Modular Approach of Learning Robot Grasp and Manipulation

submitted by

Bidan Huang

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Sciences

October 2014

Summary

In this thesis numerous seminal results are proved which will decisively shape the future development of the subject.

Contents

Chapter 1

Introduction

Grasping and manipulation are essential skills for service robots. Equipped with these skills, robots would be able to provide great assistance to humans in many aspects in daily life from hospital to household environment. Grasping and manipulation has been extensively studied for more than two decades (). In industry robot gripers and fixtures have been widely used for fast and accurate operations. Outside industry, however, there is still not a universal robust solution for grasping nor manipulation in human dominated environment.

The main challenge of robot grasping and manipulation comes from the large variety of tasks and the complicated dynamics of the robot-environment interaction. A versatile service robot is expected to be able to handle many tasks in human daily life, from simple pick-and-place task to multifinger dexterous manipulation task like writing and using tools. Different tasks have different instructions and constraints. Programming each of them by hand coding is painstaking. Further, grasping and manipulation are contact tasks, for which handling contacts between the robot end-effector and the environment is essential. The dynamics of the contacts is usually complicated. It involves the study of friction and material. To analyze the dynamics of contact tasks requires both a deep understanding of the task, the mechanics of the robot and control theory. It is infeasible for the end user to program such tasks.

To tackle this problem, robot learning has been proposed as an alternative to an analytical solution. Learning by demonstration (also called imitation learning and programming by demonstration) has been extensively studied as a promising and user-friendly approach to build robot intelligence (Schaal et al., 2003; Dillmann, 2004; Billard et al., 2006; Calinon and Billard, 2007). This approach is data-driven. It aims to program the robot to extract the success pattern of a particular task from the demonstration data (either from teaching or self-exploration) and to encode this pattern. This approach allow us to model strategies for tasks without deriving the complex dynamics of the environment. The strategies are usually encoded by statistical models allowing certain level of noise. It is particularly useful for the tasks that analytical expression

of the system is hard to derive such as contact tasks.

Although the learning by demonstration approach provides a user-friendly method for the end-user to program robot, learning grasping and manipulation tasks is still challenging. Even for the same task, the planning or control strategy can be different according to the task context. A single model is not adequate for these tasks.

In this thesis, we propose an approach to further reduce the task complexity: the modular approach. This approach focuses on the problem of decomposing a complex task to small subsections and developing solutions for each subsection separately. These solutions are then combined together to provide an integrated solution of the task. The benefit is to transfer a complex problem into many smaller problems, the solutions of which are easier to find.

The modular approach is particularly suitable for tasks involve different contexts or requiring multiple strategies. While switching between multiple modules allows the robot to quickly adapt to a changing environment, combining the modules allow the robot to generate new skills to adapt to new contexts. We apply this approach to the problem of grasping and manipulation tasks, to simplify the learning problem of contact tasks and to build an easy-to-use interface for teaching a robot. This dissertation introduces different ways to modularize tasks and combining the modules to accomplish the tasks. It provides a framework to model the modules by statistical models via a learning approach. We work shows that the modular approach in robot grasping and manipulation is not only theoretically attractive but also a practical method.

In the next section, we provide an brief overview in Section 1.1 of the use of modular approach in robotics. We first show the study of modularity in artificial intelligence (AI) and control theory and then show the application of modularity in robotics as the intersection of those two realms. In Section 1.2 and 1.3, we outline the contributions of this dissertation and present its organization.

1.1 Existing modular approaches

Robotics is an interdisciplinary area. It is an intersection of many fields in engineering and cognitive science. Two of the most important fields in robotics are AI and control theory. While AI concentrates on the high level perception and action planning, control theory focus on robustly and stably delivering the robot to the desired state. Modular approaches have been independently studied in these two areas and shown to be effective for developing autonomous and intelligent systems.

Modularity in AI AI is a field of studying how to enable machines to have animal level intelligence (Brooks, 1991). Modular approaches in AI is inspired by two factors: the evidence of modularity in cognitive science and the efficiency of modular approach in software engineer-

ing. As a research that aim to produce animal level intelligence in machines, a branch of AI studies the source of the intelligence, e.g. neuroscience and psychology, and tries to mimic the mechanisms. In both neuroscience and psychology, evidences show that brain and mind have some modularized structure (Fodor, 1983; Peretz and Coltheart, 2003; Barrett and Kurzban, 2006; Sztarker and Tomsic, 2011). It is suggested that the modularity in brain and mind helps animal to organize the functionalities and handle complex situations. These evidences motivate researchers in AI to develop modular based architecture for machine intelligence. In the other hand, from the software engineering point of view, modular approach is an effective way of building large complex system. It is widely used for separating the functionality of a program into independent modules, such that each contains everything necessary to execute only one aspect of the desired functionality. Therefore building a complicated intelligence system inevitably prefers a modular approach. Many forms of modularity has been proposed to study different aspects in AI (Bryson, 2005).

