Noname manuscript No.
(will be inserted by the editor)

A Modular Approach to Learning Manipulation Strategies from Human Demonstration

Authors · Authors

the date of receipt and acceptance should be inserted later

Abstract Object manipulation is a challenging task for robotics as the physics involved in object interaction is complex and hard to express analytically. Here we introduce a modular approach for learning a manipulation strategy from human demonstration. After recording a human demonstrating the task in different contexts, we perform modular decomposition of the control strategy, using phases of the recorded actions to guide segmentation. Each module represents a part of the strategy, encoded as a pair of forward and inverse models. All modules contribute to the final control policy; their recommendations are integrated via a system of weighting based on their own estimated error in the current task context. We validate our approach by demonstrating it on a robot platform. The task is opening a bottle cap. We show that our approach can modularize an adaptive control strategy to generate appropriate motor commands for the robot to accomplish the complete task.

1 Introduction

With robots moving into human-centered environments such as households and offices, human-like motor skills are becoming increasingly desirable. In everyday life, object manipulation is one of the most commonly used manual skills. Object manipulation includes a large category of activities ranging from the simple pick-and-place task to complicated dexterous manipulation.

Here we provide a framework for learning a human object-manipulation skill and transfering it to a robot. Generally, manipulation tasks are very difficult, due to the complicated contact situations between the manipulator and the object, and the changing kinematic and dynamic contexts that result. Humans can perform these skilled tasks and adapt to changes in context without difficulty. At the heart of this skill is prediction (Flanagan et al, 2006). Studies from neuroscience suggest that humans develop internal models for motor control, which allow us to predict the future state of the environment. By comparing the predictive state with the actual sensory state, the internal models monitor the progression of tasks, and launch any corresponding motor correction and motor reaction required to adapt to anything unexpected.

Inspired by this concept, we propose an approach to learning human adaptive control strategies. This strategy is encoded with a modular model, where each module includes a forward model for context estimation, and an inverse model for motor command generation. From multiple human demonstrations, we extract a set of strategies, each of which takes charge of one specific task context. The internal forward and inverse models are learnt within each module with a representation that can be easily transfered to a robot. When the robot executes a similar task, the forward models estimate the context of the task and 'contectualize' the inverse models, allowing them to generate the proper commands.

Our work contributes a framework composed of both automated and bespoke components for creating the modular representation of human adaptive control-strategies and to transfer these learnt internal models to a robot. To verify our approach, we use an *Opening Bottle Caps* task as an experiment. An adaptive control strategy is required here, because the friction between the bottle's and the cap's surfaces has multiple phases. We demonstrate the modularized version of the human control strategy in this task on a robot, which is used to open both familiar and novel bottles.

The rest of this article is organized as follows: Section 2 provides an overview of related work; Section 3 presents our approach of learning a multiple-module model of a human manipulation strategy. The experiments on the opening-

Address(es) of author(s) should be given

bottle-cap task and their result are shown in Section 4, along with details of the hardware specifications and the experimental setup. Section 5 concludes this work with a discussion and a look towards future work.

2 Related Work

In this section, we give a overview of area of robot learning manipulation task and modular approaches.

2.1 Learning Manipulation Tasks

Demonstration based learning has been extensively studied (Calinon et al, 2007; Dillmann, 2004; Kulić et al, 2012) 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 as a result the control strategy is hard to derive. Modeling expert's demonstration of strategies has been used as an alternative to the analytical solution.

Two major forms of demonstration are used in teaching manipulation tasks: kinematics teaching and tele-operation. In kinesthetic teaching, human directly contact with the robot and guide their movements to accomplish a task (Korkinof and Demiris, 2013; Pais and Billard, 2014; Pastor et al, 2011; Li et al, 2014). 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 at most. To control multi-finger hands, some researchers use tele-operation (Bernardino et al, 2013; Kondo et al, 2008; Fischer et al, 1998). 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 to the human demonstrator during manipulation.

In some studies, the human demonstrate manipulation tasks with their own bodies (Asfour et al, 2008). 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 needs to be transferred to robots. Various mapping methods have been proposed (Hueser et al, 2006; Asfour et al, 2008; Do et al, 2011; **?**), while human correction (Calinon and Billard, 2007; Sauser et al, 2011; Romano et al, 2011) and self-correction via learning (Huang et al, 2013a) are proposed as alternative solutions. In general, how to effectively transfer human skills to robots skill remains a challenge.

We propose a method to allows the subject to perform natural feedback control strategies in demonstration, while the strategy can then be easily transfer to any robot platform. In our task demonstration, a human wears tactile sensors and directly interacts with objects. The demonstration is expressed from an object centric viewpoint. The object centric viewpoint (Okamura et al, 2000; Jain and Kemp, 2013; Li et al, 2014) considers a manipulation task from the object's perspective. This suggests that the goal of a manipulation task is to produce a desire object movement rather than a robot end-effector movement. Our approach takes this principle and learns a control strategy for producing a desired object behavior. The demonstrated strategy expressed from the object perspective can then be transferred to a robot platform by covering the exert force to robot joint torque.

With the object centric viewpoint, we need to learn the correlation between the exerted force of the object and the desired object motion. A classic model of this correlation is impedance (Howard et al, 2010; Wimböck et al, 2012). Given the desired impedance of a task, we can compute proper motor commands for the robot to accomplish it. Fix impedance control is limited to simple tasks. In many manipulation tasks such as opening a bottle cap, variable impedance is required: at the beginning we need a large impedance to break the contact between the bottle and the cap, and later we need a small impedance to drive the cap smoothly. For such tasks fix impedance control will either lead to task failure or cause hardware damage. However, computing the impedance for a given task involving variable impedance is difficult. In many cases the impedance is roughly approximated by a linear model, but this is inadequate for nonlinear tasks.

Variable impedance can be learnt by human physically correcting the robot impedance, i.e. wiggling the robot arm, in different stages of the task (Kronander and Billard, 2012). For learning manipulation, however, wiggling the robot fingers will interrupt the task and may cause task failure. Variable impedance can also be learnt by the reinforcement learning algorithm Policy Improvement with Path Integrals (PI^2) with a task specific cost function (Buchli et al, 2011). Designing this cost function requires insight into the task and usually is difficult. In our approach, we directly model the correlation of the exerted force and the object motion by a nonlinear statistical model.

Among the approaches mentioned above, single model is built for the entire task. For tasks involving of multiple phases of dynamics, e.g. friction, single control strategy may be inadequate. To handel varying task contexts, robots operating in human centric environment need to be equipped with multiple strategies. In the next section we

give a brief overview of the multiple model approach (modular approach).

2.2 Modular Approach

Modular approach is used in adaptive control and its benefit has been long discussed (Jacobs et al, 1991; Narendra and Balakrishnan, 1997). 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 (Khalil and Dombre, 2004) are inadequate for these tasks, as instability or error may occur during the optimization of the model variables. To fast adapt, the multiple model approach (Narendra et al, 1995), referred as modular approach here, is proposed. In this approach, multiple controllers are designed, each of which in charge of a certain task context. During control, the task context is estimated online and the corresponding controllers are activated. Some recent work (Fekri et al, 2007; Kuipers and Ioannou, 2010) presents promising modular based approaches. It has also been shown to be an effective architecture of building intelligent systems (Bryson, 2004). In robotics, it is particular useful for tasks in non-stationary environments (Sugimoto et al, 2012).

The modular approach we take is inspired from MOSAIC (Haruno et al, 2001). MOSAIC (Modular selection and identification for control) is a paradigm of multiple module control, where each module is composed of a forward model and an inverse model. The forward models are responsible for estimating the task context in real time, and the inverse models are used to generate appropriate motor command for the current context. The inverse models are weighted by the accuracy of the estimations of their corresponding forward models. The final motor command is the linear combination of the commands factored by their weights.

We take the paradigm of MOSAIC but implement the multiple modules model in our own manner. In the early work, Wolpert and Kawato (1998) use Artificial Neural Network (ANN) to encode the internal models, i.e. the forward models and the inverse models. The variance of a forward model, which decides how much the multiple modules collaborate, has to be manually tuned. The later work (Haruno et al, 2001) fixes this hand tuning problem by modeling the transition between modules by a Hidden Markov Model (HMM) and optimizing the variance with the Expectation Maximization algorithm (EM). In this method the forward models are approximated by linear systems. To solve the hand tuning problem of the variance but do not restrict the complexity of the internal models, we encode our internal models with the Gaussian Mixture Model (GMM) (Cohn et al, 1996). Training the GMM by the EM algorithm, we compute the optimal values of the models' parameters. GMM has been shown to be an efficient model in capturing the nonlinearity of data (Calinon and Billard, 2007; Sauser et al, 2011; Huang et al, 2013b), this allows us to approximate more complex internal models.

