Nonprehensile Manipulation via Multisensory Learning from Demonstration

by

Ku Jin Shin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechantronics Engineering

Waterloo, Ontario, Canada, 2018

Abstract

Dexterous manipulation problem concerns control of a robot hand to manipulate an object in a desired manner. While classical dexterous manipulation strategies are based on stable grasping (or force closure), many human-like manipulation tasks do not maintain grasp stability, and often utilize the intrinsic dynamics of the object rather than the closed form of kinematic relation between the object and the robotic fingers. Such manipulation strategies are referred as nonprehensile or dynamic dexterous manipulation in the literature.

Nonprehensile manipulation typically involves fast and agile movements such as throwing and flipping. Due to the complexity of such motions (which may involve impulsive dynamics) and uncertainties associated with them, it has been challenging to realize nonprehensile manipulation tasks in a reliable way. In this paper, we propose a new control strategy to realize practical nonprehensile manipulation tasks using a robot hand. The main idea of our control strategy are two-folds. Firstly, we make explicit use of multiple modalities of sensory data for the design of control law. Specifically, force data is employed for feedforward control while the position data is used for feedback (i.e. reactive) control. Secondly, control signals (both feedback and feedforward) are obtained by the multisensory learning from demonstration (LfD) experiments which are designed and performed for specific nonprehensile manipulation tasks in concern. We utilize various LfD frameworks such as Gaussian mixture model and Gaussian mixture regression (GMM/GMR) and hidden Markov model and GMR (HMM/GMR) to reproduce generalized motion profiles from the human expert's demonstrations. The proposed control strategy has been verified by experimental results on dynamic spinning task using a sensory-rich two-finger robotic hand. The control performance (i.e. the speed and accuracy of the spinning task) has also been compared with that of the classical dexterous manipulation based on finger gating.

Acknowledgements

I would like to thank my supervisor, Dr. Soo Jeon, for his guidance and support throughout my school years. Without him, I wouldn't have made a decision to come to grad school. Thank you for trusting and providing with a great opportunity to develop skills in robotics and control theory. This work would not have been possible without him.

I would also like to thank all my colleagues at Dr. Soo Jeon's lab. The last two years were more fun because of them.

Lastly, I would like to thank my family for their unlimited love and support.

Dedication

I would like to dedicate my thesis to my beloved family.

Table of Contents

List of Tables

List of Figures

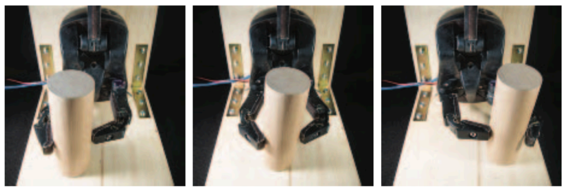
Chapter 1

Introduction

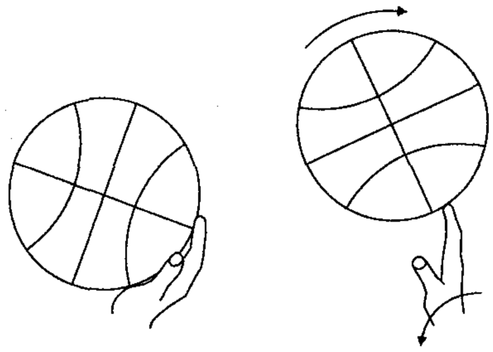
Everyday tasks such as ball throwing, finger flickering and coin flipping with human hands all require to adjust the object's position or orientation by manipulating fingers in desired manner. Producing such tasks with robots refers to *dexterous manipulation* problem in robotics. Human-like dexterous manipulation of robotic hands has long been recognized as a critical challenge to the next generation of robots [71, 28]. Much effort has been made to realize some manipulations such as regrasping, in-grasp manipulation, finger gating, finger pivoting/tracking, rolling and sliding [64]. Historically, grasp stability has usually been considered as a basic requirement of dexterous manipulation with multifingered robotic hands [15].

Instead of relying on grasp stability, an alternative approach is to manipulate objects without maintaining a stable grasp; such manipulation is referred as *nonprehensile (or dynamic dexterous) manipulation* [66]. Nonprehensile manipulation offers several potential benefits over conventional approach such as increased dexterity and increased workspace [62]. This enables robots to perform more human-like manipulations such as pushing, throwing and flipping [78]. The key idea of nonprehensile manipulation is the utilization of the intrinsic dynamics of the target object even when it is disengaged with the robotic hand.

There have been some notable attempts to realize nonprehensile manipulation with robotic hands [10, 36], most of which have relied on mathematical modeling of the hand and the object as well as the dynamics between them. Despite some successes, the complexity of dynamic models and uncertainties associated with them have been a major technical hurdle to take the dexterous manipulation to the next level. Besides, the control design can become a bit inefficient with such model-based approaches due to difficulty in handling



(a) Prehensile manipulation task: object rolling [89]



(b) Nonprehensile Manipulation: ball throwing [61]

Figure 1.1: Example of prehensile and nonprehensile manipulation tasks.

variabilities arising from changing environment (e.g. variability in objects and/or task goals).

1.1 Motivation

In order to perform human-like nonprehensile manipulation on robot hands, we have initially studied off from currently established methods including model-based method and iterative learning method. Model-based manipulation method requires to find the mathematical model of the robot, the object, and the interaction between them (i.e. friction and contact model). This method is heavily studied over the past two decades and still remains as a difficult problem to tackle [59, 92, 47]. Such modeling methods are well described in the following textbooks [30, 68]. Even though the modeling of robots itself is precise and well-established, modeling of the interaction between the object and robots still remain as a big challenge because they heavily relay on the friction and contact states (material, surface roughness, contact type, contact position, etc.). Also, the states of models have various uncertainties often with a stochastic nature, which frequently limit our capability to estimate them [91]. Through tactile and haptic sensors, these issues can somehow be mitigated and thus significantly improving manipulation performances. Nevertheless, success was limited to certain configurations.

Another popular method is to use iterative learning methods, in particular iterative learning control (IRC) [16] and reinforcement learning (RL) [85]. These methods allow to improve the control performance without understanding the precise model of the system. Main idea of both methods is performing repetitive runs to improve the control strategy through each iteration. The fundamental concept is adapted from how human learns to perform a task much better after repetitively performing the same task. IRC develops from a simple control law that calculates the new control input based on the current error signal from the previous run. Hence, it requires the knowledge of the desired trajectory. In contrast to IRC, RL does not require the desired trajectory. Instead, it requires to form a reward (or cost) function to measure the performance of the current run. Defining a good reward function is essential to allow RL to converge to its optimal control law. It seems that both approaches work reasonably well for learning and performing the nonprehensile manipulation tasks. However, both methods require hundreds of iterations which can easily damage the physical robot. Furthermore, it is often very computationally heavy due to the nature of the system (i.e. continuous system, many state variables).

Motivated by recent advances in machine learning techniques, this thesis presents a new control strategy for the nonprehensile dexterous manipulation. More specifically, we put forward two main ideas in this thesis to overcome the previously mentioned difficulties. Firstly, we employ recent techniques of *learning from demonstration* (LfD) for dexterous manipulation. To the best of authors' knowledge, LfD has not yet been implemented for the type of nonprehensile manipulation tasks that we consider in this paper. Secondly, in doing so, we make the explicit use of multiple modalities of sensory data in formulating the control law for the LfD. In fact, these two ideas parallel our own intuition of how human dexterity develops [49, 52];

1. Human can easily learn new skills by observing demonstrations from an expert and reproducing them.

2. Human dexterity is highly dependent on rich sensing (proprioceptive, vision, tactile, etc.).

As one of the most active areas of research in robotics in these days, LfD has seen a significant progress in recent years [17]. The LfD has also been called in the literature as imitation learning (IL), programming by demonstration (PbD), or apprenticeship learning (AL) [23]. The LfD is an autonomous learning strategy through which a human can configure a robot to perform a complex task by providing a good set of demonstrations of the desired task. This technique has been shown to hold a great potential for controlling the robot hand without direct programming or configuring it; hence people who are not expert at robot programming can easily configure the robot to perform desired behaviour by simply demonstrating to it.

Thanks to rapid advances in sensing technologies, modern robotic systems are, more and more, being equipped with various sensors such as motor encoders, strain gauges, tactile sensors, high-speed vision camera or combination of these. The concept of sensory-rich control (or motion control using multimodal sensory data) has been implemented in some classical control applications [46, 63], but its practical use for dexterous manipulation is yet to come. In the later section, we review the up-to-date work on in-hand manipulation, LfD, and perception systems.

1.2 Related Works

1.2.1 In-hand Manipulation

In-hand manipulation has long been studied in literature, one of earlier works of which includes the work by Salisbury [80]. A good review on the topic can be found in [71] and [14]. Despite such a long history, the problem of manipulating position and orientation of objects with human-like dexterity still remains as a big challenge [13, 68, 28]. The representative types of dexterous manipulation tasks that have been achieved so far using an anthropomorphic robot hand include: re-grasping/finger gating [84] and sliding/rolling [64]. Re-grasping/finger gating involves the sequence of steps of grasping and releasing the object. The main drawback of this manipulation is that the additional support plane is required for holding the object when it is detached from the robot. Also, it takes more time to manipulate the object due to the large number of sequences of actions required to achieve the desired task. Sliding/rolling has been pursued as an alternative strategy to perform in-hand manipulation. It may enable faster and more agile manipulations in some cases by allowing the object to slide or roll against the fingers (often without grasp stability) during the manipulation [29]. Many in-hand manipulation approaches pursue the model-based planning perspective, i.e. the grasp planning and the motion planning. Almost all the methods presume that the analytical descriptions of both hand and the object are known, not to mention the force-closure condition [15]. In contrast, we attempt, in this paper, to come up with a general control strategy that can achieve highly dexterous manipulation tasks without precise description of the models of the hand and the object.

1.2.2 Learning from Demonstration (LfD)

Learning from Demonstration (LfD) gained attention in the field of manufacturing robotics at the beginning of 1980s [17]. The first approach to this problem was simply recording the demonstrated data through the joint encoder and torque sensors while a human demonstrates through a teleoperation, and playing back to perform the same task on a robot itself. After many work was done in the field of LfD, the following pipeline was established to be a generalized scheme for the LfD [26].

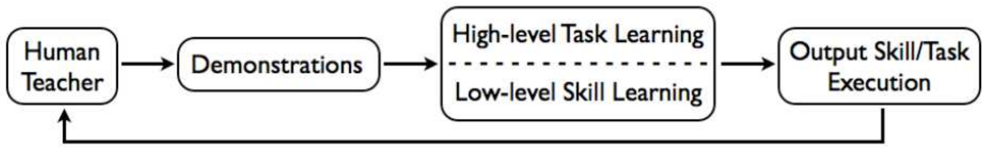


Figure 1.2: Learning by Demonstration pipeline.

From human demonstrations using vision, data gloves, virtual reality, kinesthetic teaching, and many more, the data is processed such that it can be generalized to find the trajectory that is general and noise-free from the data through high-level task learning or low-level skill learning. In other context, these are also referred as learning in symbolic level and trajectory level, respectively. High-level (or symbolic level) learning allows robot to learn hierarchy, rules and loops based on the pre-defined motion elements, which enables robot to learn new skills based on the primitive skill sets that it knows. In contrast, the low-level (or trajectory level) learning allows robot to learn the direct mapping from the action desired to required input signal. Through this learning scheme, robot can learn the generic representation of motion through encoding various sensor inputs. However, reproducing such complex action is difficult.

Incorporating demonstration data from human experts into robot task control has been extensively investigated in recent years. There exist many different approaches, but one view is that they fall under one of two main approaches; reinforcement learning (RL) method or LfD method. RL requires pre-defined reward or cost function information. Coming up with well-defined function is crucial since this governs the performance of the learning. LfD is a more common approach to allow robot to perform human task from demonstration data. Many use the probabilistic approach that is encoding task variables using probability density function (pdf) and reproducing through a regression technique.

Symbolic Learning (High-level Task Learning)

Many earlier work in LfD dealt with symbolic learning. The task at a symbolic level is described by the sequential or hierarchical organization of a discrete set of primitives that are pre-determined or extracted from pre-defined rules [23]. Examples of such works can be found in [35, 70, 72, 37]. The basic structure of the symbolic learning approach is outlined as follows: demonstration, task representation, task generation and execution. From the expert's demonstration through kinesthetic learning or teleoperation, the task is segmented with known pre-defined actions and formed to a hierarchical tree. From the tree, the generalized task sequence is achieved then the robot executes the task. To refine the robot's task, two main methods are used: providing more demonstration or expert's feedback. By combining multiple demonstration into a single hierarchical tree, the robot can generalize the task more and is able to compensate any variability during the task execution. From the expert's feedback, robot can incorporate any constraints during the task generalization/reproduction.

Although this symbolic learning process require iteration and multiple demonstrations to teach a robot a desired task, the learning speed was much faster against brutal-force trial-error approach. The main disadvantage of the above approach is that it largely rely on the prior knowledge to pre-defined symbolic representations.

Trajectory Learning (Low-level Skill Learning)

Low-level learning refers to learning a specific task or action from demonstration data acquired from various sensors. Terms such as motor skill learning and primitive action learning are used interchangeably. All the learning methods can be viewed as a supervised learning as the main goal is to find the mapping that will reproduce the desired motion from the demonstration data (training set). Hence, some of the early works include the use of Neural Network and Inductive Logic. However, finding a direct mapping from the sensor inputs to the desired motor input failed to acquire the trajectory that is free from any other source of errors or disturbances. These includes human demonstration error, sensor noises, and environmental disturbances. Therefore, modern approaches focus more on finding a generalized trajectory from multiple demonstrations. Modern approaches are based on probabilistic modeling (such as Gaussian Mixture Regression (GMR) and Hidden Markov Models (HMM)), Dynamic Movement Primitives (DMPs) or combination of them.

Probabilistic Approach Gaussian mixture model and Gaussian mixture regression (GMM/GMR) technique is the most widely used probabilistic approach in the field [22]. There also exist some extensions of GMM/GMR such as task parameterized version of GMM (TP-GMM) [20]. Most of the problems for LfD involves learning the position trajectory of the robot to perform the desired task; however, some researchers expanded the idea to force-based manipulation tasks [33, 60, 56, 76]. In this paper, we try to combine the advantages of both the position-based and the force-based GMM/GMR.

HMM is another popular probabilistic tool in LfD framework. HMMs have been widely used in recognizing human generated information data such as speech recognition [73], harndwriting recognition [38], and sign language recognition [83]. Due to its capability of capturing both spatial and temporal variations, many LfD frameworks were developed. Encoding of the robot motion trajectory with HMM can be found in [40, 57, 88]. Many researchers utilized HMM to perform LfD due to the fact that the human demonstration is time-varying, not perfect, and sensory processes are noisy. Tso and Liu [88] utilized HMM to select the best robot trajectory amongst a number of demonstration cycles of the peg-in-the-hole assembly task. Yang et al. [90] used HMM to encode the trajectory that human demonstrates on a teleoperation-controlled space robot. Human gesture was leaned using HMM and newly created data was classified using the trained model for verification. By using HMM, a Mimesis theory was developed for the primitive skill of imitative learning on humaniod robot [44]. For reproducing the generalized motion, key point spline fitting, Gaussian Mixture Regression, HMM trajectory were studied.

Dynamic Systems Models Another common approach of trajectory learning is to apply the theory of dynamical systems. Various approaches are studied including sequenced Linear Dynamical Systems (LDS) [34], Stable Estimator of Dynamical Systems (SEDS) [50], Implicit Dynamical Systems [53] and Dynamic Movement Primitives (DMP) [43, 42]. Among these methods, DMP has proven to be very effective when the learning is done by observing the expert's demonstration. This method uses nonlinear differential equations to form control policies in trajectory formation, which allows the learned dynamical system to represent the whole flow field rather than a single trajectory. The main idea behind the DMP is to use a set of first order systems and transform them into nonlinear dynamic system by adding nonlinear forcing terms. As the first order system works as a spring-damper system, the system guarantees convergence to a goal state. DMP has been used in imitating reaching movements [43, 81], manipulation tasks [51], and obstacle avoiding [42]. However, the DMP still has a few drawbacks: it is hard to fully control the generated motion profile and it cannot impose constraints on the motion.