Modularity in control Modular approaches are used in adaptive control and their benefit has been long discussed (Jacobs et al., 1991; Narendra and Balakrishnan, 1997). They are used to solve the control problem in a dynamic environment, where changes can happen rapidly or discontinuously. Classic adaptive control approaches such as model identification (Khalil and Dombre, 2004) are inadequate for these environment, as instability or error may occur during the optimization of the model variables. To quickly adapt the multiple model approach, referred as modular approach here, has been proposed by Narendra et al. (1995). In this approach, multiple controllers are designed, each of which is in charge of a certain task context. During control, the task context is estimated online and the corresponding controllers are activated. When the task context changes, the system automatically switches to another strategy that is suitable for handling the current context. This ensures the system reacts quickly enough to adapt to the environment.

Application of modular approaches in robotics Briefly speaking, modular approaches in AI mainly target decomposing tasks to simplify the design of agents, while in control theory mainly aim for building a fast adaptive control policy. In robotics, modular approaches is used for both of these two purposes. Roboticists usually focus on more specific tasks such as grasping and walking and try to develop robust and stable plans to accomplish tasks. In fact, the difficult focus groups, e.g. grasping and walking, is itself a modular approach: the high level modularity divides the committee into different groups that each try to provide a generic solution for the specific task. Further, even for the same focus group, modular approach is used to bring down the complexity of design and increase the flexibility of the planning. Some of the most well known modular approaches in robotics is motion primitive for motion

planning (Ijspeert et al., 2002; Inamura et al., 2004; Kulić et al., 2008; Peters and Schaal, 2008), hand synergies (Santello and Soechting, 2000; Gabiccini et al., 2011; Gioioso et al., 2013), eigen-grasp (Ciocarlie and Allen, 2009) and grasp by shape primitives (Miller et al., 2003; Huebner et al., 2008) and etc.

In conclusion, modular approaches are widely used in robotics. They are mainly used to tame the complexity of high level task planning and low level strategy selection. However, how to modularize a task in order to facilitate the robot learning is rarely discussed in literature and remains a open problem.

1.2 Our modular approach in robot grasping and manipulation

The definition of a module varies by discipline. Here we define a module as a functional unit that takes certain inputs and provides certain outputs. The computation from the inputs and the outputs is independent to other units. Although the concept of modularity in cognitive science is still in debate, its efficiency in software design is well recognized. In this thesis, we do not try to argue the role of modularity in animal brains. We simply take the concept and exploit its effectiveness in programming robots to do tasks. The tasks we discuss here are primitive tasks that can be described by a simple language such as "grasp" and "open" and no further subtask needs to be decomposed. Therefore the modularity we study is task-specific: multiple modules serve one task and each module serves one task context. We hence call our modularity "task level modularity". Not all primitive tasks are in need of modular approach. Some simple tasks such as "close your eyes" have generic solution. However, in grasping and manipulation, this is usually not the case. As discussed before, the contacts between the robot end-effector and the environment makes the system hard to analyze and the large variety of tasks makes it hard to find an universal solution. In our studies, we explore a few possible ways using modular approach to tame these problems.

We apply the modular approach in the three main domains of grasping and manipulation: grasp planning, manipulation force control and reaching. These three tasks have different challenges and require different ways to modularize. For grasp planning, we modularize the strategy by the object shape and propose a method to quickly plan grasps for novel objects. For manipulation, we modularize the control policy by task context and equip the robot with human level adaptive skill. For reaching, we modularize the movement by human command, which builds an understanding base between robots and humans by language and allows the human user to easily teach robot new motion primitives.

These three approaches enable the robots to accomplish tasks that are complex but can be pre-planned, need to adapt in real time, or need to follow human instructions.

Grasp planning: modularize by perception (Chapter 3) The first contribution is modularity in multifinger grasp planning. Previous researches in robot grasping focus on synthesizing grasps analytically, using precise and accurate model for the objects (Sahbani et al., 2011). Those approaches are usually computational expensive for the high degree of freedom of the multifinger robot hand and the universal representation of the object, which usually have many variables. To tackle this problem, we modularize grasping by the shape of the objects. In our approach, we first focus on fast generates grasps for familiar objects and then extend the approach to generate grasps for novel objects. In the first part, we learn statistical model for the feasible grasps of a familiar object. This distribution is then used to quickly generate grasps. In the second part, a novel object is represented as a compound of shape primitives, e.g. sphere, cylinder and box. The grasp distribution of these shape primitives are pre-trained and each act as a module. We combine the grasp distributions of the shape primitives to form a new grasp distribution for the novel objects. When combining, the conflicting parts between modules are excluded. This approach does not require a general and accurate representation of the object. As grasps can be planned quickly, fast correction can be done for small modeling error. The first part of the work is published in ICRA 2013 (Huang et al., 2013b).

