Learning Human Manipulation Strategy by A Modular Approach

Abstract

Object manipulation is a challenging task for robot as the complicated physics involves in object interaction is hard to be expressed analytically. In this work we introduce a modular approach to learn the human manipulation strategy by demonstration. The human demonstrator demonstrated the task in different contexts and we group these demonstrates according to their control strategies. Forward and inverse model pairs are learnt from each group to encode the motor control policy. We validate our approach on a robot platform with a opening bottle cap task. We show that our approach can correctly identify the number of modules of a given task and generate proper motor commands for the robot to accomplish the task.

1. Introduction

With robots moving into human-centered environments, such as household, working office, it becomes more and more desirable to endow robots with the human-like ability. In everyday life, object manipulation is one of the most commonly used skills, which includes a large category of activities ranging from the simple pick-and-place task to the complicated dexterous manipulation task.

Generally, these manipulation tasks are very difficult due to the complicated contact situations and the changing kinematic and dynamic contexts. Despite this, humans can perform such skilled tasks and adapt to the changes without difficulty. At the heart of this skill is prediction [1]. Studies from neuroscience suggest that human develop internal models for motor control, to predict the next state of the environment. By comparing the predictive state with the actual sensory state, the internal models monitor the progression of the tasks and launch the corresponding motor correction and motor reaction. One hypothesis of this mechanism is MOSAIC [2], which is a modular model composed by multiple pairs of forward model and inverse model.

Inspired by this model, we propose a modular approach that encodes human manipulation skill and transfer to a robot. From multiple human demonstrations, we extract a set of control strategies. Each strategy take charge of one task context, composed by a forward model for context estimation and an inverse model for command generation. When a robot executes a similar task, the forward models estimate the context of the task based on the sensory data and "contextized" the inverse models to generate proper command that drives the object to the desire state.

To verify our approach, we use *Opening Bottle Caps* as an experimental example. This task is complex in the sense that the friction between the bottle and the cap surfaces has multiple phases and the transition between them is hard to predict. Adaptive control strategy is required for this task. To this end, we use our modular approach to learn the strategy from human and implementing it on a robot to open learnt and unseen bottles.

The rest of the paper is organized as follows: Section 2 provides an overview of the related work; Section 3 presents our approach of learning a multiple module model of human manipulation strategy; an experiment of a opening bottle cap task and its result is shown in Section 4, along with details of the hardware specifications and the experimental setup; Section 5 concludes this work with an outlook of future work.

2. Related Work

In this section, we review related literature in the area of robot learning manipulation task and modular approach.

2.1. Learning Manipulation Tasks

Demonstration based learning has been extensively studied in literatures [3, 4, 5] as a promising approach to build robot intelligence. Learning manipulation tasks is one of the main application of this approach. The physical properties of a manipulation task is hard to express analytically, and hence the control strategy is hard to derive. Modeling expert's demonstration of good strategies is an alternative solution to manipulation tasks.

In learning manipulation tasks, kinematics teaching and tele-operation are two major forms of demonstration. In kinesthetic teaching, human directly contact with the robot and guide their movements to accomplish a task [6, 7, 8, 9]. The trajectory of movements and contact force are recorded by the robot sensors. This method is simple and effective but limited in the number of controllable end effectors. While a manipulation task usually involves multifinger movement, a human can kinematically operate one finger with each hand and hence two fingers simultaneously for the most extend. To control multi-finger hands, some researchers use teleoperation [10, 11, 12]. This usually relies on data gloves and motion capture system to sense human hand-arm motions. The human motion is mapped to robots to generate motions and interact with the environment. In fine manipulation tasks, the robot platforms are usually restricted to anthropomorphic hands for better mapping. All of these methods provide no direct force feedback during manipulation to the human demonstrator.

In some studies, the human demonstrate manipulation tasks with their own bodies [13]. With direct interaction with the object the human demonstrator is able to perform the task most naturally and with a more delicate control strategy. The task information captured from these human demonstrations is needed to be transferred to robots. Various mapping methods are proposed in many literatures [14, 13, 15], while human correction [16, 17, 18] and self-correction via learning [19] are proposed as alternative solutions. In general, how to effectively transfer human skill to robot skill remains a open problem.

In our approach, human perform the task with direct interaction with the object to allow them perform natural feedback control strategies. We transfer these strategies to robot by taking an object centric approach. The object centric viewpoint [20] considers a manipulation task from the object's perspective, which suggests the goal of learning a task is to reproduce the same object behaviour. Taking this principle, an object level approach is proposed in [9] to learn object impedance in grasping and manipulation tasks. Our approach takes the same principle and learns the control strategy to operate the objects that will produce the same object behaviour. This strategy expressed from the object perspective can be transfer to any robot platform.

The aim of an object centric approach is to learn the correlation between the exert force of the system and the desire object motion. A classic way to describe this correlation is impedance [21, 22]. Giving desired impedance of a task, we can compute proper motor commands for the robot to accomplish it. Computing the impedance for a given task is difficult, however, especially for the tasks involving variable impedance. In many cases the impedance is roughly approximated by a linear model, but this is inadequate for nonlinear tasks. In [23] variable impedance is learn by the reinforcement learning algorithm PI^2 with a task specific cost function. Designing this cost function requires insight into the task and often is also laborious.

In our approach we take the advantage of human demonstration and we directly model the correlation of the exert force and the object motion by the Gaussian Mixture Model (GMM) [24]. GMM has been shown to be an efficient model in capturing the nonlinearity of data [25, 17, 16]. Our choice of GMM has other merits which will be discussed in later sections.

Single control strategy is inadequate for task with changing contexts. Robots operating in human centric environment need to be equipped with multiple strategies to handle multiple contexts. In the next section we give a brief overview of the modular approach in control.

2.2. Modular Approach

In manipulation tasks, context changing is a common phenomenon due to object interactions. These changes are often rapid or discontinuous. Classic adaptive control approaches such as model identification [26] are inadequate for these tasks, as instability or error may occur dur-

ing the searching of new optimal values of the model variables. A solution to fast adaption is the multiple model approach [27]. In this approach, multiple controllers are designed, each of which is appropriate for a certain task context. During control, the task context is estimated online and the corresponding controllers are activated. The benefit of this approach has been long discussed in the control community [28, 29]. Some recent works [30, 31] present promising modular based approaches. Multiple model approach also has a wide range application in robot control. It has also been shown to be an effective architecture of building intelligent systems [32, 33]. It is particular useful for tasks in non-stationary environment [34].

The approach we take is a modular approach inspired from MOSAIC [2]. MOSCAIC (MOdular SeleCtion and Identification for Control) is a bio-inspired paradigm of multiple module control. Fig. 3 illustrates the workflow. Each module is a pair of forward and inverse model. At each time step, the forward models estimate the current context and compare it with the actually context. The accuracy of the estimation decided the weight of the corresponding inverse models. The final motor command is the linear combination of the commands generated from each inverse models factored by their weights. Compare to the switching modular method [29], i.e. only one module will be activated and used to generate motor command, the linear combination of the modules requires less number of modules to approximate the system dynamics.

In [35] the forward model and inverse model is united to a single one. This is because for that particular task there is always only one action (a_t) can map the current task state (s_t) to the desired task state (s_{t+1}). However, in many cases this mapping is not unique and hence the inverse model have to be built with extra variables to resolve the non-uniqueness. In our approach we build the forward and inverse model separately for each module.

Our approach is based on the MOSAIC paradigm but with different modeling methods: we use GMM to represent the forward and inverse model rather than neuron network (NN). The main reason is that neuron network can not directly estimate the likelihood of one query point. With NN, the the variance in each forward model, which is an essential variable to control how much the modules cooperate, has to be hand tuned. Training GMM with the Expectation Maximization (EM) algorithm, we find the optimal values of the variances and hence avoid hand tuning.

Further, none of the modular approaches above provide a method to identify number of modules needed in a certain task. It is usually predefined by human experience or by the number of different setups. We solve this problem by a data driven method. We cluster the demonstration data with an hierarchical approach and model each cluster as a module. This solution can be applied to find the number of modules for "modularizable tasks". To the best of our knowledge, this is the first realization of the multiple model in an object manipulation task with a real robot, learnt from human demonstration.

3. Methodology

Our goal is to acquire a modular based control policy for an object manipulation task from human demonstration. To this end, we take a three-step approach:

1. Human demonstrating a task in different contexts (Section 3.1).
2. Extracting human control strategies for different contexts and build multiple internal models(Section 3.2).
3. Use the multiple models to compute motor commands for a robot(Section 3.3).

Figure 1 shows an overview of our framework.

3.1. Human demonstration

The first step is recording human demonstration of a task. Based on the object centric principle, we collect the object trajectory and its driven force. These data can be accessed by motion capture system, force-torque sensor and wearable haptic device. Fig. 2 shows a few sensors we used in the opening bottle caps task. Details will be explained in section 4.1.

In the demonstration, the teacher perform a task a number of times to generate enough data for capturing the key features of it. Further, the teacher perform the task under different system kinematics and dynamics configurations, e.g. friction conditions, in order to explore how human adaptive to different task contexts. These different configurations are chosen to cover a wide range of different contexts. For example, in a opening bottle cap task the demonstration of opening the tightest bottle, within the capability of the learner, is included. Details will be explained in Section 4.1.