Extending the idea of DMP, reinforcement learning methods are used to optimize the learned DMP trajectories. It utilizes Policy Improvement through Path Integration (PI^2) algorithm to track the immediate cost of the trajectory throughout the movement and calculates the cost-to-go at each point to find the optimal trajectory of the given task [86]. This method has been efficiently applied to various areas including manipulation task [48] and robot dog walking [86].

1.2.3 Perception Systems

Multisensory Integration in Dexterous Manipulation

Control laws for dexterous manipulation have largely relied on the precise knowledge of the position of the manipulating fingers and/or the target object. Other types of sensory data adopted in recent years include the tactile and vision [74, 93]. Closing a feedback loop with such sensors may allow the control law to be able to adapt to unknown object properties [9]. In particular, the tactile sensory information can play an important role in the process of manipulation by detecting both the direction and the magnitude of the contact force between the object and the robot fingers. Examples of the utility of tactile information in in-hand manipulation tasks can be found in [89, 58, 65]. As is the case for humans, the visual sense can form another important sensory feedback in dexterous manipulation. Visual servoing, or vision-based robot control, is well-established in the literature and can enable precise manipulation with both fully-actuated hands [4, 25] and underactuated hands [24].

Perception Systems in LfD

Sensors play an essential role to observe the teacher's demonstration in LfD framework. Based on the method of skill transfer from human to robot, the right perception system is used [5]. The most widely used hardware is based on vision system, which includes camera and optical tracking systems [8]. This reflects how human eye sees the demonstration and understands the task. However, such vision-based system has limitations due to the limited resolution of the vision system and it is difficult to solve the correspondence problem of mapping the task from human demonstration to robot. Proprioceptive sensors provide information about internal state of the robot (e.g. motor encoders), and this is useful for demonstration approaches such as kinesthetic teaching and teleoperated demonstration [17]. For tasks that involves interaction with the surrounding utilizes force-based perception (e.g. manipulation tasks). Through force/torque sensors, the robot can learn about the reference force/torque profile and tactile data about its contact. This ensures robot not only to understand the surroundings, but also to safely interact with the environment.

1.2.4 Reinforcement Learning in Dexterous Manipulation

As artificial intelligence is getting popular in recent research works, specific learning technique called reinforcement learning (RL) [85] is applied to solve various dexterous manipulation tasks. RL problem is modeled using Markov decision process (MDP), which provides generic mathematical model abstraction to a sequential decision making problem of known states. The goal of RL is to find a control policy to maximize the user-defined reward function through iterations. Various method of RL is applied to dexterous manipulation tasks. These methods includes model-free deep RL [94], model-based deep RL [55, 39] and behavioural cloning [67]. Many works were done in a simulation environment as they are easy to setup and perform many iterations. However, these simulated model does not reflect the real system, hence some works were also done to transfer the policy learned from the simulated environment to actual experimental setups [79, 45]. Although many works were done to enable dexterous manipulations through RL, the work on nonprehensile manipulation is rare as most of the work was done for prehensile tasks.

1.3 Objective

In this work, we examine the performance of nonprehensile manipulation based on the LfD approach using a custom-built robot hand. The robot hand is equipped with various sensors including tactile (or finger tip force) and vision as well as position encoders. The proposed LfD framework can be divided into three sub-tasks:

1. Motion profile learning
2. Task reproduction
3. Control parameter learning

During the motion profile learning, the expert's demonstration data is collected using the finger tip force sensor and the position encoder. Using two probabilistic modeling techniques including Gaussian mixture model (GMM) and hidden Markov model (HMM), demonstration data is generalized with multiple Gaussian distributions. Using the Gaussian mixture regression (GMR), the position and the finger tip force data have been processed to realize the generalized position and the force signals, which are then inserted to the controller as reference signals. The task reproduction allows the robot to reproduce the motion profile obtained from LfD framework. The generated desired position signal is used for the feedback control while the desired contact force signal is injected as a feedforward command. After successfully performing the desired motion, any additional control parameter will be learned through the control parameter learning process. The control parameter can be optimized through iteration steps.

As an exemplary task, we realized a nonprehensile task of spinning and stopping a circular disk with two fingers, the goal of which is to make the disk rotate as fast and closely to the desired angle as possible. The detailed experimental setup and the task description is described in the next section.

1.4 Experiment: Nonprehensile Spinning Task

1.4.1 Experimental Setup

The hardware platform for this research is the custom-built planar robot hand, which is depicted in Figures 1.3 and 1.4. This robot has two fingers each of which has two degrees of freedom (2 DOF) consisting of two joints and two links. In order to mimic the sensory-rich behavior of human, the robot is equipped with multiple sensors; each joint is equipped with the encoder, each arm is a force sensor (strain gauge) and the finger tip has a 3D tactile sensor attached at the end-effector. The vision camera at the top of the system (see Figure 1.4) captures the manipulation scene and measures the configuration of the object in realtime. All the hardware is connected to LabVIEW Real-Time target (manufactured by National Instruments Inc.) that is running at 1 kHz except for the vision loop which is running at 12.5 Hz ($T_v = 80$ ms). The detailed hardware specification is shown in Table 1.1.

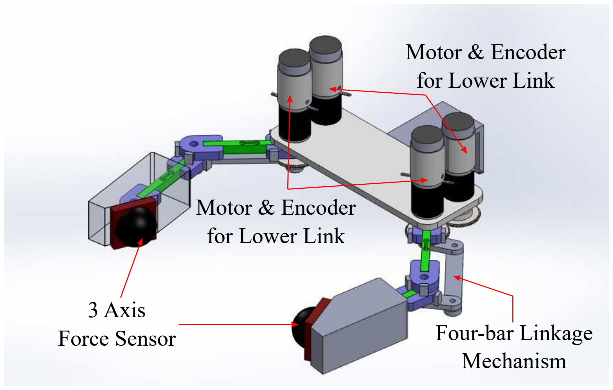


Figure 1.3: 3D schematic of the robot hand

As shown in Figure 1.4, the object is mounted to a table bearing to reduce the friction against the ground as much as possible. An image identifier tag is attached on top of the object to allow vision camera to track the orientation of the object. The cylindrical

Table 1.1: Hardware Specifications

Component	**Manufacturer / Model**	**Specification**
Geared Motor Set	Maxon Motor (222053, 201937, 201937)	Max speed: 9270 rpm, Rated torque: 11.6 mNm, Gear ratio: 84:1, Encoder resolution: 512 ppr
Strain Gauge	Strain Measurement Device (S220)	Max load: 6 lbs
3 Axis Force Sensor	OnRobot (OMD-30-SE-100N)	Nominal capacity: 100 N (F_z compression), ± 25 N (F_{xy})
Vision Sensor	Basler (cA2000-340km)	Resolution: 2048px$\times$1088px

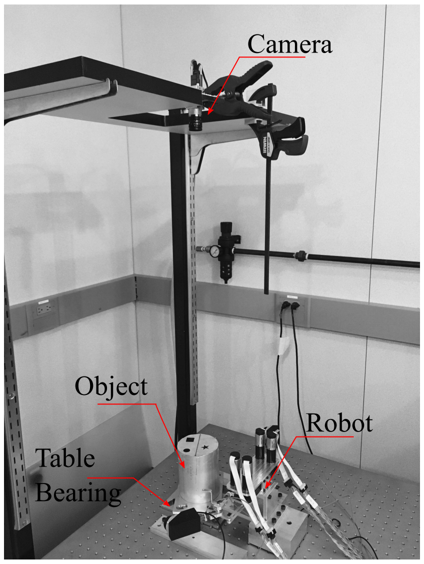


Figure 1.4: Overall configuration of experimental setup

object is made of aluminum with its rotational inertia of around 0.057 $kg \cdot m^2$. Using our experimental setup, the maximum rotation to be achieved by a single step force closure manipulation is $\sim \pm 25°$.

1.4.2 Nonprehensile Task: Nonprehensile Spinning

The nonprehensile task that we are considering in this thesis is nonprehensile spinning task. Through a single stage force closure manipulation, where the robot is in contact with the object all the time during the manipulation, the maximum rotation angle is $\sim \pm 25°$ in our current experimental setup. Hence, in order to achieve object rotation greater than $\pm 25°$, the robot is required to re-grasp and perform the manipulation repeatedly until the desired manipulation is reached. Through the proposed control system in Section 1.3, we would learn to perform nonprehensile spinning task to achieve greater angle object rotation in precise and faster manner. The nonprehensile manipulation for spinning proceeds in three steps as shown in Figure 1.5; impulsive spinning, free rotation and catching. The spinning move is the main task to generate the fast rotational motion in a nonprehensile way. Thus, we utilizes the LfD framework to learn the impulsive spinning motion. The catching motion completes the nonprehensile manipulation task, and it is critical to find the right time to catch the object. We employ iterative learning method to find the proper time to re-grasp the object to stop at a desired angle.

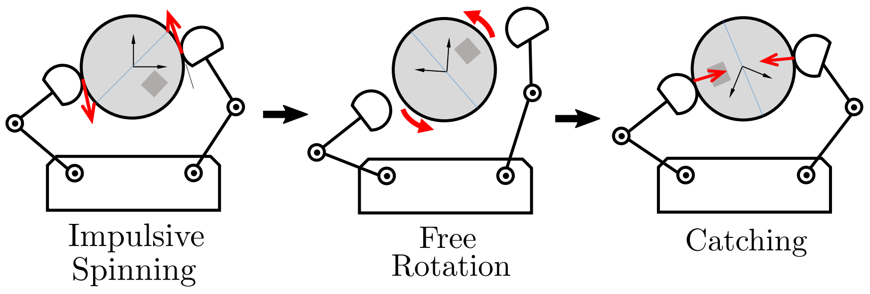


Figure 1.5: Control procedure for nonprehensile spinning.

1.5 Organizations

The rest of this thesis is organized as follows. In Chapter 2, mathematical background in probabilistic LfD frameworks and robot kinematics is presented. Based on the available

LfD tools, we extend its application to nonprehensile motion profile learning with sensory-rich robot in Chapter 3. Then, the reproduced generalized trajectory is then evaluated on the real robot through combination of feedback and feedforward controller in Chapter 4. Finally, learning unknown controller parameters are studied in Chapter 5 to complete the nonprehensile manipulation system via multi-sensory learning from demonstration.

Chapter 2

Mathematical Background

In this section we overview the mathematical tools for learning from demonstration (LfD) framework and robot manipulator model. Specifically, the most popular probabilistic modeling tool of Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM) is outlined. Gaussian mixture regression (GMR) technique is also reviewed as a popular method to retrieve the generalized trajectory from the encoded model. Finally, modeling of the 2-link robot manipulator is discussed, which we utilize in controller design.

2.1 Notation

Let us first introduce some common notations used in later sections.

$$\begin{aligned}
K &: \text{number of Gaussians in the model, indexed with } k \in \{1, \dots, K\} \\
M &: \text{number of demonstrations, indexed with } m \in \{1, \dots, M\} \\
T &: \text{number of datapoints in a single motion profile, indexed with } t \in \{1, \dots, T\} \\
N &: M \times T = \text{total number of datapoints in entire demonstrated motion profile,} \\
&\quad \text{indexed with } n \in \{1, \dots, N\} \\
D &: \text{number of state variables in a single datapoint} \\
\xi_t \in \mathbb{R}^D &: \text{single datapoint with } D \text{ state variables} \\
\xi_{1:T} &: \{\xi_1, \dots, \xi_T\} = \text{single observed motion profile} \\
\xi &: \{\xi^1_{1:T}, \dots, \xi^M_{1:T}\} = \text{entire demonstration motion profile set}
\end{aligned}$$

Θ^{GMM} : Gaussian Mixture Model (GMM)

Θ^{HMM} : Hidden Markov Model (HMM)

2.2 GMM/GMR Framework

2.2.1 Gaussian Mixture Model (GMM)

A Gaussian mixture model (GMM) is a probabilistic model that assumes that all of the datapoints are generated from a mixture of a finite number of Gaussian distributions [23]. Such modeling technique is useful when trying to find a trend from multiple data sets. This can be seen as a generalized version of the k-means clustering technique [41], but may be considered more general in terms of flexibility in choosing the covariance between the Gaussian distributions. Some widely used application of GMM includes speech recognition [12] and multiple object tracking [31].

Model Definition

GMM can estimate the probability density distribution of the samples, where the estimated model is the weighted sum of several Gaussian models. Denoting by K the number of component Gaussian distributions, the probability that the D-dimensional t-th datapoint $\xi_t \in \mathbb{R}^D$ belongs to the GMM can be written as

$$p(\xi_t) = \sum_{k=1}^{K} p(k) p(\xi_t|k) \tag{2.1}$$

where $p(k) = w_k \in [0,\, 1]$ is the mixing coefficients (or prior probability of the k-th Gaussian), and $p(\xi_t|k)$ is the conditional probability density function (pdf). More specifically,

$$\begin{aligned} p(\xi_t|k) &= \mathcal{N}(\xi_t|\mu_k, \Sigma_k) \\ &= \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} e^{-\frac{1}{2}(\xi_t-\mu_k)^\intercal \Sigma_k^{-1}(\xi_t-\mu_k)} \end{aligned} \tag{2.2}$$

where $\mu_k \in \mathbb{R}^D$ and $\Sigma_k \in \mathbb{R}^{D\times D}$ denote the center and the covariance matrix for the k-th Gaussian, respectively. Thus, the GMM can be characterized by the set of parameters $\Theta^{\text{GMM}} = \{w_k, \mu_k, \Sigma_k\}_{k=1}^K$. The prior probability w_k acts as a weighting factor for each Gaussian model and it must satisfy $\sum_{k=1}^K w_k = 1$.

2.2.2 Parameter Learning: Expectation-Maximization (EM) Algorithm

These set of unknown parameters can be found using the standard expectation-maximization (EM) algorithm, which is basically the maximum likelihood estimation (MLE) of the mixture parameters that is performed iteratively [32].

Let us first define a posterior probability (called responsibility, $\gamma_k(\xi)$) for a given data-point value ξ. Using the Baye's rule,

$$\begin{aligned}\gamma_k(\xi) = p(k|\xi) &= \frac{p(k)p(\xi|k)}{p(\xi)} \\ &= \frac{w_k \mathcal{N}(\xi|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_k \mathcal{N}(\xi|\mu_j, \Sigma_j)}\end{aligned} \tag{2.3}$$

As the name of the algorithm suggests, the EM algorithm iterates over two steps: E-step and M-step. At each cyle, E-step evaluates the reponsibilities using the current parameter values, and then M-step re-estimates the parameters by maximizing the joint distribution of the data and the hidden variable. In other words, new data points are drawn from the given GMM and the GMM parameters are newly calculated through MLE step. The initial parameters are usually approximated using k-means clustering algorithm [41].

E-step:

$$\forall k, n, \text{ calculate } \gamma_k(\xi_n) \text{ with the current parameters.}$$

M-step:

$$\begin{aligned} w_k^{new} &= \frac{\sum_{n=1}^{N} \gamma_k(\xi_n)}{N}, \\ \mu_k^{new} &= \frac{\sum_{n=1}^{N} \gamma_k(\xi_n)\xi_n}{\sum_{n=1}^{N} \gamma_k(\xi_n)}, \\ \Sigma_k^{new} &= \frac{\sum_{n=1}^{N} \gamma_k(\xi_n)(\xi_n - \mu_k^{new})(\xi_n - \mu_k^{new})^{\intercal}}{\sum_{n=1}^{N} \gamma_k(\xi_n)} \end{aligned}$$

After calculating the new parameters, the log-likelihood value, $\mathcal{L}(\xi|\Theta^{\text{GMM}})$, is calculated to compare against the previous log-likelihood value, such that if the increase in the log-likeliood is small, then the iteration stops.

$$\mathcal{L}(\xi|\Theta^{\text{GMM}}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} w_k \mathcal{N}(\xi_n|\mu_k, \Sigma_k) \right\} \tag{2.4}$$

Therefore, the criteria of stopping the iteration is represented as $\frac{\mathcal{L}^{new}}{\mathcal{L}} < C$ where C is the threshold.

2.2.3 Gaussian Mixture Regression (GMR)

From the learned GMM, we can reproduce a generalized trajectory through the Gaussian mixture regression (GMR) process [23].

