boldsymbol{y}$ that maximizes the rewards facing situation $\boldsymbol{x}_c$ based on imitation trajectory $\boldsymbol{y}_D$.

Algorithm 2 Rewards Learning for Movement Adaptation

```
Initialize $\boldsymbol{w}^{(0)} = [\boldsymbol{w}_D^{(0)T}, \boldsymbol{w}_C^{(0)T}, \boldsymbol{w}_E^{(0)T}]^T$
for Iteration $i = 0$ to $T_l$ do
    Task Context and Environment Perception: $\boldsymbol{x}_c^{(i)}$
    Movement Imitation:
        $\boldsymbol{y}_D^{(i)}, \boldsymbol{\Sigma}_D^{(i)} \leftarrow p(\boldsymbol{\mathcal{T}}|\boldsymbol{x}_c^{(i)})$
    Movement Adaptation:
        $\boldsymbol{\pi}^{*(i)} = \arg\max_{\boldsymbol{\pi}} f(\boldsymbol{y}, \boldsymbol{x}_c^{(i)}, \boldsymbol{y}_D^{(i)}; \boldsymbol{w}^{(i)})$
        $\boldsymbol{y}^{(i)} \leftarrow \boldsymbol{\pi}^{*(i)}$
    Movement Execution: $\boldsymbol{y}^{(i)}$
    if User Provides Feedback: $\bar{\boldsymbol{y}}^{(i)}$ then
        $\alpha^{(i)} = 1/\sqrt{i}$
        $\boldsymbol{w}_D^{(i+1)} = \boldsymbol{w}_D^{(i)} + \alpha^{(i)}(\boldsymbol{\phi}_D(\bar{\boldsymbol{y}}^{(i)}, \boldsymbol{y}_D^{(i)}) - \boldsymbol{\phi}_D(\boldsymbol{y}^{(i)}, \boldsymbol{y}_D^{(i)}))$
        $\boldsymbol{w}_C^{(i+1)} = \boldsymbol{w}_C^{(i)} + \alpha^{(i)}(\boldsymbol{\phi}_C(\bar{\boldsymbol{y}}^{(i)}) - \boldsymbol{\phi}_C(\boldsymbol{y}^{(i)}))$
        $\boldsymbol{w}_E^{(i+1)} = \boldsymbol{w}_E^{(i)} + \alpha^{(i)}(\boldsymbol{\phi}_E(\bar{\boldsymbol{y}}^{(i)}, \boldsymbol{x}_c^{(i)}) - \boldsymbol{\phi}_E(\boldsymbol{y}^{(i)}, \boldsymbol{x}_c^{(i)}))$
        Weights Projection:
            $\bar{\boldsymbol{w}}^{(i+1)} = [\boldsymbol{w}_D^{(i+1)T}, \boldsymbol{w}_C^{(i+1)T}, \boldsymbol{w}_E^{(i+1)T}]^T$
            $\boldsymbol{w}^{(i+1)} = \arg\min_{\boldsymbol{w} \in \boldsymbol{C}} \|\boldsymbol{w} - \bar{\boldsymbol{w}}^{(i+1)}\|^2$
    else   $\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)}$
    end if
end for
```

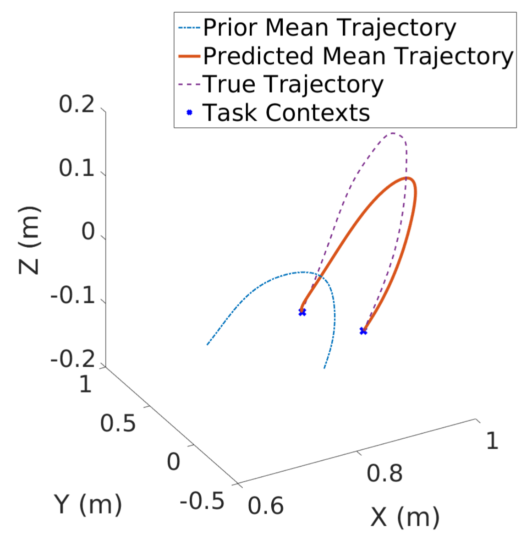
3.5 Experiments

To validate the system described above, we design and conduct the following experiments on a Baxter humanoid platform. The Baxter robot is asked to do manipulation tasks such as cleaning on a table top, with the surface as $\boldsymbol{S} = (0, 0, -0.1)$, the leftmost location as $\boldsymbol{B}_1 = (0, 0.8, 0)$ and the rightmost location as $\boldsymbol{B}_2 = (0, -0.8, 0)$ in robot spatial space described in meters. It needs to learn transferring the manipulated object between different locations while avoiding obstacles in desired manners.

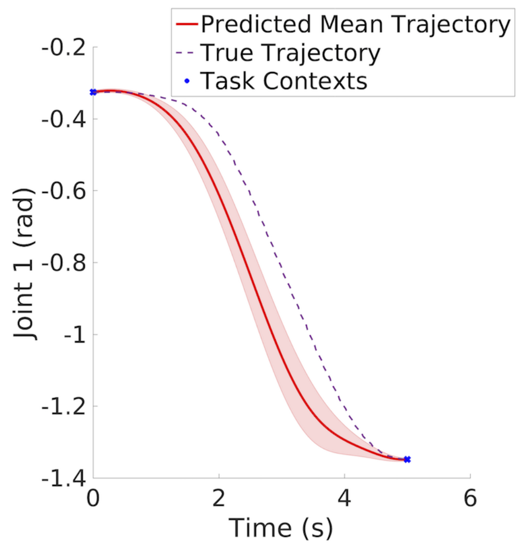
During an off-line learning phase, the robot learns the movement skill from multiple kinethestic demonstrations with no obstacles on the table. During the online learning stage, a variety of obstacles are located randomly on the table, and we assume the robot can obtain their locations from perception modules. The system learns iteratively to adapt the movement skill in novel situations such as *with different manners avoiding* obstacles, and at the same time it *follows the similar movement pattern* from off-line demonstrations.

3.5.1 Movement Imitation

In the first stage of the experiments, we have our robot learn off-line the movement skill from demonstrations. All trajectories are sampled discretely and normalized to $T = 200$ steps for transferring movement in joint space, and the left arm of the Baxter has $d = 7$ degrees of freedom. The training trajectories are encoded by ProMPs with $n = 10$ Gaussian basis functions so that the movement skill can be generalized to different initial and target states.



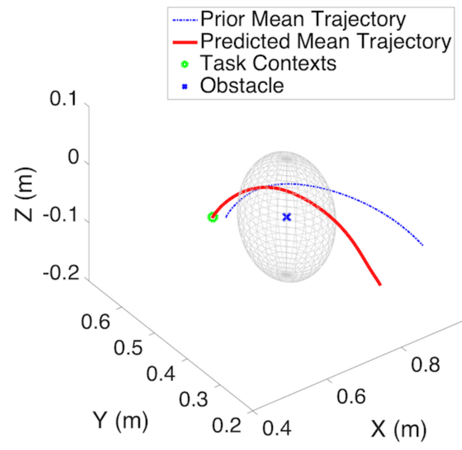
(a)



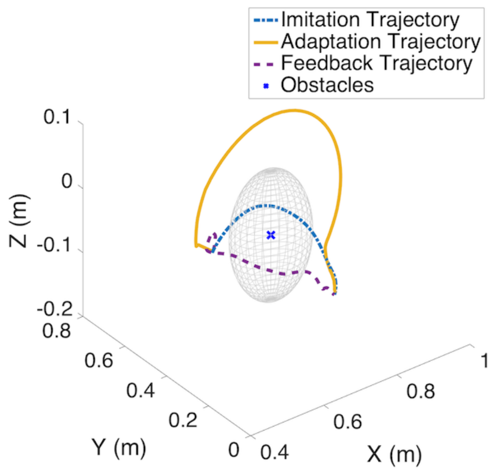
(b)

Figure 3.4: Movement Imitation with ProMPs for Transferring Task: (a) Imitation trajectory predicted based on prior movement and task contexts in spatial space; (b) Imitation trajectory for joint s_0 in joint space, shaded area indicating the predicted variance.

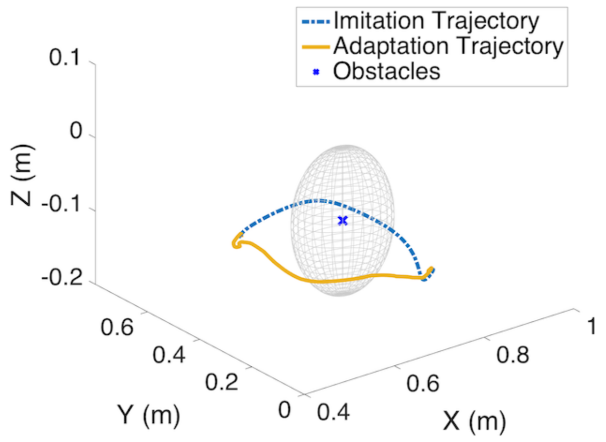
Fig. 3.4(a) shows an example of our generated imitation trajectory in spatial space for new task contexts using ProMPs. Fig. 3.4(b) shows the corresponding imitation trajectory of joint s_0 in joint space. The blue crosses here are desired new initial and target states, and the shaded area is the estimated variance for the imitation trajectory, which reflects the variations of demonstrations. True trajectory here means a trajectory recorded from user demonstration in the testing scenario for comparison. It is not hard to see that the predicted mean of the imitation trajectory well generalizes to new initial and target states and follows the same movement pattern as the prior mean trajectory learned from demonstrations. Therefore, the robot can perform the task well by following this imitation trajectory if there are no obstacles or other safety constraints.



(a)



(b)



(c)

Figure 3.5: Learning to Adapt Movement for Transferring a Leaking Bottle: (a) Movement Imitation failed to avoid the obstacle; (b) Movement adaptation with initial weights successfully avoided the obstacle by a path above it but has a potential danger of spilling water, feedback trajectory is provided afterwards; (c) Movement adaptation for a different situation with new task contexts and obstacle locations, with updated weights after learning from feedback trajectory, successfully avoids the obstacle through a path around. Corresponding execution on the Baxter platform is given by Fig. 3.2.