In another work the forward model and inverse model are united to a single model (Petkos et al, 2006). For that particular task the action (a_t) taking the current task state (s_t) to the desired task state (s_{t+1}) is always unique. However, in many cases this mapping is not unique and hence the inverse model has to include extra variables in order to resolve the non-uniqueness. To take a more general approach, we build the forward and inverse model separately.

In our model the final motor command is the sum of the outputs of all modules. Compare to the switching modular method (Narendra and Balakrishnan, 1997), where only one module will be activated to generate motor command per time step, our approach requires less number of modules to approximate the system dynamics.

Despite the many discussions of the modular approach, its application in robotics is not yet wide-spread. One main challenge is how to modularize a given task, i.e. determine the number of modules and decompose the task. Similar problem is studied in the research of motion primitives. Kulić et al (2008) use a hierarchical clustering method to extract primitives from human motion sequence. Different cut off parameters are tested to evaluate the trade off effect between facilitating quick group formation and introducing misclassification. Our modularize approach is similar to this and one step further. We cluster the demonstration data with an hierarchical method and regard each cluster as one module. Instead of hand tuning the cut off parameter, we determine its value by the variance of the data. This provides us with a proper grouping of the data, which can generate proper motor command for control.

Figure. 3 illustrates the workflow of our approach. To the best of our knowledge, our work is the first realization of the modular approach in learning an object manipulation task with a real robot.

3 Methodology

This section presents the method we used to modularize human demonstration of manipulation tasks. 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).

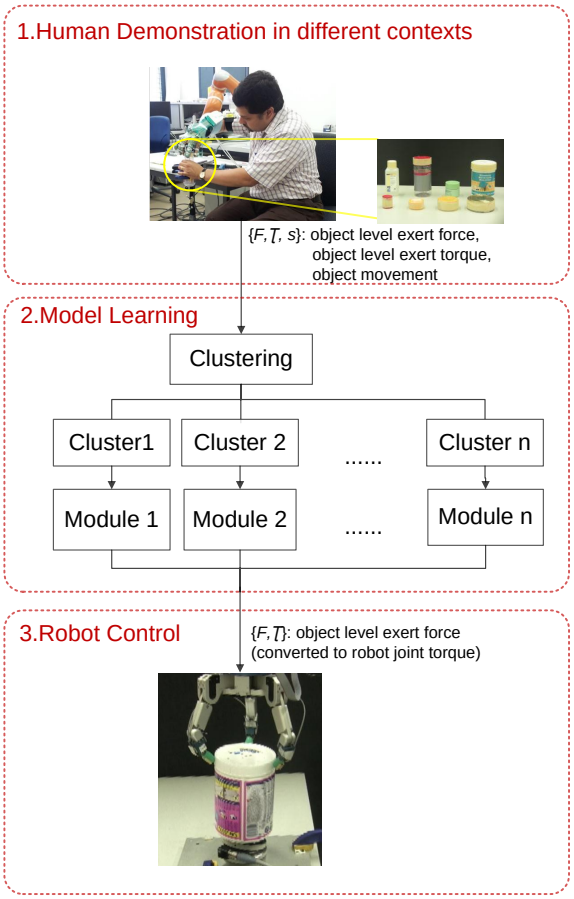
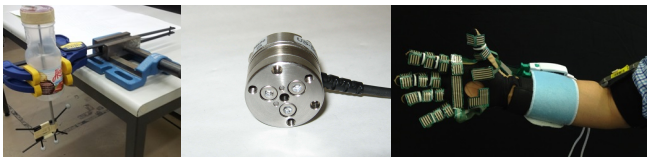


Fig. 1: System overview. Our system takes a three-step approach. 1) Human demonstrate a task in different contexts. In the opening bottle cap experiment, the demonstrations are done with different bottles and caps. The object level exert force and torque, the object movements are used for training. 2) Clustering human control strategies. Each cluster is modeled as one module. 3) Use the multiple modules to compute motor commands to control the robot.



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

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

3. Use the multiple modules 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. Figure. 2 shows a few sensors we used in the opening bottle caps task. Details will be explained in section 4.1.

In the demonstrations, the demonstrator performs a task a number of times to generate enough data for capturing the key features of it. Further, the demonstrator performs the task under different conditions, e.g. friction conditions, in order to explore how humans adapt 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. These wide range demonstrations are then used to learn a multiple module model. Details will be explained in Section 4.1.

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 demonstrator'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 and torque on the object, the robot joint torques can be deduced by the Jacobian matrix (Okamura et al, 2000). To this end, we focus on the force-torque-displacement tuple: $\{F, \tau, s\}$ demonstrated in the task, where F is the exert force in all directions including the grip force, τ is the exert torque in all directions and s is the object displacement. In later sections, we refer $\{F, \tau\}$ as the motor command (action) with notation $\{a\}$. In each demonstration, a time series of the tuple is recorded.

3.2.2 Decide number of modules

Due to object interaction, a manipulation task frequently encounter abrupt changes of the system dynamics, e.g. transfer between statuses with no contact and with contact, between statuses driven by static friction and by driven dynamic friction. Different strategies should be used to handle different dynamics and hence a multiple module model is desired. Our approach is to extract strategies from demonstration and build one module for each of the strategies.

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 regard different setups as the same task context and handle them 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. 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 (Berndt and Clifford, 1994).

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 setup as a reference. As mentioned above, under the same setup 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 setups, is used as the threshold for the clustering.

Many of the clustering methods require the specification of the number of clusters. In our case, however, the number of clusters is an unknown variable. Therefore we use the hierarchical agglomerative clustering method (Willett, 1988) to group our data. Agglomerative clustering is a method that merges similar 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

```
Init(): Make each time series a cluster
mergeable = true
function MERGE(all clusters, distance matrix)
    while mergeable is true do
        mergeable = false
        for each cluster do
            ClusterA = current cluster
            ClusterB = nearest neighbor of ClusterA
            if distance(ClusterA,ClusterB) < clustering threshold
then
                Merge ClusterB into ClusterA
                mergeable = true
            end if
        end for
    end while
end function
```

When the clusters can not be merged further, we define the number of modules for 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 a specific task context.

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 demonstrator. The demonstrator 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.

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 experiment, s is the one dimensional cap angular displacement 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 3.3.1). At the other hand, as a generative model GMM is able to generate new data from the model, i.e. generate motor commands. This is done by the *Gaussian Mixture Regression* (GMR). Table 1 explains the encoding process of GMM and the computation process of GMR.

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 3.3.1). Figure 3 shows the basic control flow of a forward-inverse model pair.

We encode the forward model Ω_F by the joint distributions of the current system state (object displacement), previous system state and the previous motor command, i.e. $p(s_t, s_{t-1}, a_{t-1} \mid \Omega_F)$, and similarly encode the inverse model Ω_I by the joint distributions of the current system state, the desired next system state, previous motor command and the current motor command, i.e. $p(s_t, s^*_{t+1}, a_{t-1}, a_t|\Omega_I)$. The previous motor command a_{t-1} is necessary for the inverse model. In some tasks, the system status can remains unchanged for a certain period until the exert force reaches a threshold to change it. This will cause degeneracy in the inverse model and we hence include the previous motor command in the model to tackle it.

By clustering the training data into different groups (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.

With a GMM, the joint distribution Ω of a set of variables $\{\eta\}$ is expressed as a sum of N Gaussian components:

$$p(\eta \mid \Omega) = \sum_{n=1}^{N} \pi_n p(\eta \mid \mu_n, \Sigma_n) = \sum_{n=1}^{N} \pi_n \frac{1}{\sqrt{(2\pi)^D \mid \Sigma_n \mid}} e^{-\frac{1}{2}(\eta-\mu_n)^\top \Sigma_n^{-1}(\eta-\mu_n)} \quad (1)$$

where π_n is the prior of the n-th Gaussian component and the μ_n, Σ_n the corresponding mean and covariance, D the number of the variables.
Gaussian Mixture Regression (GMR) allows us to estimate the conditional expectation value of a variable η^e given a query point η^q where $\{\eta\} = \{\eta^q, \eta^e\}$. To compute this expectation value, first we define:

$$\mu_n = \begin{pmatrix} \mu_n^q \\ \mu_n^e \end{pmatrix} \qquad \Sigma_n = \begin{pmatrix} \Sigma_n^{qq} & \Sigma_n^{qe} \\ \Sigma_n^{eq} & \Sigma_n^{ee} \end{pmatrix} \quad (2)$$

Secondly we compute the expected distribution of η^e from the $n-th$ component:

$$\hat{\mu}_n = \mu_n^e + \Sigma_n^{eq}(\Sigma_n^{qq})^{-1}(\eta^q - \mu_n^e) \quad (3)$$

$$\hat{\Sigma}_n = \Sigma_n^{ee} - \Sigma_n^e q(\Sigma_n^{qq})^{-1}\Sigma_n^{qe} \quad (4)$$

Finally, all the N Gaussian components are taken into account, and the expectation value of variable η^e is computed as the mean $\hat{\mu}^e$ with the covariance $\hat{\Sigma}^{ee}$:

$$\hat{\mu}^e = \sum_{n=1}^{N} \beta_n \hat{\mu}_n \qquad \hat{\Sigma}_n^{ee} = \sum_{n=1}^{N} {\beta_n}^2 \hat{\Sigma}_n \quad (5)$$

where

$$\beta_n = \frac{\pi_n p(q|\mu_n^q, \Sigma_n^{qq})}{\sum_{n=1}^{N} \pi_n p(q|\mu_n^q, \Sigma_n^{qq})} \quad (6)$$

Note that in a multiple module model, different module may have different number of Gaussian components.