Dexterous manipulation: modularize by action (Chapter 4) The second contribution is about manipulation. 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. After human demonstrates a task in different contexts, we modularize the control strategies according to the contexts. Strategy in each module is encoded by a pair of forward and inverse models. All modules contribute to the final control policy, according to their estimation errores of current task context. We validate our approach on a robot platform with a opening bottle cap task. We show that our approach can modularize the adaptive control strategy to generate appropriate motor commands for the robot to accomplish the task. Fast estimation of the current task context and choosing the proper module enables the robot to react to changes of environment. This work is submitted to a robotics journal.

Motion primitive: modularize by language (Chapter 5) The third constribution is about learning reaching motion primitive for manipulation task. In this work, we develop an easy-to-use human interface for teaching and commanding a robot to do manipulation tasks. The human-demonstrated manipulation motion primitives are initially encoded by statistical models. The models are then projected to a topological space where they are labeled by language description of their properties. We explore the unknown area in this space by interpolation between the models. New motion primitives are thus generated from the unknown area to meet

new manipulation scenarios. Human commands are understood by matching with the labels of the motion primitives. Human can give new commands during execution to correct improper robot behavior. Here we make use of the modularity nature of human language to modularise robot motion. This work is published in ROBIO 2013 (Huang et al., 2013a).

1.3 Organization of the thesis

This dissertation has 6 chapters. Chapter 2 gives an overview of existing modular approaches in robotics, discuss its benefits and challenges and describe the framework of my approach. Chapter 3 to 5 detail our works in learning grasp planning, manipulation and reaching motion. We discuss the advantage of modular approach in grasping and manipulation tasks and the potential to extend it to more areas. Chapter six discuss the achievement of our work and summarizes the contribution.

Chapter 2

Related work

This chapter gives an overview of the relative research areas: robot grasping and manipulation, imitation learning and modular approaches. In Section 2.1 we overview the studies in robot grasping and manipulation, outline the current challenges in the area. In Section 2.2, we introduce the technique of robot imitation learning (program by demonstration) and particularly look at its applications in robot grasping and manipulation. In Section 2.3 we first discuss the motivation of modular approaches and its biological inspiration. We then give a brief review on modular approaches in control theory (multiple module adaptive control). The final part of this section focus on the applications of modular approaches in robotics, especially in grasping and manipulation. Figure 2-1 depict the structure of this section.

2.1 A review of robot grasping and manipulation

Robot grasping and manipulation researches aim to enable robots with human level ability of handling objects. Grasping and manipulation are usually put in to the same research category and studied by the same robotic community, as they both try to tackle the "contact tasks", which use robot hand (end-effectors) to get physical contacts and interact with the target objects. Robot grasping focuses on how to stabilize the target objects with the support from the robot hand. This involves the problem of where and how to place the contact points between the robot hands and the objects. Robot manipulation focus on delivering the target objects from the current state to a desired state, which involves the problem of how to apply forces and torques on the object to achieve the desired state. Besides these two problems, one problem is often discussed by the same community – the reaching problem. How to move the robot hand to reach the object so that the planned grasps or manipulation strategy can be achieved, for example making contacts on the right places to pick up a box, is the problem studied in reaching. In the later three section, we will present an overview on these three topics.