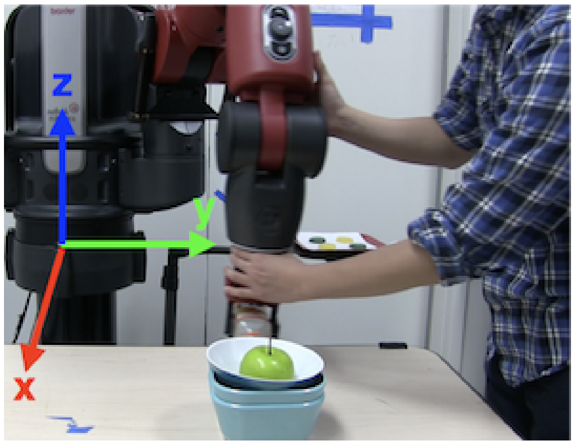
3.5.2 Learning Adaptation

We consider the situation, where the robot, while facing the task of transferring a leaking bottle, finds a bowl filled with food as obstacle on the table. We assume that the bowl's center location $\boldsymbol{O}_1$ and minimum safety distance d_1 are obtained through perception.

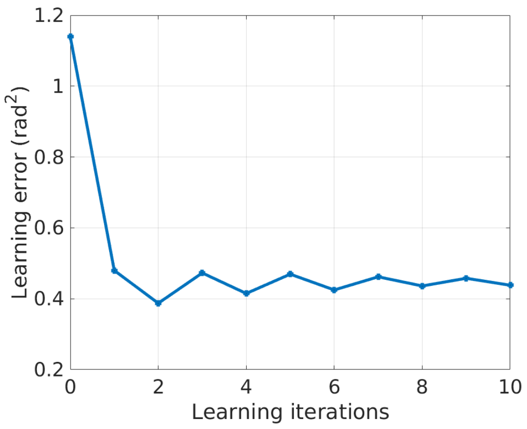
For movement adaptation, we set the prediction horizon $T_p = 11$ in the model predictive control and select system matrices $\boldsymbol{A} = 0.9 \cdot \boldsymbol{I}, \boldsymbol{B} = \boldsymbol{C} = \boldsymbol{I}$ to make the system stable in the prediction window as suggested in [39]. The limits of joints could be found from the Baxter hardware specification. The minimum safety distance to the table boarder is set as $d_{min} = 0.1$. And the weights for reward function are initialized to $\boldsymbol{w}_D = 30 \cdot \mathbf{1}, \boldsymbol{w}_C = 10, \boldsymbol{w}_E = \mathbf{0}$. We apply the native Matlab Gradient-based optimization method *fmincon* to solve the optimization at each time step.

Fig. 3.5(a) shows the output from movement imitation for transferring the leaking bottle, which failed to avoid the obstacle even though the trajectory generalizes to a novel initial and target states. Fig. 3.5(b) shows the movement adaptation with initial weights. There is no preference specified in the reward function about how to avoid obstacles or take safety considerations about boarders. Therefore, even though the adapted trajectory could avoid the obstacle successfully, it may be not an ideal trajectory.

To learn the user preference, we then provide feedback via kinethestic demonstration illustrated in Fig. 3.6(a) and the feedback trajectory is shown in Fig. 3.5(b)



(a)



(b)

Figure 3.6: Rewards Learning from User Feedback for Transferring Leaking Bottle: (a) User feedback via kinethestic demonstration; (b) Learning curve for adaptation under the same feedback.

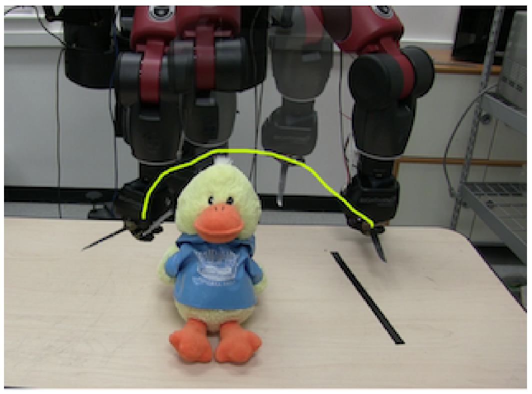
as dashed line to indicate user preferences. Following Algo. 2, the robot iteratively updates the rewards weights based on the user feedback. Weights are limited via projection in the feasible set $\boldsymbol{C}$ where $\boldsymbol{w}_D \in [1, 100]^7, \boldsymbol{w}_C \in [1, 100], \boldsymbol{w}_E \in [0, 100]$ except that the last two parameters in $\boldsymbol{w}_{O,k}$ indicating preferred deviation direction could be $[-100, 100]$. To quantitatively validate the performance of our method in movement adaptation, we consider the metric of cumulative error between the adapted trajectory and the feedback trajectory $e(i) = \frac{1}{T} \sum_{t=0}^{T} \left(\bar{\boldsymbol{y}}^{(i)}(t) - \boldsymbol{y}^{(i)}(t) \right)^2$ as the learning error at iteration i. Since the metric is affected by different situations such as obstacles' locations, we consider the feedback trajectory as fixed and let the robot iteratively learn several times to see how it performs, and we record the "learning curve" under the same feedback. From Fig. 3.6(b), we can see that the error decreases and converges after several iterations, and it only requires a few iterations to achieve an adapted trajectory as desired preference according to the feedback.

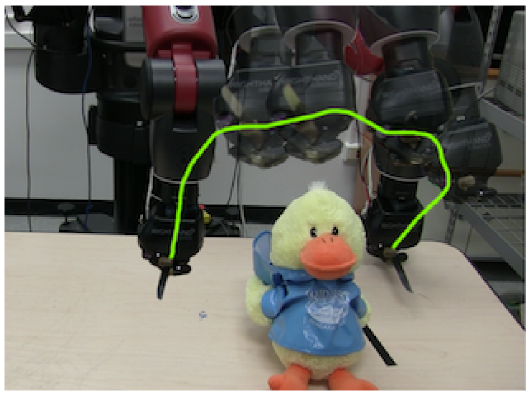
After learning, the robot uses the updated weights for movement adaptation in a different situation with novel initial/target states and obstacles' locations. Fig. 3.5(c) shows the adapted trajectory based on the updated weights after one iteration, where it successfully avoids the obstacle via the desired direction.

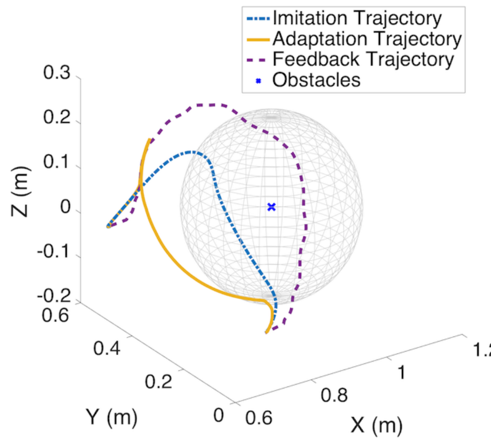
In a second scenario where a robot is transferring a knife around some fragile obstacle, the user may prefer the robot to avoid the obstacle above it instead of around it. With the same methods here, we could also generate adapted trajectories as shown in Fig. 3.7(a) and Fig. 3.7(c) for initial weights. With the user provided feedback trajectory, the robot successfully learns the user specified preferences for movement adaptation and generates the improved adapted trajectories for different situations as shown in Fig. 3.7(b) and Fig. 3.7(d).

3.6 Summary

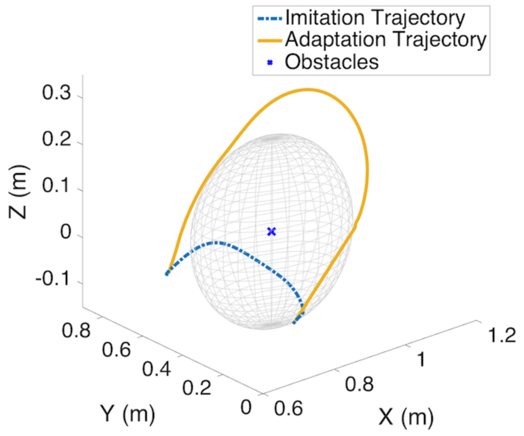
We presented a framework for learning to adapt robot end effector movement for manipulation tasks. The proposed method generalizes offline learned movement


(a)


(b)



(c)



(d)

Figure 3.7: Baxter Learning to Adapt Movement for Transferring Knife: (a) (c) Movement adaptation with initial weights using a path around the duck doll successfully avoided it but risked scratches; afterwards feedback trajectory is provided for adaptation preferences; (b) (d) Movement adaptation for different situations, with updated weights after learning from feedback trajectory, successfully avoided the duck doll using a path above it as desired.

skills to novel situations considering obstacle avoidance and other task-dependent constraints. It adapts the imitation trajectory generated from demonstrations, while maintaining the learned movement pattern and considering variations in the geometry. Here we considered as variations, avoiding obstacles with movements in desired directions, and keeping certain distances for a safety margin within a workspace.

The methods also provides a way to learn how to adapt the movement in on-line interactions with user's feedback.

Another interesting way to incorporate environmental constraints would be to consider visual information of objects and the environment as an indication of the preferences for movement adaptation. For instance, the deviation direction for avoiding a knife could be directly inferred from the location and orientation of its blade from visual input. A possible extension is to further investigate the possibility of directly learning the preferences to adapt movement from visual perception for the task context.

Chapter 4: Dynamic Motion Planning for Sequential Tasks with Subgoals Learning

In the first two problems, we studied how to learn the movement of endeffectors from human demonstrations and how to learn online adaptation to new environmental constraints according to user's preference. However, in order to adapt robots to various complex manipulation tasks, which is a sequence composed of pre-learnt primitive actions and their subgoals, we need to investigate a planning and learning framework that allows the robot to select good subgoals for sequences of primitive actions, and then plan and execute the task. At the same time, the robot should be able to refine its actions by planning/execution and re-planning/re-execution during practice.