Let us assume that the t-th data point ξ_t consists of the input $\xi_t^{\mathcal{I}}$ and the output $\xi_t^{\mathcal{O}}$, i.e. $\xi_t = \text{col}\left(\xi_t^{\mathcal{I}}, \xi_t^{\mathcal{O}}\right)$. From Equations (2.1) and (2.2), the probability for ξ_t to belong to the learned GMM is

$$p(\xi_t) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\xi_t|\mu_k, \Sigma_k)$$

where the mean vector μ_k and the covariance matrix Σ_k can be partitioned as

$$\mu_k = \begin{bmatrix} \mu_k^{\mathcal{I}} \\ \mu_k^{\mathcal{O}} \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_k^{\mathcal{I}} & \Sigma_k^{\mathcal{I}\mathcal{O}} \\ \Sigma_k^{\mathcal{O}\mathcal{I}} & \Sigma_k^{\mathcal{O}} \end{bmatrix} \tag{2.5}$$

The superscripts $\mathcal{I}$ and $\mathcal{O}$ represent the input and the output variables, respectively. Then, the GMR process involves the prediction of the distribution of the output data $\xi_t^{\mathcal{O}}$ for the k-th Gaussian when the input data $\xi^{\mathcal{I}}$ is given. Specifically, using the conditional probability distribution, we have

$$p\left(\xi_t^{\mathcal{O}}|\xi_t^{\mathcal{I}}, k\right) = \mathcal{N}\left(\hat{\xi}_{k,t}, \hat{\Sigma}_{k,t}\right) \tag{2.6}$$

where $\hat{\xi}_{k,t}$ and $\hat{\Sigma}_{k,t}$ are the predicted mean and the predicted covariance of $\xi_t^{\mathcal{O}}$ for the k-th Gaussian given the input $\xi_t^{\mathcal{I}}$. Using the conditional probability, they are given by

$$\begin{aligned} \hat{\xi}_{k,t} &= \mu_k^{\mathcal{O}} + \Sigma_k^{\mathcal{I}\mathcal{O}} \left(\Sigma_k^{\mathcal{I}}\right)^{-1} \left(\xi_t^{\mathcal{I}} - \mu_k^{\mathcal{I}}\right), \\ \hat{\Sigma}_{k,t} &= \Sigma_k^{\mathcal{O}} - \Sigma_k^{\mathcal{O}\mathcal{I}} \left(\Sigma_k^{\mathcal{I}}\right)^{-1} \Sigma_k^{\mathcal{I}\mathcal{O}} \end{aligned}$$

Then, the complete GMM can be obtained by summing up Equation (2.6) over k as

$$p\left(\xi_t^{\mathcal{O}}|\xi_t^{\mathcal{I}}\right) = \sum_{k=1}^{K} h_{k,t}\mathcal{N}\left(\hat{\xi}_{k,t}, \hat{\Sigma}_{k,t}\right) \tag{2.7}$$

where $h_{k,t} = p\left(k|\xi_t^{\mathcal{I}}\right)$ is the probability that the k-th Gaussian distribution is responsible for $\xi_t^{\mathcal{I}}$;

$$h_{k,t} = \frac{p(k)p\left(\xi_t^{I}|k\right)}{\sum_{i=1}^{K} p(i)p\left(\xi_t^{I}|i\right)} = \frac{\pi_k\mathcal{N}\left(\xi_t^{\mathcal{I}}; \mu_k^{\mathcal{I}}, \Sigma_k^{\mathcal{I}}\right)}{\sum_{i=1}^{K} \pi_i\mathcal{N}\left(\xi_t^{\mathcal{I}}; \mu_i^{\mathcal{I}}, \Sigma_i^{\mathcal{I}}\right)} \tag{2.8}$$

By using the linear transformation property of Gaussian distributions, the conditional expectation $p\left(\xi_t^{\mathcal{O}}|\xi_t^{\mathcal{I}}\right)$ can be approximated by a single Gaussian distribution $\mathcal{N}\left(\hat{\xi}_t, \hat{\Sigma}_t\right)$ where $\hat{\xi}_t$ and $\hat{\Sigma}_t$ are the weighted sums, respectively, of $\hat{\xi}_{k,j}$ and $\hat{\Sigma}_k$ through $h_{k,j}$ over k [22]. In the end, the sequence of $\hat{\xi}_t$ represents the desired trajectory from the GMR with its uncertainty (or variability) encoded by $\hat{\Sigma}_t$ for each data point j.

2.2.4 Dynamic Time Warping (DTW)

The GMM/GMR approach explained above is capable of capturing spatial variability. However, it does not encapsulate well the temporal variation within the data set. The dynamic time warping (DTW) algorithm is a method proposed to measure the similarity between two temporal sequences and to align them in a more consistent way [27]. Preprocessing datapoints with DTW is known to estimate the GMM parameters more precisely so that more concrete GMR trajectory can be reproduced. Hence, the DTW is widely applied in the fields where temporal sequences are used, such as video, audio, and graphics data.

A basic idea of DTW is as follows [23]. Given two trajectories, ξ and $\bar{\xi}$, of length T, consider the distance between two datapoints of temporal index k_1 and k_2, i.e. $h(k_1, k_2) = \|\xi_{k_1} - \bar{\xi}_{k_2}\|$. Then, the DTW determines the warping path, $S = \{s_l\}_{l=1}^{L}$ for L elements of $s_l = \{k_1, k_2\}$ such that its cumulative distance $\gamma(k_1, k_2)$ is successively minimized by the induction process;

$$\begin{aligned}\gamma(k_1, k_2) &= h(k_1, k_2)\\ &\quad + \min\left[\gamma(k_1 - 1, k_2 - 1), \gamma(k_1 - 1, k_2), \gamma(k_1, k_2 - 1)\right]\end{aligned}$$

with the initial value $\gamma(1,1) = 0$. As will be discussed later, time alignment through the DTW step is particularly important in implementing GMR for nonprehensile manipulation tasks because the contact force is often impactive (i.e. a large force is applied within a short duration of time).

2.3 HMM/GMRa Framework

2.3.1 Hidden Markov Model (HMM)

The hidden Markov model (HMM) is another popular statistical model that describes the evolution of observable events that depend on the internal factors, which are not directly observable. Such modeling is widely used in the field of LfD as it is capable of modeling both the spatial and the temporal variations in data [23]. One can simply think of it as an extension of GMM with the transition characteristic between the Gaussian states.

Model Definition

An HMM with K number of states is characterized as $\Theta^{\text{HMM}} = \{\{a_{ij}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ [73]:

- $\{a_{i,j}\}$, the state transition probability, with $1 \leq i, j \leq K$
- Π_i, the initial state probability vector, with $1 \leq i \leq K$
- μ_i, Σ_i, the distribution of the observations of each state is represented with multivariate Gaussian, which is shown with $\mathcal{N}(\xi_t|\mu_i, \Sigma_i)$

To put it simply, the HMM represents how the observation datapoints are mapped from the hidden Gaussian states.

2.3.2 Forward Variable

Forward variable is the probability to be in state i at time step t and the partial observation $\xi_{1:t} = \{\xi_1, \xi_2, \ldots, \xi_t\}$, and it is defined as

$$\alpha_{t,i}^{\text{HMM}} = p(s_t = i, \xi_{1:t}) \tag{2.9}$$

where s_t indicates the hidden state at time step t. This variable can be easily calculated through induction.

$$\alpha_{t,i}^{\text{HMM}} = \big(\sum_{j=1}^{K} \alpha_{t-1,j}^{\text{HMM}} a_{ij}\big)\mathcal{N}(\xi_t|\mu_i, \Sigma_i) \tag{2.10}$$

starting from:

$$\alpha_{1,i}^{\text{HMM}} = \Pi_i \mathcal{N}(\xi_1|\mu_i, \Sigma_i) \tag{2.11}$$

2.3.3 Backward Variable

In a similar manner, we can define the probability of the partial observation $\{\xi_{t+1}, \ldots, \xi_{T-1}, \xi_T\}$ knowing that we are in state i at time step t. This is called the backward variable which is defined as

$$\beta_{t,i}^{\text{HMM}} = p(\xi_{t+1:T}|s_t = i) \tag{2.12}$$

and it can also be calculated by induction:

$$\beta_{T,i}^{\text{HMM}} = 1 \tag{2.13}$$

$$\beta_{t,i}^{\text{HMM}} = \sum_{j=1}^{K} a_{ij}\mathcal{N}(\xi_{t+1}|\mu_j, \Sigma_j)\beta_{t+1,j}^{\text{HMM}} \tag{2.14}$$

2.3.4 Parameter Learning: Baum-Welch Algorithm

The parameter estimation of HMM can be done through a modified version of Expectation-Maximization algorithm, or the Baum-Welch algorithm [11]. Similar to Section 2.2.2, the algorithm iterates over two steps: E-step and M-step. At each cycle, E-step infers the hidden states given the HMM parameters and the re-estimated the parameters given the data in M-step.

The K hidden state HMM's parameters (Θ^{HMM}) can be estimated with the observation information of length T , $\xi_{1:T} = \{\xi_1, \ldots, \xi_T\}$. In order to simplify the calculation we define the following variables:

$$\begin{aligned}\gamma_{t,i}^{\text{HMM}} &= p(s_t = i|\xi_{1:T}) \\ &= \frac{\alpha_{t,i}^{\text{HMM}}\beta_{t,i}^{\text{HMM}}}{\sum_{k=1}^{K}\alpha_{t,k}^{\text{HMM}}\beta_{t,k}^{\text{HMM}}}\end{aligned} \tag{2.15}$$

$$\begin{aligned}\zeta_{t,i,j}^{\text{HMM}} &= p(s_t = i, s_{t+1} = j | \xi_{1:T}) \\ &= \frac{\alpha_{t,i}^{\text{HMM}} a_{ij} \mathcal{N}(\xi_{t+1} | \mu_j, \Sigma_j) \beta_{t+1,j}^{\text{HMM}}}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{t,k}^{\text{HMM}} a_{kl} \mathcal{N}(\xi_{t+1} | \mu_l, \Sigma_l) \beta_{t+1,l}^{\text{HMM}}}\end{aligned} \tag{2.16}$$

where α and β are the forward and backward variables described in previous sections. Using Equations (2.15) and (2.16), the parameters of the HMM model $\Theta^{\text{HMM}} = \{\{a_{ij}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ can be calculated iteratively. For the hidden state i (we will drop HMM superscript here),

$$\Pi_i^{new} = \gamma_{1,i} \tag{2.17}$$

$$a_{ij}^{new} = \frac{\sum_{t=1}^{T-1} \zeta_{t,i,j}}{\sum_{t=1}^{T-1} \gamma_{t,i}} \tag{2.18}$$

$$\mu_i^{new} = \frac{\sum_{t=1}^{T} \gamma_{t,i} \xi_t}{\sum_{t=1}^{T} \gamma_{t,i}} \tag{2.19}$$

$$\Sigma_i^{new} = \frac{\sum_{t=1}^{T} \gamma_{t,i} (\xi_t - \mu_i^{new})(\xi_t - \mu_i^{new})^\intercal}{\sum_{t=1}^{T} \gamma_{t,i}} \tag{2.20}$$

2.3.5 GMR for HMM

From the definition of the HMM, applying classical GMR approach as mentioned before is possible, yet the temporal information is not utilized in the classical GMR method. In order to utilize the temporal information that we characterized in HMM, we use a modified version of GMR called GMRa [21, 77]. The classical formulation is similar to Equation (2.7). Instead of using the weighting factor h, we use α, the forward variable in HMM. Hence the new equation becomes

$$p\left(\xi_t^{\mathcal{O}} | \xi_t^{\mathcal{I}}\right) = \sum_{k=1}^{K} \alpha_{k,t} \mathcal{N}\left(\hat{\xi}_{k,t}, \hat{\Sigma}_k\right) \tag{2.21}$$

2.4 Modeling of Robot Manipulators

In this section, we outline the mathematical model of general 2-link manipulator. The schematic diagram can be found in Figure 2.1.

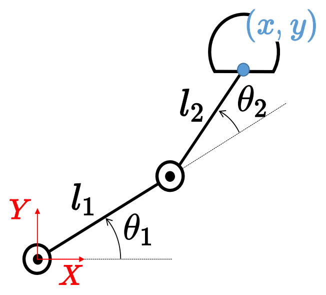


Figure 2.1: 2 link manipulator schematic diagram

2.4.1 Forward Kinematics

The end-effector position (x, y) relative to the global coordinate axis (X, Y) can be written as

$$\begin{aligned} x &= x_0 + l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ y &= y_0 + l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{aligned} \tag{2.22}$$

2.4.2 Inverse Kinematics

Unlike forward kinematics, the inverse kinematics problem may have multiple solutions, a unique solution, or no solution in general. It can be easily solved in the planar two link robot by utilizing the polar coordinates.

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \alpha &= \cos^{-1}\left(\frac{l_1^2 + \_2^2 - r^2}{2l_1l_2}\right) \\ \beta &= \cos^{-1}\left(\frac{r^2 + l_1^2 - l_2^2}{2l_1r}\right) \end{aligned} \tag{2.23}$$

$$\begin{aligned} \theta_1 &= \tan^{-1}(\frac{y}{x}) \pm \beta \\ \theta_2 &= \pi \pm \alpha \end{aligned}$$

where the sign used to calculate θ_1 and θ_2 must be the same.

2.4.3 Jacobian

Jacobian describes the relationship between two different coordinate systems. The Jacobian is given by a set of partial differential equations. From Equation (2.22),

$$J = \frac{\partial p_e}{\partial \theta} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{2.24}$$

In this case, the Jacobian describes the relationship between the end-effector position (which we will denote $p_e = [x, y]^\intercal$) and the set of joint angles (which we will denote $\theta = [\theta_1, \theta_2]^\intercal$). Using Jacobian, the velocities of the end-effect are related to those of joint angles by

$$\dot{p_e} = J\dot{\theta}$$

In addition to velocity relationship, using the energy-force definition, the joint torque can be related to the end-effector force through the same Jacobian.

$$F_e = J^\intercal \tau \tag{2.25}$$

where F_e is the end-effector force.

Chapter 3

Motion Profile Learning from Demonstration

In this chapter, we outline different methods to allow robot to learn nonprehensile manipulation task from the expert's demonstrations while utilizing robot's rich sensing ability. Demonstrations are provided to the robot through kinesthetic teaching and various sensors record the demonstrated information. Then through probabilistic modeling described in Section 2, these motion profiles are encoded to reproduce a generalized motion profile. The overall flow of this learning system is shown in Figure 3.1. We perform three different LfD approaches including GMM/GMR, GMM+DTW/GMR and HMM/GMRa. In the next section we outline demonstration collection, various LfD framework implementations and experimental results for nonprehensile spinning task described in Section 1.4.2.