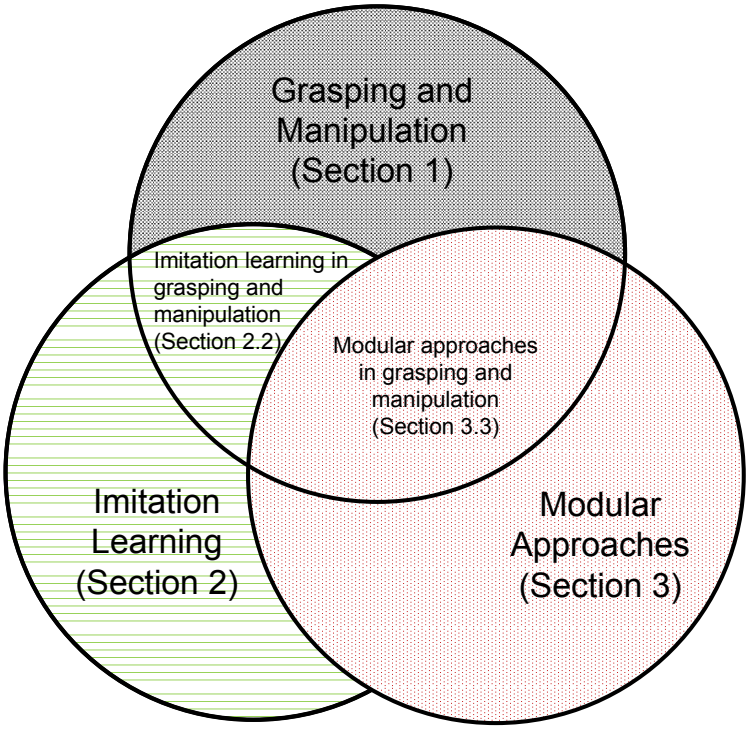


Figure 2-1: Structure of literature review

2.1.1 Robot grasp planning

Given a robot hand and object, there are infinite ways to grasp the object. These grasps have different performances and functionalities. Grasp planning is usually formulated as an optimization problem of grasp performance by finding the contact point locations or robot hand configuration. This technique is call optimal grasp synthesis. The most important criteria in the optimization is the stability of the grasp. In the robot grasping literature, two of the most extensively used mechanisms for guaranteeing grasp stability are the force-closure and form-closure criterion (Nguyen, 1987). A grasp is said to achieve force-closure when the fingers can apply appropriate forces on an object to produce wrenches in any direction (Salisbury Jr, 1985). Form-closure is a stronger condition than force closure, which can only be achieved if a grasp is force closure with frictionless contact points (Dizioğlu and Lakshiminarayana, 1984).

To measure grasp stability qualitatively, the concept of grasp quality is introduced. Various grasp quality metrics are proposed for different purposes. Starting from the idea of minimizing the sum of the contact forces, (Li and Sastry, 1988; Kirkpatrick et al., 1992; Ferrari and Canny, 1992) propose different measurements of the grasp quality based on the hand wrench space. These metrics are "object-centric", i.e. they only consider the contact point locations and the object geometry, while the robot hand configuration is not taken into account. Miller and Allen (1999) take one step further: they use a simulation method to compute the grasp quality of a given object and robot hand configuration. They later develop the physical simulator GraspIt! for grasp quality analysis (Miller and Allen, 2004). Our work in grasp planing described in the Section 3 is based on this simulator.

Optimal force-closure grasp synthesis concerns determining the contact point locations so that the grasp achieves the most desirable performance in resisting external wrench loads. Based on the grasp quality concept, some approaches optimize an objective function according to a pre-defined quality criterion (Zhu and Wang, 2003; Zhu and Ding, 2004) in the grasp configuration space. These approaches do not take into account the kinematics of the hand, which is difficult to achieve. To bridge this gap, Khoury et al. (2012) propose a one shot grasp synthesis approach that formulates and solves the problem as a constraint-based optimization.

Multi-finger grasps usually involve a large number of degrees of freedom. Searching the grasp space for an optimal grasp requires massive computing time considering the huge amount of possible hand configurations. To solve this problem, imitation learning and modular approaches are introduced to constraint the searching space. The relative literatures are reviewed in the Section 2.2 and Section 2.3

The above methods are for static grasp planning that relies on precise and accurate object models. These methods are well suited in the controlled industrial environments, for example picking up aligned boxes from the assembly line. However, they are not very applicable for service robots working in human dominated environments. For this reason, in recent years the

research is shifting to tackle the problem of maintaining the grasp stability in dynamics and cluttered scenes. These studies include handling uncertainty and noise in the perceptual data and handling unseen (novel) objects and unforseen situations. To tackle the former problem, one approach is to take the uncertainty and noise into account in the planning and generate robust grasps (Brost, 1988; Zheng and Qian, 2005; Hsiao et al., 2011a). Besides synthesis, grasping motion is also studied (Kehoe et al., 2012), where the uncertainty is handled by the complaint finger motions. For grasping novel objects, different general object shape representations are proposed. The most studied representations are 2D or 3D local features such as edge, contour and color (Saxena et al., 2008; Detry et al., 2009; Kroemer et al., 2010), combination of shape primitives (Miller et al., 2003; Huebner et al., 2008; El-Khoury and Sahbani, 2010) and exclusive mathematics representation of the global object surface geometry and topology (El-Khoury et al., 2013; Pokorny et al., 2013). Local features allow quick computation of grasps on subpart of an object, while the global representations allow global search of good grasps with large computation expenses. Planning grasps for novel objects effectively and robustly is remains a challenge.

2.1.2 Robot Manipulation

A classic approach of manipulation is impedance control (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.

TO BE EXTENDED

2.1.3 Reaching motion planning

Reaching motion is another key component in the robot grasping and manipulation problem. Given a computed stable grasp, how to deliver the robot hand to the desired position and form the desired hand posture is the question to answer in this study. This is not a simple path planning problem for the robot arm, but a high dimensional planning problem taking the multiple finger movement into account. At one hand, most studies try to plan a motion to avoid pre-manure collisions between the hand and the object. To this end, the finger movement and the arm movement always need to couple in order to ensure the fingers clutch at a proper moment (Shukla and Billard, 2011), and curve around the object to form the desired grasps (Kroemer et al., 2011). To increase the robustness of a grasp, the uncertainty in perception is also taken into account (Stulp et al., 2011). At the other hand, however, some researches study how to produce "pre-mature" contacts with the object. Chang et al. (2010) study the human "pre-grasp" movements such as sliding a coin to the table edge in order to pick it up, and rotating the handle of a pan to a proper position to grasp it. These methods largely increase the chance of successfully execute a grasp by changing the object status.