4.1 Motivation

In robotics, learning complex tasks is very challenging due to the inherent high-dimensional continuous state and action spaces. A promising idea to tacle this challenge is to use elemental behaviours as action primitives to compose more complex behaivour. Many robot tasks can be decomposed into elemental movements in temporal domain according to their common structure while each primitive could be specific to the task. For example, opening a door and opening a bottle both can be decomposed into reaching and turning movements, while opening a microwave needs a pulling movement after reaching. Therefore, we could naturally consider a sequential task as a sequence of action primitives and their subgoals. Each action primitive has its own subgoal parameters for successful execution of the whole task such as where to reach and how much angle to pull open. Different task specifications give the robot different sequences of action primitives. To let the robot execute those versatile action sequences in the new environment, which differs from the one in the demonstration phase, we could have a planning algorithm generate motion trajectories for each primitive action that considers the task model, i.e, the action sequence to plan, the environment constraints, i.e, geometry of obstacles and objects from perception, and the embodiment constraints, i.e, kinematics of the robotic arms. At the same time, due to the changing nature of the environment, such as different locations and even different geometry of target objects, the subgoals of the action primitives need to be changed to satisfy the different constraints. Therefore, we propose a dynamic motion planning framework for such sequential tasks, which learns subgoals distribution of action primitives through re-planning and re-execution and adapts learned subgoals in the motion planning.

Take the opening task as a case study here. A daily robot needs to open the door handles, microwaves, bottles, drawers, boxes and all the different objects in order to achieve the manipulation task. On one hand, this problem requires robots to be adaptive enough to generate motion sequences for different task logic models due to different objects, such as turning opening for bottle caps and pull opening for microwaves. On the other hand, among the sequential task specifications, there is a sequence of action primitives with different subgoals, which needs to be adapted during the execution. Therefore, the robot needs to refine autonomously the sequence of action primitives to achieve the task goal.

In this work we propose an approach of dynamical motion planning for sequential manipulation tasks with subgoals learning. Fig. 4.1 gives an overview of our framework. The main contributions of this work are: 1) A system to generate a robot's movements according to sequential task specifcations with adapted subgoals for each primitive action in the task, fulfilling perception constraints in a new environment and constraints from the embodiment of the robot itself; 2) An approach for robot learning to improve the sugoals of each primitive action by updating the parameters of its distribution through re-planning and re-execution trials; 3) An implementation of the planning schema for the task of "openning a microwave" involving a sequence of primitive actions and subgoals. We validate the implementation on a humanoid platform (Baxter), and the experimental results support our claims.

4.2 Related Work

There are several studies on robot learning of complex behaviours. Bentivenga [44] proposed a framework to decompose complex behaviour into primitives and learn them from observations. To deal with the variety of situations and environments that robots may face, one approach is that we could provide users

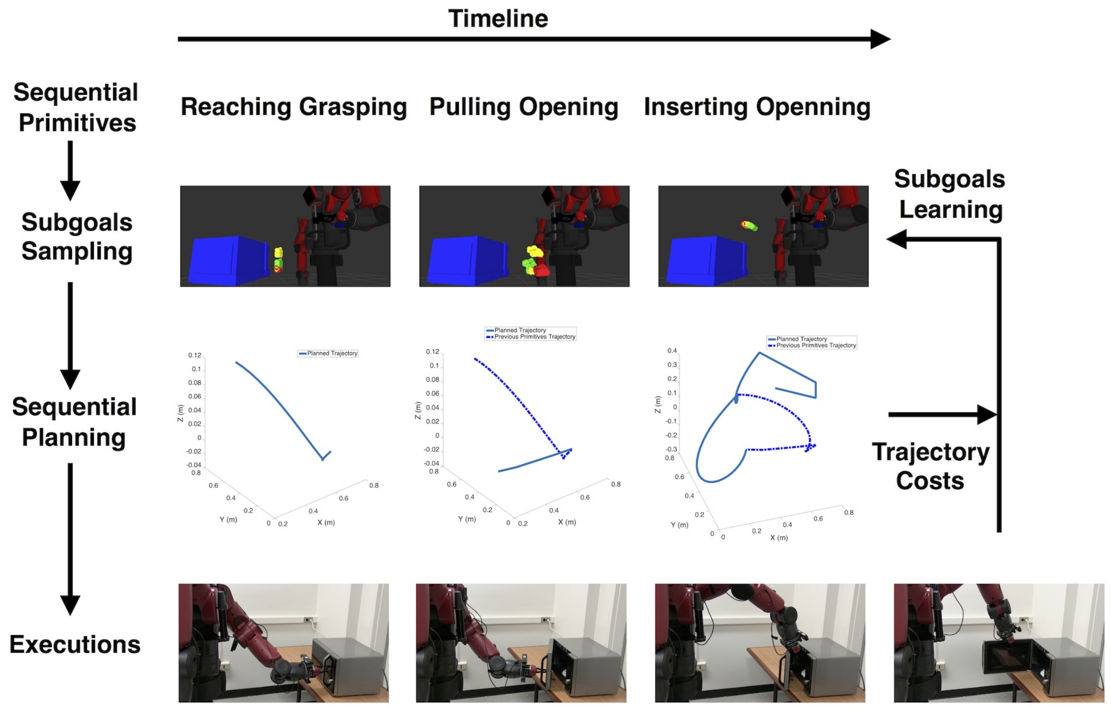


Figure 4.1: Overview of dynamical motion planning framework for sequential tasks with subgoals learning. Opening microwave is an example task shown here.

with simple methods for programming robots that do not require the skill of an expert. For this reason, learning from demonstration (LfD) has become a popular alternative to traditional robot programming methods, aiming to provide a natural mechanism for quickly teaching robots. There are lots of approaches that apply LfD at the symbolic level for task learning. One common way is to segment and encode the task according to sequences of predefined actions, described as symbols, then regenerate the sequences of these actions through planning techniques. [45] is one of the first papers to learn task-execution plans from demonstrations. The robot learns a hierarchical task plan for an object manipulation task by observing the movements of the teacher's hand through visual sensor, where the objects and primitive actions are already known to the robots.

Nicolescu et al. [46] presented a graph-based representation of a task, where each node of the graph represents a different action primitive described in terms of preconditions and postconditions. The robot is capable of generalizing from multiple demonstrations by task segmentation and supervised execution phases. Ekvall et al. [47] proposed a task-level planning approach to learn an abstract task goal from demonstrations and use a symbolic planner in run time to choose best action policy according to environment state. They learn spatio-temporal constraints on object manipulations and relations between objects to generailize states and therefore task goal. In [48] robot learns spatial relations among objects as concepts in conceptual spaces and then learns high level task goal described in first order predicate logic, which is used in symbolic planner to reproduce the goal.

Akihiko et al. [49] explored the pouring skills by modeling robot behaivour as primitive action selection and behavioral parameters selection through planning and learning methods where the entire pouring process is modelled as hierarchical finite state machine where the primitve actions are manually defined. Robot is able to learn how to select appropriate primitive actions and parameters as well to adapt to variations of pouring task from practice feedback.

Reinforcement learning techniques have been applied to a range of robotic problems in both high-level control [50] and low-level motor control [51]. Broadly, reinforcement learning can be categorized as model-free methods, which directly learn a control policy from data measurements interacting with environment [52], and model-based methods [53], which optimizes the policy under some model of the system dynamics, where the model could be given or learned. Model-based methods can be substaintially more sample efficient and typically achieve the fastest learning times, but they require a model representation that can be used to learn an accurate estimate of the true dynamics, which closely depends on given task. da Silva et al. [54] learned a mapping from task parameters to skill parameters through low-dimensional subspace extraction and used that for parameters initialization followed by policy improvement. It considerably improves the generalization of the skills but leads to sample-inefficient policy updates since they work almost independently except initialization of policy search. To relax this limitation, Neumann et al. [55] integrated skill generalization and improvement in reinforcement learning algorithm by proposed contextual policy search methods that learns a policy choosing the control parameters in accordance the context vector.

For learning motion primitive goals, [56] proposed an approach to optimize the goal parameters of a dynamic movement primitive(DMP) through reinforcement learning, by iteratively approximating a continuous value function in state space. The robot learns to pour liquid into a container in a 2-D parameter space, as only allowing two of the six degrees of freedom (DOFs) at the end-effector pose. [57] further determined the moving goal of a movement such that a prespecified velocity vector is achieved when coming into contact with a table tennis ball. [58] extends the policy improvement with path integrals algorithm to simultaneously optimize shape and goal parameters. It also applies the algorithm to learn subgoals of sequence of primitive actions, where each primitive action is represented as DMP. In our work, we propose to learn subgoal distributions for sequence of primitive actions for motion planning.

On the other hand, sampling-based methods have been highly successful for computing feasible (and optimal) motion plans for a wide variety of robots, including manipulators with many degrees of freedom. Most of motion planners are aiming to minimize metrics such as Euclidean distance in the workspace or configuration space, but more and more methods have investigated incorporating more general task-based cost functions. [59] extended rapidly exploring random trees (RRTs) [60], a common used sampling-based method for computing feasible and obstacle-avoiding trajectories. The extended RRT algorithm samples only inside a user-specified number of standard deviations of a mean demonstrated trajectory. [61] proposed asymptotically optimal motion planning algorithms such as probabilistic roadmaps star(PRM*) that guarantee asymptotic optimality. [62] integrated a learned hidden Markov Model(HMM) representing a task with a sampling-based motion planner, PRM*, to achieve optimal motion plans for the given task model. During task execution, their motion planner quickly searches in the Cartesian product of the task model and a probabilistic roadmap for a plan with features most similar to the demonstrations given the locations of the task-relevant objects. Our work is to connect a sequence of probabilistic roadmaps, each one corresponds to a primitive action, by using learned distribution of subgoals and therefore bias the planner to select subgoals with higher successful rate of executing given task.