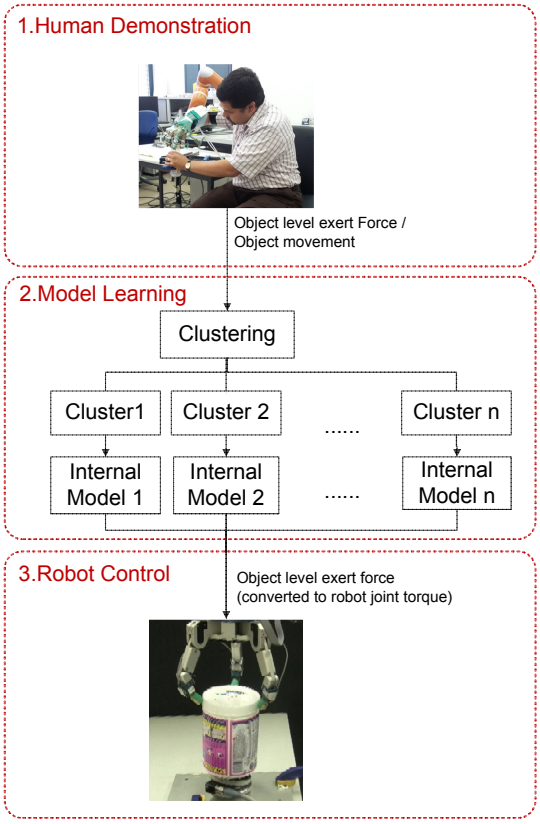


Figure 1: System overview.

(a) Optitrack markers attaching to a cap (b) Force torque sensor (c) Texscan tactile sensors mounting to a glove

Figure 2: Sensors used in the human demonstration of opening a bottle cap task.

3.2. Learning a Multiple-Module Model

In this section we detail our modeling method, explaining how do we model human manipulation strategy and how to choose the number of modules of a task.

3.2.1. Object centric manipulation strategy

As mentioned in Section 2, one of the challenges in imitation learning is the correspondence problem, i.e. how to map the teacher's motions to the robot's motions so that they produce the same effects, e.g. reaching the same point. In a object manipulation task, however, the goal is to deliver the object from the current state to a desired state. During this process the movement of the manipulator is bounded by the movement of the object. It is more important to imitate how human apply forces to achieve object's desired movement, than to imitate the human limb movement.

Therefore, our model encodes a force and torque profile rather than the end effector movement trajectory. The imitation learning objective here is not to find a policy for the end effector movement but to find a policy that maps the force and torque to the object movement. This policy allows the robot to efficiently acquire new behaviors to accomplish the task. Giving the robots' kinematics and the desired exert force on the object, the joint torques can be deduced by the Jacobian matrix [20]. To this end, we focus on the force-torque-displacement tuple: $\{F, \tau, s\}$ demonstrated in the task. In later sections, we refer $\{F, \tau\}$ as the motor command with notation $\{a\}$. In each demonstration, a time series of the tuple is recorded.

3.2.2. Decide number of modules

Due to the involvement of object interaction, a manipulation task frequently encounter abrupt changes of the system dynamics, e.g. transfer between status of no contact to contact, of static friction and dynamic friction. Different strategies should be used to handle different dynamics. We achieve this goal by using a modular approach. Multiple modules are learned from the demonstrations under different task contexts.

Different tasks may need different number of modules. In human demonstration the same task is demonstrated with a few different setups to explore how human adapt to them. However, the number of setups does not necessary equal to the number of modules needed in the task. Human may handle a few different contexts with the same

control strategies. In order to find a proper number of modules, we need to differentiate different types of strategies. The differences can be reflected from the different pattern of the force-torque-displacement tuple. Hence we differentiate the patterns in a data driven manner: cluster force-torque-displacement tuple. Data in the same cluster is considered to be governed by the same strategy. The number of clusters is the number of modules and each module is encoded by one model.

The goal of clustering is to separate a set of data to a few groups according to their similarities. The first step of clustering is to measure the similarities, i.e. the distances, between different data points. Different from an usual clustering problem, the data we need to cluster are not a set of single data points but a set of time series. Here we use the Dynamic Time Warping technique (DTW) to measure the distance between each pair of time series [36].

Dynamic time warping is suitable for measuring the similarity between two time series, which may have different speeds or durations. It warps the data in the time dimension and finds the optimal match between the time series. The similarity is computed as the average distance between the corresponding points in two series.

The similarity (distance) between each pair of time series is computed by DTW and produce a distance matrix. In this distance matrix, each element contains a measurement of the similarity between two time series. We then cluster these time series into a few groups by a threshold of the similarity. This threshold is set by using the variance of the data from the same context as a reference. As mentioned above, under the same context a task is demonstrated a few times. These demonstrations are presumed to be handled with the same strategy and hence belong to the same cluster. The variance of these demonstrations give a reference of a proper variance of a cluster. The largest variance, across the variance of all contexts, is used as the threshold for the clustering.

Most of the clustering methods require the specification of some variables, such as the number of clusters. Without knowing the number of clusters at the first place, we use an hierarchical agglomerative clustering method [37]. Agglomerative clustering is a method that merges data iteratively until the stop criteria satisfied, which does not require a predefined number of clusters. Our clustering method is described as follow:

1. At the beginning, each single time series is considered to be one cluster.
2. Compute the distances between each pair of clusters.
3. Starting from the first cluster, find its nearest cluster. We define the distance between two clusters to be the average distance across all the time series pairs in them. If the distance to the nearest cluster is smaller than the threshold, merge these two cluster.
4. Move to the next cluster. Repeat the last step for the rest of the clusters.
5. A new set of clusters are formed by the last few steps, move to the next hierarchy and repeat the step 2 to 4 until no new clusters can be formed.

Pseudocode of the complete algorithm is shown in Algorithm 1.

Algorithm 1 Agglomerative Hierarchical Clustering

```
1: Init(): Make each time series a cluster
2: mergeable = true
3: function MERGE(all clusters, distance matrix)
4:     while mergeable is true do
5:         mergeable = false
6:         for each cluster do
7:             ClusterA = current cluster
8:             ClusterB = nearest neighbor of ClusterA
9:             if distance(ClusterA,ClusterB) < clustering threshold then
10:                Merge ClusterB into ClusterA
11:                mergeable = true
12:            end if
13:        end for
14:    end while
15: end function
```

When the clusters can not be merged further, we discover the number of modules require in this task: it is the number of the remaining clusters. Each cluster is used as a module. The pattern of the data in a cluster represents a strategy of handling specific task contexts.

3.2.3. Learning Models

After identifying the number of modules, we build models for each of the module. In this section, we explain the way we encode human manipulation strategy.

During demonstrations, we constantly acquire the object displacements and the force and torque applied by the teacher. The teacher is the only source of exert force and torque of the system. The relationship between the exert force and torque and their resulting object displacement shows the dynamic characteristic of the task.

In our approach, we model the correlation of the force and the displacement with GMM. The task dynamics is hence encoded as a joint distribution of the object status displacement s and the action a taken by human $p(s,a,|\Omega)$. In our task, s is the one dimensional cap angular displacment and a is the one dimensional exert torque and grip force (section 4). Modeling the distribution by GMM allows us to capture the nonlinearity in the data, as well as to compute the likelihood of a query data point in the model. This provides an good estimation of the reliability of the module in the current task context, which is crucial in choosing the correct modules for control (discussed in Section 3.3.2). At the other hand, as a generative model GMM is able to generate new data from the model, i.e. generate motor commands.

With a GMM, the joint distribution of the variables is expressed as a sum of N Gaussian components,

$$p(s,a \mid \Omega) = \sum_{n=1}^{N} p_n p(s,a \mid \mu_n, \Sigma_n) \tag{1}$$

where p_n is the prior of the n-th Gaussian component and the μ_n, Σ_n the corresponding mean and covariance as:

$$\mu_n = \begin{pmatrix} \mu_{s,n} \\ \mu_{a,n} \end{pmatrix} \qquad \Sigma_n = \begin{pmatrix} \Sigma_{ss,n} & \Sigma_{sa,n} \\ \Sigma_{as,n} & \Sigma_{aa,n} \end{pmatrix} \tag{2}$$

We aim to build a model closely simulates human motor strategy in order to make the best use of the human data. A forward model is held to anticipate the outcome of the motor command, while an inverse model is held to generate motor commands to take the current system state to the next state. The discrepancy between the anticipation of the forward model and the actual feedback is used to correct the motor command generated from the inverse model (Section 3.3.2). Figure 3 shows the basic control flow of a forward-inverse model pair.