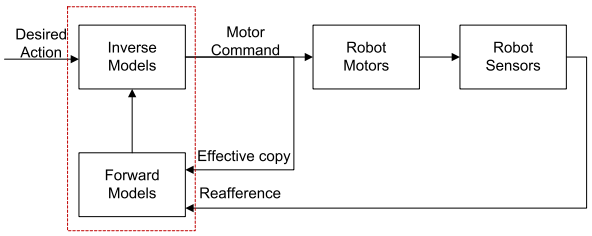
Table 1: Encoding process of GMM and computation process of GMR

3.3 Multiple module adaptive control

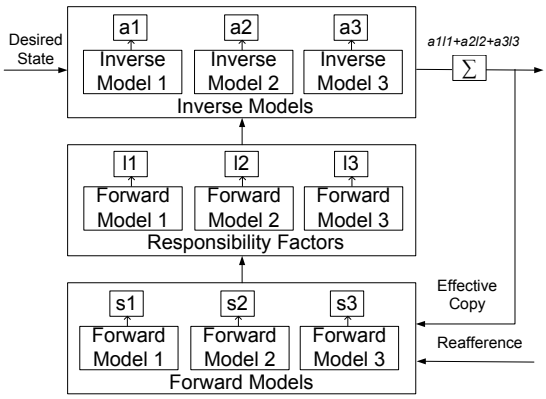
Once the number of modules is found and the multiple pair of forward and inverse models are learnt, they are used to compute motor commands for task execution. We consider the human motor system acts upon by motor command a_t at time t with current system status s_t. A function f maps a_t and s_t to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

$$s_{t+1} = f(s_t, a_t) \quad (7)$$

The goal of the controller is to generate a motor command a_t that bring the current system status from s_t a desired state



(a)



(b)

Fig. 3: Control flow diagram of forward-inverse model in motor control. (a) System overview. Pairs of forward and inverse models work together to generate final motor command. The mechanism inside the red box is shown underneath. (b) A example of a 3 modules model. The forward models predict current task context (s1, s2, s3) and estimate the accuracy of their prediction (l1, l2, l3). These accuracy is called "Responsibility Factors" as they decide how much responsibility each inverse model should take in the final command. The Inverse models generate commands (a1, a2, a3) and the final command is the summation of them factorized by their responsibility factor (a1l1+a2l2+a3l3).

s^*_{t+1}:

$$a_t = g\left(s^*_{t+1}, s_t\right) \tag{8}$$

Equation 7 represents the forward model and Equation 8 represents the inverse model. In the modular approach, it takes two steps to compute the motor command a_t:

1. Anticipate the sensory output and compute the responsibility factor λ_t
2. Compute motor command by each inverse model and compute the final motor command a_t

3.3.1 Responsibility factor

In a modular approach, choosing the proper modules to control the system is a crucial step. For this we rely on the responsibility factors, which act as the weights of the inverse models. The responsibility factor is a measurement of the reliability of using one module to represent the current system context.

With the k-th forward model we can anticipate the current state $\hat{s}^k_t$ by the *GMR* (Table 1):

$$\hat{s}^k_t = E\left(s_t | s_{t-1}, a_{t-1}, \Omega^k_I\right) \tag{9}$$

By comparing the anticipated current state $\hat{s}^k_t$ with the actual current state s_t detected by the sensors, we can evaluate how good the k-th module represents the current system. The actual current state, previous state and the previous motor command form a data point $\eta_t = \{s_t, s_{t-1}, a_{t-1}\}$. As the forward models are built as GMMs, it is easy to compute the likelihood of one data point belongs to a particular model: $p(\eta_t|\Omega^j_F)$. The discrepancy between $\hat{s}^k_t$ and s_t is embedded in this likelihood and hence in practice we only compute the $p(\eta_t|\Omega^j_F)$ and skip $\hat{s}^k_t$. The responsibility factor of the k-th inverse model is the likelihood of the data point η_t belongs to the k-th module, normalized by the total sum:

$$\lambda^j_t = \frac{p(\eta_t|\Omega^j_F)}{\sum_{k=1}^K p(\eta_t|\Omega^k_F)} \tag{10}$$

where K is the number of modules [1]. At every time step, we compute the responsibility factor for each module. The final motor command at that time step is the linear combination of the commands generated from each inverse model multiplied by their responsibility factors.

3.3.2 Generate motor command by Inverse Model

The motor command a^k_t for the k-th inverse model is computed by GMR with the steps explained in Table 1. At each time step, the responsibility factors λ^k_t weight its corresponding inverse model: the higher the responsibility is, the more response the inverse model takes in the control. The final motor command generated by this multiple model system is:

$$a_t = \sum_{k=1}^K \lambda^k_t a^k_t = \sum_{k=1}^K \lambda^k_t E\left(a_t \mid s^*_{t+1}, s_t, a_{t-1}, \Omega^k_I\right) \tag{11}$$

These three steps are all 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.

[1] In the case that the dominator is very close to zero, the whole control process will be terminated as it indicates that the model is used on a different task.

4 Robot Experiment

The proposed multiple module approach is implemented on a real robot system (the 7 DOF Light Weight KUKA robot arm and the 4 DOF Barrett Hand) for a particular manipulation task: opening bottle caps. The target of this task is to unscrew a tighten cap until it can be lifted from the bottle. 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.

The friction between the bottle and the cap plays an important role in the task: it largely determines the exert torque required to open the cap. However, the friction, and the way it changes from screwed to unscrewed, varies among different bottles.

Estimating the friction coefficient (FCO) solely according to the material is difficult, as it is effected by many factors such as the load force, movement velocity, contact surface situation, composition of the material, temperature and etc. (Gustafsson, 2013). A deterministic control strategy based on the value of FCO is not practical in this task. A small estimation error in the FCO may produce either too small torque, which leads to task failure, or too large torque, which may cause hardware damage. Therefore an adaptive control strategy is desired in this task. We use our multiple module approach to model the adaptive strategy.

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 but not an easy one for robot. Before the task begins, human does not possess any information about the tightness of the cap. This information can only be estimated once the task is started. 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. This plan can only be make in real time as the contact surface condition changes along the task process. Human have to cope with these uncertainties and adapt to the changes. Figure 4 shows three different patterns of human control strategies for three different contexts. This task requires an adaptive strategy that controls the turning torque, griping force and the displacement of the cap. Learning from human demonstration allows us to gain such a control strategy without fully analyzing 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.

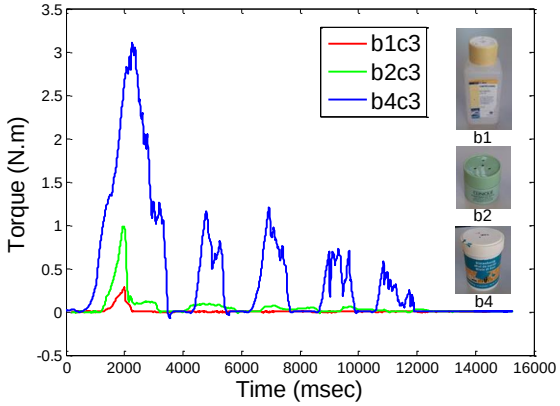


Fig. 4: Exert torque for opening three different bottles.

4.1.1 Demonstration in different task contexts

The experiment starts with human demonstration. In order to explore different task contexts, we demonstrated the task with different setups, which are the combination of four different plastic bottles ($b1-b4$) and four different plastic caps ($c1-c4$) (Figure. 5). According to the surfaces condition of the bottles and the caps, the difficulty of opening the bottles varies. $b1-b4$ are labeled by increasing difficulty. The bottle $b1$ is the easiest one, which originally contains body lotion. We lubricate bottle $b1$ with its body lotion to make it even easier. The bottle $b4$ is the most difficult one and it originally contains honey which is very sticky. We leave honey on the surfaces of $b4$ to make it more difficult. The difficulty is estimated qualitatively. It is judged according to the friction coefficient between the contact surfaces. Generally speaking, the friction coefficient between lubricated surfaces is smaller than between dry surfaces, while between smooth surfaces is smaller than between sticky surfaces [2]. The $c1-c4$ are labeled by the increasing diameters of the caps.