4.3 Formulation

A sequential task specification can be naturally decomposed into distinct phases and subgoals. Within the action primitive paradigm, each phase corresponds to an action primitive, and the subgoals correspond to the goal parameters of the primitives. Therefore, we represent the task as sequentially organized action primitives, $\{p_1, \cdots, p_N\}$, each primitive p_i has subgoal parameter g_i. As the following primitive depends on the subgoal of the preceding one, the goal parameters have to be adapted carefully to the states of robots and environments so that the whole sequence could be planned and executed successfully. We consider that the states of robot is $q \in C$, where $C \subseteq \mathbb{R}^d$ is the d-dimensional feasible configuration space of the robot. The robot also has a perception module to sense environment states s_e, which consists of L task-relevant objects. Therefore, we represent the system state by combining the robot states and the environment states $\boldsymbol{s} = [q, s_e]$. In order to successfully generate complete motion for this sequential task, our framework has three steps: 1) Subgoals Reinforcement Learning, in which we learn the subgoal parameters $\boldsymbol{g} = \{g_i\}$ for given initial system states $\boldsymbol{s}$ by iterative improvements, thereby for a set of different initial states $\{\boldsymbol{s}^j\}$, we could learn a corresponding set of goal parameters $\{\boldsymbol{g}^j\}$; 2) Subgoals Supervised Learning, in which we learn the subgoal policy $\boldsymbol{g}(\boldsymbol{s})$ using training samples generated from reinforcement learning results for different initial states. Therefore, we could generailize the subgoal policy to different situations. 3) Sequential Motion Planning, in which we use the learned subgoal policy to construct a spatiotemporal graph, where each primitive action is a subgraph, to find a shortest path (minimized costs) for the whole sequential task. The block diagram of the system is presented in the Fig. 4.2.

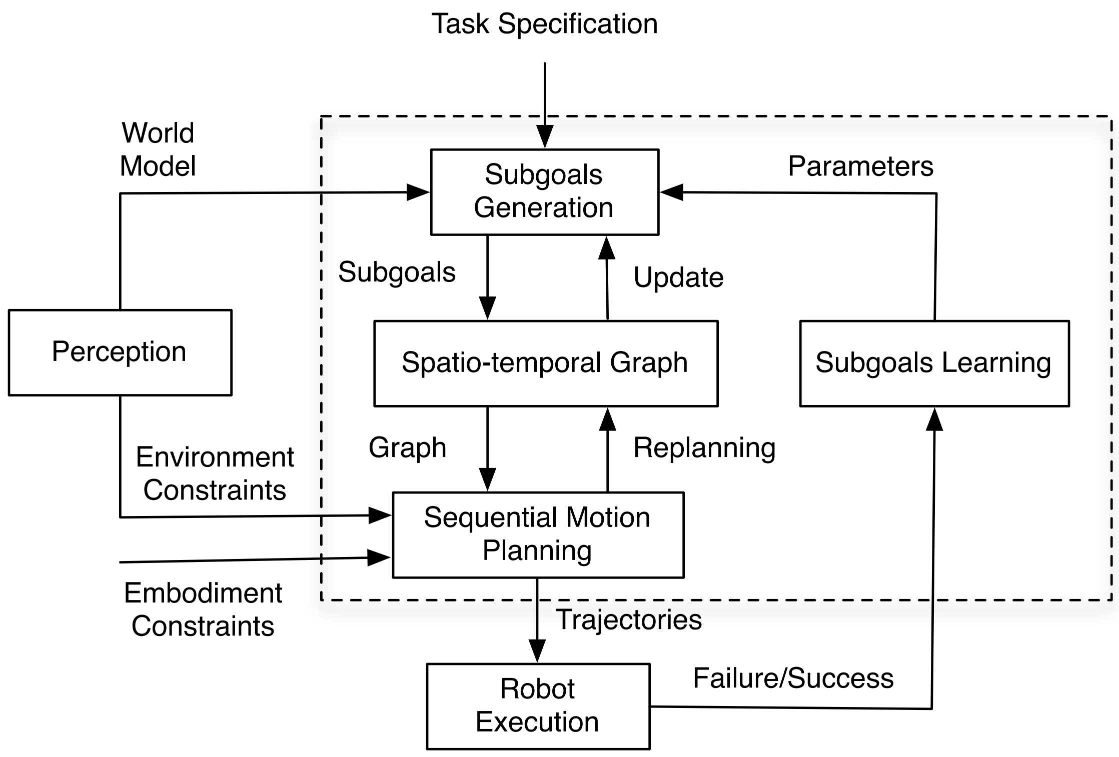


Figure 4.2: Block diagram of dynamic motion planning for sequential task with subgoals learning

4.3.1 Primitive Motion Planning

For each action primitive, we generate the trajectory through general motion planning framework. Each primitive motion planning problem is defined as a tuple $< C, E, h, F, e_0, e_t >$, where:

- C is the space of possible configurations of a robot (the joint space for robotic manipulator),
- E is the space of possible poses (Cartesian coordinates and orientation) of the

robot's gripper,

- $h : C \rightarrow \{0, 1\}$ is a bollean function that determines whether or not a configuration of the robot is in collision based on geometric shape of environment which is introduced by perception,
- e_0, e_t are the initial and target poses of the robot's gripper in its workspace E,
- $F : C \rightarrow E$ is forward kinematic defined by the geometry shape of the robotic arm and maps from a joint configuration to a pose of the robot end-effector.

Solving motion planning in mathematical form is to find a feasible path in C space defined by $\gamma : [0, T] \rightarrow C$ which satisfies following constraints:

$$\gamma_0 = F^{-1}(e_0) \tag{4.1}$$

$$\gamma_T = F^{-1}(e_t) \tag{4.2}$$

$$||\gamma_i - \gamma_{i-1}|| \leq \delta, \forall i \in [1, T] \tag{4.3}$$

$$\gamma_i \in C, \forall i \in [0, T] \tag{4.4}$$

$$h(\gamma_i) = 0, \forall i \in [0, T] \tag{4.5}$$

As a further step, the optimal motion planner could not only find a feasible path but also the optimal on (minimized costs). Many motion planning approaches could be used to solve this problem, such as probablistic roadmap star (PRM*) [61]. A roadmap is a graph in which vertices represent the states of the robot and edges represent feasible local plans between these states. In the simplest case these local plans are just straight line trajectories in configuration space. It constructs a roadmap as undirected graph and finds the shortest path as follows:

1. Randomly draw N configurations $q_0, \cdots, q_N$ from configuration space C without collision with obstacles via rejection sampling.

2. Edges are constructed between all configurations q_i and q_j for which $||q_i - q_j||_d < \epsilon$ if a feasible local plan can be found. Each edge has cost for trajectory between two vertices as user defined cost function.

3. Find the shortest path between initial and target states using Dijkstras algorithm.

However, for task with sequential action primitives, as different action primitives may have different constraints and subgoals, this motion planning algorithm is not sufficient as they could not be constructed into a single graph. Therefore, we propose to connect the primitive graphs, where each of them is a probabilistic roadmap for a single primitive action, by considering the distributions of subgoals for successful execution of the task. We will firstly present our approach for learning subgoals distribution of the given task.

4.3.2 Subgoals Reinforcement Learning

For a given task model, which is a fixed sequence of action primitives $\{p_1, \cdots, p_N\}$, and given initial system states $\boldsymbol{s}$, we assume the subgoals distribution for successful exectuion of the task is a Gaussian distribution with mean parameters to be learned and known independent standard covariance. We firstly initialize mean parameters for subgoals $\boldsymbol{g} = \{g_i\}$ for corresponding primitive actions $\{p_i\}$, then we could sequentially generate motion segment trajectories $\{\mathcal{T}_i\}$ for the action primitives using primitive motion planning and then concatenate segments as full motion trajectory $\boldsymbol{\mathcal{T}} = \{\mathcal{T}_i\}$ for the sequential task.

The aim of reinforcement learning here is to tune the mean parameters for subgoals $\boldsymbol{g}$, i.e goal parameters for each primitive in action sequence, such that they minimize a cost function regarding to the generated trajectory $\boldsymbol{\mathcal{T}}$ from motion planning. In this work, we consider the generic trajectory cost function for each segment of trajectory $\mathcal{T}_i$ as following:

$$J(\mathcal{T}_i) = \phi_{T_i} + \textstyle\sum_{t=0}^{T_i-1} r_t \tag{4.6}$$

$$\phi_{T_i} = w_T \cdot (1 - success) \tag{4.7}$$

$$r_t = \begin{cases} w_c \cdot \|y_{t+1} - y_t\|^2 & \text{if } \ success \\ 0 & \text{if not } \ success \end{cases} \tag{4.8}$$

where J is the finite horzion cost over the trajectory $\mathcal{T}_i$, which is discretized to T_i steps as $\{y_0, \cdots, y_{T_i}\}$. This cost consists of a terminal cost ϕ_{T_i}, which is a binary term penalizing the failure cases for the task, and an immediate cost r_t, which prefers a shorter path over long one if the task is successfully executed. w_T and w_c are hyper weight parameters for adjusting terminal cost and immediate cost.

However, since the subgoal parameter g_i is also determining the start state of next action primitive p_{i+1} in the sequence, and may thus have influence on the cost of planning and executing the subsequent primitives. Therefore, we optimize goal parameters g_i with respect to the cost of the current action primitive, as well as the costs of the rest of the action primitves in the sequence.

We formalize this by denoting a sequence trajectory $\mathbf{\Gamma}_i = [\mathcal{T}_i, \cdots, \mathcal{T}_N]$ which is the trajectories starting from action primitive p_i till the end of the task p_N, and we can compute the sequential costs for it, i.e, the total cost of the current trajectory and all subsequent trajectories in the sequence.

$$J(\mathbf{\Gamma}_i) = \sum_{d=i}^{N} J(\mathcal{T}_d) \tag{4.9}$$

Exploration of this reinforcement learning is done by planning and executing the sequential trajectory K times, each time with slightly different subgoal parameters $\{\boldsymbol{g}\}_k$ sampled from its distribution $\boldsymbol{N}(\boldsymbol{\mu}_g, \mathbf{\Sigma}_g)$. Each different parameters may generate (or fail to plan/execute) different trajectories $\{\boldsymbol{\mathcal{T}}\}_k$, which lead to different costs. We refer to the planning and execution of such an action sequence as a "episode", and the set of K exploration trials as an "iteration". Given the costs and sampled subgoal parameters of the K trials of episodes, policy improvement methods then update the mean parameter vector $\boldsymbol{\mu}_g$ such that it is expected to generate movements that lead to lower costs.