The forward model Ω_I is encoded by the joint distributions of the current system state, previous system state and the previous motor command, i.e. $p(s_t, s_{t-1}, a_{t-1} \mid$

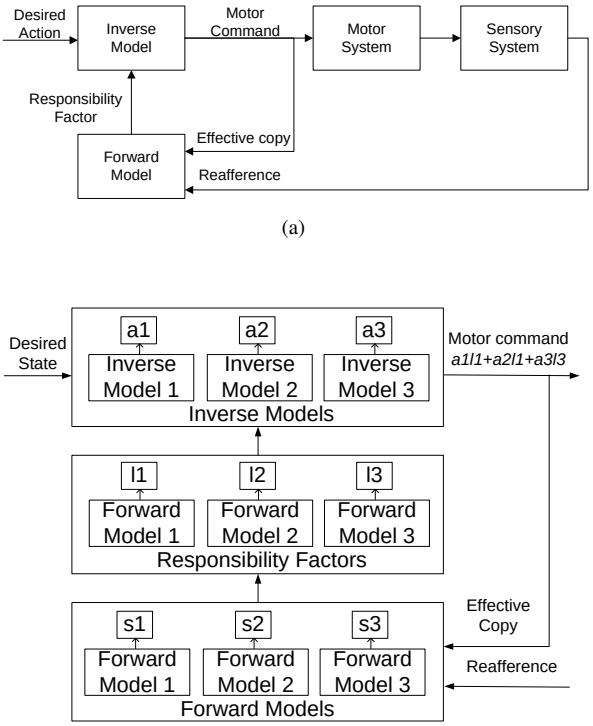


Figure 3: Control flow diagram of forward-inverse model in motor control.

Ω_I). For a given object displacement and a motor command, the Gaussian Mixture Regression (GMR) provides a close-form solution to compute the anticipating object displacement s_t, i.e. $p(s_t|s_{t-1}, a_{t-1}, \Omega_I)$. The inverse model Ω_f is encoded by the joint distributions of the current object displacement s_t and the desired object displacement s^*_{t+1}. In some tasks, the initial status of the system remains unchanged for a certain time until the exert force or torque is big enough to change it. This will cause degeneracy in the inverse model. To solve this problem we include the previous motor command into the model, i.e. $p(s_t, s^*_{t+1}, a_{t-1}, a_t|\Omega_F)$.

By clustering the training data into different groups as discussed in Section 3.2.2, we are able to discover the number of different patterns, i.e. number of modules. We train one GMM on each of the modules to encode the different changing patterns of the task context.

3.3. Multiple modular adaptive control

Once the number of modules is found and the multiple pair of forward and inverse models are learnt, they are used to compute motor commands for task execution. We consider the human motor system acts upon by motor command a_t at time t with current system status s_t. The resulting system status at time $t+1$ then is

$$s_{t+1} = f(s_t, a_t) \tag{3}$$

The goal of the controller is to generate a motor command that bring the current system status from s_t a desired state s^*_{t+1}:

$$a_t = g\left(s^*_{t+1}, s_t\right) \tag{4}$$

According to the discussion in Section 3.2.3, equation 3 represents the forward model and equation 4 represents the inverse model. It takes three steps to compute the motor command a_t:

1. Compute expected system state $\hat{s}_t$ by each forward model
2. compute responsibility factor λ for each module
3. Compute motor command by each inverse model and compute the final motor command a_t

3.3.1. Anticipate sensory output by forward model

With the k-th forward model we can estimate the current status $\hat{s}_t$ by

$$\hat{s}^k_t = E\left(s_{t-1}, a_{t-1} \mid \Omega^k_I\right) \tag{5}$$

This equation is to predict the environment status based on the observation and prediction on the controller's influence on the system. The expectation values of the current system status of the k-th module is computed by the *Gaussian Mixture Regression* (GMR). The computation is as follow.

With the sensory input $\{s_{t-1}, a_{t-1}\}$ as a query point q we define:

$$\mu^k_{q,n} = \begin{pmatrix} \mu^k_{s_{t-1},n} \\ \mu^k_{a_{t-1},n} \end{pmatrix} \tag{6}$$

$$\Sigma^k_{qq,n} = \begin{pmatrix} \Sigma^k_{s_{t-1}s_{t-1},n} & \Sigma^k_{s_{t-1}a_{t-1},n} \\ \Sigma^k_{a_{t-1}s_{t-1},n} & \Sigma^k_{a_{t-1}a_{t-1},n} \end{pmatrix} \tag{7}$$

and GMR then uses:

$$\hat{\mu}^k_{s_t,n} = \mu^k_{s_t,n} + \Sigma^k_{s_t q,n}(\Sigma^k_{qq,n})^{-1}(q - \mu^k_{q,n}) \tag{8}$$

$$\hat{\Sigma}^k_{s_t s_t,n} = \Sigma^k_{s_t s_t,n} - \Sigma^k_{s_t q,n}(\Sigma^k_{qq,n})^{-1}\Sigma^k_{q s_t,n} \tag{9}$$

Finally, all the N Gaussian components of the k-th module[1] are taken into account and the current sensory data $\hat{s}_t$ is predicted as the mean $\hat{\mu}_{s_t}$ with the covariance $\hat{\Sigma}_{s_t,s_t}$ according to:

$$\hat{\mu}_{s_t} = \sum_{n=1}^{N} \beta_n(q)\hat{\mu}^k_{s_t,n} \tag{10}$$

$$\hat{\Sigma}_{s_t s_t,n} = \sum_{n=1}^{N} \beta_n(q)^2 \hat{\Sigma}^k_{s_t s_t,n} \tag{11}$$

where

$$\beta_n(q) = \frac{p_n p(q|\mu^k_{q,n}, \Sigma^k_{qq,n})}{\sum_{n=1}^{N} p_n p(q|\mu^k_{q,n}, \Sigma^k_{qq,n})} \tag{12}$$

[1] Different modules may have different number of Gaussian components.

3.3.2. Responsibility factor

In a multi-module approach, choosing the proper modules to compute the motor command is crucial. For this we rely on the computation of the responsibility factor. The responsibility factors act as the weight of the modules. Each module has its responsibility factor at every time step, the final motor command at that time step is the linear combination of the commands generated from each module multiplied by their responsibility factors. The responsibility factor is a measurement of the reliability of using one model to represent the current system contexts.

As the models are built as GMMs, it is easy to estimate the likelihood of one data point belongs to a particular module. The actual current state from the sensory feedback, the previous state and the previous motor command forms a query point $\{s_t, s_{t-1}, a_{t-1}\}$. The responsibility is computed by the likelihood of the current query data point in each module, normalized by the total sum:

$$\lambda_t^j = \frac{p(s_t, s_{t-1}, a_{t-1} \mid \Omega_I^j)}{\sum_{k=1}^{K} p(s_t, s_{t-1}, a_{t-1} \mid \Omega_I^k)} \tag{13}$$

where K is the number of modules.

3.3.3. Generate motor command by Inverse Model

The motor command a_t^k for the i-th inverse model is computed by GMR with the same steps described in Section 3.3.1. The responsibility factors λ^j act as the weights of each model in the control system. The higher the responsibility is, the more response the model takes in the control system. Therefore, the final motor command generated by this multiple model system is

$$s_t = \sum_{k=1}^{K} \lambda^k a_t^k = \sum_{k=1}^{K} \lambda^k E\left(s_{t+1}^*, s_t, a_{t-1}^k \mid \Omega_I^k\right) \tag{14}$$

These three steps are computed with a close form solution. This ensures that this system can react quickly to the changes in the environment by adjusting the responsibility factor.

4. Robot Experiment

The proposed multiple model approach is implemented on a real robot system (the KUKA robot arm and the Barrett Hand) for a particular manipulation task: opening bottle caps. The target of this task is to unscrew a tighten cap. This task is chosen as it is a common task in human daily life and at the same time a complex one from the control point of view. Before the task begins, human does not possess any information about the tightness of the cap. During the task, human constantly update the motor commands, i.e. how much torque to apply to the cap and how much force to grip the cap, according to the sensory feedback of the status of the context. This plan can not be make in advance as the dynamic configuration of each bottle is different and it changes throughout the duration of the task. Human have to cope with these uncertainties and adapt to the changes. In the later sections, we will explain the experiment details and show that the multiple module approach is able to acquire human adaptive control policy and enable the robot to master this manipulation task.

4.1. Human demonstration and experimental setup

Opening bottle cap is a common task for human however a difficult one for robot. The friction between the bottle and the cap, and the way they change from screwed to unscrewed, vary among different bottles. This requires multiple controlling strategies to open them. Hence a multiple modular approach is suitable for this task, which includes at lease two different control strategies for handling static friction and kinetic friction. Figure 4 shows three different patterns of human control strategies for three different contexts.

The task involves the control of the turning torque, griping force according to the displacement of the cap. As the changes of the friction coefficient is nonlinear during the unscrewing process, it is hard to build an analytical model for it. Human can easily open most of the bottles in daily life. Learning from human demonstration will allow us to gain a controlling strategy without writing the analytical expression of the dynamics of the whole system.

In each demonstration, data from first time the finger touch the cap to the cap is finally open and lifted was recorded. Opening bottle cap is a cyclic task. Each cycle includes three stages: reaching, turning and releasing. In our experiments, four to six cycles need to be completed to open the bottles. During the reaching and releasing stages, no torque nor griping force is applied to the cap and the cap remains still. During the turning stages, human continuously apply torque to the cap and it starts moving once the friction is overcome.