2.2 A review of imitation learning

This section give a brief introduction of robot imitation learning and then review its application in robot grasping and manipulation.

2.2.1 Robot imitation learning

Since the first study on robot imitation learning (Friedrich et al., 1996), this approach has become one of most popular research area in robotics. It is considered to be an designer-friendly approach to teach robot new tasks. The aim of imitation learning, referred as program by demonstration in some literatures, is to enable robot to learn new skills by observing human demonstrations and reuse these skill in similar tasks. Imitation learning is always referred as "Learn by Demonstration (LbD)" or "Program by Demonstration (PbD)". In recent years, this approach has been extensively studied (Calinon et al., 2007; Calinon, 2008; Dillmann, 2004; Kulić et al., 2012) as a promising approach to build robot intelligence.

TO BE EXTENDED

2.2.2 Robot learning grasping and manipulation

As discussed in the Section 2.1, conventional grasp and manipulation planning methods suffer from the curse of dimensionality. Learning technique have been introduced to avoid the complexity of computing kinematical constraints guaranteeing stable grasps. Briefly speaking,

robot grasping has two learning sources: imitation learning from human demonstration and learning from data collected from the simulation. In imitation learning, some researchers use datagloves for human demonstration. The human hand configuration is then mapped to an artificial hand workspace and the joint angles (Fischer et al., 1998; Ekvall and Kragic, 2007), or hand preshapes (Kyota et al., 2005; Pelossof et al., 2004; Ying et al., 2007) are learnt. Some other researchers use stereoscopy to track the hand when a demonstrator is performing a grasp (Hueser et al., 2006) or to match the hand shape to a database of grasp images (Romero et al., 2008). For long term automatic learning, markerless methods to track human hand and arm movements in the approaching and grasp execution are studied (Ekvall and Kragic, 2007; Do et al., 2009). These learning based approaches succeed in taking into account the hand kinematics and generate hand preshapes that are compatible with the object features. Human grasp postures are usually mapped to robot hand postures in fixed schemes, according to the shape of the object and the type of grasp human choose. The learn from simulation method get around this mapping step: it directly generate grasps with the robot hand mechanical constraints. For a given object shape and a robot hand, thousands of grasps are generated in the simulator and later used as training data. Pelossof et al. (2004) use a discriminative model Support Vector Machine to learn the correlation between the grasp configuration and grasp quality while Huang et al. (2013b) use a generative model Gaussian Mixture Model to learn the distribution of force closure grasps. Both models are used to generate new grasps. Grasp training data can also be generated in a real robot platform rather than simulator (Herzog et al., 2014). However this method is much more time consuming and hence it focus on finding a way to maximize the use of the grasping experience, i.e. generalizing grasping strategies to novel objects.

To further reduce the complexity of grasping problem, modular approaches are used. This will be discussed in the Section 2.3.

Besides reducing the complexity of grasping problem, learning approaches are also used to tackle those common problems appear in human environment: uncertainty and noise in perception data, novel objects and unforeseeable situations. Ekvall and Kragic (2007); Stulp et al. (2011) study human grasp motion and try to learn how human choose the approach vector that is robust to pose uncertainty. With the same principle, the human grasp postures are studied and mapped to robot hands in order to grasp objects of which the pose is not estimated with high confidence (Tegin et al., 2009). With the recent advance of tactile sensing technology, many attempt to include the tactile sensory data in assessing the grasp stability. By estimating the grasp stability after the grasp execution, failure further actions on the object can be avoided. Bekiroglu et al. (2011) integrate the information of the object shape primitive, approach vector, tactile data and hand joint configuration to estimate a grasp quality. In the later work, contact point locations are also taken into account (Dang and Allen, 2012, 2014b). The support vector

machine (SVM) is the most used model in this task.

The tactile feedback based methods are "robot-centric", i.e. they do not rely on the pre-defined object shape. Hence these methods can be easily applied to estimate grasp quality for novel objects. To generate grasps for novel objects, the mechanisms of both human perception and action are studied. Detry et al. (2009) study the human Early-Cognitive-Vision (ECV) and use it as a feature to associate with grasps. The ECV feature includes colour and edge information and hence can be used to generate grasps for novel objects. El-Khoury et al. (2007) try to imitate human mechanism of representing objects and segment objects to a set of superquadric shape primitives. The mechanism of human choosing the grasp component is then learnt by a Neural Network (El-Khoury and Sahbani, 2010).