In each iteration, the set of exploration parameters $\{\boldsymbol{g}\}_{k=1,\cdots,K}$ is sampled at the beginning of the iteration. Then we can use the sequential cost of the trajectory $J(\{\mathbf{\Gamma}_i\}_k)$ to compute corresponding probabilty $P(\{\mathbf{\Gamma}_i\}_k)$. The motivation behind this is that as the effect of $\boldsymbol{g}$ remains constant during execution, there is no temporal dependence of $\boldsymbol{g}$ on the cost. Thus, probability-weighted averaging could be used to update the mean parameter μ_{g_i} for each primitive in the action sequence as following:

$$P(\{\boldsymbol{\Gamma}_i\}_k) = \frac{e^{-\frac{1}{\lambda}J(\{\boldsymbol{\Gamma}_i\}_k)}}{\sum_{l=1}^{K}[e^{-\frac{1}{\lambda}J(\{\boldsymbol{\Gamma}_i\}_l)}]} \tag{4.10}$$

$$\mu_{g_i} = \sum_{k=1}^{K}[P(\{\boldsymbol{\Gamma}_i\}_k)\cdot\{g_i\}_k] \tag{4.11}$$

4.3.3 Subgoals Supervised Learning

After improvements of iterations in subgoals reinforcement learning, we could get the goal parameters $\boldsymbol{g}^j$ tuned for given initial system state s^j. In order to generalize such subgoals for different situations, we use supervised learning technique as an outer loop learning method here.

Given a set of different initial system states $\{\boldsymbol{s}^j\}$, for each sample situaion $\boldsymbol{s}^j$, we could use above reinforcement learning techinque to tune the subgoal mean parameters $\boldsymbol{\mu}_g^j$. Then we could get a set of expected means for subgoals $\{\boldsymbol{\mu}_g^j\}$ as labels corresponding to different initial states. Thus, we assume the linear model regarding to feature vector $\boldsymbol{\phi}(\boldsymbol{s})$ with weight vector $\boldsymbol{w}$ as following:

$$\boldsymbol{\mu_g} = \boldsymbol{w}\cdot\boldsymbol{\phi}(\boldsymbol{s}) \tag{4.12}$$

The feature vector $\boldsymbol{\phi}(\boldsymbol{s})$ is task-dependent according to sequence of primitive actions. For example, in “pick-place” task, we have action sequence {reaching, grasping, transporting, placing, withdrawing}, corresponding subgoal parameters are reaching distance offset to the object, grasping angle respect to the obejct, transporting location offset to the target area, placing angle and the withdrawing distance offset. And the features are relative distance and direction from object to robotic hand, relative location of target area to the object. The cost function we are using for training is the residual sum of squares between the observed responses and the predicted approximations.

In overall, the algorithm for our subgoals learning is as following:

Algorithm 3 Subgoals Learning for Sequential Manipulation Task

```
for Training situations: $j = 0$ to $M$ do
    Initialize system state $\boldsymbol{s}^j$
    Initialize subgoal parameters mean $\boldsymbol{\mu}_g^{(0)}$ and covariance $\boldsymbol{\Sigma}_g$
    for Iteration: $l = 0$ to $L$ do
        Randomly sample $K$ subgoal vectors $\{\boldsymbol{g}\}_k^{(l)}$ from distribution $\boldsymbol{N}(\boldsymbol{\mu}_g^{(l)}, \boldsymbol{\Sigma}_g)$
        for Episode: $k = 1$ to $K$ do
            Generate trajectory $\{\boldsymbol{\mathcal{T}}\}_k^{(l)}$ through primitive motion planning
            Collect sequential costs $J(\{\boldsymbol{\Gamma}_i\}_k^{(l)})$ for each primitive $i = 0, \cdots, N$:
              $J(\{\boldsymbol{\Gamma}_i\}_k^{(l)}) = \sum_{d=i}^{N} J(\{\mathcal{T}_d\}_k^{(l)})$
        end for
        Compute probability weight for each sample trajectory:
          $P(\{\boldsymbol{\Gamma}_i\}_k^{(l)}) = \frac{e^{-\frac{1}{\lambda}J(\{\boldsymbol{\Gamma}_i\}_k^{(l)})}}{\sum_{p=1}^{K}[e^{-\frac{1}{\lambda}J(\{\boldsymbol{\Gamma}_i\}_p^{(l)})}]}$
        Update subgoal parameters mean for each primitive action in the sequence:
          $\mu_{g_i}^{(l+1)} = \sum_{k=1}^{K}[P(\{\boldsymbol{\Gamma}_i\}_k^{(l)}) \cdot \{g_i\}_k^{(l)}]$
    end for
    Collect tuned subgoal mean vectors $\boldsymbol{\mu}_{\boldsymbol{g}}^j$
end for
Compute weights $\boldsymbol{w}$ with samples $\{\boldsymbol{s}^j, \boldsymbol{\mu}_{\boldsymbol{g}}^j\}$
```

With iterations of learning, we could learn subgoals distribution $\boldsymbol{N}(\boldsymbol{\mu}_{\boldsymbol{g}}(\boldsymbol{s}), \boldsymbol{\Sigma}_g)$ which can guide the subgoals decision to highly likely successful regions.

4.3.4 Sequential Motion Planning

For sequential motion planning, our method first builds a spatiotemporal roadmap, following the idea from [62], in which the edge costs between subsequent primitive actions are set based on the learned subgoal distribution. During task execution, we efficiently update the roadmap edge costs and perform searches on the roadmap to plan and replan the sequential trajectory for changing environment.

Recall that in primitive motion planning, for each action primitive, we construct a spatial roadmap. In order to accommodate dependence on the time step in the sequential task, we have a temporal roadmap, where primitive actions in the task correspond to vertices in the graph, and edges represents the transitions in the action sequence.

Therefore, we construct a spatiotemporal roadmap, a directed graph that combines the information in the spatial and temporal roadmaps. The vertices of the spatiotemporal roadmaps are each defined by a pair composed of a vertex from the spatial roadmap and a vertex from the temporal roadmap. The set of edges are given by the vertex-wise union of edges in the spatial and temporal roadmaps Fig. 4.3.

For each subgraph of primitive action, edges in the roadmap are assigned costs based on the lengths of the local plans they represent as defined in trajectory cost function. Among the transitions of primitive actions, we are aiming to find transitions of subgoals which are most likely to successfully perform the task, so we choose edge costs based on learned subgoals distribution $\boldsymbol{N}(\boldsymbol{\mu_g}(\boldsymbol{s}), \boldsymbol{\Sigma}_g)$, which can guide the graph seach to paths with subgoals that have higher probability to

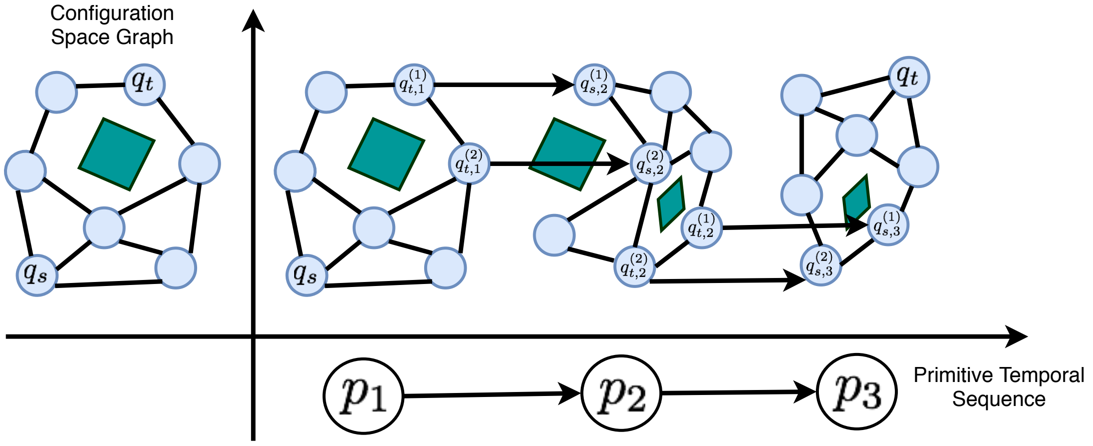


Figure 4.3: Spatiotemporal roadmap constructed from the configuration space graph and the primitve sequence of the task.

succeed in the sequential task. Specifically we define a notion of cost which, when minimized, maximizes probability of successfully executing the sequential task as learned in the subgoals distribution.

By the construction of spatiotemporal roadmap, each edge are between two primitives p_i and p_{i+1} with same configuration q. The configuration q is the target state of primitive p_i and start state of primitive p_{i+1}, thus could be determined according to the subgoal parameter g_i. Our edge cost is defined based on the negative log probability of subgoal for transition from primitive p_i to primitive p_{i+1}, which is as following:

$$-log(P(q|p_i, p_{i+1})) = -log(P(g_i|p_i, p_{i+1})) \tag{4.13}$$

$$P(g_i|p_i, p_{i+1}) \sim \boldsymbol{N}(\mu_{g_i}(s), \Sigma_{g_i}) \tag{4.14}$$

Then, we could use graph search algorithm to find shortest path that minimizes the sum of the edge costs, which means to minimize spatial paths and to maximize the joint probability of successful transitions using sampled subgoals in the task.

During execution, we update the roadmap using the latest sensed information and query roadmap in a closed-loop manner. In each replanning cycle, the spatial roadmap is firstly expanded to add more waypoints in order to be sufficient to produce high quality plan, particularly when the environment changes dramatically. Then, roadmap edge costs are re-evaluted using latest system states and thus updated probability distributions of subgoal parameters.