We chose to vary the setups in surface condition and cap size as these are the main variances between different bottles effecting the control strategy. The intention is to see

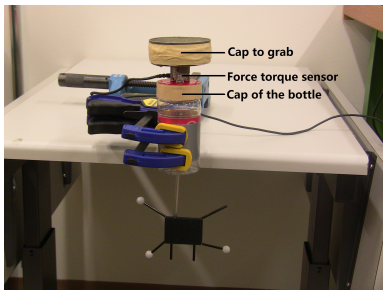
[2] Precise value of friction coefficient between plastics varies by type of the plastic. According to an internet resource, the dry dynamic friction coefficient between plastic-plastic surface is 0.2-0.4 and the lubricated dynamic friction coefficient is 0.04-0.1 (http://www.tribology-abc.com/abc/cof.htm)

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

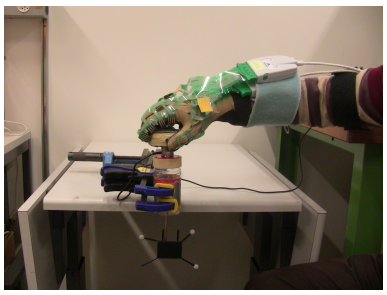
	 Cap 1 25*mm*	 Cap 2 42*mm*	 Cap 3 56*mm*	 Cap 4 80*mm*
 Bottle 1			b1c3	
 Bottle 2			b2c3	
 Bottle 3	b3c1	b3c2	b3c3	b3c4
 Bottle 4			b4c3	

Table 2: Different setups of bottles and caps for demonstration. Bottle 1 to 4 are in increasing order of the difficulty to open. Cap 1 to 4 is in increasing order of the cap sizes, whose diameters are shown.

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 (Figure. 6). To investigate the effects of different caps and different bottles separately, we conduct two groups of demonstrations: a fix bottle with 4 different caps ($b3c1, b3c2, b3c3, b3c4$) and a fix cap with 4 different bottles ($b1c3, b2c3, b3c3, b4c3$). Demonstrations on the first group allow us to explore human grasping strategies with different cap sizes. Demonstrations on the second group allow us to explore human control strategies in adapting to different bottle condition. In total, we have seven different setups for the human demonstration (Table 2).



(a)



(b)

Fig. 6: Experimental setup for the task of opening a bottle cap. (a) Setup b3c4: bottle 3 combined with cap 4 (cap to grab). A force-torque sensor is mounted between the "cap of the bottle" and the "cap to grab", so that the exert force and torque can be measured. 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 demonstrator 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[3] for exert torque (τ);
2. OptiTrack[4] for cap displacement (s);
3. Tekscan[5] 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

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

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. It is mounted between the bottle and the cap (Figure. 6). During the task, the demonstrator 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 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 (Figure. 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 (Figure. 4.2). The method of mounting the sensors to the glove is detailed in (de Souza et al, 2014). 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.

Exert torque 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.

Object displacement From the OptiTrack, the cap's displacement is originally 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.

Grip Force 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. Figure. 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.[6] 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 Figure. 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 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 we need to break at this stage is decide by the static FCO. Once the cap starts to move, the FCO between bottle and cap transits to kinetic FCO, which

[6] 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.

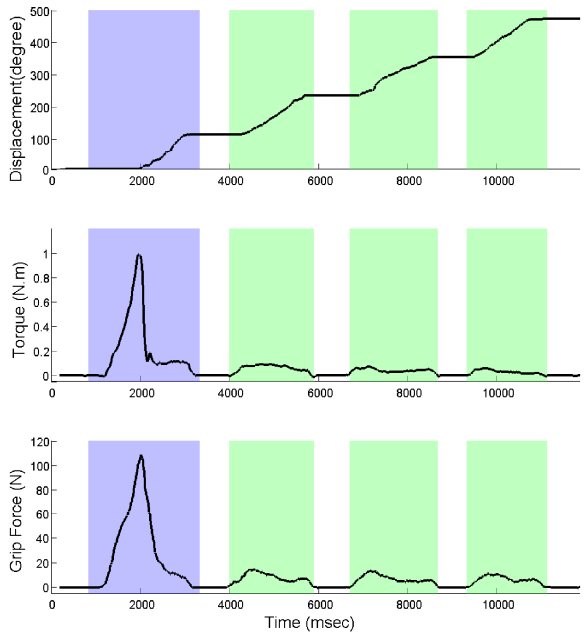


Fig. 7: Aligned data of all three channels. Highlighted parts mark the turning process: blue blocks denote the first cycle, i.e. the phase I, and green blocks denote the later cycles, i.e the phase II. Phase I is significantly different from the phase II.

is usually smaller than the static FCO for the same surface condition. 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, 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

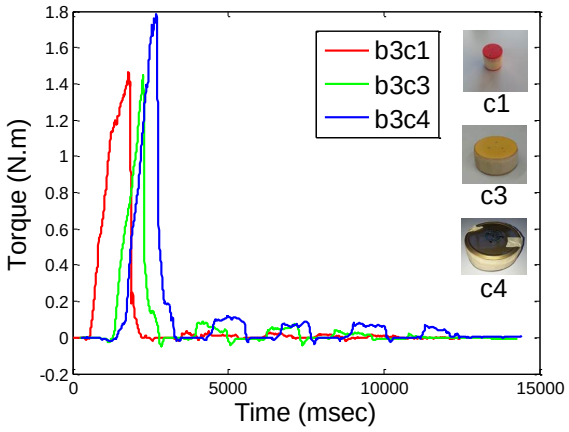


Fig. 9: Exert torque for opening bottle b3 with three different cap sizes.

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 (Figure. 8). As can be told from the heatmap, the trials with the same setup and in the same cycle are very similar to each other. Hence we regard these trials represent the same control strategy and use their variance as the criterion of the clustering. From this heatmap we can also see that within the same cycle, the trials with the same bottle but with different caps, e.g. $b3c1, b3c3$ and $b3c4$, are similar to each other. In the first cycle, the trials with the same cap but with different bottles, e.g. $b1c3, b2c3, b3c3$ and $b4c3$, are significantly different from each other. In the the later cycles, this difference decrease gradually. This result shows that in the opening bottle cap task, the surface condition between the bottle and the cap plays an important role in the control strategy, while the role of cap size is relatively minor. Figure 9 shows three trials of opening bottle b2 with different sizes of caps. It can be seen that their patterns are similar.

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 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 $b3c1, b3c2, b3c3, b3c4, b2c3$ and phase II of $b4c3$, 24 time series;