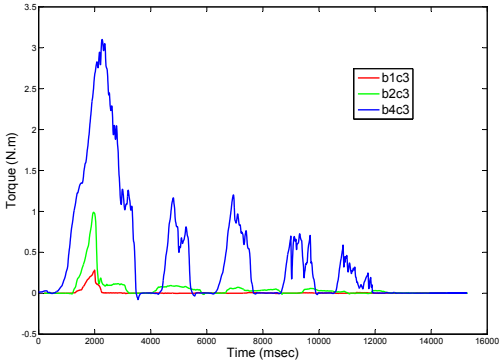


Figure 4: Exert torque for opening three different bottles.

Figure 5: Bottles and caps for human demonstration. From left to right: b1 c1, b2 c2, b3 c3, b4 c4

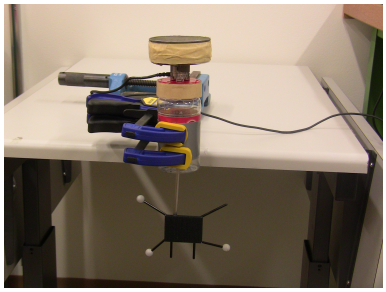
4.1.1. *Demonstration in different task contexts*

The experiment starts with human demonstration. In order to explore different task context, we demonstrated the task with different setups, which are the combination of four different bottles ($b1-4$) and four different caps ($c1-4$) (Fig. 5). According to the surfaces situation of the bottles and the caps, the difficulty of opening the bottles varies. We lubricate the cap and bottle surfaces of $b1$ to make it easy, and leave adhesive materials on the surfaces of $b4$ to make it difficult. $b1-4$ are labeled by increasing difficulty, while $c1-4$ are labeled by the increasing diameters of the caps.

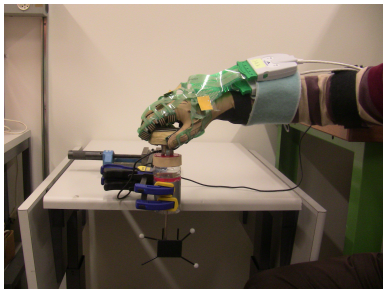
We chose to vary the setups in friction coefficient and cap size as these are the main variances between different bottles effecting the control strategy. The intention is to see how does these two variables effect human behaviour. To this end, we "combine" the bottles and the caps by mounting the caps onto the "actual caps" of the bottles (Fig. 6). In this way, we have seven different setups for the demonstration (Table 1). Demonstrations on the set of setups with the same cap and different bottles, i.e. $b1c2, b2c2, b3c2, b4c2$, allow us to explore human strategies in adapting to different friction coefficients. Demonstrations on the set of setups with the same bottle and different caps, i.e. $b3c1, b3c2, b3c3, b4c3$, allow us to explore human strategies in this task with different cap sizes which result in different grasping strategies.

	 Cap 1	 Cap 2	 Cap 3	 Cap 4
 Bottle 1			b1c3	
 Bottle 2	b2c1	b2c2	b2c3	b2c4
 Bottle 3			b3c3	
 Bottle 4			b4c3	

Table 1: Different setups of bottles and caps for demonstration. Bottle 1 to 4 is in increasing order of the friction coefficients. Cap 1 to 4 is in increasing order of the cap sizes.



(a)



(b)

Figure 6: Experimental setup for the task of opening a bottle cap. (a) Setup b2c4: bottle 2 combined with cap 4. A force-torque sensor is mounted between the bottle can the cap to measure the exert force and torque. A set of Optitrack markers are connected with the cap to record the displacement of it. The bottle is fixed on a table. (b) Human demonstrating opening a bottle cap. To avoid extra torque, only one hand is used during the demonstration. Human grip the cap from the top and apply torque to the system.

4.1.2. Sensors

In each setup the teacher demonstrates the task of opening bottle cap three times. Before each demonstration, the bottle is tighten with the cap with the same scale of tightness. In total we recorded 21 sets of demonstrations. In this section, we explain how did we record these demonstrations by sensors.

As explained in section 3.2.1, we focus on the tuple $\{\tau, F, s\}$ of the task. Three different set of sensors are used in the experiment to capture them:

1. Force torque sensor[2] for exert torque (τ);
2. OptiTrack[3] for cap displacement (s);
3. Tekscan[4] for exert force (F).

Data from these three sensors stream from three different channels. Due to the hardware limitation the raw data steam from different channels do not come at the same time, and are not recorded in a regular frequency. To synchronize the data, we produce a synchronization signal at the beginning of each demonstration: the demonstrator tap on the cap three times. The movement of the hand and impulses on the cap produce simultaneous pulses in all three channels. After recording, the data from different channels are synchronize by aligning the synchronization signal.

In this task, the turning torque is the essential variable. This is measured and recorded by a ATI force torque sensor (Fig **??**). It is mounted between the bottle and the cap (Fig. 6). During the task, the teacher grab the cap on the top of the force-torque sensor and apply torque to open the bottle mounted below the sensor. As the bottle is fixed on the table, the movement of the cap is restricted on the rotation along the bottle's axis. Under the approximation of zero angular momentum, the reading of the sensor shows the force and torque applied on the cap. Besides the torque, force applied to the z-axis direction is also recorded for the purpose of synchronization (Section 4.2).

We track the displacement of the cap by a motion tracking system OptiTrack. The OptiTrack system track movement by the infrared reflecting markers attached on the

[2] https://www.ati-ia.com/
[3] http://www.naturalpoint.com/optitrack/
[4] http://www.tekscan.com/

object. In order to avoid obstacle of the demonstration, we attach markers to a stick, which is fixed to the cap from one end and the other end coming out from the bottom of the bottle (Fig. 6). We also recorded the human hand movement, by tracking the markers attached to the human hand. The movement of human hand is used later for synchronization (Section 4.2).

During the task, human also apply grip force on the cap in order to grasp it firmly for turning. This force can not be sensed by the force torque sensor. Therefore, we used a pressure sensor (Tekscan Grip System) for measuring the human grip force. The Tekscan Grip System is a flexible tactile pressure sensor that can be built into a glove. It has 18 patches of sensors to covert the human hand front surface. For manipulation human use not only the front surface, but also the side surface of our fingers. In order to measure the force applied by those surfaces, we mount two sets of Tekscan Grip System sensors onto a glove to cover also the side surfaces (Fig. 4.2.2). The method of mounting the sensors to the glove is detailed in [38]. With different sizes of the caps or in different stages of the task, the way human grasp the cap varies, e.g. using 2 fingers to grip the smallest cap c1 and using 4 fingers to grab the biggest cap c4. The used patches in each grasps are recorded. In the computation of the total grip force, only the used patches are taken into account. All patches are calibrated to read in the unit of $N{\cdot}m$.

4.2. Data Analysis

In this section we explain how we manage the raw data and extra training data. The raw data from the three sensors stream in three separated channels. They have different formats and hence are handled differently.

4.2.1. Optitrack

With 3 markers attached with the object, OptiTrack is able to track both the object's translation and rotation. This is expressed in the position vector and the rotation matrix. To compute the angular displacement of the cap by the ration matrix of the cap, and compute the hand movement by the position vector of the hand. The accumulated angular displacement is used to learn the model and the hand movement is used to synchronize the data.

4.2.2. Force-Torque Sensor

As the movement of the cap is restricted to the rotation along the z-axis, we concern only the torque applied in this direction. Other concerned dimension is the force applied in the z direction. The three taps on the cap before each demonstration will create three pulses in the z direction and hence is used for synchronization.

4.2.3. Tekscan

As mentioned in previous section, we use two set of Tekscan to cover the front and the size of the human hand. This enable the demonstrator to use any grasp they like for the task, not restricted to using the first three fingers as most of the other grasp experiment. In each type of grasp, the reading from the patches contacting with the cap are summed and multiplied by their surface area to compute the total grip force.

Data from these three channels is synchronized by aligning the synchronization pulses. The time of the last detected pulse is set to the zero reference point. After synchronization we re-sample all the temporal sequences to 100Hz. Hence each single data point is synchronized. Finally, we filtered the noise by a low pass filter. Fig. 7 shows an example of the data from three different channels.

In this task we focus on the turning stage of each cycle. More specifically, we focus on the data starts from the moment that the fingers contact the cap and end at the moment that the turning is finished and the cap is released. The reaching and releasing cycles do not involve contact with the environment and hence are not concerned here. In order to collect data from only the turning cycles, we trim the data by the contact single: only the sequence with non-zero contact force will be kept.[5] The trimmed sequences are labeled by their setups and the order of their appearance, e.g. the 1st cycle of the bottle 1 with cap 3 is labeled by $b1c3_1$.

As can be seem from Fig. 7, there are dramatic difference between the cycle one and the rest of the cycles: the exert force and torque are much higher in the first cycle than in the other cycles. This is caused by the difference

[5] In this task the segmentation is done manually. The data can also be segmented by other algorithms but here we do not focus on task segmentation.