Human environment is dynamics and full of perturbations. These perturbations can not be foreseen and can only be handled when happen. Learning approach is also used here to provide methods for quick adaptation. Methods are proposed to simplify the generation of grasp such that moving object can be caught (Harada et al., 2008; Kim and Billard, 2012; Huang et al., 2013b) Beside using visual features, tactile sensors can provide additional useful information which is not accessible by vision. Many methods for quick adaption to the actual contact conditions are proposed (Hsiao et al., 2010, 2011b; Kazemi et al., 2012; Sauser et al., 2011; Li et al., 2014a).

2.3 A review of modular approaches

This section first review the modular approaches studied in cognitive science and control theory very briefly, and then concentrate on modular approaches in robotics.

2.3.1 Modular approaches in cognitive science

One typical hypothesis of modular model in motor control is MOSAIC: the Modular Selection and Identification of Control. It 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 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.

TO BE EXTENDED

2.3.2 Modular approaches in control

The use of modular approaches in the control theory is different from in AI. As discussed above, in the discipline of AI, the modularity is in needed when building a large complex versatile system. In the discipline of control theory, at the other hand, the modular approaches are used to handle the adaptive control problem, which is usually referred as the multiple model adaptive control (MMAC). Adaptive control is the control method that the controller change itself to adapt to the changes in the control condition. An commonly used example is the controller of an airplane needs to adapt to the reduction of the weight of the oil. Conventional adaptive control methods rely on state estimation. The controller try to estimate the changes of the system dynamics and then modulate its control parameters to adapt to the changes. For frequently changing environment, however, the period of modulation of the control parameters may cause a transient error, where strong fluctuation can downgrade the performance and damage the hardware. MMAC is used to reduce the transient error by conducting a fast adaption. A MMAC system is usually composite by a few different controllers, each particularly designed for one control condition. During the control process, the environment is monitor in realtime and one or more controllers suitable for this environment are activate to generate the control command. When the system encounters a sudden change, it will adapt to it by activating another set of controllers. It does not need to re-optimise the control parameters and hence the transient error is reduced.

MMAC can date back to the 1970s. Athans et al. (1977) use multiple Kalman filters in controlling equilibrium flight, to handle sensor errors and to reconstruct the state variables in different flight conditions. The final adaptive control signal is computed by the linear combination of the control signal generated by each model, weighted by the associated probability. Later, a switching MMAC is proposed and its stability is studied (Fu and Barmish, 1986). Narendra and Balakrishnan (1994) use MMAC to improve the performance of the controller in multiple environments, particulary to reduce the transient error that caused during the transition of the control parameters from one set of optimal values to another set of optimal values. They later used the neural networks to build models for the nonlinear system (Narendra et al., 1995; Narendra and Balakrishnan, 1997). This controller is implemented in a robot manipulator to follow a predefined trajectory and shows improved performance compared to single model control.

To apply MMAC to a practical control problem, the first step is to design how many modules to use and how to decompose the problem space. For linear plants, this problem is addressed by Anderson et al. (2000). They use the concept of Vinnicombe distance to decompose the space. Firstly, an initial random starting point is chosen, where a controller is determined. The controller finds its boundary in the neighborhood where its control is acceptably accurate. At the boundary, a new starting point is chosen and a new controller is determined. This pro-

cess continue until the whole space is covered. Based on this method, **?** propose an approach to recognize new condition and learn new controls online to adapt. These methods, however, only perform well for learn plants. How to apply MMAC in nonlinear systems remains a challenge.

In robot control, MMAC have many applications for conducting a task in the frequent varying environments. These changing environments can be caused by many factors such as object interactions. Works on this trend include Petkos et al. (2006) learning multiple inverse models for controlling robot to follow a trajectory with different workload on the arm; Nakanishi et al. (2013) proposing time-based switching method for robot system with variable stiffness actuation to handel the different phases of interaction with the environment; the "eMOSAIC" (Sugimoto et al., 2012) to bring the MOSAIC from simulation to real robot control. In the last work, the performance of MOSAIC under large observation noise is improved by using an optimal control technique. The method is implements on the 51 DOF humanoid robot CB-i for squatting task and carrying load task. This is by far as we know the first MMAC implemented on a real robot.

Despite the remarkable theoretical accomplishments and many successful applications of MMAC, its application in controling service robot is not flourishing. On one hand it is because robotics always involve nonlinear control problems, of which the MMAC has not a principle solution. On the other hand, MMAC controller itself is difficult to design. Control problems in robotics are highly task specific and the service robots are expected to handle a huge amount of tasks. Hand designing MMAC for these tasks is not cost effective.

2.3.3 Modular approaches in robotics

In the previous two sections we list a few applications of the modular approaches in robotics from the AI and control prospectives. Modular approaches in robotics go further. In recent years, studies in modular approach have been highly active especially in robot motion planning, grasp planning, manipulation planning. This is mainly due to the trend that we are trying to move robots from industrial controlled environment to human dominated environment, where the robots have to handel dynamic and complex situations. In this section, we will give a overview on modular approaches in motion planning. Applications in grasp planning and manipulation will be reviewed in detail in the Section 2.3.4.