The overall algorithm is as following:

Algorithm 4 Planning and Replanning for Sequential Task

```
With learned weight parameters $\boldsymbol{w}$ for subgoal mean parameters
while task is not finished do
    Collect task relevant objects and environment states $\boldsymbol{s}$ with Perception
    Generate subgoals mean value: $\boldsymbol{\mu_g} \leftarrow \boldsymbol{w} \cdot \boldsymbol{\phi(s)}$
    Compute subgoals distribution $P(\boldsymbol{g}|\boldsymbol{s}) \sim \boldsymbol{N}(\boldsymbol{\mu_g}, \boldsymbol{\Sigma}_g)$
    Construct and update spatiotemporal graph for sequential task with edge costs
    Find shortest path $\boldsymbol{\mathcal{T}}$ starting from current state in the spatiotemporal graph
    Execute the trajectory until next perception updating cycle
end while
```

4.4 Experiments

To validate the system described above, we design and conduct the following experiments on a Baxter humanoid platform. The Baxter robot is asked to do manipulation tasks such as openning microwave on a table top, with different initial state such as the microwave initial location and orientation. Given task specification of openning the microwave in steps of primitive actions {*reaching grasping, pulling* *openning, inserting openning*$\}$, the robot needs to learn subgoals distribution of each primitive action for different initial states and therefore to successfully plan and execute the sequential task in new situations.

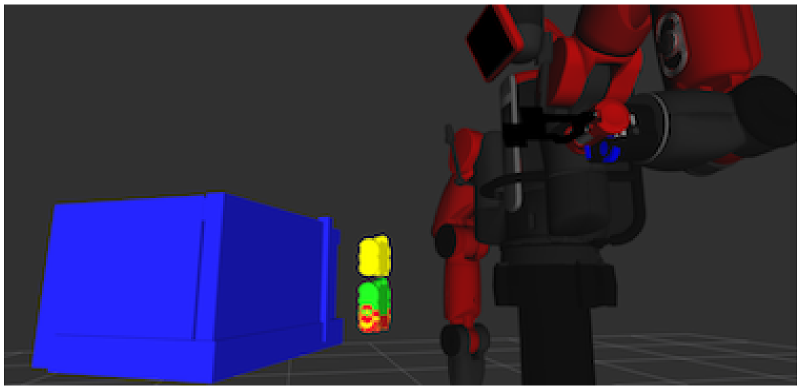
4.4.1 Subgoals Learning

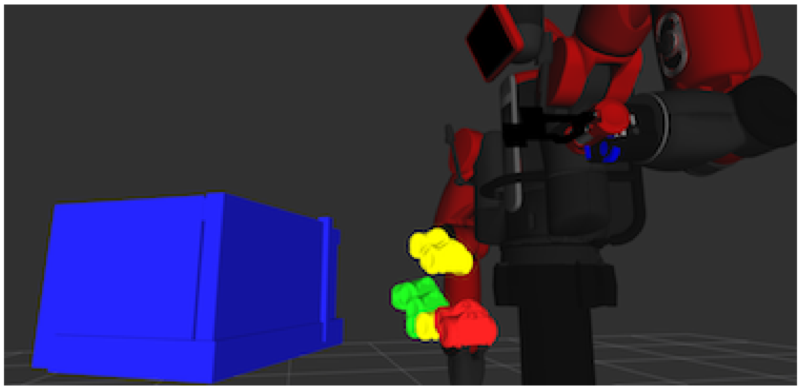
During the learning phase, the robot learns the subgoals distribution for the given sequential task. Assuming the microwave is on a flat tabletop, we set the initial location of the microwave is changing at $[(0.7, 0.1, -0.1) - (0.9, 0.3, 0.1)]$ in robot spatial space described in meters, and its rotation angles along $z - axis$ (Yaw) is varying from $[-0.5, 0.5]$ in rads facing to the robot. We set the grid step as $5cm$ for each axis and 0.25 rads for Yaw angle. Therefore, we have 625 different initial states for training.

For given sequential task specification, we have subgoals for primitive actions $\{p_1, p_2, p_3\}$ as following. p_1 is primitive action for reaching and grasping the handle of the microwave, and the subgoal for this action is defined as g_1, the z-axis offset relative to the center location of the handle. This subgoal indicates where to grasp for the handle and affects if the task could be successfully planed and executed due to different initial locations of the microwave. We could imagine that the robot should be grasping the upper part of the handle if the microwave is in a lower height position for higher successful rate. p_2 is primitive action of pulling openning with subgoal defined as g_2, the angles between the desired opened door and microwave frame to indicate how much to partially open the door. p_3 is the action of inserting gripper between partially opened microwave door and its frame, then pushing the door to fully opened. Its subgoal parameter is defined as g_3, which is the pitch angle of gripper for inserting and pushing the door.

Firstly, for each different initial state of the microwave, we initialize the mean values of subgoals as $(p_1 = -0.1, p_2 = 0.8, p_3 = 0)$ and use reinforcement learning to update the mean values iteratively. During each iteration, we used $K = 10$ samples for generating trajectories and collecting sequential costs corresponding to different sampled subgoal parameters. Then we compute probability weight for each sample and update subgoal parameters. Fig. 4.4 presents the subgoals for each primitive action during learning iterations. The color of markers indicates the costs of sequential costs. Decreasing costs are shown in color changing from red to green. Fig. 4.4.1 shows the learning result that the sequential cost of the trajectories decreases along with updating iterations under different weight parameters λ in calculating the probability weight. Here we found that with $\lambda = 1$ the algorithm can get a reasonable convergence rate in our scenario during learning iterations.

Secondly, we collect the learned subgoal parameters for each scenario and we split the data into training and testing sets by ratio of 4:1. Then we use standard linear regression to find parameters $\boldsymbol{w}$ by minimizing the residual sum of squares between the observed responses in the training dataset, and the responses predicted by the linear approximation. The features we are using in this task are locations and orientations of the microwave handle in the coordinate system relative to the robot torso.


(a)


(b)

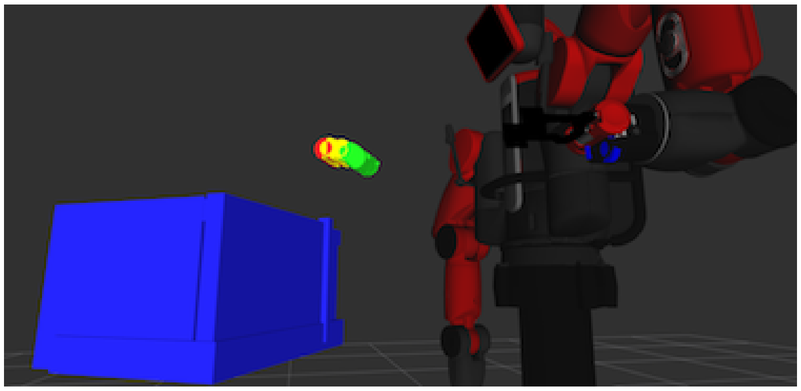

(c)

Figure 4.4: Subgoals in Learning Iterations for Opening Microwave: (a) Subgoals for primitive action reaching grasping, which indicates location offset to grasp handle; (b) Subgoals for primitive action pulling openning, which indicates pulling angle for openning door; (3) Subgoals for primitive action inserting openning, which indicates rotation angle for inserting gripper. Colors of markers are changing from red to green indicating the decrease of trajectory cost.

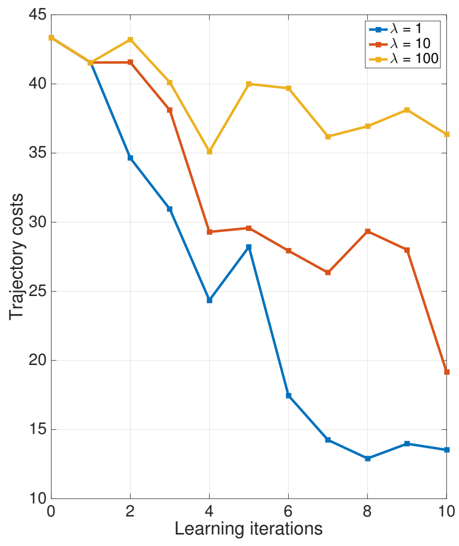
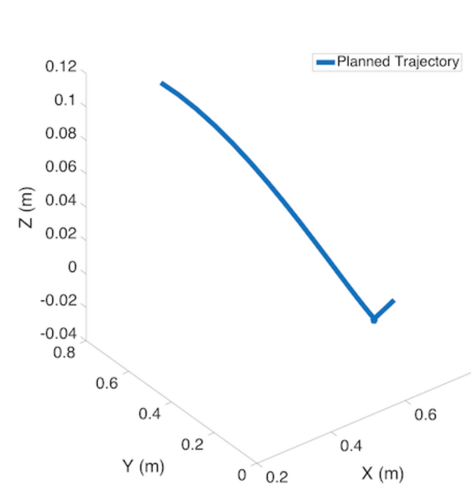


Figure 4.5: Subgoals Learning Curve for Opening Microwave with Different Weight Parameters λ.

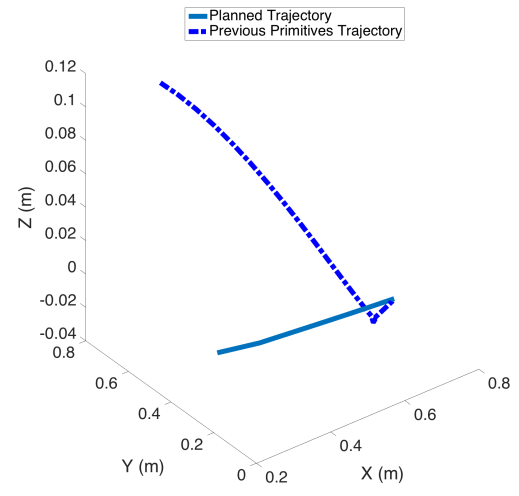
4.4.2 Sequential Motion Planning

During the testing phase, the robot is facing to a different initial state where the microwave is located in random place with random orientation. It needs to reason correct subgoals according to the newly perceived environment and successfully plan and execute the sequential task.