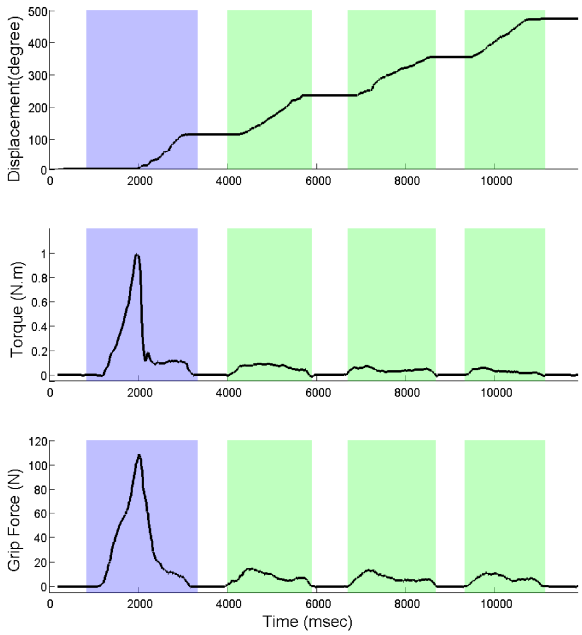


Figure 7: Aligned data of all three channels. Highlighted parts mark the turning process: blue blocks denote the first cycle, i.e. the phase I, and green blocks denote the later cycles, i.e the phase II. Phase I is significantly different from the phase II.

between the static friction and the kinetic friction. At the beginning of the task we have to first break the contact between the bottle and the cap. The friction at this stage is driven by the static friction coefficient. Once the cap starts to move, the friction coefficient between bottle and cap transits to kinetic friction coefficient, which is usually a few times smaller than the static friction coefficient for the same surface. As a result, the torque and hence the grip force required to turn the cap decrease in the later cycles. This phenomenon implies that at lease two modules are needed for this task. In the later section we will discuss these two phases separately and refer the cycle one as "phase I" and the later cycles as "phase II".

In different demonstrations, the number of cycles used to open the cap is different, varying from four to six. The pattern of the later cycles are similar as the demonstrator just repeat the same strategy to rotate the cap. For training, we take the first four cycles from each of the demonstration. This results in 84 time series in total for the learning.

4.3. Learning Modules

In this section, we explain how do we encode the training data into a few different modules. As mentioned in Section 3.2.2, the first step is to cluster the data and find out the number of modules required in this task.

4.3.1. Data clustering

To cluster the 84 time series $Q\{s,\tau,F\}$ obtained from human demonstration, we first computed the distance between each pair of them by the DTW technique. As this task is time independent, "warping" of the data in the dimension of time does not effect the control policy encoded in the time series. The distances between each pair of the time series is shown in the heatmap (Fig. 8).

As mentioned before, the demonstration of each setup is repeated three times and the data from the same setup and same cycle are presumed to belong to the same cluster. To set a threshold for clustering, we check the distances between the time series come from the same setup and the same phase. The largest distance found is 0.04 (normalized) from the $b3c2$ phase 4. We add a 10% margin on this (resulting to 0.044) and use it as the threshold of clustering. The time series distance less than the threshold are grouped into the same cluster. We use the hierarchical agglomerative clustering (Section 3.2.2) to merge

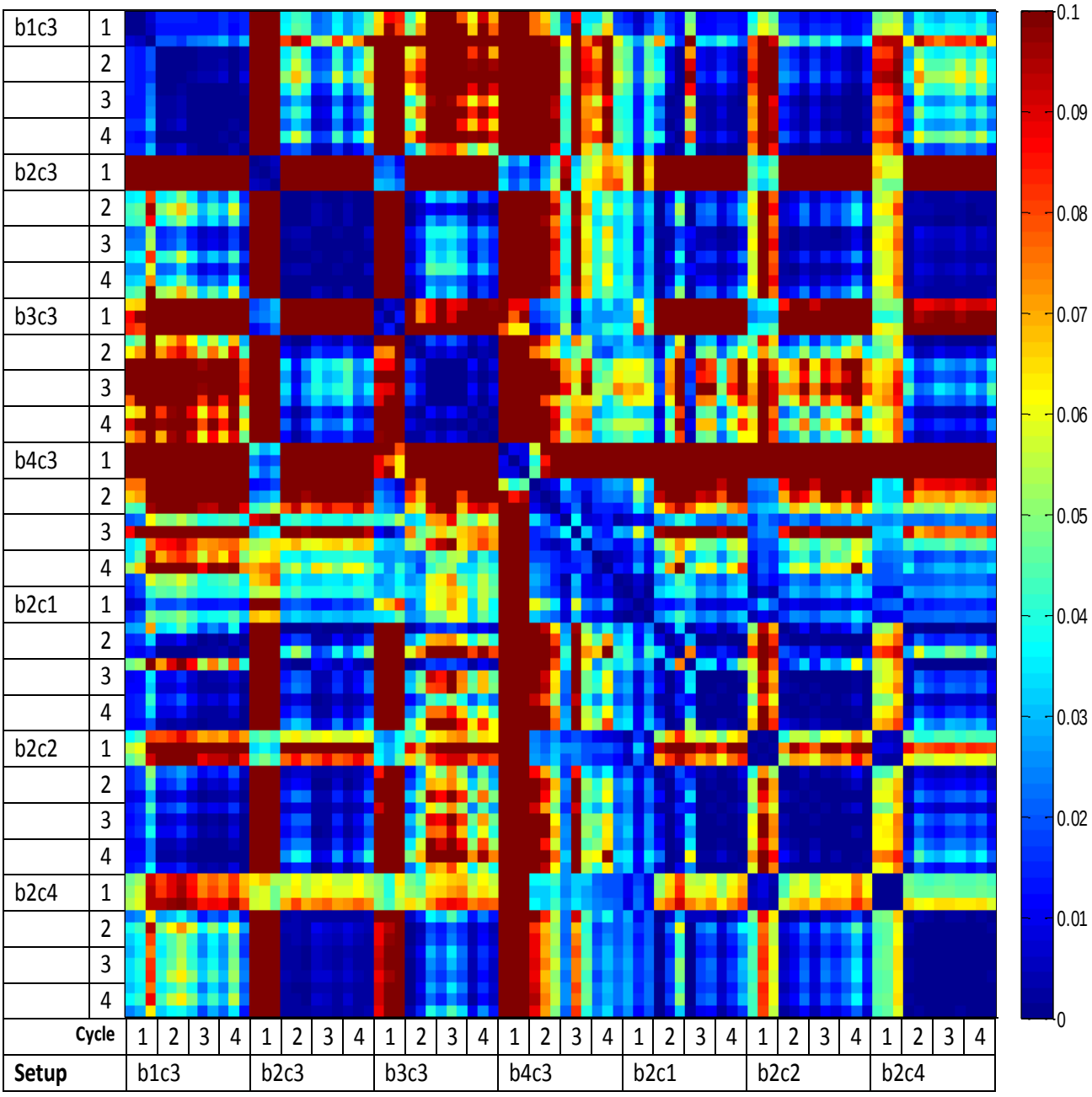


Figure 8: A heatmap representation of the distances between each pair of time series. Demonstrations are done with each setup for three times. The first four cycles are used for training. In total 84 time series are used as training data.

the data into different clusters. After 5 times of merging, the clusters are not mergeable and 3 clusters remains.

These three clusters contain the data from:

1. phase I of $b4c3$ (most difficult bottle), 3 time series;
2. phase I of $b2c3, b3c3, b2c1, b2c2, b2c4$ and phase II of $b4c3$, 24 time series;
3. phase I of $b1c3$ (easiest bottle) and phase II of the other setups, 57 time series.

The result of clustering is shown in Table 2. This result suggests that human use three different strategies for opening bottles: one for handling the phase I of the most difficult bottle with adhesive materials on the bottle and cap surfaces; one for handling the phase I of most bottles and the phase II of the most difficult bottle; and one for handling the phase I of the lubricated bottle and the phase II of the other bottles. The size of the cap turn out to be playing a less important role in the control strategies. According to this result, we encode these three clusters separately.

4.3.2. Learning Modules

We encode the data in each of the module by GMM. As explained in Section 3.2.3, a forward model and an inverse model are built for each module. The forward model is encoded by the joint distribution $p\{s(t), s(t-1), a(t-1) \mid \Omega_F\}$, while the inverse model is encoded by $p\{s(t), s(t+1), a(t), a(t-1) \mid \Omega_I\}$. For each model, the number of Gaussians is determined by the BIC. We use 25 Gaussian for cluster 1, 40 for cluster 2 and 15 for cluster 3. Their BIC tests are shown in Fig 9.

4.4. Generating motor command for manipulation

Our approach is independent of robot system and can potentially be applied to any robot. We choose to implement this work with a Barrett hand mounted on a KUKA lightweight robot as they are available in our lab. We implement the multiple module system on this platform to enable the robot opening bottle caps.

In this experiment, we control the wrist joint (last joint of KUKA) for producing torque to turn the bottle cap. A force torque sensor is fixed under the bottle to provide torque feedback. Each finger of the Barrett hand is mounted with a *Syntouch*[6] tactile sensor, which is cali-

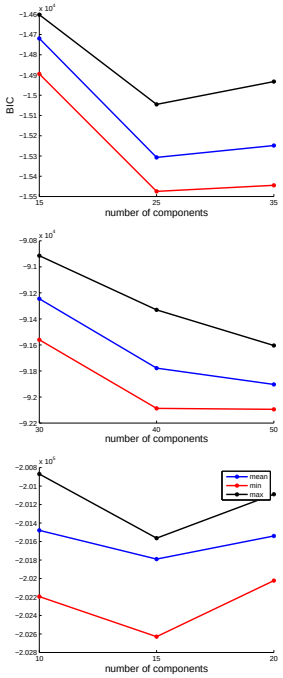


Figure 9: BIC test result for clusters. (a) Cluster 1, (b) Cluster 2, (c) Cluster 3.