Modularities in robotics always refer to "primitives", such as "object shape primitive", "motion primitive" and "manipulation primitive". Among those one of the most extensively studied areas is the motion primitives. To build a versatile service robot that can work in a human dominating environment and assistant human, high level behavior planning is required. This means robots need to be equipped with the ability to plan a sequence of movements that fulfil a commanded task, such as "clean the table" and "put the food into the fridge". Conventional way of motion planning is done by an search in a high dimensional space formed by the

numerous degree of freedoms of the robot. The number of possible solutions to accomplish a task is nearly infinite.

This redundancy is useful. In reality various constraints, such as avoiding obstacles, may be added to the task. Due to the redundancy, we are able to find feasible solutions under multiple task constraints. However, this redundancy also makes planning difficult as the searching space is extremely large. One of the pervasive approach is to do optimization for the task with constraints that mathematically equivalent to the task constraints. The drawback of this optimization approach is that defining an proper cost function and proper constrains of the task is not easy. This requires the robot use to process a certain amount of knowledge in mathematics and mechanism, as well as a deep understanding of the task.

As an alternative, modular approaches is used to reduce the searching space, without getting rid of good solutions. To this end, the concept of motion primitive is introduced into robotics. This is an concept from the neuroscience research. Neuroscientist find evidences suggest that the vertebrate motor system generate motions by combining a small number of motor primitives (Mussa-Ivaldi et al., 1994; Mussa-Ivaldi, 1999; Bizzi et al., 2008; Grillner, 2011). These show the modularized mechanism running in brains: each motor primitive is one module, the combination of many modules generate the complex behavior.

This idea inspires roboticists to develop simple motion primitives and use them as substrates to develop complex behaviors. In robot motion planning, motion primitives are defined as the most elementary motions, each of which serves one particular purpose. A common way to generate motion primitives is extracting them from human demonstrations: motion sequences demonstrated by human are discretized to a sequence of motion primitives. Modularized by the motion primitives, task planning problem is brought from a huge high dimensional searching space to a finite discrete space. They can be reused in other tasks as functional units.

Many literatures have discussed the motion primitives. Robot motion primitives are learn from human. These approaches mainly focus on three problems, which are also the typical problems in a modular approach: how to model the motion primitives, how to extract motion primitives from a complex motion sequence and how to combine them to form a complex behavior.

In the studies of the first problem, many roboticists encode the motion primitives with statistical or analytical models, which can be modulated in some extend by varying the parameters according to the requirements of a certain task. The Hidden Markov Model (HMM), mixture models such as Gaussian Mixture Model (GMM) and the dynamical systems represented by a set of nonlinear differential equations are the most used modeling methods for motion primitives. HMM is used to encode temporal motions (Inamura et al., 2004; Kulić et al., 2008; Takano and Nakamura, 2008; Lee and Ott, 2010; Huang et al., 2013a). For time independent motions, Gribovskaya et al. (2010); Khansari-Zadeh and Billard (2010) use GMM to model

multiple human demonstrations in the state space, while Ijspeert et al. (2002, 2003); Schaal et al. (2005); Peters and Schaal (2008) use nonlinear differential equations to capture an observed behavior in an attractor landscape. The later is referred as the Dynamical Movement Primitives (DMP), of which the design principle and roadmap is reviewed in (Ijspeert et al., 2013).

Many of the algorithms mentioned above obtain the motion primitives from manual segmentation of motions. However, it is still not clear to us that how many motion primitives we need to compose all the human daily behaviors and what these primitives should be. To obtain these primitives, demonstrating all primitives or manually extracting motion primitives from demonstrations are not practical. Even if a library of motion primitives existed, to learn a complex behavior from human demonstration, a robot still need to recover the motion primitives from demonstrated motion sequence. Hence, a general automatic mechanism to extract motion primitives is required.

To this end, segmentation of a motion sequence (Takano and Nakamura, 2006; Pais et al., 2013) and clustering of data (Kulic et al., 2009; Kulić et al., 2012) are the most used techniques. These approaches usually rely on a carefully chosen threshold to decide when to segment and stop clustering. A method is to set boundaries on the kinematic variables such as the velocity: Fod et al. (2002) segment a sequence when a Zero Velocity Crossing (ZVC) is observed. Takano and Nakamura (2006) perform the segmentation according to the correlation among short motions. They first divide the sequence to a set of short notes. When a new motion is demonstrate, they segment it at the moment that the difference between the predicted next note and actual observed one is larger than a threshold. 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. Pais et al. (2013) extract the primitives according to the variances of the motions in a few demonstrations for a same task. Many other approaches have been proposed to extract motion primitives according to their task requirements. All of these approaches target to extract a set of motion primitives that are independent functional units and generalized enough to be reused in many tasks. With these pre-defined motion primitives, online recovery of a sequence of motion primitives is feasible. With the presumption of an existence of a motion primitive library and reduce the segmentation problem to a online motion recognition problem Meier et al. (2011).