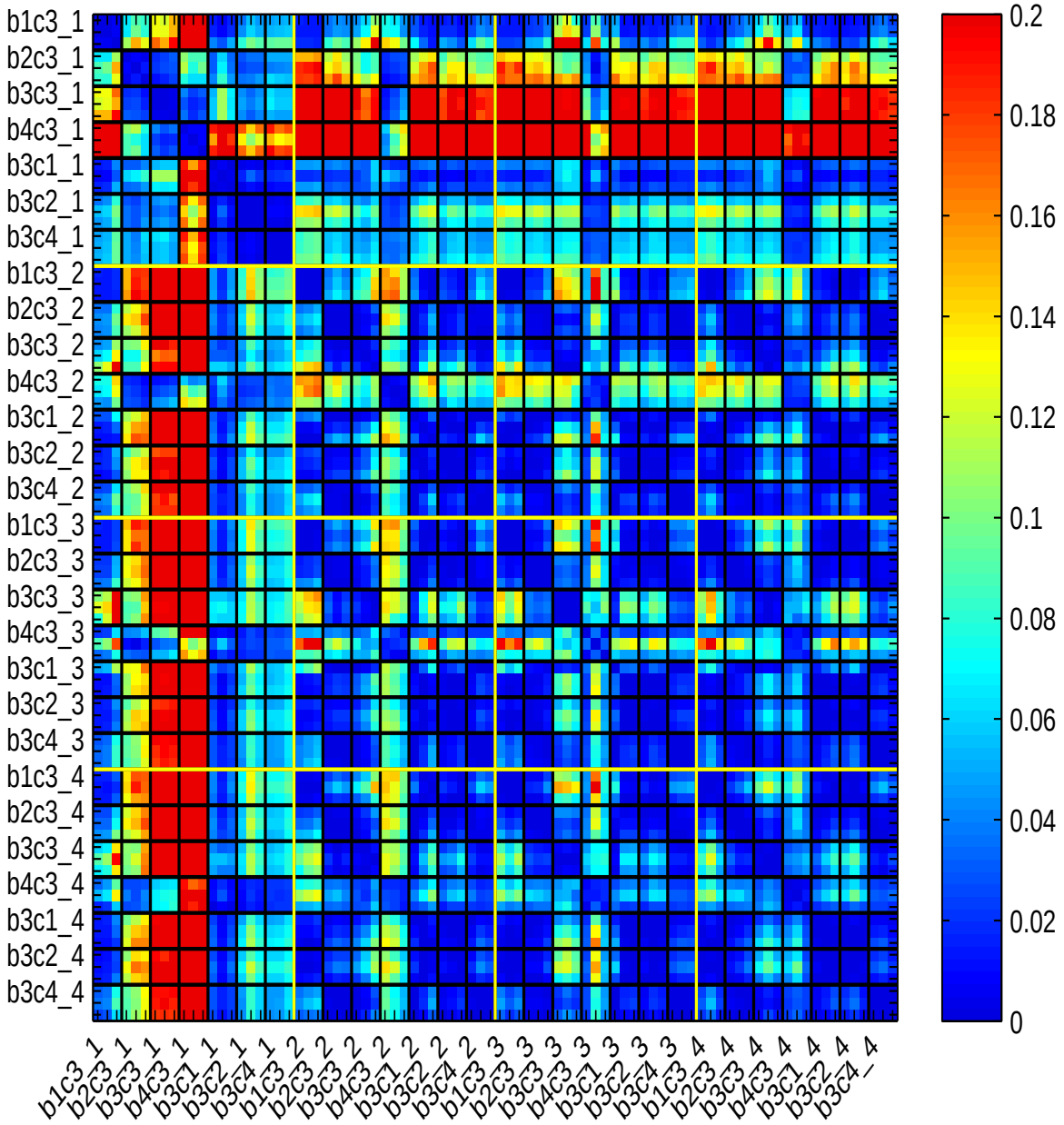


Fig. 8: A heatmap representation of the distance matrix of 84 time series (7 setups × 4 cycles × 3 trials). The labels are in the format of "setup_cycle". For example, "b1c3_1" represents the first cycle of the b1c3 setup. The yellow lines divide the x and y axis by the 4 cycles and hence form 16 big blocks. In each big block, the black lines divide the x and y axis by the 7 setups and hence form 49 small blocks.

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 3. 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.

		 Cap 1	 Cap 2	 Cap 3	 Cap 4
 Bottle 1	Phase I			(b1c3) Cluster 3	
	Phase II			Cluster 3	
 Bottle 2	Phase I			(b2c3) Cluster 2	
	Phase II			Cluster 3	
 Bottle 3	Phase I	(b3c1) Cluster 2	(b3c2) Cluster 2	(b3c3) Cluster 2	(b3c4) Cluster 2
	Phase II	Cluster 3	Cluster 3	Cluster 3	Cluster 3
 Bootle 4	Phase I			(b4c3) Cluster 1	
	Phase II			Cluster 2	

Table 3: Clustering result

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 $p\{s_t, st+1, a_t, a_{t-1} \mid \Omega_I\}$. For each model, the number of Gaussians is determined by the Bayesian information criterion (BIC). We use 25 Gaussian for cluster 1, 40 for cluster 2 and 15 for cluster 3. Their BIC tests are shown in Fig 10.

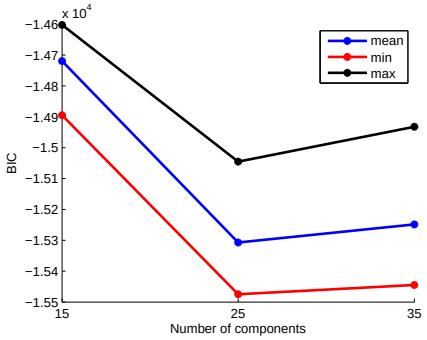
4.4 Generating motor command for manipulation

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.

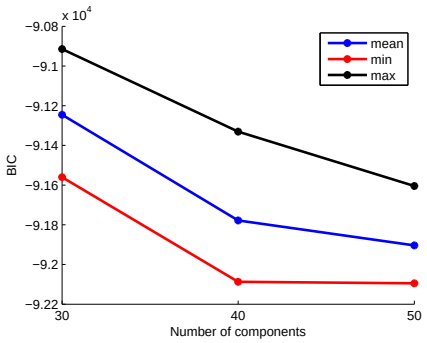
In this experiment, we control the wrist joint (last joint of KUKA) for producing torque to turn the bottle cap. A

Algorithm 2 Control Algorithm

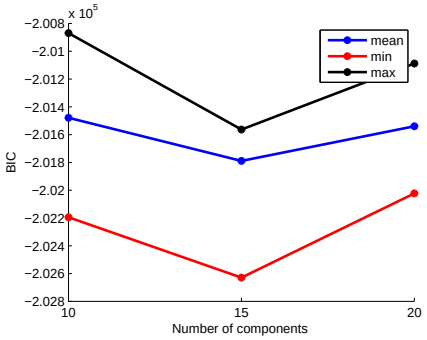
```
for r = 1:4 do
    REACHING(): Robot move to initial position
    function TURNING()
        Read previous sensor information {s_{t-1}, τ_{t-1}, F_{t-1}}
        for k=1:3 do
            ŝ^k = FORWARD(s_{t-1}, T_{t-1}, Ω_F^k)
        end for
        for k=1:3 do
            λk = ResponsibilityFactor(ŝ^k, 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} = Σ_{k=1,2,3} λk{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
```



(a) Cluster 1. Optimal number of Gaussians is 25.



(b) Cluster 2. Optimal number of Gaussians is 40.



(c) Cluster 3. Optimal number of Gaussians is 15.

Fig. 10: BIC test result for clusters.

force torque sensor is fixed under the bottle to provide torque feedback. Each finger of the Barrett hand is mounted with a *Syntouch*[7] tactile sensor, which is calibrated 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 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 validated the algorithm to control cap opening in our robot. We first tested the ability of the system to open 2 of the bottles seen during training (b1 and b4). We then tested the generalization capacity of the system by opening two bottles (b5 and b6) not seen during training. Bottle b1 and b4 are the easiest and most difficult bottle to open in the training set. Bottle b5 is a large bottle, which is hard for human to grab and open. Bottle b6 is a glass bottle with a plastic cap. The surface interaction between these two materials is not demonstrated. As the Barrett hand is significantly larger than a human hand, $b1, b4, b6$ are mounted with $c5$ (the cap of $b5$ with diameter $110mm$) on the top to ensure a firm grasp. In total 4 different setups are used in the experiment: $b1c5, b4c5, b5c5$ and $b6c5$. As discuss above, the size of the cap has minor effect on the control strategy. Therefore we expect the setups $b1c5$ and $b4c5$ will result in a similar behavior as those of $b1c3$ and $b4c3$ in the training. The experimental results and demonstration snapshots are shown

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

in figures 11- 14 [8]. Figure 15 is a similar plot to figure 4, that aligns the exert torque of the 4 experiments.

In each experiment we record the cap displacement, exert torque, and the responsibility factors of all three modules. Bottle b1 is the easiest bottle to open in the training set, the control policies of both phase *I* and phase *II* are grouped into cluster 3. As a result, in the b1 experiment the cluster 3 takes most responsibility (Figure. 11).

Bottle b4 is the most difficult bottle to open in the training set and it's phase *I* requires more than 3 *Nm* (Figure. 4). Due to the smooth contact surfaces between the Barrett hand and the cap, it is difficult to apply 3 *Nm* torque to the cap without slipping. To avoid damaging the robot, we test the b4 phase *II* only: the cap is loosely screwed on the bottle. Without knowing this, in the experiment the robot is able to properly estimate the current task context. As can be seen from the figure 12, different from b1, the dominate cluster is the cluster 2 which corresponds to the b4 phase II. This performance would be hard to achieved by a deterministic system based on expected values for friction coefficients.

Bottle b5 is a novel one but is made of similar material (plastic) to the trained bottles. A very similar torque profile to b2 and b3 is generated for b5: phase *I* is sharp, while phase *II* is flatten and significantly smaller than phase *I* (b2: Figure. 4, b3: Figure. 9, b5: Figure. 13). This is because b5 has dry contact surface as b2 and b3, and b1 is lubricated and b4 is attached with sticky material, i.e. honey.

Bottle b6 is also a novel one but with novel surface materials (plastic and glass) [9]. It's torque profile is different from what we observed in training set. Despite this, b6 is open with this torque profile generated by the three learnt modules.

With the above four different setups, the modular model adapts accordingly and successfully generate torque command to open the bottles. Successful cap opening is achieve when the cap is unscrewed far enough that it can be lifted up. Though no prior information is provided about the bottles, the task contexts are properly estimated and "contextized" motor commands are generated to unscrew the caps. These experiments show that our multiple modular approach is indeed effective in manipulation tasks.

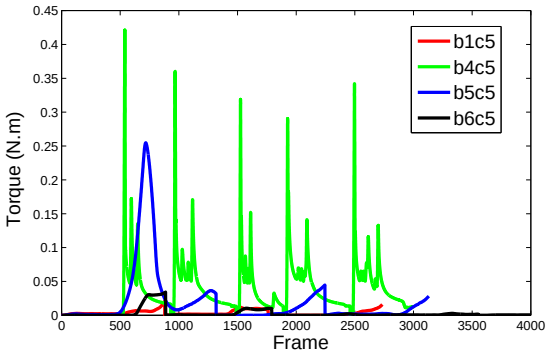


Fig. 15: Robot exert torque for opening four bottles: b1 b4 b5 b6. Time is warped and shifted for displace purpose.