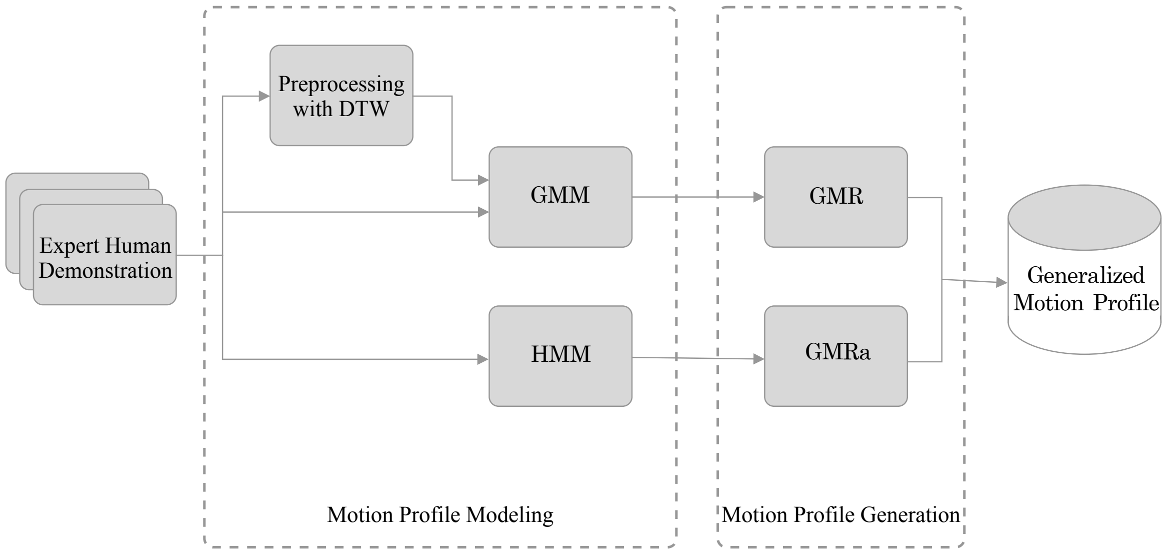

Expert Human Demonstration
Preprocessing with DTW
GMM
HMM
GMR
GMRa
Generalized Motion Profile
Motion Profile Modeling
Motion Profile Generation

Figure 3.1: Learning System flowchart

3.1 Demonstration Collection

Human expert's demonstration data may be collected through different ways. Examples include visual motion tracking system [54], physically guided by humans through kinesthetic teaching [77] or teleoperation by human through remote controller [69]. In this work, we focus on *kinesthetic teaching*, in which a human teacher guides the robot in performing the skill by physically holding it [3]. Kinesthetic teaching has been chosen for its advantage over other demonstration techniques such as direct compensation of the kinematic constraints at the first hand, and no correspondence problem due to a direct relationship from the demonstration to the sensor readings on the robot. Also, due to the limited degrees of freedom of the robot (limited to planar motion), there was no external force (ie. gravity) compensation required when performing the kinesthetic teaching. Figure 3.2 shows how the kinesthetic teaching was performed by the expert for our research. As mentioned in Section 1.4.2, the LfD framework is employed to allow robot to learn the joint angle and the end-effector force motion profiles for the impulsive spinning motion. Hence, the expert's demonstration only focused on the impulsive spinning.

Total of M demonstrations are performed to the robot and the collection of datapoints is represented with $\xi = \{\xi^m_{1:T}\}^M_{m=1}$ where $\xi^m_{1:T} = \{\xi^m_1, \xi^m_2, \ldots, \xi^m_T\}$ is the collection of data-

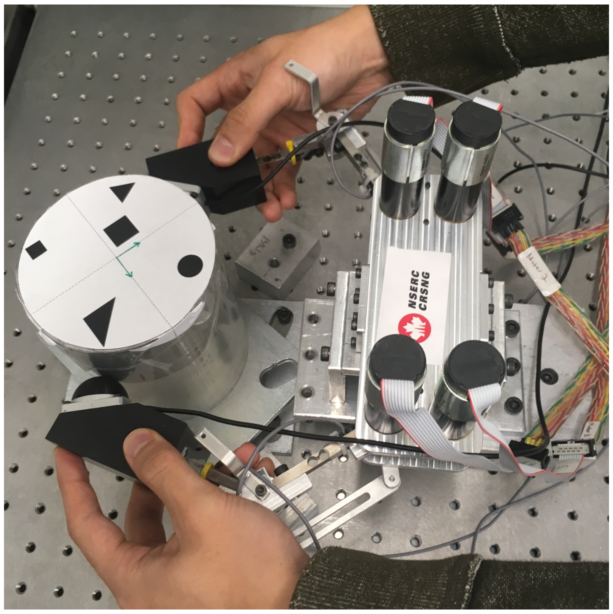


Figure 3.2: Kinesthetic Teaching

points in a single demonstration (m) and T represents the total number of datapoints in a single demonstration. Each datapoint $\xi_t^m \in \mathbb{R}^8$ is a vector constructed by collecting time index, position and force profiles that are gathered from the robot's equipped sensors.

$$\xi_t^m = \text{col}\left((\xi_i)_t^m, (\xi_\theta)_t^m, (\xi_f)_t^m \right)$$

where $\xi_i \in \mathbb{R}$ denotes a time index, $\xi_\theta \in \mathbb{R}^4$ denotes a vector of joint angles and $\xi_f \in \mathbb{R}^4$ denotes a vector of end-effector forces. Figure 3.3 illustrates these data variables on the schematic diagram. In the following sections, we show how we utilize collected demonstration information to learn the motion profile for nonprehensile manipulation task through GMM/GMR and HMM/GMRa framework.

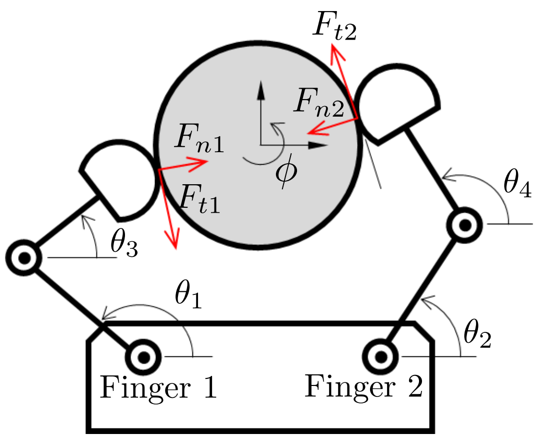


Figure 3.3: Schematic diagram of the robot system with data variables.

3.2 GMM/GMR

Among many approaches to solving LfD problem, GMM/GMR approach offers an advantage of resulting smooth generalized trajectory and not requiring many training samples. Another noticeable advantage of using probabilistic approach is that any outliers can be removed from the demonstration [23].

3.2.1 Training with GMM

The recorded data from the demonstration is then modeled with a GMM. As mentioned in Section 2.2.1, we select K number of Gaussians to represent the entire motion profile parameterized by $\Theta^{\text{GMM}} = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$. To learn the parameters, we follow the EM-algorithm mentioned in Section 2.2.2 with concatenated demonstration dataset shown as $\xi = \{\xi_{1:T}^{m}\}_{m=1}^{M}$.

Selecting Number of Gaussians

The choice of K, the number of Gaussians, is critical when finding a probabilistic model of the given datapoints. Selecting small number of Gaussians results in a model that does not represent the motion profile correctly, and a large number of Gaussians may cause the model to over-fit the data, which may result in losing robustness in the generalized motion profile. Popular methods of selecting a proper number of Gaussians include Akaike information criterion (AIC) [2], Bayesian information criterion (BIC) [82] and the consistent AIC (CAIC) [18]. These selection methods are described as the following equations.

$$\begin{aligned}\text{AIC} &= -2\log L + 2\nu \\ \text{BIC} &= -2\log L + \nu\log(N) \\ \text{CAIC} &= -2\log L + \nu\left(\log(N) + 1\right)\end{aligned} \tag{3.1}$$

where L is the log-likelihood of the model against the training set, ν is the free parameters in the model selection and N is the total number of the training datapoint. From the equations we can observe that all the expression on the right side have two terms, the first term for measuring the goodness-of-fit and the second term for penalizing model complexity. All the listed methods showed consistent performance over small number of free parameters (ν), however as the model complexity grows the performance fails [18]. Hence, for our purpose we use a slightly modified version of BIC method by introducing a scaling factor (λ) to the second term, which can be shown as:

$$\text{BIC}_\lambda = -2\log L + \lambda\nu\log(N) \tag{3.2}$$

where $\nu = (K(D^2 - D)/2 + 2D + 1) - 1$, which is the number of free parameters in the selected number of Gaussian mixtures. The λ value was chosen to be 0.01 in our work.

Temporal Alignment via DTW

The time alignment of both position and force can play a crucial role when performing nonprehensile manipulation. This is because the end-effector force must be applied to the object at the right position of the robot fingers to maximize the effectiveness of the nonprehensile manipulation. Hence, it is very important to align the demonstrated trajectories in time through the DTW algorithm as shown in Section 2.2.4. Since a single DTW algorithm cannot align more than two time series at the same time, we set a single reference trajectory and perform DTW multiple times with the rest of the trajectories. We evaluate the effectiveness of DTW by performing GMM/GMR with and without DTW on the same demonstrated datapoints.

3.2.2 Motion Profile Reproduction with GMR

After the demonstration data are modeled, smooth motion and force profiles are found using GMR. As explained in Section 2.2.3, given a joint probability distribution of training data, $p(\xi_i, \xi_\theta, \xi_f)$, the GMR estimates the conditional probabilities for the position $(p(\xi_\theta|\xi_i))$ and the force $(p(\xi_f|\xi_i))$. Then, taking the expectation of the conditional probability results in smooth motion and force profiles along the time space.

3.3 HMM/GMRa

Another popular method of solving LfD system is modeling the demonstration data with HMM. In this section we show how the same demonstrated datapoints can be modeled without time index which results in a probabilistic model that compensated both temporal and spatial variations. Then we show how HMM can also be used for reproducing the generalized motion profile through modified version of GMR referred as GMRa.

3.3.1 Training with HMM

Similar to GMM we select K number of Gaussians to parameterize the motion profile represented with $\Theta^{\text{HMM}} = \{\{a_{ij}\}_{j=1}^{K}, \Pi_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ as described in Section 2.3.1. Since HMM is capable of handling temporal variations, we do not need the time index term ξ_i so our datapoint vector reduces to $\xi_t^m = \left[(\xi_\theta)_t^m, (\xi_f)_t^m\right]^\intercal$. When training for HMM, we've divided the dataset into two, where one dataset corresponds to the left finger of the robot (finger 1) and the other dataset is for the right finger of the robot (finger 2) as labeled on Figure 3.3. This is based on our assumption that the correlation is small between the left finger dataset and the right finger dataset.

In the absence of time index, we can use the position information as the input and the force information as the output. Therefore, we end up with two HMM models $\Theta_{\text{left}}^{\text{HMM}}$ and $\Theta_{\text{right}}^{\text{HMM}}$ which are trained with datasets $\xi_{t,\text{left}}^m$ and $\xi_{t,\text{right}}^m$, respectively. For selecting the number of Gaussians, we use the similar criterion mentioned in Section 3.2.1. For calculating the likelihood, we sum up Equation (2.15) for all K Gaussian states.

3.3.2 Trajectory Reproduction with GMRa

The HMM has largely been used for recognizing the human demonstrated motion due to its capability of handling temporal and spatial variation. There have also been many attempts to reproduce trajectory from the HMM data. Specific examples include averaging approach [23], keypoint encoding [7], and spline fitting approach [19]. In our work, we simply extended the GMR for motion profile reproduction from HMM. With the trained HMM, the generalized motion profile is generated using GMRa algorithm shown in Equation (2.21). From the pre-defined motion profile as input variables, the corresponding end-effector forces are calculated.

3.4 Experimental Result

For nonprehensile spinning task mentioned in Section 1.4.2, we have collected total of 4 demonstration datasets ($M = 4$) with each having a trajectory length of $T = 3000$. Hence, the total datapoints of $N = M \times T = 12000$ are used for the training purpose. As we control the robot position by position tracking, we use the encoder readings directly to our position trajectory. However, the tactile sensor measures the contact forces at a fixed coordinate on the sensor, which is different from the contact force measured at the contact point. So, the original force readings are converted to normal and tangential components at the contact point. We utilized the geometrical relationship to find the contact points, but this may also be solved through vision camera. In learning the force profile, we only used the tangential force. There are several reasons for this: the task is limited to only rotation, hence the normal direction force does not play much role in spinning. Furthermore, since the object center is fixed, the normal force does not reflect the actual object force.

3.4.1 Gaussian Number Selection

To select the proper number of Gaussians, we conducted EM-calculation on the same demonstration dataset with varying the number of Gaussian models (K). We utilize the modified BIC calculation as described in Equation (3.2) and the change of BIC number respect to the varying number of Gaussians is shown in Figure 3.4. As we aim to select the number of Gaussians that well represents the training data with the minimum number of free parameters, we select K that corresponds to the lowest BIC number. Therefore, we have selected $K = 5$ as the modified BIC number. This tells that 5 Gaussian models are suitable to fit the given training data with a reasonable number of the free parameters. The selected number of Gaussians is used throughout all the experiments.

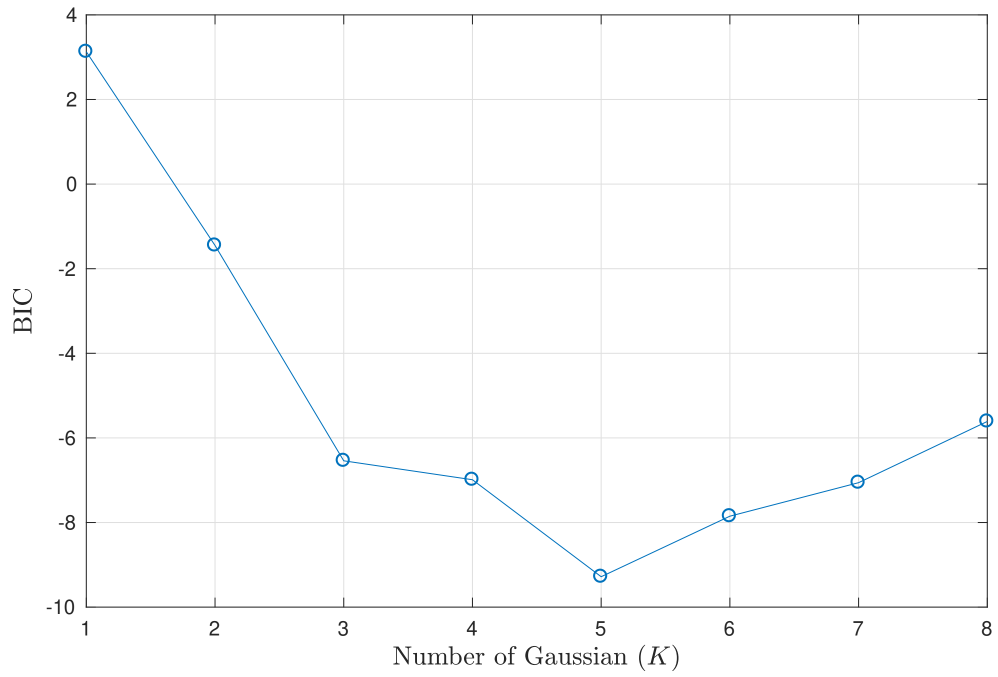


Figure 3.4: Number of Gaussian models vs modified BIC number

3.4.2 GMM for Nonprehensile Spinning Task

The following figures show how Gaussian mixture models are encoding the demonstrated motion profiles. Figure 3.5 shows the result of GMM with the raw demonstrated input and Figure 3.6 shows the result of GMM with the time-warped inputs (i.e. pre-processed with DTW). The demonstrated motion profiles are represented with dashed lines and the corresponding GMM is represented with the ellipses. By looking at both figures, one can see that the selected number of Gaussians ($K = 5$) successfully encode all the trajectory data. We can also see that pre-processing the data with DTW allows GMM to have smaller variances when encoding the same motion profile.