[6] http://www.syntouchllc.com/

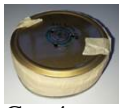
		 Cap 1	 Cap 2	 Cap 3	 Cap 4
 Bottle 1	Phase I			(b1c3) Cluster 2	
	Phase II			Cluster 3	
 Bottle 2	Phase I	(b2c1) Cluster 2	(b2c2) Cluster 2	(b2c3) Cluster 2	(b2c4) Cluster 2
	Phase II	Cluster 3	Cluster 3	Cluster 3	Cluster 3
 Bottle 3	Phase I			(b3c3) Cluster 2	
	Phase II			Cluster 3	
 Bootle 4	Phase I			(b4c3) Cluster 1	
	Phase II			Cluster 2	

Table 2: Clustering result

Algorithm 2 Control Algorithm

for r = 1:4 **do**
  REACHING(): Robot move to initial position
  function TURNING()
    Read previous sensor information $\{s_{t-1}, \tau_{t-1}, F_{t-1}\}$
    for k=1:3 **do**
      $\hat{s}^k$ = FORWARD($s_{t-1}, T_{t-1}, \Omega_I^k$)
    end for
    for k=1:3 **do**
      λk = ResponsibilityFactor($\hat{s}^k, s_t$)
    end for
    Read current sensor information $\{s_t\}$
    for k=1:3 **do**
      $\{a^k\}$ = INVERSE($s*_{t+1}, s_t, a_{t-1}$)
    end for
    $\{a_t\} = \sum_{k=1,2,3} \lambda k \{a^k\}$
    Add compensational torque to τ_t
    Execute motor command $\{a_t\}$
    RELEASING(): Release the cap;
  end function
end for
while LIFTCAP() is false **do**
  REACHING();
  TURNING();
  RELEASING();
end while

brated to provide contact force information, for the grip force feedback. The cap displacement is measured by the wrist joint displacement, assuming that there is no slip between the fingers and the cap.

The target bottle is fixed on the top of a table with it's cap tighten. The robot is placed above it on a distance that allows a proper grasp on the cap. The Barrett hand then close the fingers until the bottle cap is touched. This position is recorded as the initial position, where the cap displacement is marked as zero. In the experiment we focus on the turning cycle. The releasing and reaching cycles are programmed by opening the fingers and restoring to the initial position.

We first test the model with the trained bottles and then test with two new bottles. With each bottle, the turning-releasing-restoring cycles are repeated four times. Data stream from the sensors are filtered to $100Hz$. Once the turning cycle starts, the forward models take the torque and displacement at the last time step as input, compute the expected displacement of the current time step. These expected displacements are compared with the actual displacement measured at the sensor to evaluate the reliability, expressed as a normalized responsibility factor, of each module. The inverse models take the current displacement, desired next displacement and the previous force and torque as input to compute the a proper action (force and torque) to take on the cap. Each of the three outputs is multiplied with its responsibility factor, and the final output is the sum of the factorized three outputs (Algorithm 2).

In implementation on a real robot, we found that without putting any restriction of the responsibility factor, it can change rapidly. This is caused by the environmental noise in the sensory input and results in instability of the control system. We apply a low pass filter on the responsibility factor to reduce the fluctuation. This filtering implies that the real dynamics does not switch from module to another with high frequency, which consists with the character of our task.

Before apply the final output on the robot, a compensational torque is added to it in order to compensate the slag causing by the distortion of the robot hand during turning. The control algorithm described above is shown in algorithm 2.

4.5. Experiment results

We implemented this algorithm with two bottles (b1 and b4) in the training set and then two bottles have not been used for training. The experimental results and demonstration snapshots are shown in figures 10- 13. The experiments show that our multiple modular approach is indeed effective in learning manipulation tasks.

5. Conclusion and Discussion

In this paper we proposed a modular approach for learning manipulation task from human demonstration. We find out the number of modules needed in a task by hierarchical clustering. From each cluster we use forward and inverse model pairs to model the motor control mechanism. The forward models are to predict the effect of the previous motor command, while the inverse models compute a motor command to bring the current state to a desired state. The statical approach enables us to estimate the reliability of the inferences of each module under the current context. The final motor command is the sum of the weighted command from each module. Our experiments verify that by this modular approach, the robot can automatically recognize the current task context and compute motor commands to accomplish a manipulation task, i.e. opening bottle caps.

There are many promising directions of further studies of this work. The first is to apply this approach to more contact and friction relative tasks and learn a more general human control strategy to handle the instability caused by friction. To extend our approach to learn tasks involve multiple steps, one could also integrate it with task segmentation technique, to break down the task into atomic steps and recognize the steps needs modular approach. So far we have implement the approach with a task controlled in three dimensions and by three modules. How does the number of modules change according to the dimension of the task is another useful information to reason about.

To summary, tasks involve multiple phases or contexts are hard to implement by a single model. By clustering the control strategies and corresponding task contexts, we are able to focus on each task subspace and build local models. Modular architecture is a practical approach for modeling these tasks. As manipulation usually involves multi-phase friction and multi-body interaction, learning manipulation tasks with a modular approach can simplify the modeling problem in a large extend.

[1] J. R. Flanagan, M. C. Bowman, R. S. Johansson, Control strategies in object manipulation tasks, Current opinion in neurobiology 16 (6) (2006) 650–659.

[2] M. Haruno, D. M. Wolpert, M. Kawato, Mosaic model for sensorimotor learning and control, Neural computation 13 (10) (2001) 2201–2220.

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

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

[5] D. Kulić, C. Ott, D. Lee, J. Ishikawa, Y. Nakamura, Incremental learning of full body motion primitives and their sequencing through human motion observation, The International Journal of Robotics Research 31 (3) (2012) 330–345.

[6] D. Korkinof, Y. Demiris, Online quantum mixture regression for trajectory learning by demonstration, in: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, 2013, pp. 3222–3229.

[7] A. L. Pais, A. Billard, Encoding bi-manual coordination patterns from human demonstrations, in: Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, ACM, 2014, pp. 264–265.

[8] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, S. Schaal, Skill learning and task outcome prediction for manipulation, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 3828–3834.

[9] M. Li, H. Yin, K. Tahara, A. Billard, Learning object-level impedance control for robust grasping and dexterous manipulation, in: Proceedings of International Conference on Robotics and Automation (ICRA), 2014., (accepted).

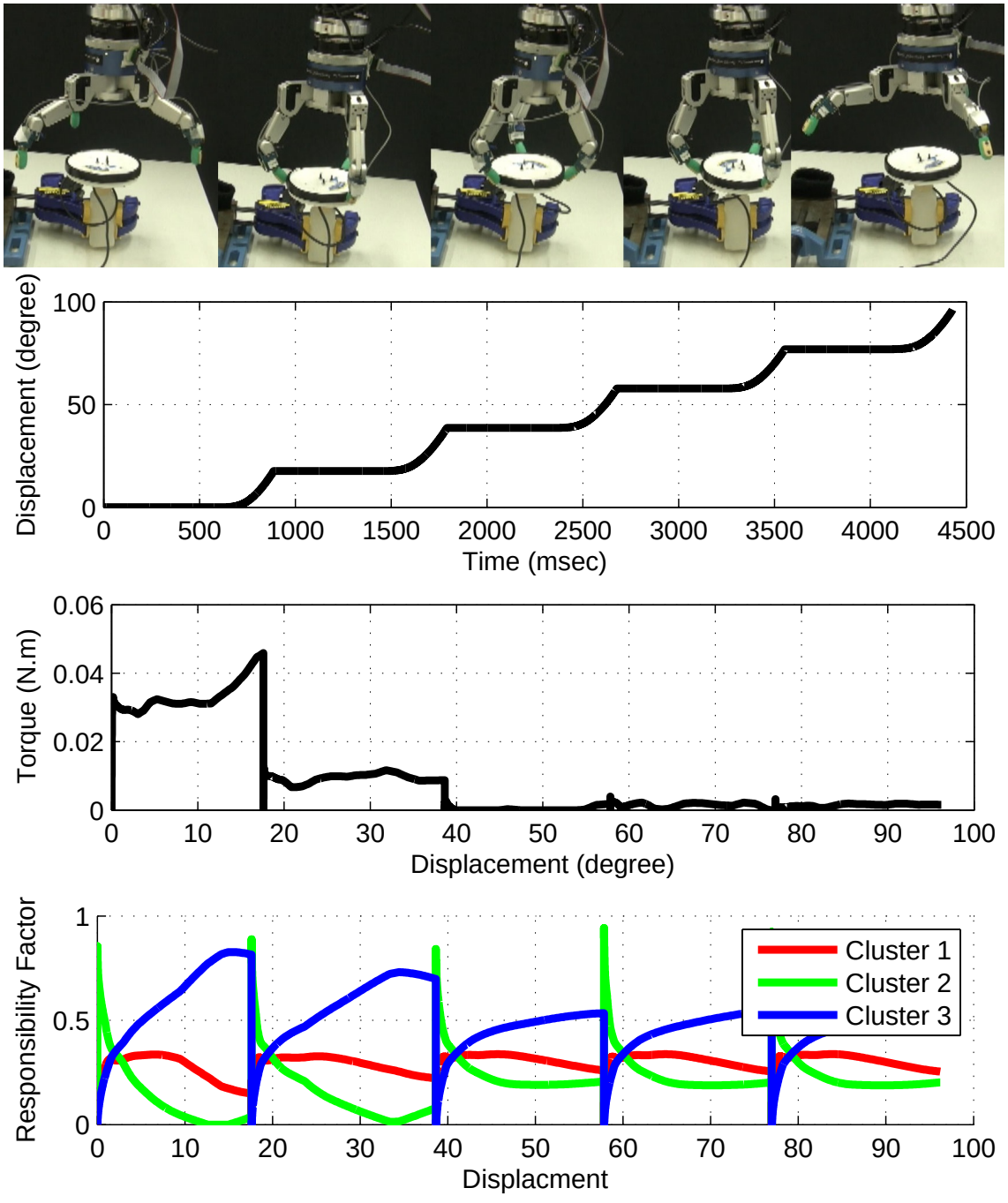


Figure 10: Robot demonstration on opening b1.