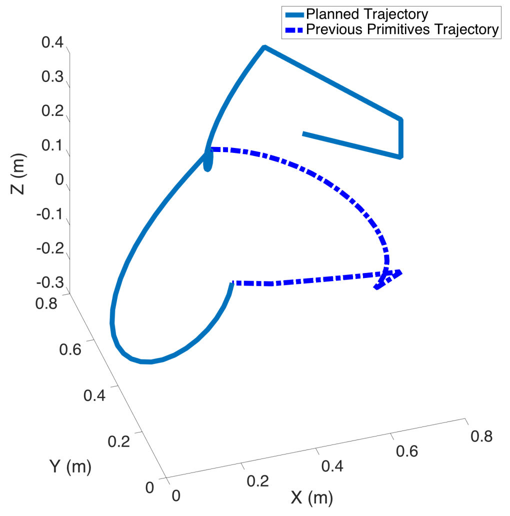
According to the learned distribution of subgoals, we sampled 10 subgoal states for each primitive action and construct the spatiotemporal graph where different temporal components are corresponding to sequential primitives in the task specification. The temporal components are connected through the subgoal states with weights of log probability of its distribution. The higher weight indicates lower successful rate for planning and executing the task. Fig. 4.6 shows the planned trajectories for each primitive action. Fig. 4.7 visualizes the results in RVIZ during



(a)

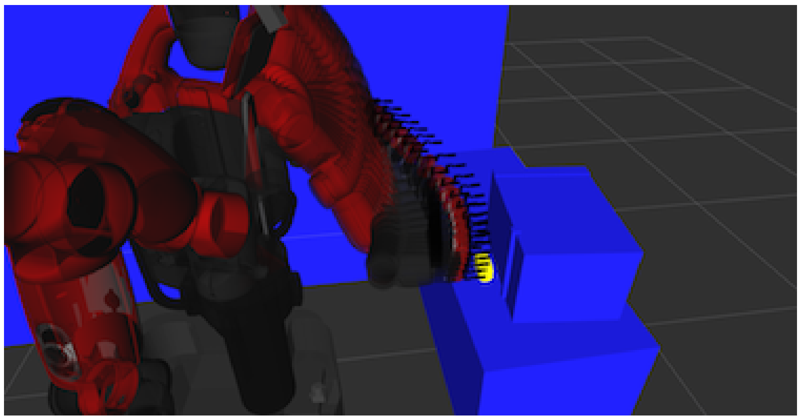


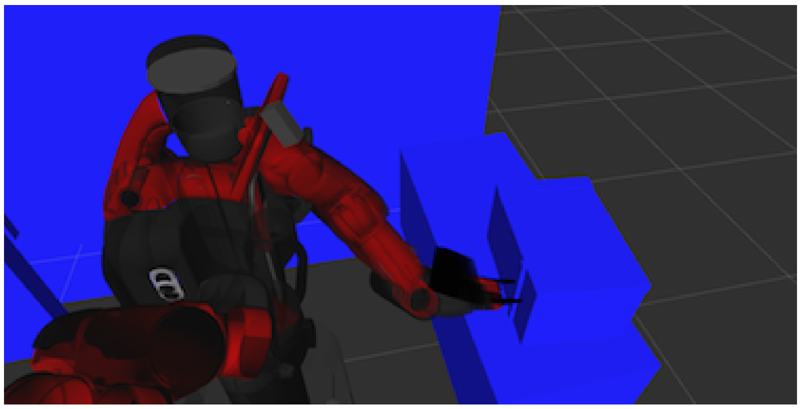
(b)



(c)

Figure 4.6: Sequential Motion Planning Results for Openning Microwave: (a) Planned trajectory for primitive action reaching grasping; (b) Planned trajectory for primitive action pulling openning; (c) Planned trajectory for primitive action inserting openning.


(a)


(b)

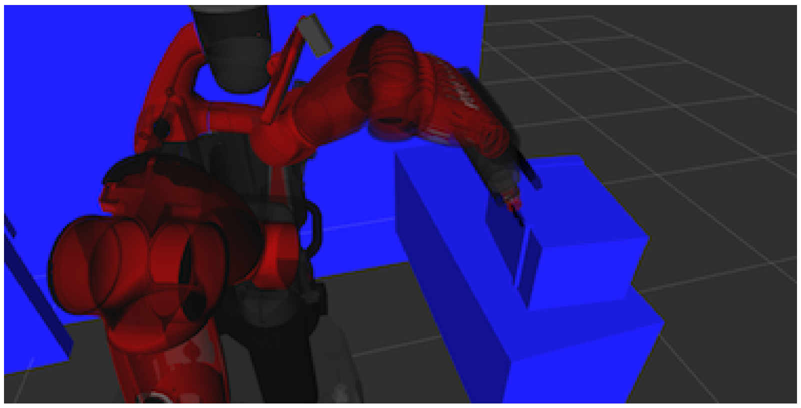
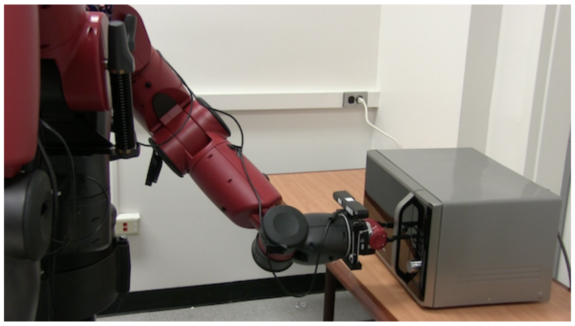
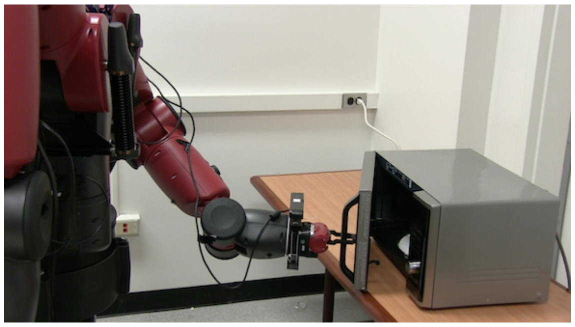

(c)

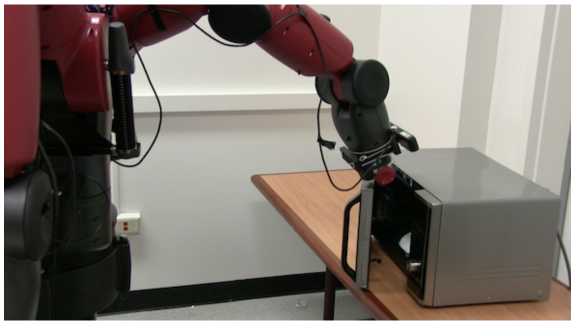
Figure 4.7: Planning Results in Simulation for Openning Microwave: (a) Planned trajectory for primitive action reaching grasping; (b) Planned trajectory for primitive action pulling openning; (c) Planned trajectory for primitive action inserting openning.



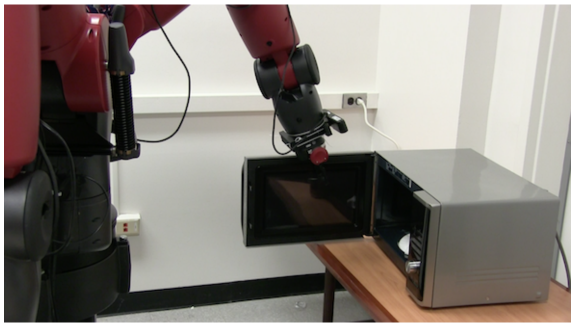
(a)



(b)



(c)



(d)

Figure 4.8: Baxter Execution for Openning Microwave: (a) Execution results after reaching grasping; (b) Execution results after pulling openning; (c) Execution results after inserting gripper. (d) Execution results after fully openning door.

simulation environment. The real execution of Baxter for the openning microwave task is shown in Fig. 4.8.

4.5 Summary

We presented a framework for dynamical motion planning of sequential manipulation tasks with learning subgoals for primitive actions. The proposed method generates robots' movements according to sequential task specifications by considering the learned subgoal distributions for each primitive action in the task, while fulfilling perception constraints in a new environment and embodiment constraints of robot itself. It gives us better successful rate in the task planning and execution as the planning algorithm could adapt subgoals according to learned experiences. The robot is learning subgoals of each primitive action by iteratively improving the parameters of its distribution trhough re-planning and re-execution trials. We illustrated the framework using the Baxter platform for the task of "openning a microwave" involving a sequence of primitive actions and subgoals.

As potential extension work, one could consider better generalizations of subgoals learning with complex features and methods to handle more complex tasks that require integration of task and motion planning such as different type of microwaves that needs to be opened in different sequence of actions. It is also interesting to investigate methods to make replanning faster, including using demonstrated trajectories or pre-plans to boost sampling in spatiotemporal graph construction and path search.

Chapter 5: Conclusions

In this dissertation, we proposed three research problems to explore the learning methods of robots in the setting of manipulation tasks. Most of daily tasks that the robots need to perform are manipulation tasks. In order to have robots perform those tasks, they will need to be capable of adapting to new scenarios as they may be facing to different objects and tasks. Hence, it is impractical to preprogram all the skills into these robots. Instead, such robots should have the capability to learn the skills for manipulating objects autonomously.

To begin with, we firstly enable robots the capability to learn basic skills so that it can perform the task in new situations. However, even simple tasks, like cutting a cucumber, may be realized in thousands of different ways. An intuitive solution is to decompose the task into smaller primitive actions so that the robot can learn the skills from observing a human. Therefore, in the first problem, we firstly study hand movement learning from human demonstrations. For practical purposes, we propose a system for learning hand actions from markerless demonstrations, which are captured using the Kinect sensor. The proposed algorithm autonomously segments an example trajectory into multiple action units, each described by a movement primitive, and constructs a task-specific model. Using proposed method, we learn a generative model of a human's hand task such as cutting from observations. Similar movements for different scenarios can be generated, and performed on Baxter Robots. The proposed method provides a potentially fully automatic way to learn hand movements for humanoid robots from demonstrations, and it does not require special hand motion capturing devices. With that, the robots could learn and transfer the demonstrations to a library of skills and therefore generate similar movements for different scenarios to perform the task.