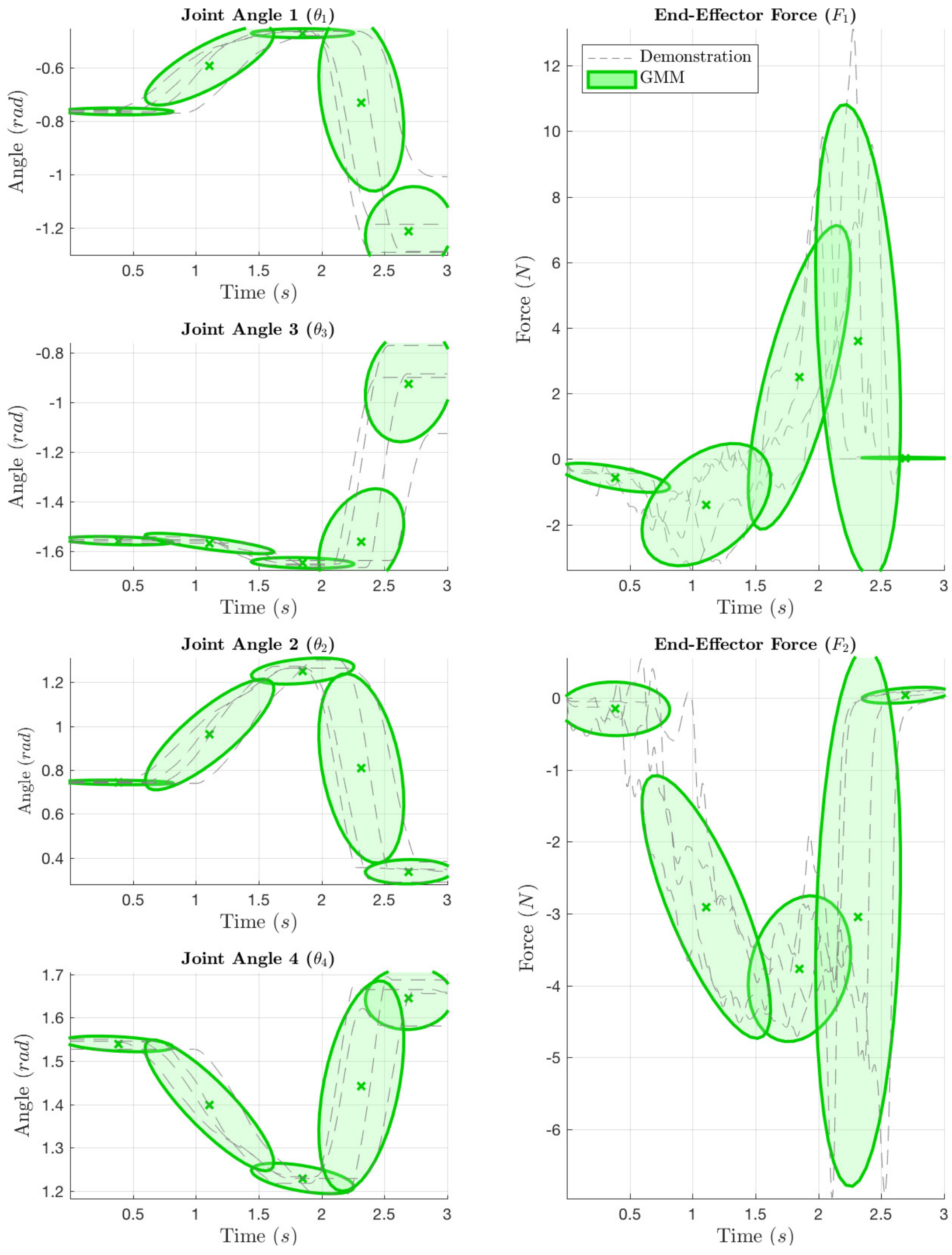


Figure 3.5: GMM

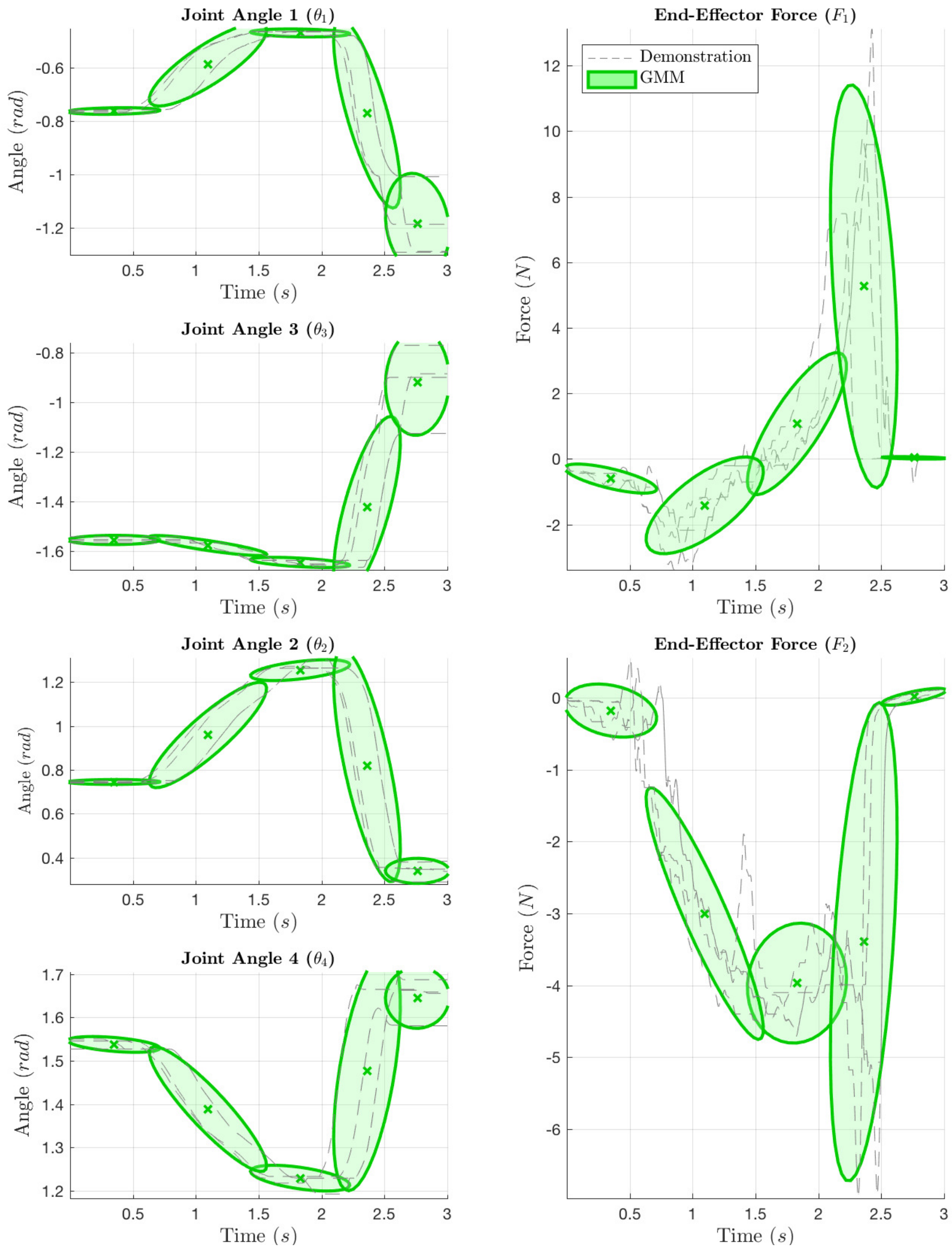


Figure 3.6: GMM+DTW

3.4.3 HMM for Nonprehensile Spinning Task

As described in Section 3.3.1, two HMM models $\Theta^{\text{HMM}}_{\text{left}}$ and $\Theta^{\text{HMM}}_{\text{right}}$ were trained with given demonstration data. Figures 3.7 and 3.8 show the HMM models for left and right finger, respectively. For each figure, it shows the Gaussian states between the joint angles and end-effector force on the left side of the figure. Instead of using the time index as we did in the previous section with GMM, we represented the same motion profile only with joint angles and the end-effector forces. Specifically, the joint angles are used as the input datapoints and end-effector forces are treated as the output datapoints. Hence in the plots below, we have joint angles as the x-axis and end-effector forces as the y-axis. Also, the right side of Figures 3.7 and 3.8 shows the HMM state transition between the states that are color matched with the figures shown in the left. The opacity of the arrow shows the transition probability of between the states (i.e. darker arrow corresponds to the higher probability of transition). To simplify the convergence, we have forced the transition probability model to be left-to-right topology.

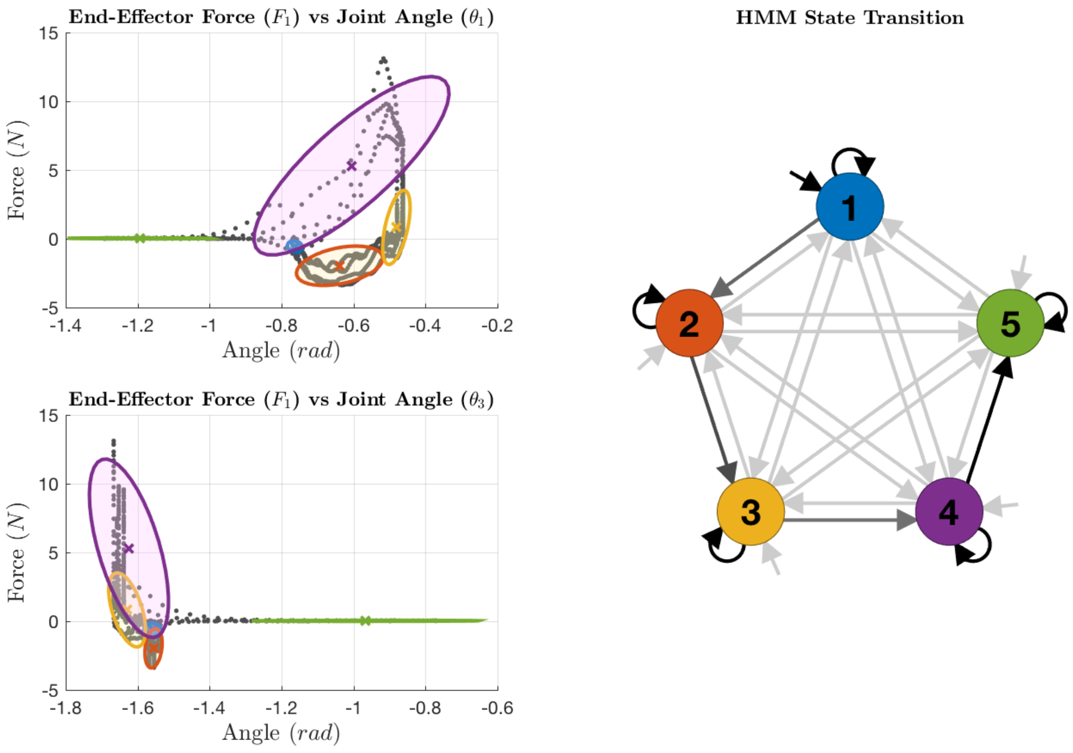


Figure 3.7: HMM for left finger

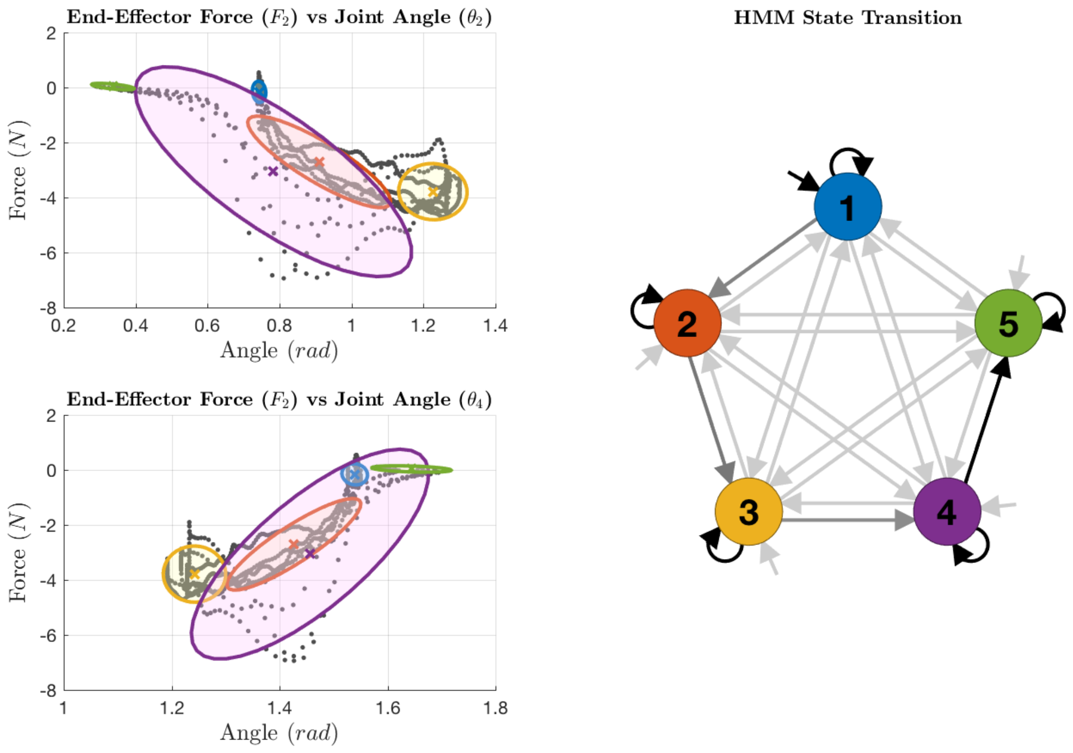


Figure 3.8: HMM for right finger

3.4.4 Generalized Motion Profile for Nonprehensile Spinning Task

From the encoded models ($\Theta^{\text{GMM}}, \Theta^{\text{GMM+DTW}}, \Theta^{\text{HMM}}_{\text{left}}, \Theta^{\text{HMM}}_{\text{left}}$) calculated in the previous sections, generalized motion profiles are reproduced as described in Section 3.2.2 and 3.3.2. For GMMs the input to the regression model was the time sequence ($t_{0:3s}$). For the HMM models we had to use joint angles as an input to the model to retrieve the corresponding finger-tip force output; hence, we have used generalized joint angles generated by GMM for the regression calculation. Results are summarized in Figure 3.9.

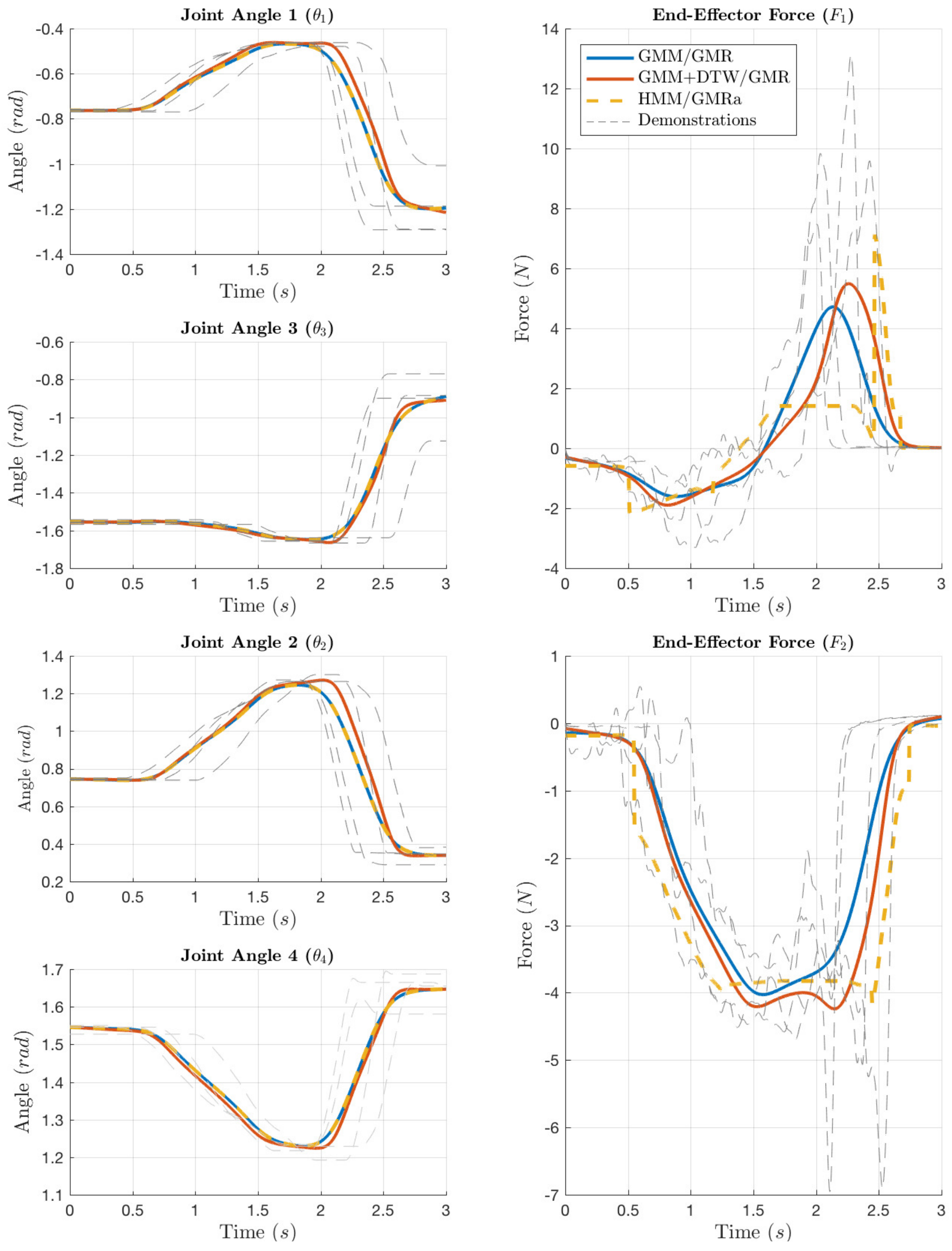


Figure 3.9: Reproduced generalized trajectories

3.4.5 Discussion

In this section, we were able to collect $M = 4$ demonstrations to calculate multiple generalized motion profiles for nonprehensile spinning task through different LfD methods. The length of the motion profile is $T = 3,000$ time steps, which is equivalent to $3s$ as each time step is $0.001s$. For the model training, the iteration count for convergence of GMM, GMM+DTW and each HMM models (left and right) took 77, 75, 22 and 13, respectively. By comparing the two figures Figure 3.5 and 3.6, one can see that the covariance of the GMM gets smaller with time-warped input, which implies that the temporal variation has been compensated by utilizing DTW process. As we can see from the results in Figure 3.9, the generalized trajectories all share the similar behaviour for both angle and force profiles. One key point to note is that using HMM, the impulsive behaviour of the spinning is well-established through the reproduction stage. The narrow peak on F_1 profile illustrates this, which is compared to GMM/GMR methods. The performance of each motion profile will be compared in the next section when we use them for experiment with the same controller.