5 Conclusion and Discussion

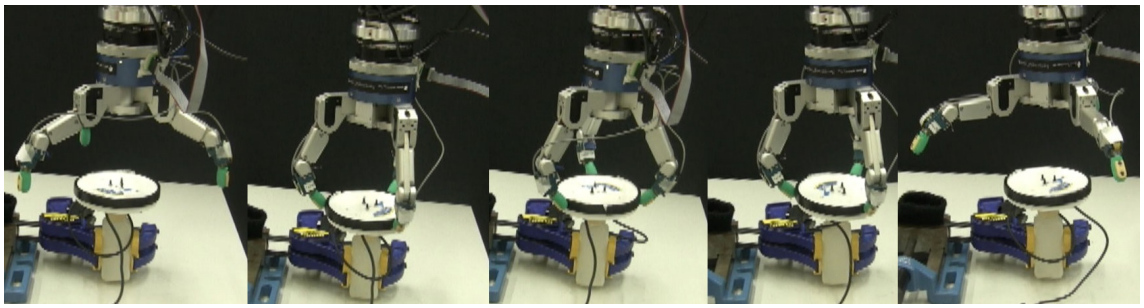
In this paper we proposed a modular approach for learning manipulation task from human demonstration. We discover the number of modules needed in a task by hierarchical clustering. From each cluster we use forward and inverse model pairs to model the motor control mechanism. The forward models predict 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 statistical 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 weighed command from each module. With an object centric viewpoint, the learnt human internal models can be easily transfer to robot. Our experiment verifies that by this modular approach, the robot can automatically recognize the current task context and compute proper motor commands to accomplish a manipulation task, i.e. opening bottle caps.

Our approach is applicable to manipulation tasks that require adaptive control strategy. It has a few benefits compare to the pervasive methods for adaptive control, e.g. classic model identification adaptive control and reinforcement learning. By imitating the human behaviors, we do not need to derive the system dynamics nor the cost function of the tasks, which involve deep insight of the task and can be painstaking. The difficulty of modeling an adaptive strategy is further reduced by a modular approach: dividing the large state space into several subspaces, where the local strategies can be approximated more accurately. With this approach, we divide the the complex human strategy into a few modules, and combine them to generate contextized motor commands.

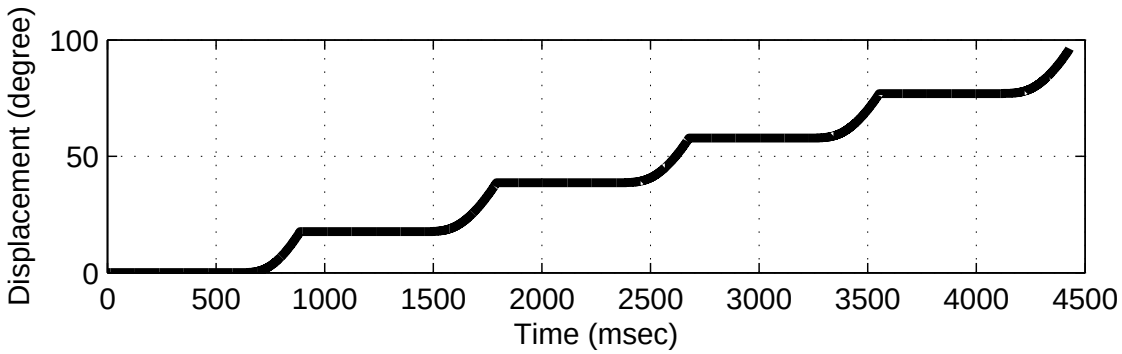
Our object centric approach is a practical approach for teaching a robot manipulation tasks that require proprioception. This allows human demonstrating the task with physical contact with the object, and hence have direct feedback from their own sensory system. We bypass the problem of direct mapping human movement to robot movement by expressing the strategy from an object centric viewpoint. This can largely benefit learning manipulation tasks such as impedance control task, as measuring human mus-

[8] Demonstration videos are available at http://www.cs.bath.ac.uk/ bh325/opencap.rar

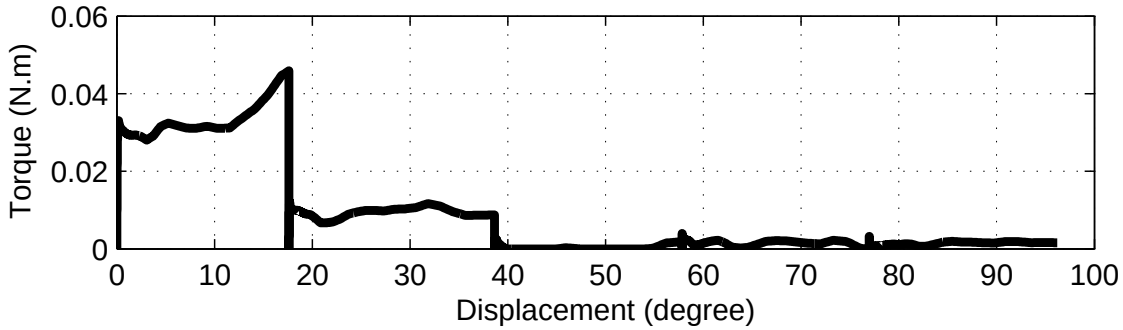
[9] A common way of measuring the FCO of a material is measuring it against metal: the static FCO between glass and metal is 0.5-0.7, while between two polythene and steel is around 0.2. This implies that the plastic and glass are indeed very different in FCO. There is not an universal measurement of the FOC between plastic and glass.



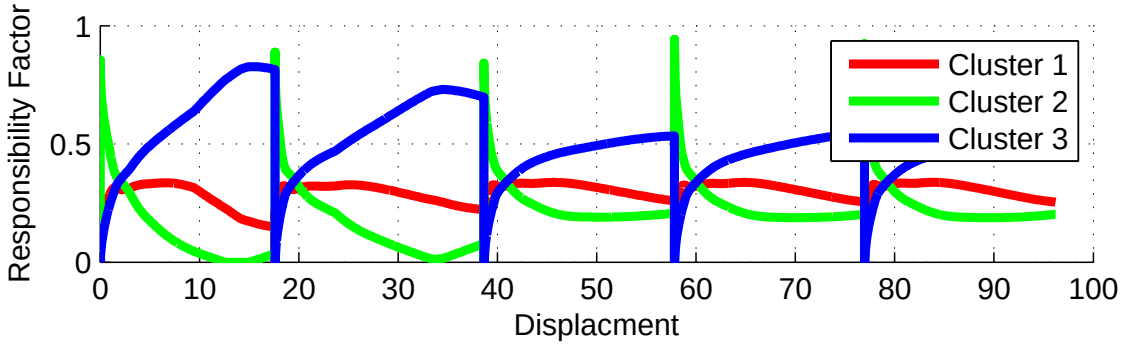
(a) Snapshots for robot opening b1 demonstration.



(b) Cap displacement during the demonstration.

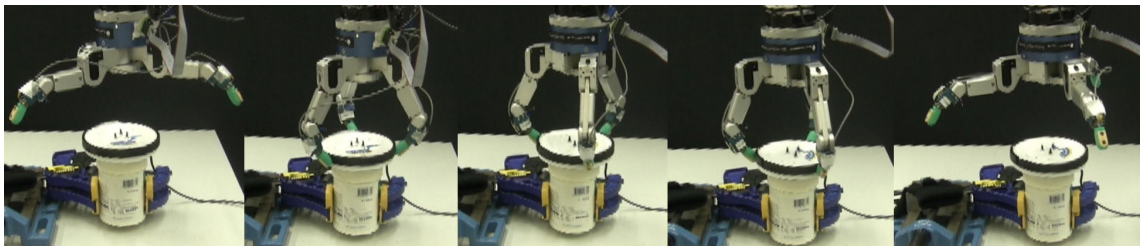


(c) Exert torque against cap displacement during the demonstration.