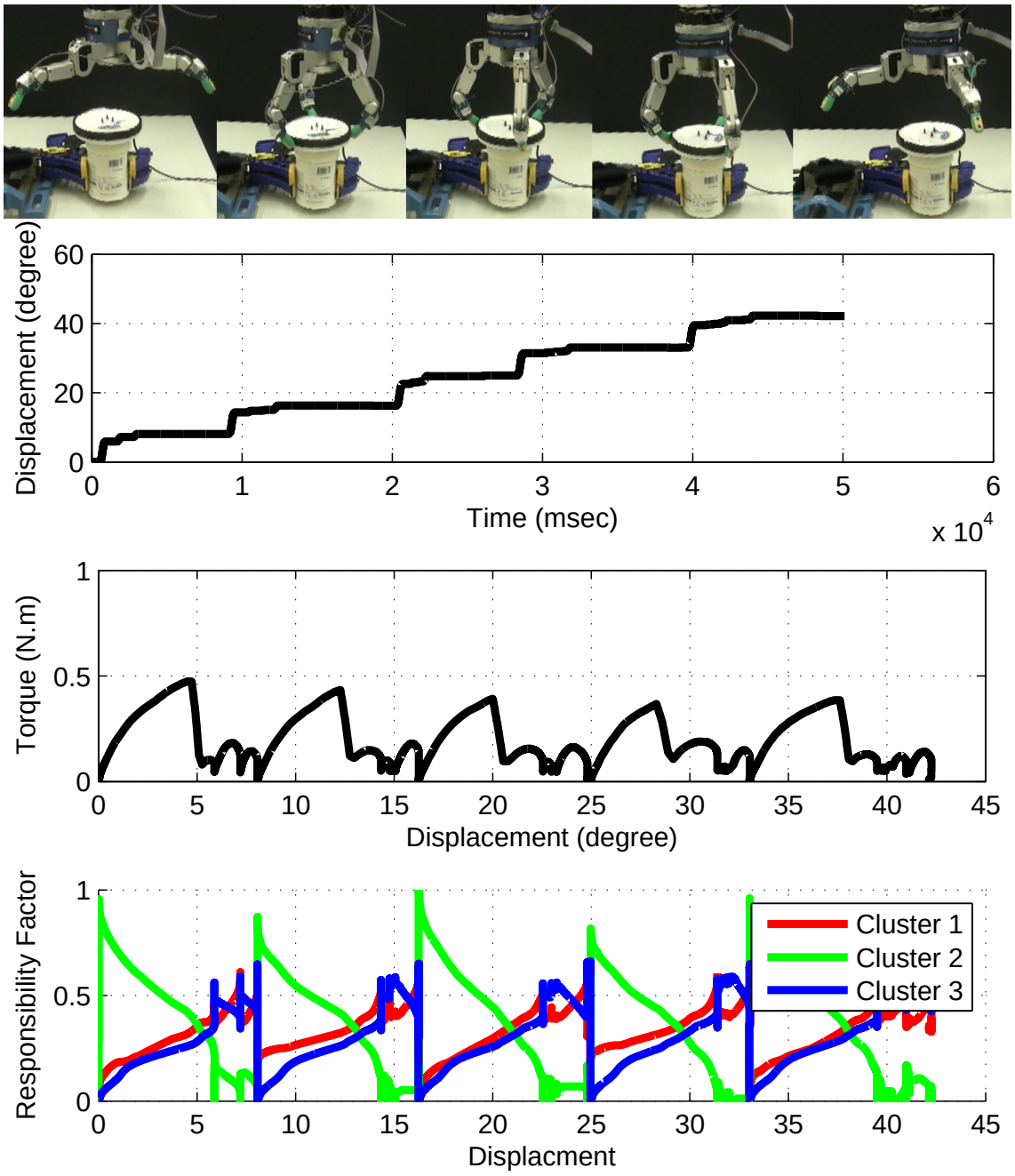


Figure 11: Robot demonstration on opening b4.

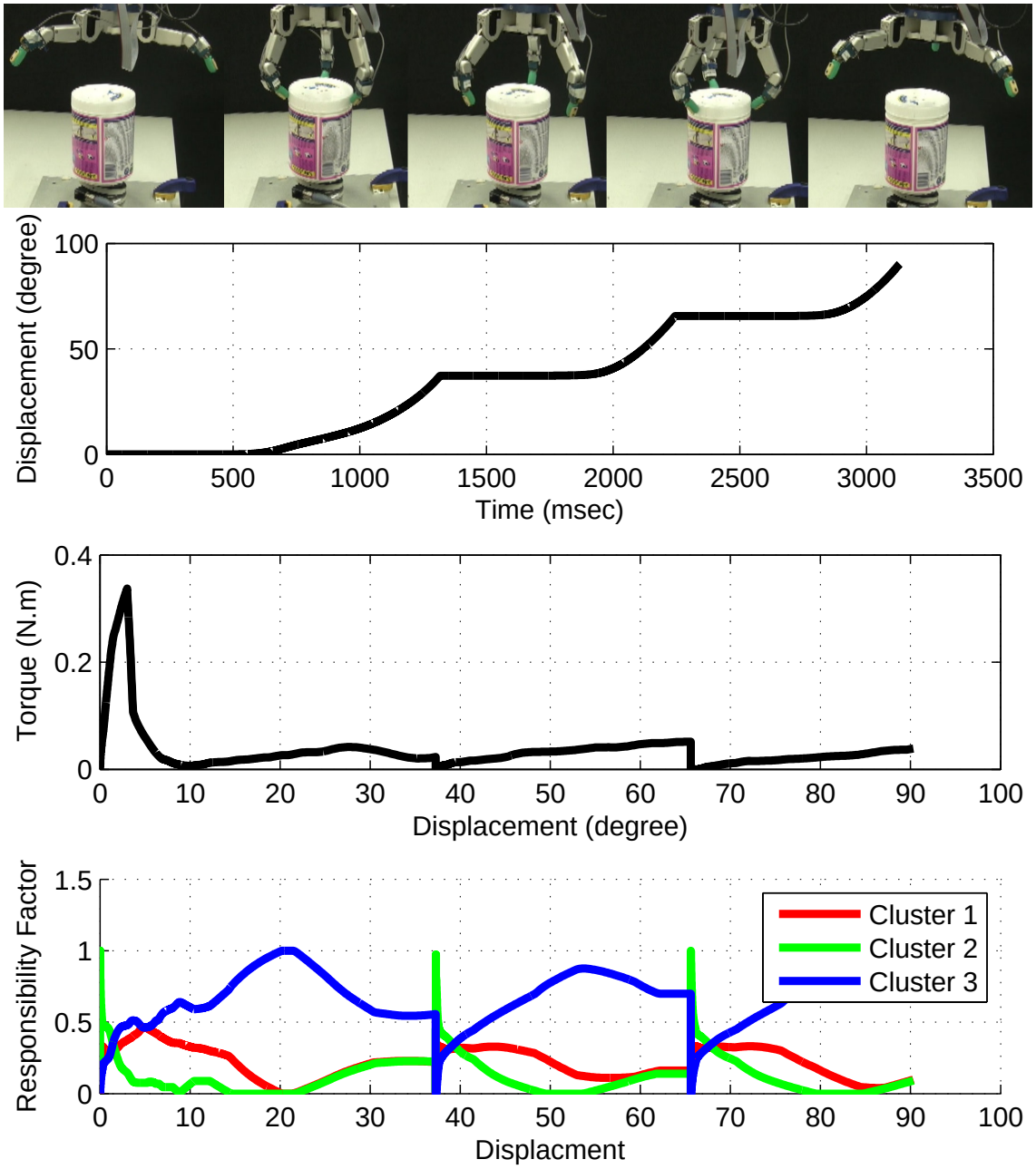


Figure 12: Robot demonstration on opening a new bottle (b5).