Chapter 4

Reproduction of Nonprehensile Spinning Task

In this section, we develop the controller to implement the generalized motion profiles that were generated in the previous chapter. As nonprehensile manipulation task utilizes both position and force profiles, we have selected to use combination to two controllers to control the entire system. Specifically, we have implemented the feedback control for the position profile and the feedforward control for the force profile.

4.1 Controller Design

After successful reproduction of generalized motion profiles in Chapter 3, these motion profiles are applied to a robot hand as control inputs, which are denoted by θ_d and F_d. Since the motion is represented with the joint angles, a simple PD feedback controller is capable of generating the desired finger motion. To perform nonprehensile manipulation, additional end-effector forces are often needed. Hence, we perform hybrid control by adding feedforward term to the PD position controller. To provide the desired end-effector force during the manipulation, we use the Jacobian transpose control to calculate the required feedforward portion of the joint torque. Thus, the control input (τ_{input}) to the robot hand is given as

$$\tau_{input} = \tau_{ff} + \tau_{fb}$$

where τ_{ff} and τ_{fb} are the feedforward and the feedback control terms, respectively, which are given by

$$\begin{aligned}\tau_{ff} &= J^{\intercal}(\theta)F_d \\ \tau_{fb} &= K_P(\theta_d - \theta) + K_D(\dot{\theta}_d - \dot{\theta})\end{aligned} \tag{4.1}$$

where $J \in \mathbb{R}^{4\times4}$ denotes the Jacobian matrix, and $K_P \in \mathbb{R}^{4\times4}$ and $K_D \in \mathbb{R}^{4\times4}$ are diagonal matrices that contain the proportional and derivative gains for each joint. The control input $\tau_{input} \in \mathbb{R}^4$ is a vector that is fed into the motor of each joints. The block diagram of the controller for the nonprehensile manipulation is depicted in Figure 4.1.

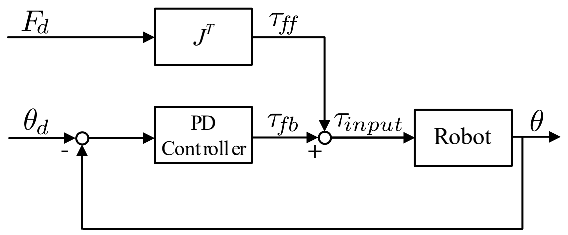


Figure 4.1: Controller for the nonprehensile manipulation

4.2 Experimental Result

We performed several experiments to validate the performance of the proposed controller design to the custom-built two finger robot introduced in Section 1.4.1. After the controller design verification, the generated motion profiles for the nonprehensile spinning task from different LfD framework are fed into the robot to compare the performance of each generation methods. We first evaluate the performance of feedback and feedforward controller separately. This seemed to be a valid approach for evaluating the controller as the control input to the robot hand is the superposition of the two. Once the performance of each controller design is verified, the reference control signals established in Section 3.4.4 are used as the controller input signals and performance of each nonprehensile manipulation task will be compared.

4.2.1 Valication of Feedback Controller for Position Tracking

The position feedback controller is a simple PD controller that utilizes the error between the desired and reference angles and their derivatives. The selected controller gains are

$K_P = \text{diag}(20, 20, 20, 20)$ and $K_D = \text{diag}(1500, 1500, 1500, 1500)$. The chosen reference signal is a smooth spline profile that allows robot to pre-shape the grasp position from its zero position. The experimental results are shown in Figure 4.2. It includes the plots of desired and actual joint angle response plotted together and a single plot to show the errors between them. We can see that the position tracking is well performed as the error is in the range of 2 to 3 degrees. By adding the integration term, the steady state error can be reduced, but the further controller design was not implemented as the performance was suitable for our use.

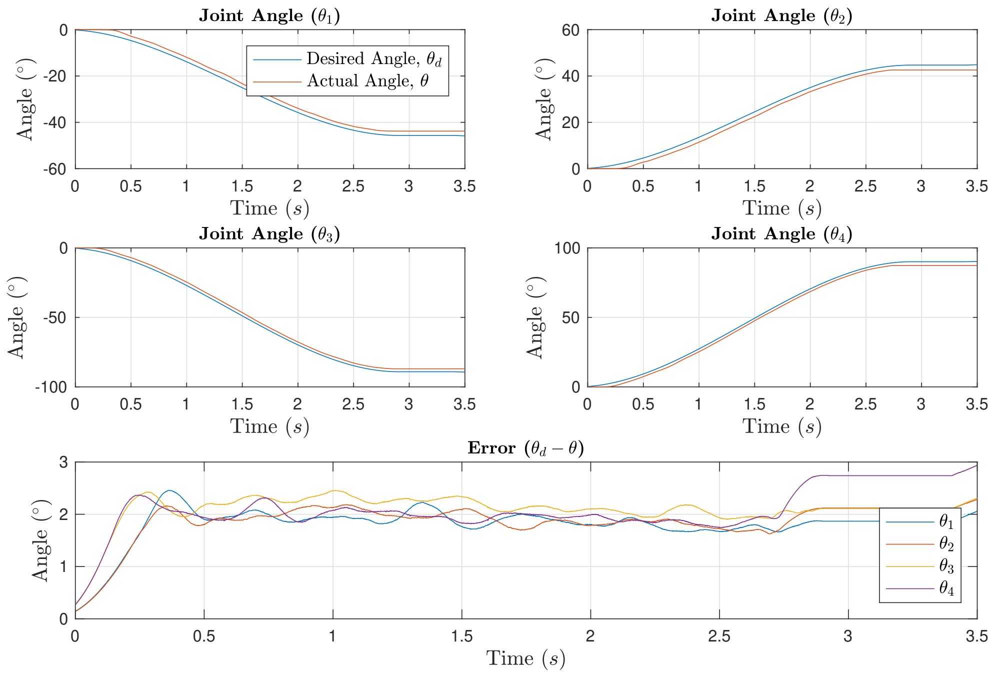


Figure 4.2: Position controller performance

4.2.2 Validation of Feedforward Controller for End-Effector Force

Similar approach was used to verify the performance of the feedforward controller. The robot's finger tip was placed against the stationary object and the normal directional force was applied towards the object in a stepwise command (i.e. a series of step signals that increase every $1s$ by $0.5N$). The experimental results are shown in Figure 4.3. The top figures show the system responses against the desired command signal and the bottom figures illustrate the error responses. The average error magnitude sits around $0.2N$, the performance of which is adequate for our purpose.

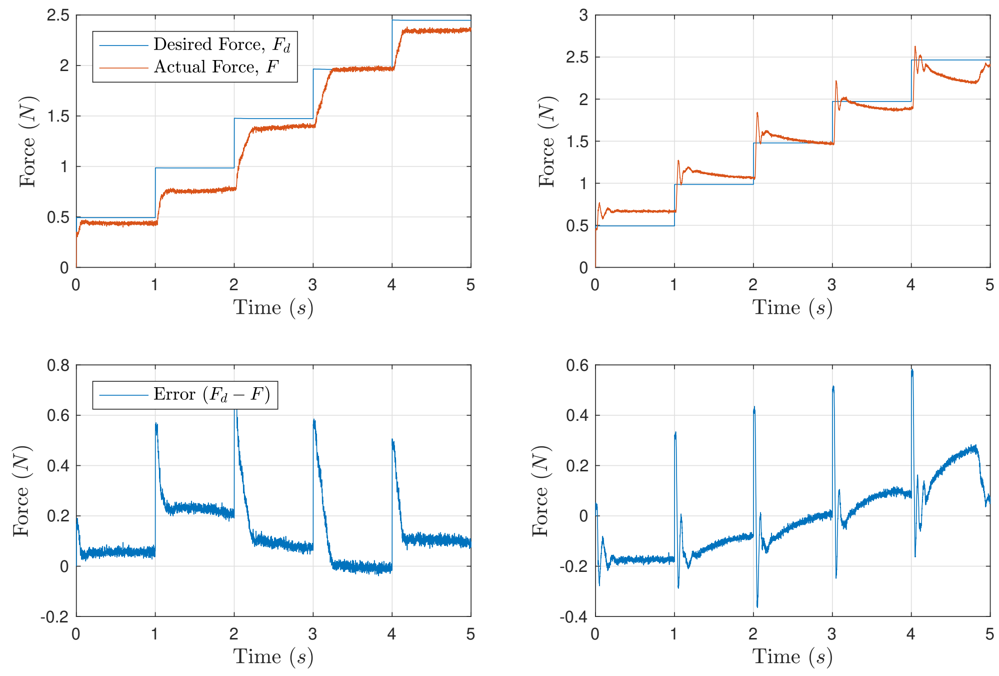
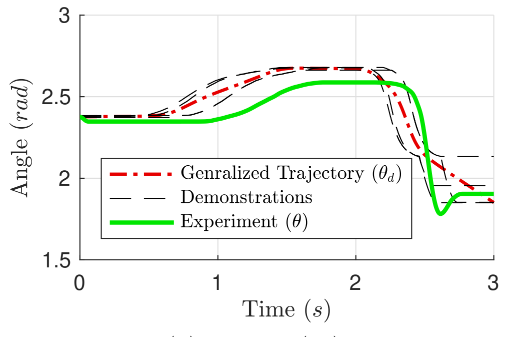


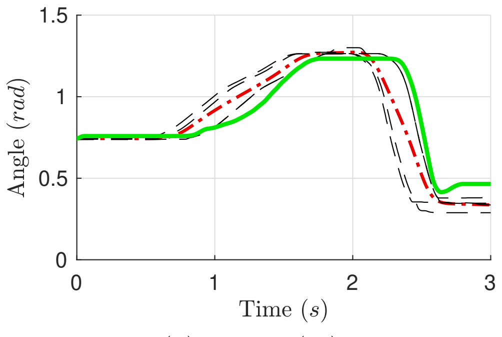
Figure 4.3: Stairway force step for feedforward controller verification

4.2.3 Evaluating Reproduced Generalized Motion Profiles

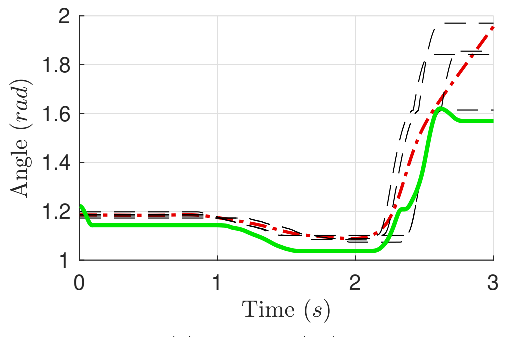
The reference control signals established in Section 3.4.4 are used as the controller input signals in this section to compare the performance of each motion profile. To describe the performance of the proposed controller in nonprehensile manipulation task, the reproduced generalized motion profiles and the actual experimental output are shown in Figure 4.5 (force) and Figure 4.4 (position). For the demonstration purpose, we are only showing the results from GMM+DTW/GMR on these figures as the other methods result a similar behaviour. Figures 4.5a, 4.4a and 4.4c correspond to Finger 1 while Figures 4.5b, 4.4b and 4.4d correspond to Finger 2. (See Figure 3.3). The thin dotted lines represent the demonstration data from kinesthetic teaching (4 of them in each plot). The thick dash-dot line is the generalized trajectory obtained through GMM+DTW/GMR. The generalized trajectories in Figures 4.5 and 4.4 are implemented to the controller in Figure 4.1, and the resulting signals are depicted as thick solid lines in Figures 4.5 and 4.4. Note that the



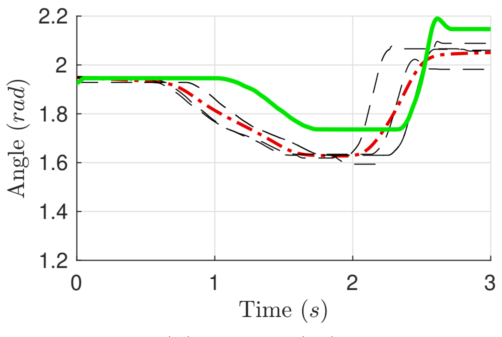
(a) Joint 1 (θ_1)



(b) Joint 2 (θ_2)

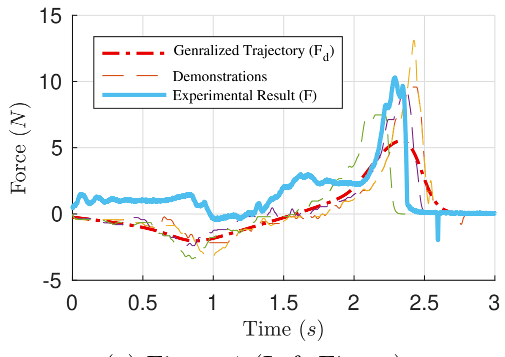


(c) Joint 3 (θ_3)

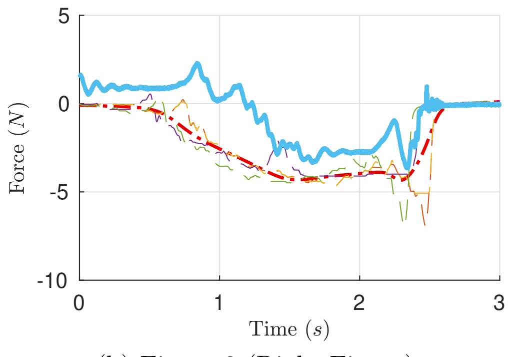


(d) Joint 4 (θ_4)

Figure 4.4: GMR trajectories for position variables.



(a) Finger 1 (Left Finger)



(b) Finger 2 (Right Finger)

Figure 4.5: GMR trajectories for tangential forces.

fingers are in contact with the object from around $0.8s$ until around $2.7s$.

The performance of the reference input tracking for position and force is not as good as that shown in Figure 4.2 and Figure 4.3 although the desired nonprehensile spinning motion has been achieved by the robot. The main source of error for position tracking may be attributed to ignoring any kinematic constraints during the learning process and a possible source of error for the force tracking is postulated to come from subtracting out the tangential force created by the motion of the robot. Although the robot was not able to reproduce expert level of nonprehensile manipulation task, the robot showed a significant improvement over classical manipulation approach based on regrasping.

We have collected 10 trials for each motion profiles and recorded the total rotated angle from the starting position. We used the total rotated angle as the performance measure to compare how closely the generated profiles follow the expert's demonstration. The collected data are summarized in Table 4.1 and Figure4.6.