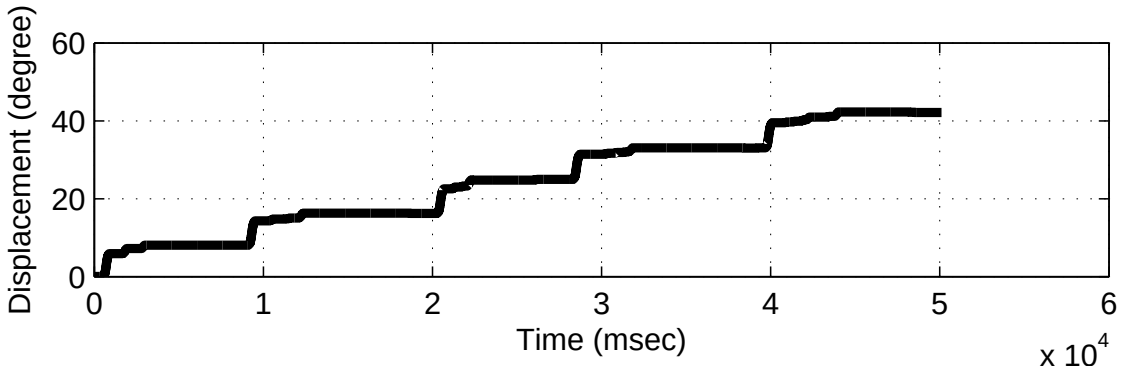


(d) Responsibility factor against cap displacement of each module.

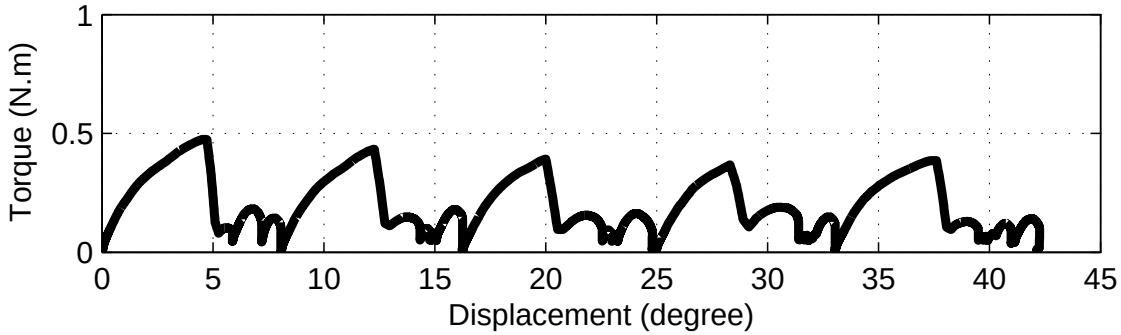
Fig. 11: Robot demonstration on opening b1.



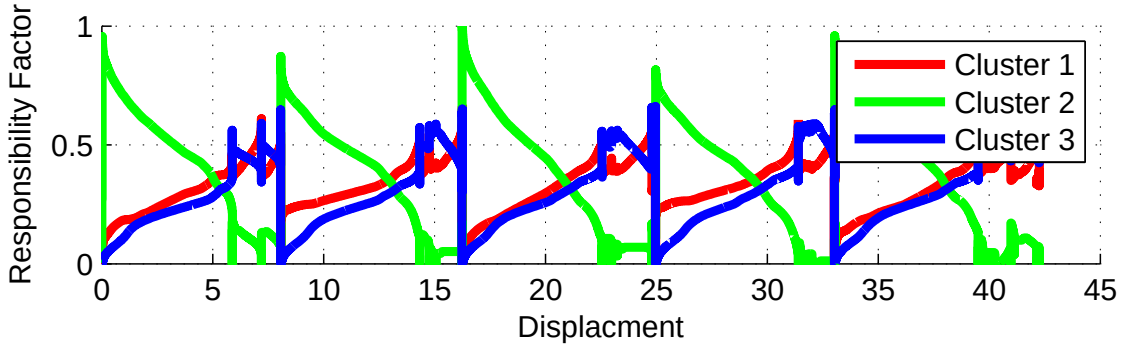
(a) Snapshots for robot opening b4 demonstration.



(b) Cap displacement during the demonstration.

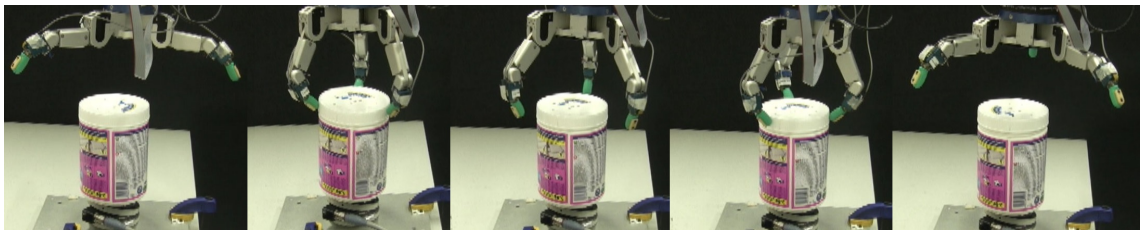


(c) Exert torque against cap displacement during the demonstration.

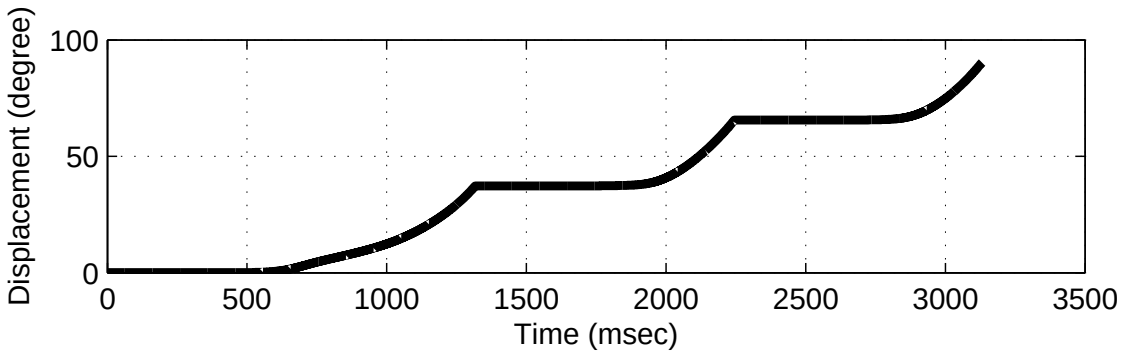


(d) Responsibility factor against cap displacement of each module.

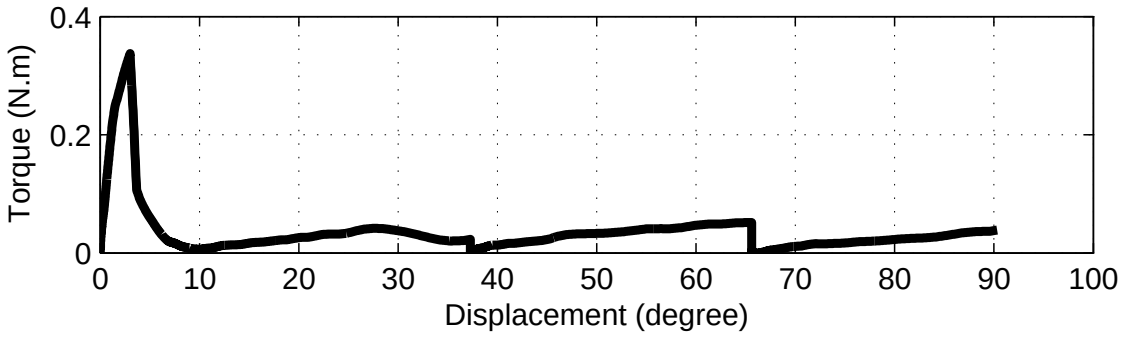
Fig. 12: Robot demonstration on opening b4.



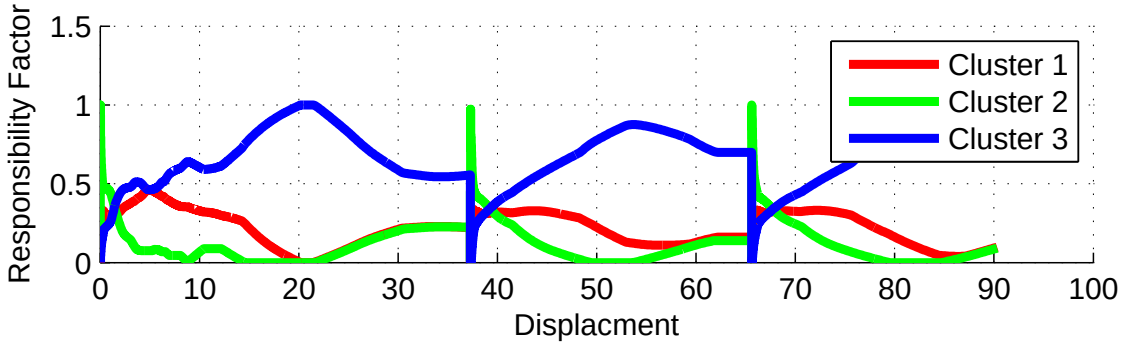
(a) Snapshots for robot opening b5 demonstration.



(b) Cap displacement during the demonstration.