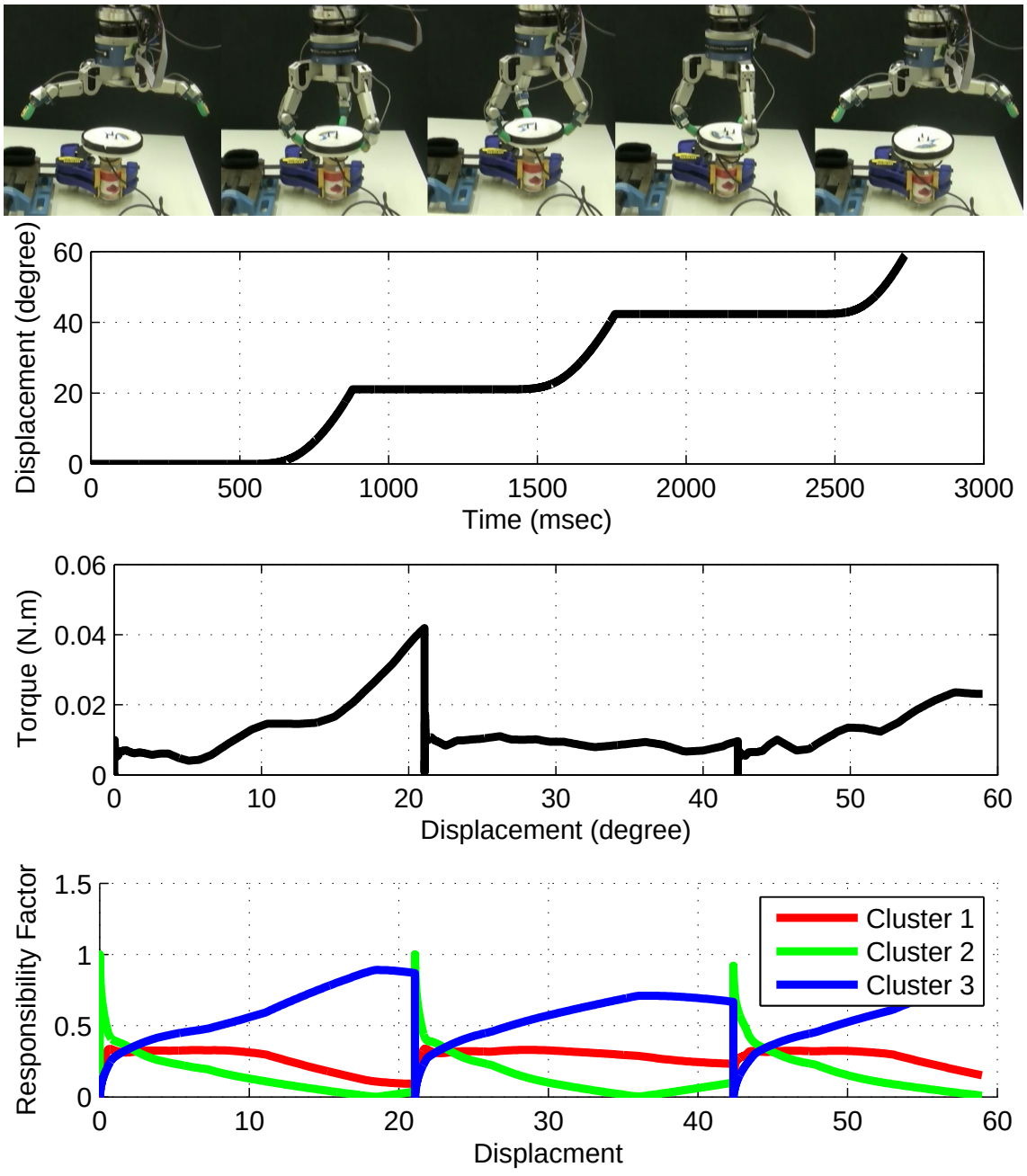


Figure 13: Robot demonstration on opening a new bottle (b6).

[10] A. Bernardino, M. Henriques, N. Hendrich, J. Zhang, Precision grasp synergies for dexterous robotic hands, in: Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on, IEEE, 2013, pp. 62–67.

[11] M. Kondo, J. Ueda, T. Ogasawara, Recognition of in-hand manipulation using contact state transition for multifingered robot hand control, Robotics and Autonomous Systems 56 (1) (2008) 66–81.

[12] M. Fischer, P. van der Smagt, G. Hirzinger, Learning techniques in a dataglove based telemanipulation system for the dlr hand, in: Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, Vol. 2, IEEE, 1998, pp. 1603–1608.

[13] T. Asfour, P. Azad, F. Gyarfas, R. Dillmann, Imitation learning of dual-arm manipulation tasks in humanoid robots, International Journal of Humanoid Robotics 5 (02) (2008) 183–202.

[14] M. Do, T. Asfour, R. Dillmann, Towards a unifying grasp representation for imitation learning on humanoid robots, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 482–488.

[15] M. Hueser, T. Baier, J. Zhang, Learning of demonstrated grasping skills by stereoscopic tracking of human head configuration, in: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE, 2006, pp. 2795–2800.

[16] S. Calinon, A. Billard, Incremental learning of gestures by imitation in a humanoid robot, in: Proceedings of the ACM/IEEE international conference on Human-robot interaction, ACM, 2007, pp. 255–262.

[17] E. Sauser, B. Argall, G. Metta, A. Billard, Iterative learning of grasp adaptation through human corrections, Robotics and Autonomous Systems.

[18] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, K. J. Kuchenbecker, Human-inspired robotic grasp control with tactile sensing, Robotics, IEEE Transactions on 27 (6) (2011) 1067–1079.

[19] B. Huang, J. Bryson, T. Inamura, Learning Motion Primitives of Object Manipulation Using Mimesis Model, in: Proceedings of 2013 IEEE International Conference on Robotics and Biomimetics. ROBIO, 2013.

[20] A. M. Okamura, N. Smaby, M. R. Cutkosky, An overview of dexterous manipulation, in: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, Vol. 1, IEEE, 2000, pp. 255–262.

[21] M. Howard, D. Mitrovic, S. Vijayakumar, Transferring impedance control strategies between heterogeneous systems via apprenticeship learning, in: Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, IEEE, 2010, pp. 98–105.

[22] T. Wimböck, C. Ott, A. Albu-Schäffer, G. Hirzinger, Comparison of object-level grasp controllers for dynamic dexterous manipulation, The International Journal of Robotics Research 31 (1) (2012) 3–23.

[23] J. Buchli, F. Stulp, E. Theodorou, S. Schaal, Learning variable impedance control, The International Journal of Robotics Research 30 (7) (2011) 820–833.

[24] D. A. Cohn, Z. Ghahramani, M. I. Jordan, Active learning with statistical models, arXiv preprint cs/9603104.

[25] B. Huang, S. El-Khoury, M. Li, J. J. Bryson, A. Billard, Learning a real time grasping strategy, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 593–600.

[26] W. Khalil, E. Dombre, Modeling, identification and control of robots, Butterworth-Heinemann, 2004.

[27] K. S. Narendra, J. Balakrishnan, M. K. Ciliz, Adaptation and learning using multiple models, switching, and tuning, Control Systems, IEEE 15 (3) (1995) 37–51.

[28] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, Neural computation 3 (1) (1991) 79–87.

[29] K. S. Narendra, J. Balakrishnan, Adaptive control using multiple models, Automatic Control, IEEE Transactions on 42 (2) (1997) 171–187.

[30] S. Fekri, M. Athans, A. Pascoal, Robust multiple model adaptive control (rmmac): a case study, International Journal of Adaptive Control and Signal Processing 21 (1) (2007) 1–30.

[31] M. Kuipers, P. Ioannou, Multiple model adaptive control with mixing, Automatic Control, IEEE Transactions on 55 (8) (2010) 1822–1836.

[32] J. J. Bryson, Modular representations of cognitive phenomena in ai, psychology and neuroscience, in: IN AI, PSYCHOLOGY AND NEUROSCIENCE, VISIONS OF MIND, Citeseer, 2004.

[33] J. J. Bryson, Structuring intelligence: The role of hierarchy, modularity and learning in generating intelligent behaviour, in: D. McFarland, K. Stenning, M. McGonigle (Eds.), The Complex Mind: An Interdisciplinary Approach, Palgrave-Macmillan, Basingstoke, 2012, pp. 126–143.

[34] N. Sugimoto, J. Morimoto, S.-H. Hyon, M. Kawato, The emosaic model for humanoid robot control, Neural Networks 29 (2012) 8–19.

[35] G. Petkos, M. Toussaint, S. Vijayakumar, Learning multiple models of non-linear dynamics for control under varying contexts, in: Artificial Neural Networks–ICANN 2006, Springer, 2006, pp. 898–907.

[36] D. J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series., in: KDD workshop, Vol. 10, Seattle, WA, 1994, pp. 359–370.

[37] P. Willett, Recent trends in hierarchic document clustering: a critical review, Information Processing & Management 24 (5) (1988) 577–597.

[38] R. de Souza, S. El Khoury, J. Santos-Victor, A. Billard, Towards comprehensive capture of human grasping and manipulation skills, in: 13th International Symposium on 3D Analysis of Human Movement, 2014.