Table 4.1: Performance of nonprehensile spinning task without catching

Motion Profile Generation Method	Median Angle	Maximum Angle	Minimum Angle
GMM/GMR	170°	183°	157°
GMM+DTW/GMR	206°	221°	196°
HMM/GMRa	235°	256°	216° (outlier)

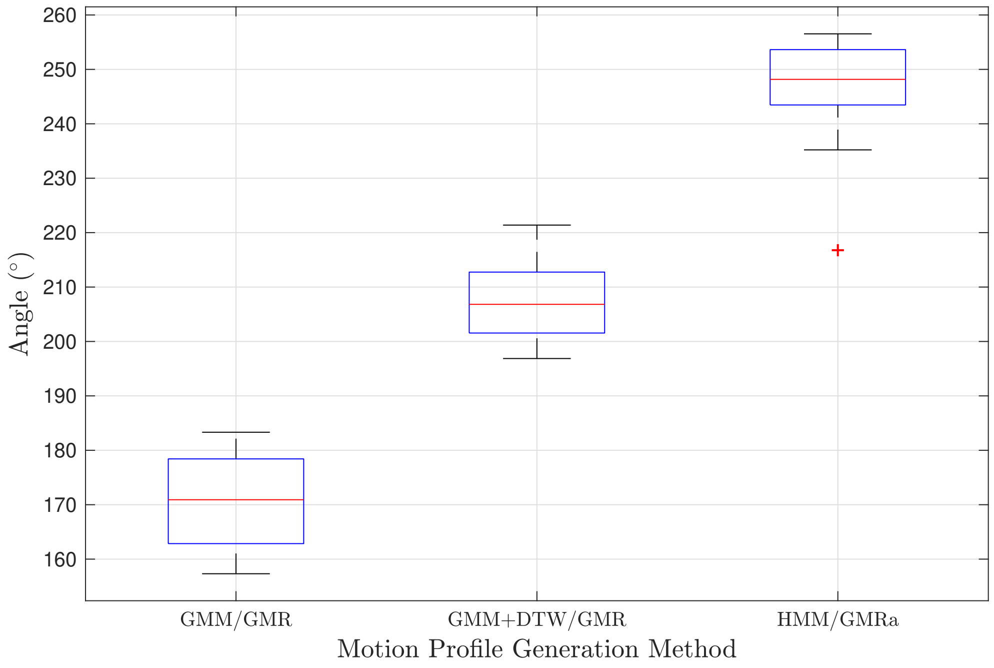


Figure 4.6: Spinning result

Discussion

Through the experiment, we have clearly verified that the proposed motion profile generation methods and the controller design allow robot to perform an object spinning task much better than the classical spinning method that utilizes grasp-stability. More specifically, through the new approach, robot can spin the object much faster and at greater angles than the traditional method of spinning which keeps the contact between object and robot's fingertips (approximately $\sim 25°$).

Although the human expert's demonstration resulted object spinning of about full 2 rotations, none of the experimental results was performing the nonprehensile spinning task at the human-level. Hence, the criteria for comparison between the experimental results was simply which motion turned the greater angle. From the experiment, it was shown that the each LfD framework provided different results for the desired task. The result shows that compensating not only the spatical variation but also the temporal variation provides much better performance for the desired task. HMM/GMRa showed the best performance as this framework is independent of the time index.

Chapter 5

Dynamic Catching through Parameter Learning

Since we can spin the object at a much greater angle in a short amount of time, we also need to catch the object at a right time to complete the manipulation goal of object spinning at a desired angle. Finding the right timing will be discussed in this section. We utilize the theory from the iterative learning control (IRC) to iteratively find the optimal timing to trigger the stopping comman to the robot in order to stop the spinning object at a desired angle.

5.1 Main Control Parameter

After the robot makes an impulsive spin, the visual feedback is used to stop the spinning object at a desired angle ϕ_d. Stoping action is made by dynamic catching (or an impulsive grasping action) by sending a simple closing command for both fingers. Denoting by $\bar{\phi}_t$ the rotation angle of the object measured by the vision at the time step t, the time to trigger the regrasping action, denoted by t_{tr}, can be determined by the following equation;

$$t_{tr} = t \quad \text{such that} \quad \bar{\phi}_t = \phi_d - \omega\,(T_v + T_a) \tag{5.1}$$

where ω is the angular velocity of the object, T_v is the sample time of vision sensor, T_a is the time duration of both fingers completing their closing action. Here, we made an assumption that the object is rotating quasi-statically, i.e. at a relatively constant angular speed (which holds true for the experiments that will be shown in the next section). However, it may also be computed online by the numerical differentiation of successive vision data for the orientation of the object.

5.2 Iterative Learning Control

From Equation (5.1), T_a is the only unknown parameter. We plan to adapt the idea of iterative learning control (ILC) method to learn the parameter from repetitive experiments. As the name suggests the key idea of ILC is learning through a predetermined hardware repetition [1]. Among various ILC methods, we are using the iterative learning scheme of "Arimoto-type"[6], given by

$$u_{k+1} = u_k + \Gamma e_k \tag{5.2}$$

where index k is the iteration number, u is the control signal, Γ is the learning gain and e_k is the error between the desired trajectory and the actual value (i.e. $e_k = x_d - x$). For our purpose, we do not seek for the control signal directly, but we prefer to learn the unknown parameter T_a. Therefore, we modify the Equation (5.2) to fit our purpose.

$$T_{a,k+1} = T_{a,k} + \Gamma \left(\phi_d - \bar{\phi}_{t,k}\right) \tag{5.3}$$

5.2.1 Experimental Result

To learn the parameter, we have selected nonprehensile spinning manipulation with desired angle of 90° to repetitively perform. By utilizing Equations (5.1) and (5.2), the unknown parameter T_a was learned. We first fixed the unchanging parameters such as $\omega = 209°/sec$, $T_v = 0.08s$ and $\Gamma = 0.001$. The angular speed of the object is set to a fixed point because the object's angle changes almost linearly right after the spinning action is made, and this remains linearly until the object spun about 200°. Hence, the angular speed can be kept fixed if the nonprehensile spinning task is done within 200°. The red line in Figure 5.1 shows the linear region of the object angle change. The slope of this line is $209°/sec$.

Through the iteration algorithm described in Equation (5.2), T_a was found to be $0.129s$ in 12 iteration steps. The change in the parameter value and the error indicating the control performance is shown in Figure 5.2. The parameter learning was stopped when there was no longer significant changes to the learned parameter. In order to verify whether the parameter was learned correctly, the nonprehensile spinning tasks were performed at various desired angles and the results are displayed in the next section.

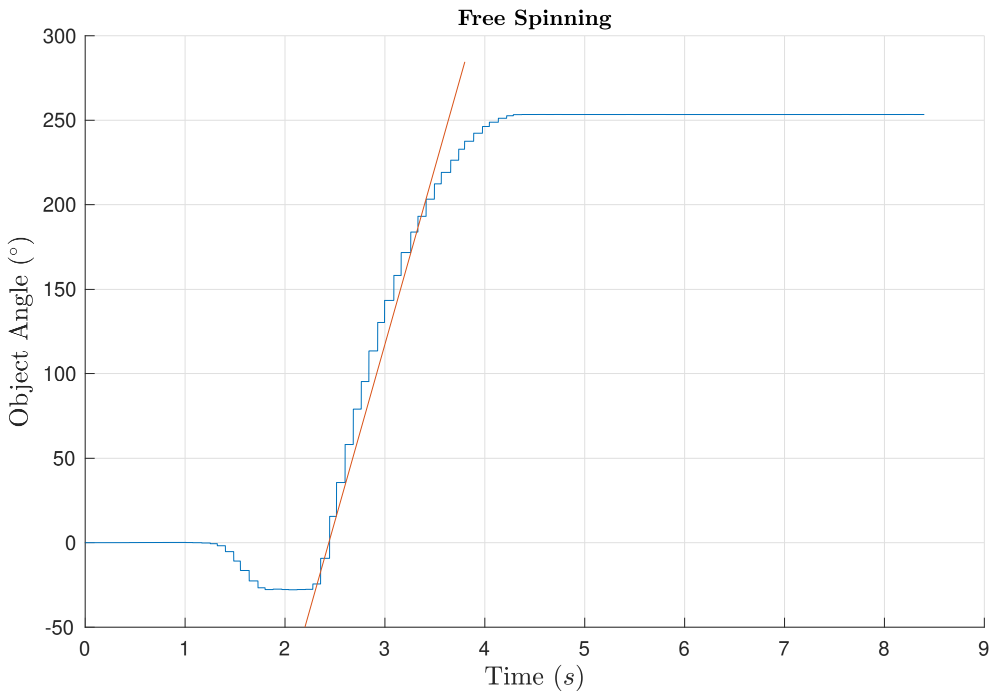


Figure 5.1: Object angles after free spinning

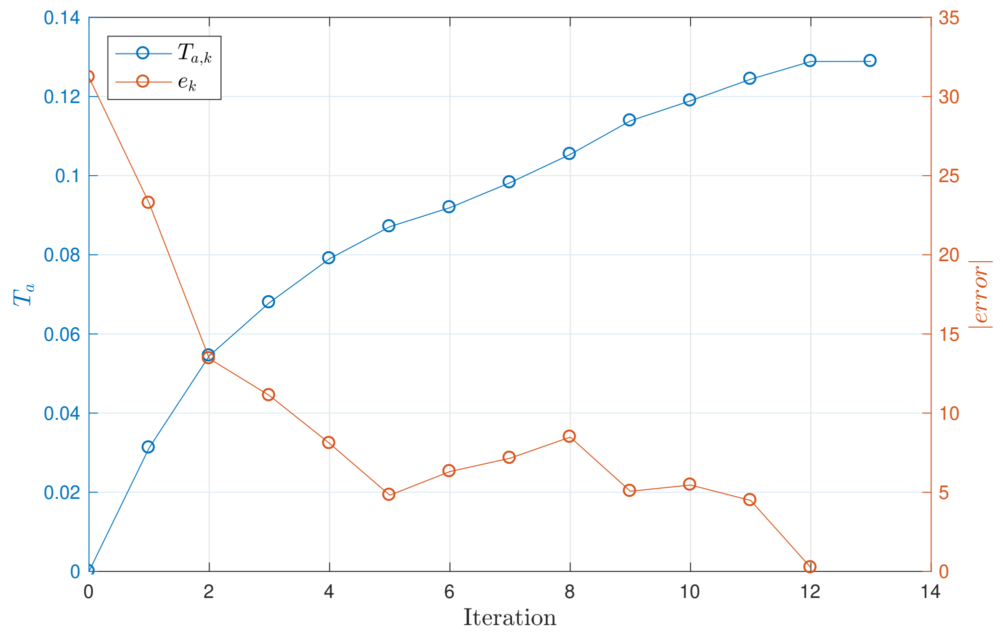
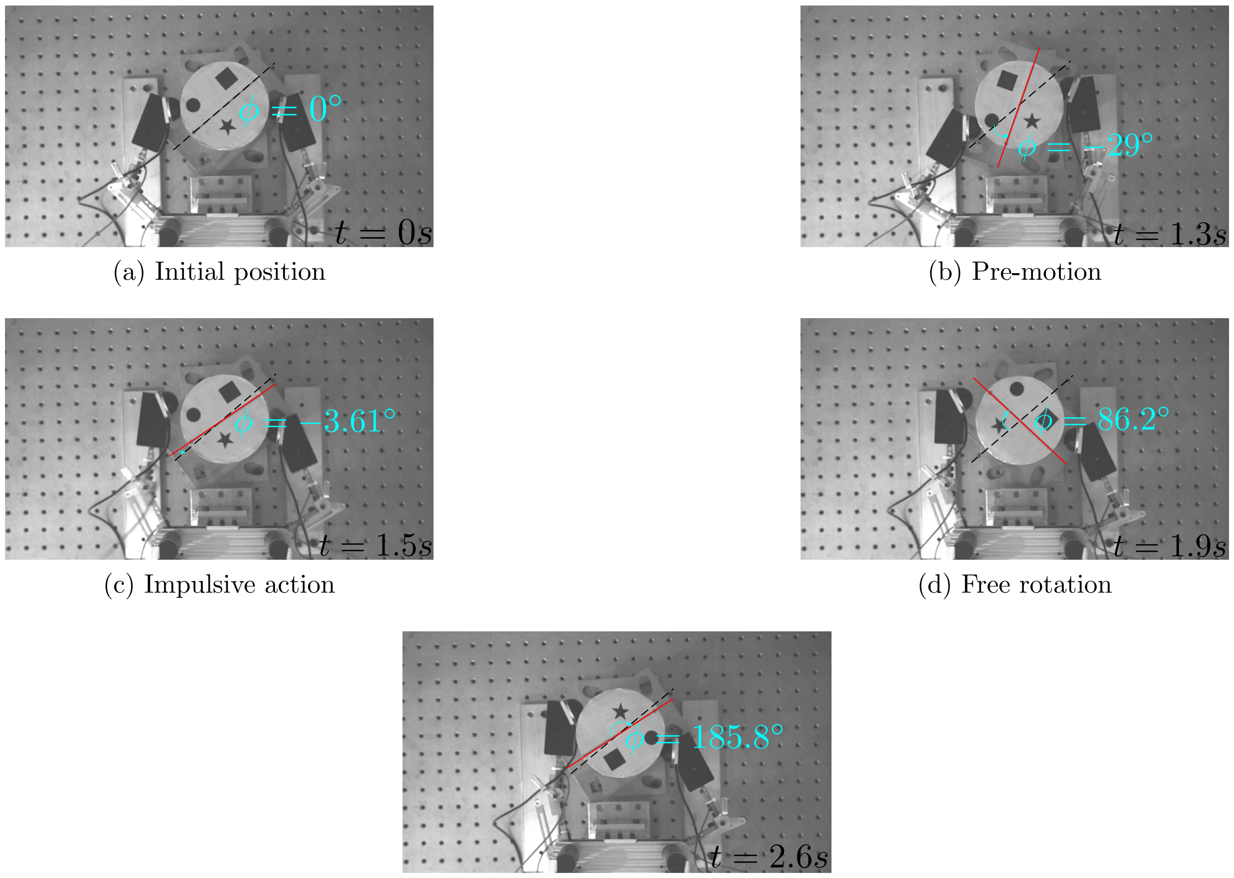


Figure 5.2: Iterative parameter learning for T_a

5.3 Complete Nonprehensile Spinning Task Experimental Result

The experiments are conducted to realize fast rotation of the object beyond the range achievable by the single step stable grasping manipulation through nonprehensile spinning method. Specifically, we chose three different desired rotation angles, $\phi_d = 90°, 120°$ and $180°$. Note that the same impulsive spinning move can be used for any ϕ_d because the particular values of ϕ_d can be achieved by different values of t_{tr} in Equation (5.1) (i.e. the time to engage in the stopping action). Thus, we obtained only one set of generalized trajectories of position and force for the spinning move. Among several generalized trajectory from the previous section, we've selected the fastest one which is from the HMM/GMRa framework.

To illustrate the whole procedure of the nonprehensile spinning manipulation, the snapshots for $\phi_d = 180°$ is shown in Figure 5.3, which are taken by the vision camera. The five snapshots in Figure 5.3 delineate different stages of motion that constitute the complete nonprehensile spinning manipulation. As shown in Figure 5.3a, at first, both fingers are moved to the object so that they grasp it, symmetrically, at the rightmost and the leftmost contact points located along the horizontal line passing through the object center. Then,



(a) Initial position (b) Pre-motion

(c) Impulsive action (d) Free rotation

(e) Catching

Figure 5.3: Snapshots of the manipulation for $\phi_d = 180°$.

both fingers are engaged in the wind-up motion (called the pre-motion) as shown in Figure 5.3b, by which the object is slowly rotated counterclockwise through a finger tip rolling to its maximum range ($\sim -29^\circ$ in this case) to be ready for the impulsive spinning move. Then, both fingers quickly fling the object to the opposite (i.e. clockwise) direction through the impulsive spinning as shown in Figure 5.3c which depicts the snapshot of the moment when both fingers have just spun the object with a quick motion. After that, a period of free motion follows as shown in Figure 5.3d. During this period, the object undergoes a constant speed rotation while the fingers remain detached from the object. Finally, as shown in Figure 5.3e, when the object is getting close to his desired rotation ϕ_d (which is 180° in this case), the fingers are engaged in the fast catching action to stop the rotation according to Equation (5.1).