Then we tackle the problem for robot generating motions adapting to new scenarios. When the robot is asked to perform a task in a new environment, it could select appropriate skills from the pre-learnt library given the task model. However, traditional motion generation methods do not consider versatile constraints from different users, tasks, and environments. For example, the robot may face to an environment where there are obstacles newly perceived while standard movement imitation learning only mimicks the motion and may collide with the obstacles. We propose a co-active learning framework for learning to adapt the movement of robot end-effectors. Our system could generalize robots' movements learned from demonstrations to fulfill constraints perceived in a new environment. By using model predictive control, we generalize the imitation trajectories to new environment with different constraints such as obstacle avoidance and safety margin. Meanwhile, it is able to adapt trajectories according to user preferences. We present an approach for robot learning preferences to adapt trajectories by updating reward weights based on users' feedback. The user thus can co-actively train the robot in-the-loop by demonstrating desired trajectories. We also implement the optimization schema for the skill of “transferring objects” which considers obstacles and different user preferences for the movements. The implementation is validated on a humanoid platform (Baxter). The proposed method generalizes offline learned movement skills to novel situations considering obstacle avoidance and other task-dependent constraints, and it also provides a way to learn how to adapt the movement in on-line interactions with user’s feedback. A possible extension is to further investigate the possibility of directly learning the preferences to adapt movement from visual perception for the task context. For instance, the deviation direction for avoiding a knife could be directly inferred from the location and orientation of its blade from visual input. With the proposed framework, the robots could learn and adapt the pre-learnt skill to new environments where different constraints are perceived.

In order to further adapt robots to perform more complex manipulation tasks, we at last present a system to generate robots’ movements according to task specifications by sequential motion planning and subgoals learning. A sequential task is naturally considered as a sequence of pre-learned action primitives, each action primitive has its own goal parameters corresponding to the subgoal. The proposed system learns the subgoals distribution of given task model using reinforcement learning by iteratively updating the parameters in the trials. The learned subgoals distribution is used to connect primitive actions, where each of them is represented as a subgraph for connectivity of robot states, therefore to construct spatiotemporal graph for sequential motion planning. As a result, by considering the learned subgoals distribution in sequential motion planning, the proposed framework could adaptively select better subgoals that lead higher probability for robot to execute the task successfully. We implement the planning schema for the task of “openning a microwave” involving a sequence of primitive actions and subgoals. And we validate the implementation on a humanoid platform (Baxter) to support our claims. The proposed approach allows the robot to learn and improve sugoals of each primitive action through re-planning and re-execution trials during the practice.

Bibliography

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

[2] Ren Mao, John S Baras, Yezhou Yang, and Cornelia Fermuller. Co-active learning to adapt humanoid movement for manipulation. *arXiv preprint arXiv:1609.03628*, 2016.

[3] F Stulp and S Schaal. Hierarchical reinforcement learning with learning movement primitives. In *IEEE International Conference on Humanoid Robots*, 2011.

[4] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[5] Sylvain Calinon, Florent D'halluin, Eric L Sauser, Darwin G Caldwell, and Aude G Billard. Learning and reproduction of gestures by imitation. *Robotics & Automation Magazine, IEEE*, 17(2):44–54, 2010.

[6] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3, 2011.

[7] Robert Krug and Dimitar Dimitrovz. Representing movement primitives as implicit dynamical systems learned from multiple demonstrations. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–8. IEEE, 2013.

[8] Freek Stulp, Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning to grasp under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5703–5708. IEEE, 2011.

[9] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1398–1403. IEEE, 2002.

[10] Volker Krüger, Dennis L Herzog, Sanmohan Baby, Ales Ude, and Danica Kragic. Learning actions from observations. *Robotics & Automation Magazine, IEEE*, 17(2):30–43, 2010.

[11] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE, 2009.

[12] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[13] Franziska Meier, Evangelos Theodorou, and Stefan Schaal. Movement segmentation and recognition for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 761–769, 2012.

[14] Ali Erol, George Bebis, Mircea Nicolescu, Richard D Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1):52–73, 2007.

[15] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[16] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *Robotics, IEEE Transactions on*, 26(5):800–815, 2010.

[17] Denis Forte, Andrej Gams, Jun Morimoto, and Aleš Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60(10):1327–1339, 2012.

[18] Tamim Asfour, Pedram Azad, Florian Gyarfas, and Rüdiger Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*, 5(02):183–202, 2008.

[19] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ int. conference on intelligent robots and systems (IROS2002)*, number BIOROB-CONF-2002-003, pages 958–963, 2002.

[20] Franziska Meier, Evangelos Theodorou, Freek Stulp, and Stefan Schaal. Movement segmentation using a primitive library. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3407–3412. IEEE, 2011.

[21] Mitesh Patel, Carl Henrik Ek, Nikolaos Kyriazis, Antonis Argyros, Jaime Valls Miro, and Danica Kragic. Language for learning complex human-object interactions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4997–5002. IEEE, 2013.

[22] J Randall Flanagan and Roland S Johansson. Action plans used in action observation. *Nature*, 424(6950):769–771, 2003.

[23] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Real time hand pose estimation using depth sensors. In *Consumer Depth Cameras for Computer Vision*, pages 119–137. Springer, 2013.

[24] Noam Chomsky. *Lectures on government and binding: The Pisa lectures*. Number 9. Walter de Gruyter, 1993.

[25] Craig G. Nevill-Manning and Ian H. Witten. Identifying hierarchical strcture in sequences: A linear-time algorithm. *J. Artif. Intell. Res.(JAIR)*, 7:67–82, 1997.

[26] Rüdiger Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2):109–116, 2004.

[27] Austin Myers, Angjoo Kanazawa, Cornelia Fermuller, and Yiannis Aloimonos. Affordance of object parts from geometric features. In *Workshop on Vision meets Cognition, CVPR*, 2014.

[28] Jens Kober, Betty Mohler, and Jan Peters. Learning perceptual coupling for motor primitives. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 834–839. IEEE, 2008.

[29] Andrej Gams, Auke J Ijspeert, Stefan Schaal, and Jadran Lenarčič. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous robots*, 27(1):3–23, 2009.

[30] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

[31] S Mohammad Khansari-Zadeh and Aude Billard. Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2676–2683. IEEE, 2010.

[32] Tetsunari Inamura, Iwaki Toshima, Hiroaki Tanie, and Yoshihiko Nakamura. Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377, 2004.

[33] Dae-Hyung Park, Peter Pastor, Stefan Schaal, et al. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 91–98. IEEE, 2008.

[34] Heiko Hoffmann, Peter Pastor, Dae-Hyung Park, and Stefan Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2587–2592. IEEE, 2009.

[35] Sylvain Calinon, Irene Sardellitti, and Darwin G Caldwell. Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 249–254. IEEE, 2010.

[36] Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 365–371. IEEE, 2011.

[37] Andrej Gams, Bojan Nemec, Auke J Ijspeert, and Ales Ude. Coupling movement primitives: interaction with the environment and bimanual tasks. *Robotics, IEEE Transactions on*, 30(4):816–830, 2014.

[38] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[39] Amir M. Ghalamzan E., Chris Paxton, Gregory D Hager, and Luca Bascetta. An incremental approach to learning generalizable robot tasks from human demonstration. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5616–5621. IEEE, 2015.

[40] Emrah Akin Sisbot, Luis F Marin, and Rachid Alami. Spatial reasoning for human robot interaction. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2281–2287. IEEE, 2007.

[41] Ashesh Jain, Shikhar Sharma, Thorsten Joachims, and Ashutosh Saxena. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, page 0278364915581193, 2015.

[42] Zhangfeng Ju, Chenguang Yang, and Hongbin Ma. Kinematics modeling and experimental verification of baxter robot. In *Control Conference (CCC), 2014 33rd Chinese*, pages 8518–8523. IEEE, 2014.

[43] Giovanni Ciná and Ulle Endriss. Proving classical theorems of social choice theory in modal logic. *Autonomous Agents and Multi-Agent Systems*, pages 1–27, 2016.

[44] Darrin C Bentivegna. *Learning from observation using primitives.* PhD thesis, Citeseer, 2004.

[45] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *Robotics and Automation, IEEE Transactions on*, 10(6):799–822, 1994.

[46] Monica N Nicolescu and Maja J Mataric. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 241–248. ACM, 2003.

[47] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):223–234, 2008.

[48] Richard Cubek, Wolfgang Ertel, and Gunther Palm. High-level learning from demonstration with conceptual spaces and subspace clustering. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2592–2597. IEEE, 2015.

[49] Akihiko Yamaguchi, Christopher G Atkeson, and Tsukasa Ogasawara. Pouring skills with planning and learning modeled from human demonstrations. *International Journal of Humanoid Robotics*, 12(03):1550030, 2015.

[50] Hendry Ferreira Chame and Philippe Martinet. Cognitive modeling for automating learning in visually-guided manipulative tasks. In *Informatics in Control, Automation and Robotics*, pages 37–53. Springer, 2015.

[51] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, page 0278364913495721, 2013.

[52] Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.

[53] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[54] Bruno Da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. *arXiv preprint arXiv:1206.6398*, 2012.

[55] Gerhard Neumann, Christian Daniel, Alexandros Paraschos, Andras Kupcsik, and Jan Peters. Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8, 2014.

[56] Minija Tamosiunaite, Bojan Nemec, Aleš Ude, and Florentin Wörgötter. Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59(11):910–922, 2011.

[57] Katharina Mülling, Jens Kober, and Jan Peters. Simulating human table tennis with a biomimetic robot setup. In *International Conference on Simulation of Adaptive Behavior*, pages 273–282. Springer, 2010.

[58] Freek Stulp, Evangelos A Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on robotics*, 28(6):1360–1370, 2012.

[59] Jonathan Claassens. An rrt-based path planner for use in trajectory imitation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3090–3095. IEEE, 2010.

[60] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[61] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[62] Chris Bowen and Ron Alterovitz. Closed-loop global motion planning for reactive execution of learned tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1754–1760. IEEE, 2014.