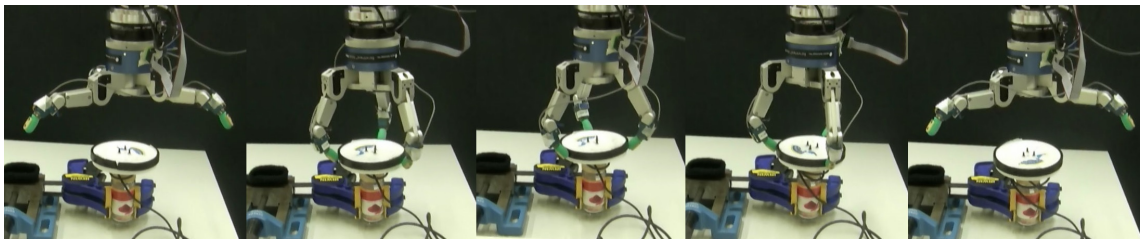


(c) Exert torque against cap displacement during the demonstration.

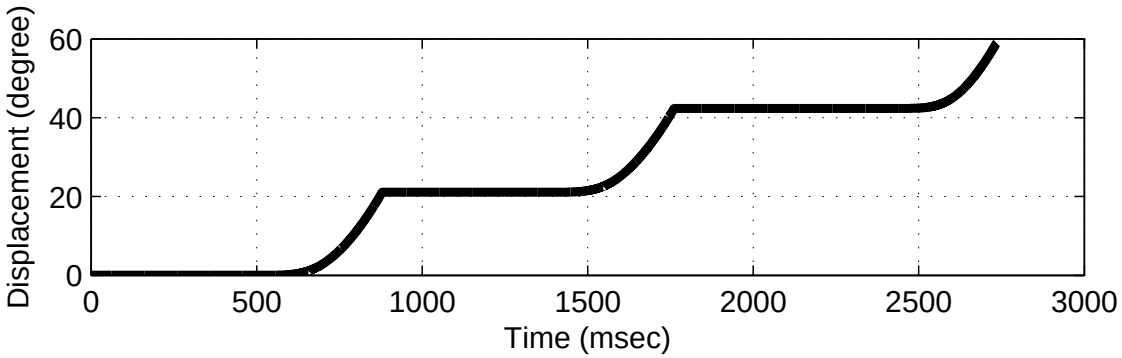


(d) Responsibility factor against cap displacement of each module.

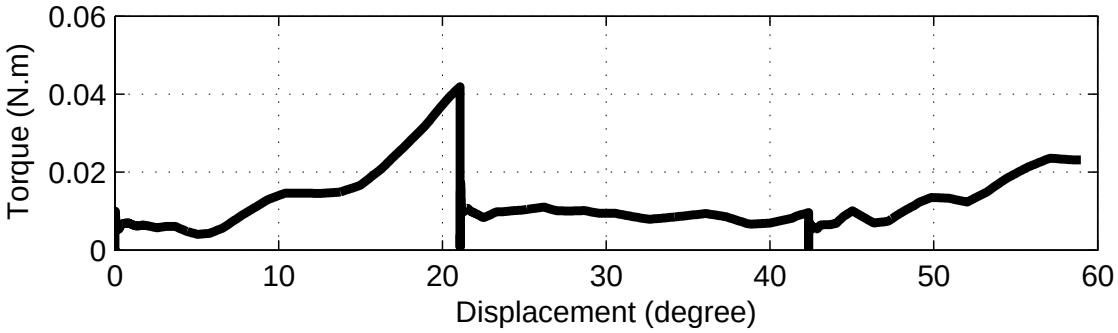
Fig. 13: Robot demonstration on opening a new bottle (b5).



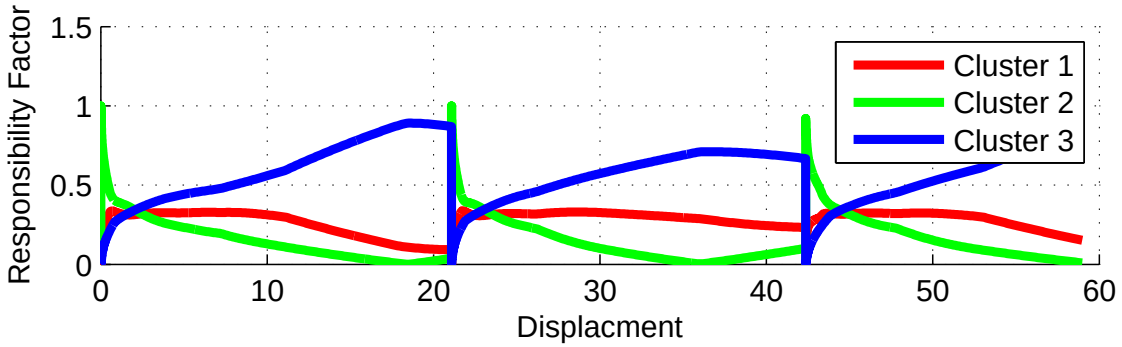
(a) Snapshots for robot opening b6 demonstration.



(b) Cap displacement during the demonstration.



(c) Exert torque against cap displacement during the demonstration.



(d) Responsibility factor against cap displacement of each module.

Fig. 14: Robot demonstration on opening a new bottle (b6).

cle impedance is hard while measuring the impedance of an object is more feasible.

We compute the final motor command by summing the output of each module. This makes an assumption that the state space is continuous. For tasks with discontinuous space, constraints have to be applied and the other control methods mentioned above are more applicable.

There are many promising directions of further studies of this work. The first is to apply this approach to other contact tasks and learn a more general human control strategy in handling the instability caused by friction. In this study, we focus on the control strategy of unscrewing the cap. We hardly analyze the effect of changing the cap size and the positioning of the fingers on the cap, which is revealed in the tactile signature. This analysis will be progressed in the future work to study task specific grasping strategy (El-Khoury et al, 2013; Dang and Allen, 2014).

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. How does the number of modules change according to the tasks is another useful information to reason about.

In summary, tasks involve multiple phases or different contexts are hard to implement by a single model. 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.

References

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

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

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, Seattle, WA, vol 10, pp 359–370

Bryson JJ (2004) Modular representations of cognitive phenomena in ai, psychology and neuroscience. In: In AI, Psychology and Neuroscience, Vision of Mind, Citeseer

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

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

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

Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. arXiv preprint cs/9603104

Dang H, Allen PK (2014) Semantic grasping: planning task-specific stable robotic grasps. Autonomous Robots pp 1–16

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

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

El-Khoury S, Li M, Billard A (2013) On the generation of a variety of grasps. Robotics and Autonomous Systems 61(12):1335–1349

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

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

Flanagan JR, Bowman MC, Johansson RS (2006) Control strategies in object manipulation tasks. Current opinion in neurobiology 16(6):650–659

Gustafsson E (2013) Investigation of friction between plastic parts. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden

Haruno M, Wolpert DM, Kawato M (2001) Mosaic model for sensorimotor learning and control. Neural computation 13(10):2201–2220

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

Huang B, Bryson J, Inamura T (2013a) Learning Motion Primitives of Object Manipulation Using Mimesis Model. In: Proceedings of 2013 IEEE International Conference on Robotics and Biomimetics. ROBIO

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

Hueser M, Baier T, Zhang J (2006) 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, pp 2795–2800

Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. Neural computation 3(1):79–87

Jain A, Kemp CC (2013) Improving robot manipulation with data-driven object-centric models of everyday forces. Autonomous Robots 35(2-3):143–159

Khalil W, Dombre E (2004) Modeling, identification and control of robots. Butterworth-Heinemann

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

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

Kronander K, Billard A (2012) Online learning of varying stiffness through physical human-robot interaction. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, Ieee, pp 1842–1849

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

Kulić D, Takano W, Nakamura Y (2008) Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. The International Journal of Robotics Research 27(7):761–784

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

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

Narendra KS, Balakrishnan J (1997) Adaptive control using multiple models. Automatic Control, IEEE Transactions on 42(2):171–187

Narendra KS, Balakrishnan J, Ciliz MK (1995) Adaptation and learning using multiple models, switching, and tuning. Control Systems, IEEE 15(3):37–51

Okamura AM, Smaby N, Cutkosky MR (2000) An overview of dexterous manipulation. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, IEEE, vol 1, pp 255–262

Pais AL, Billard A (2014) Encoding bi-manual coordination patterns from human demonstrations. In: Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction, ACM, pp 264–265

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

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

Romano JM, Hsiao K, Niemeyer G, Chitta S, Kuchenbecker KJ (2011) Human-inspired robotic grasp control with tactile sensing. Robotics, IEEE Transactions on 27(6):1067–1079

Sauser E, Argall B, Metta G, Billard A (2011) Iterative learning of grasp adaptation through human corrections. Robotics and Autonomous Systems

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

Sugimoto N, Morimoto J, Hyon SH, Kawato M (2012) The emosaic model for humanoid robot control. Neural Networks 29:8–19

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

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

Wolpert DM, Kawato M (1998) Multiple paired forward and inverse models for motor control. Neural Networks 11(7):1317–1329