The same task can be done with the classical re-grasping method. The task's objective was to rotate the object about 180° in counterclockwise direction. Due to the kinematic constraints, the robot is only capable of rotating the object about 25° while maintaining the grasp stability. Hence, to perform the 180° rotating task, robot is required to perform six regrasping motion. The resulting dynamic dexterous manipulation shows that it has a great advantage over regrasping when considering the manipulation time. The nonprehensile manipulation only requires less than $2.5s$ whereas, the regrasping takes about $12s$. The accuracy of the catching while the object is spinning is another concern to achieve the task objective. However, this can be easily dealt by performing the nonprehensile manipulation then performing the fine adjustment of the object while maintaining the stable grasping.

We conducted 10 repeated trials for each ϕ_d to evaluate the performance of the robotic hand. The performance of the nonprehensile spinning manipulation is compared with that of finger regrasping method in Tables 5.1 and 5.2. For all three values of ϕ_d, the nonprehensile manipulation completed the rotation task within 2.5 seconds as shown in Table 5.1. On the hand, it takes significantly longer time if we use the classical finger gating. Specifically, it takes about 7.5 seconds to rotate the object by 90 degrees as shown in Table 5.2. This is because the robotic fingers need to go through 4 rounds of regrasping before completing the desired rotation. Of course, the time of completion increases in portion to ϕ_d, e.g. it requires 11.78 seconds to complete 180^{circ} of rotation. Although the angle errors appear smaller for finger gating, the error values in Table 5.1 are insignificant because we can easily compensate for all these errors by incorporating a small rolling manipulation followed by the catching motion (which will take only a fraction of a second).

Table 5.1: Performance of nonprehensile manipulation to generate the desired angle ϕ_d

Desired Angle	Average final angle	Average time of completion	Mean of angle error
90°	91.63°	$2.007s$	1.63°
120°	119.4°	$2.1739s$	0.6°
180°	182.1°	$2.4612s$	2.1°

Table 5.2: Performance of finger gating to generate the desired angle ϕ_d

Desired Angle	Average final angle	Average time of completion	Mean of angle error
90°	91.03°	$7.520s$	1.03°
120°	120.8°	$10.608s$	0.8°
180°	180.4°	$11.782s$	0.4°

Chapter 6

Conclusion

6.1 Conclusion

In this thesis, we proposed the use of multisensory learning from demonstration (LfD) framework to enable nonprehensile (or dynamic dexterous) manipulation tasks with a multifingered robotic hand. The main idea is to produce generalized trajectories for both position and force using probabilistic LfD technique. Among various probabilistic LfD techniques, we have used GMM/GMR, GMM+DTW/GMR and HMM/GMRa. From the same demonstration data, we have trained different probabilistic models and retrieved the generalized motion profile through regression step. We demonstrated the performance of each LfD technique and it was shown that the HMM/GMRa performs most closely to the expert's demonstration among others. This is due to the capability of compensating for both spatial and temporal variations in the training data. Then, we have completed the task goal of spinning object at desired angle by incorporating dynamic catching action. The control parameter required for dynamic catching was learned through iterative method.

The proposed overall manipulation technique was verified through experimental tests with two-finger planar robotic hand which is controlled to spin the circular object fast and accurately. The proposed technique is also compared with the classical regrasping method based on force closure and finger gating, which showed the superiority of our approach in terms of speed and agility. We believe that the proposed framework can be generalized to other nonprehensile manipulation tasks that involve more complex dynamics between the object and the manipulating bodies.

6.2 Recommendation

As a future work, a simulation environment can be established for the robot using dynamic simulation tool such as V-REP [75] or MuJoCo [87]. Through the simulation environment more complex nonprehensile manipulation tasks can be learned and tested without the need of creating a different testing environment. Creation of such simulation environment can be extended to performing reinforcement learning (RL) to the system. The simulation environment solves the issue of physical robot getting damaged by performing repeated experiments. Through RL, robot can learn to perform the nonprehensile manipulation at a human-like level through learning from its own experience. The converging time (or learning time) of the RL can be reduced by starting from a good initial motion profile. This is where the current research can be useful for; the learned motion profile from LfD framework can be used as a good initial motion profile of RL framework.

This work can be also extended to learn prehensile manipulation tasks by utilizing both position and force sensor observations. Through this, robot can learn new tasks without the need to mathematical modeling for the task. Also, after learning several low-level manipulation skills, high-level task learning can be applied to learn more complex manipulation task from the expert's demonstration. Future tasks involves task such as pushing or ball throwing that have more than one degree of freedom as we demonstrated in our research. We believe, that our LfD framework is capable of generalizing various well-demonstrated expert information that is provided to robot.

References

[1] H. Ahn, Y. Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1099–1121, Nov 2007.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[3] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *ACM/IEEE Int. Conf on Human-Robot Interaction (HRI)*, pages 391–398, March 2012.

[4] P. K. Allen, A. T. Miller, P. Y. Oh, and B. S. Leibowitz. Using tactile and visual sensing with a robotic hand. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 676–681, April 1997.

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

[6] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems. In *IEEE Conf. on Decision and Control*, pages 1064–1069, Dec 1984.

[7] T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 40–47, Dec 2006.

[8] C. G. Atkeson, J. G. Hale, F. Pollick, M. Riley, S. Kotosaka, S. Schaul, T. Shibata, G. Tevatia, A. Ude, S. Vijayakumar, E. Kawato, and M. Kawato. Using humanoid robots to study human behavior. *IEEE Intelligent Systems and their Applications*, 15(4):46–56, July 2000.

[9] Hyunki Bae, Soo Jeon, and Jan P. Huissoon. Vision and force/torque integration for realtime estimation of fast-moving object under intermittent contacts. *ROBOMECH Journal*, 3(1):15, Jul 2016.

[10] G. Bätz, A. Yaqub, Haiyan Wu, K. Kühnlenz, D. Wollherr, and M. Buss. Dynamic manipulation: Nonprehensile ball catching. In *Mediterranean Conf. on Control and Automation (MED)*, pages 365–370, June 2010.

[11] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 41(1):164–171, 02 1970.

[12] Jacob Benesty, M. Mohan Sondhi, and Yiteng (Arden) Huang. *Springer Handbook of Speech Processing*. Springer-Verlag, Berlin, Heidelberg, 2007.

[13] A. Bicchi. Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662, Dec 2000.

[14] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 348–353, April 2000.

[15] Antonio Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research*, 14(4):319–334, 1995.

[16] Zeungnam Bien and Jian-Xin Xu, editors. *Iterative Learning Control: Analysis, Design, Integration and Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[17] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[18] Hamparsum Bozdogan. Model selection and akaike's information criterion (aic): The general theory and its analytical extensions. *Psychometrika*, 52(3):345–370, Sep 1987.

[19] S. Calinon and A. Billard. Incremental learning of gestures by imitation in a humanoid robot. In *ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI)*, pages 255–262, March 2007.

[20] S. Calinon, D. Bruno, and D. G. Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3339–3344, May 2014.

[21] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics Automation Magazine*, 17(2):44–54, June 2010.

[22] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, April 2007.

[23] Sylvain Calinon. *Robot Programming by Demonstration - a Probabilistic Approach.* EPFL Press, 2009.

[24] B. Calli and A. M. Dollar. Vision-based precision manipulation with underactuated hands: Simple and effective solutions for dexterity. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1012–1018, Oct 2016.

[25] C. C. Cheah, H. Y. Han, S. Kawamura, and S. Arimoto. Grasping and position control for multi-fingered robot hands with uncertain jacobian matrices. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 3, pages 2403–2408, May 1998.

[26] Sonia Chernova and Andrea L. Thomaz. *Robot Learning from Human Teachers.* Morgan & Claypool Publishers, 2014.

[27] Chih-Yi Chiu, Shih-Pin Chao, Ming-Yang Wu, Shi-Nine Yang, and Hsin-Chih Lin. Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation*, 15(3):446 – 466, 2004.

[28] H. I. Christensen. A roadmap for u.s. robotics: From internet to robotics. Technical report, Robotics Virtual Organization, Oct 2016.

[29] A. A. Cole, P. Hsu, and S. S. Sastry. Dynamic control of sliding by robot hands for regrasping. *IEEE Transactions on Robotics and Automation*, 8(1):42–52, Feb 1992.

[30] John J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989.

[31] P. Poornesh L. Narayana Rao N. Arun Kumar D. Hari Hara Santosh, P. Venkatesh. Tracking multiple moving objects using gaussian mixture model. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(2):2231–2307, May 2013.

[32] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[33] Z. Deng, J. Mi, Z. Chen, L. Einig, C. Zou, and J. Zhang. Learning human compliant behavior from demonstration for force-based robot manipulation. In *IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, pages 319–324, Dec 2016.

[34] K. R. Dixon and P. K. Khosla. Trajectory representation using sequenced linear dynamical systems. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 3925–3930, April 2004.

[35] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *IEEE International Symposium on Robot and Human Interactive Communication*, pages 358–363, Sept 2006.

[36] Michael Erdmann. An exploration of nonprehensile two-palm manipulation: Planning and execution. In Georges Giralt and Gerhard Hirzinger, editors, *Robotics Research*, pages 16–27. Springer London, London, 1996.

[37] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin. Robot programming by demonstration (rpd): Supporting the induction by human interaction. *Machine Learning*, 23(2):163–189, May 1996.

[38] Michel Gilloux.

[39] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning Dexterous Manipulation for a Soft Robotic Hand from Human Demonstration. *ArXiv e-prints*, March 2016.

[40] Blake Hannaford and Paul Lee. Hidden markov model analysis of force/torque information in telemanipulation. *The International Journal of Robotics Research*, 10(5):528–539, 1991.

[41] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[42] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, Feb 2013.

[43] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1398–1403, May 2002.

[44] Tetsunari Inamura, Iwaki Toshima, and Yoshihiko Nakamura. Acquiring motion elements for bidirectional computation of motion recognition and generation. In Bruno Siciliano and Paolo Dario, editors, *Experimental Robotics VIII*, pages 372–381. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[45] S. James, A. J. Davison, and E. Johns. Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task. *ArXiv e-prints*, July 2017.

[46] S. Jeon. State estimation based on kinematic models considering characteristics of sensors. In *American Control Conference (ACC)*, pages 640–645, June 2010.

[47] Y. Jia and Y. Xue. Dexterous manipulation by two fingers with coupled joints. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3172–3179, May 2018.

[48] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal. Learning force control policies for compliant manipulation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4639–4644, Sept 2011.

[49] Eric R Kandel, James H Schwartz, Thomas M Jessell, Department of Biochemistry, Molecular Biophysics Thomas Jessell, Steven Siegelbaum, and AJ Hudspeth. *Principles of neural science*. McGraw-hill New York, 2000.

[50] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, Oct 2011.

[51] J. Kober, B. Mohler, and J. Peters. Learning perceptual coupling for motor primitives. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 834–839, Sept 2008.

[52] John W Krakauer and Pietro Mazzoni. Human sensorimotor learning: adaptation, skill, and beyond. *Current Opinion in Neurobiology*, 21(4):636 – 644, 2011.

[53] R. Krug and D. Dimitrovz. Representing movement primitives as implicit dynamical systems learned from multiple demonstrations. In *Int. Conf. on Advanced Robotics (ICAR)*, pages 1–8, Nov 2013.

[54] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, 31(3):330–345, 2012.

[55] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383, May 2016.

[56] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 177–184, May 2015.

[57] C. Lee and Yangsheng Xu. Online, interactive learning of gestures for human/robot interfaces. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 4, pages 2982–2987, April 1996.

[58] M. Li, Y. Bekiroglu, D. Kragic, and A. Billard. Learning of grasp adaptation through experience and tactile sensing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3339–3346, Sept 2014.

[59] Zexiang Li, Ping Hsu, and Shankar Sastry. Grasping and coordinated manipulation by a multifingered robot hand. *The International Journal of Robotics Research*, 8(4):33–50, 1989.

[60] Y. Lin, Shaogang Ren, M. Clevenger, and Y. Sun. Learning grasping force from demonstration. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1526–1531, May 2012.

[61] K. M. Lynch and M. T. Mason. Dynamic manipulation with a one joint robot. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 359–366, April 1997.

[62] Kevin M. Lynch and Matthew T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999.

[63] C.-C. Wang M. Tomizuka, H. Cheng. Sensing rich drive trains for modern mechatronic systems: first year progress report. In *Proc.SPIE*, volume 6529, 2007.

[64] R. R. Ma and A. M. Dollar. On dexterity and dexterous manipulation. In *Int. Conf. on Advanced Robotics (ICAR)*, pages 1–7, June 2011.

[65] H. Maekawa, K. Tanie, and K. Komoriya. Tactile sensor based manipulation of an unknown object by a multifingered hand with rolling contact. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 743–750, May 1995.

[66] Matthew T. Mason. Progress in nonprehensile manipulation. *The International Journal of Robotics Research*, 18(11):1129–1141, 1999.

[67] Eduardo F. Morales and Claude Sammut. Learning to fly by combining reinforcement learning with behavioural cloning. In *Int. Conf. on Machine Learning*, pages 76–, 2004.

[68] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.

[69] B. Nemec, F. J. Abu-Dakka, B. Ridge, A. Ude, J. A. Jørgensen, T. R. Savarimuthu, J. Jouffroy, H. G. Petersen, and N. Krüger. Transfer of assembly operations to new workpiece poses by adaptation to the desired force profile. In *Int. Conf. on Advanced Robotics (ICAR)*, pages 1–7, Nov 2013.

[70] 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, 2003.

[71] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 255–262, April 2000.

[72] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):322–332, April 2007.

[73] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[74] A. A. Rizzi and D. E. Koditschek. An active visual estimator for dexterous manipulation. *IEEE Transactions on Robotics and Automation*, 12(5):697–713, Oct 1996.

[75] E. Rohmer, S. P. N. Singh, and M. Freese. V-rep: A versatile and scalable robot simulation framework. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1321–1326, Nov 2013.

[76] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell. Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1024–1030, Sept 2015.

[77] L. Rozo, P. Jimenez, and C. Torras. Robot learning from demonstration of force-based tasks with multiple solution trajectories. In *Int. Conf. on Advanced Robotics (ICAR)*, pages 124–129, June 2011.

[78] F. Ruggiero, V. Lippiello, and B. Siciliano. Nonprehensile dynamic manipulation: A survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, July 2018.

[79] F. Sadeghi, A. Toshev, E. Jang, and S. Levine. Sim2Real View Invariant Visual Servoing by Recurrent Control. *ArXiv e-prints*, December 2017.

[80] J. Kenneth Salisbury, Jr. Recent advances in robotics. chapter Kinematic and Force Analysis of Articulated Hands, pages 131–174. John Wiley & Sons, Inc., New York, NY, USA, 1985.

[81] Stefan Schaal. Dynamic movement primitives -a framework for motor control in humans and humanoid robotics. In Hiroshi Kimura, Kazuo Tsuchiya, Akio Ishiguro, and Hartmut Witte, editors, *Adaptive Motion of Animals and Machines*, pages 261–280. Springer Tokyo, Tokyo, 2006.

[82] Gideon Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 03 1978.

[83] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Int. Symp. on Computer Vision - ISCV*, pages 265–270, Nov 1995.

[84] S. A. Stoeter, S. Voss, N. P. Papanikolopoulos, and H. Mosemann. Planning of regrasp operations. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 245–250, May 1999.

[85] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[86] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Learning policy improvements with path integrals. In Yee Whye Teh and Mike Titterington, editors, *Int. Conf. on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 828–835. PMLR, May 2010.

[87] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

[88] S. K. Tso and K. P. Liu. Hidden markov model for intelligent extraction of robot trajectory command from demonstrated trajectories. In *IEEE Int. Conf. on Industrial Technology (ICIT)*, pages 294–298, Dec 1996.

[89] H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 121–127, Nov 2015.

[90] Jie Yang, Yangsheng Xu, and C. S. Chen. Human action learning via hidden markov model. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 27(1):34–44, Jan 1997.

[91] M. Yashima and T. Yamawaki. Iterative learning scheme for dexterous in-hand manipulation with stochastic uncertainty. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3166–3171, May 2018.

[92] M. Yoshida, S. Arimoto, and J. Bae. Blind grasp and manipulation of a rigid object by a pair of robot fingers with soft tips. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4707–4714, April 2007.

[93] Hanna Yousef, Mehdi Boukallel, and Kaspar Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: Physical*, 167(2):171 – 187, 2011.

[94] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. *ArXiv e-prints*, October 2018.