The intention of modeling motion primitives is using them to help with the motion planning problem. According to the task, the use of the motion primitives can be in the form of selecting, mixing and sequencing. The selecting and mixing are for adaptive behavior: robot need to select one or mix a few motion primitives according to the current task context such that it can finish the task. Selection can be decided by a pre-learnt correlation between the primitives and

the task contexts: the highest correlated primitive with the current task context is the one to choose (Takano et al., 2006). On the top of this, Daniel et al. (2013) use Relative Entropy Policy Search (REPS) to optimize the joint state-action distribution and hence choose the optimal set of parameters of the primitive. Some others choose the primitive that can result in a system state closest to the desired next system state (Hauser et al., 2008). Similar idea is used in the mixing method, where more than one motion primitives can be activated at the same time. Weight of each motion primitive is computed to make sure the resulting motion can bring the system to the desired state (Huang et al., 2013a; Sugimoto et al., 2012). From the human robot interaction prospective, robot should be able to understand human verbal commands and plan the action. Takano and Nakamura (2008) propose a method to associate morpheme words with motion primitives. This potentially enable the robots to understand human command and plan motion by parsing the sentence.

2.3.4 Grasping and manipulation by modular approaches

Modular approaches in robot grasping and manipulation to reduce the problem complexity. Modularization in grasping and manipulation are mainly done in two approaches: modularize by perception and modularize by action. Perceptual modules are mainly used in planning, while action modules are mainly used in execution.

Modularize by perception

The first step of making a plan of grasping and manipulation is observing the object. Most of grasp stability analysis are done based on the shape of an object. In human dominated environment, the possible shapes of objects to grasp and manipulate is infinite. Conventional methods to model these object are only effective in convex models. For highly non-convex shapes, local vision features such as edges and colors are used to generate grasping plans at the local areas. To generate grasp for the whole object, Miller et al. (2003) propose a modular approach, i.e. planning grasps by shape primitives. The key idea is to approximate a complex object, e.g. non-convex shape, to a set of shape primitives such as boxes, cylinders and spheres. Planning on these shape primitives is relatively easier or pre-trained. Therefore the complex planning problem is tamed to a set of simple problems. According to different purposes, different shape primitives are proposed. Miller et al. (2003) use four primitives including box, cone, cylinder and sphere; Huebner et al. (2008) use minimum bounding box to decompose an object and El-Khoury and Sahbani (2010) use superquadric as the shape primitive. These methods are based on the complete object point clouds, which may not be fully accessible in the real scenario. Methods to split objects to shape primitives and detect primitives parts are proposed, which mainly exploit the techniques in graphics such as the RANdom SAmple Consensus (RANSAC) (Garcia, 2009; Gallardo and Kyrki, 2011). Faria et al. (2012) use multiple

sensors to track human hand trajectory and tactile data, and hence extract motion primitives and contact primitives from the demonstration. These information is then merged to form a object probabilistic volumetric model, which is decomposed to multiple superquadrics.

Modularize by action

The motion primitive concept is also introduced to grasping and manipulation. Different from the reaching movement primitives discussed in the previous Section 2.3.3, of which the goals are to reach the targeted points, the grasping and manipulation motion primitives are more task-oriented, i.e. each primitive is associated with a specific impact on the environment, such as getting contact with the object and pushing the object. Therefore in literatures these primitives are sometimes referred to "task primitives". Because of the variety of tasks and their complexity, usually these task primitives are manually defined. Transitions between them are usually decided by contact events that indicate the impacts on the environment (Morrow and Khosla, 1997). Michelman and Allen (1994) propose to represent the relationship between task primitives by a finite state machines. Kazemi et al. (2012) define three task primitives for force compliant grasping of small objects from a table top. The Dynamical Movement Primitives (DMP) mentioned previously, which models desired motion by an attractor landscape, is extended to due with various problems when executing a grasp. The combination of the DMP and the Early Cognitive Vision Descriptor (ECVD) for grasp planning enable a robot to plan approaching path of the hand and the finger that avoids pre-mature contact between finger and object (Kroemer et al., 2011). Taking the object poses distribution into account, a new optimization method of the DMP is proposed to find an approaching trajectory that produce robust grasp to object pose uncertainty (Stulp et al., 2011). Later simplify version of DMP is used to learn movement goal and hence can quickly change the end point location to adapt to the object shape (Stulp et al., 2011, 2012).

A few frameworks are proposed to model and organize the task primitives. Laaksonen et al. (2010); Felip et al. (2013) propose a hierarchical framework to solve the embodiment problem of sharing experience among different robot platforms. This is done by defining task primitives in an abstract layer and an embodiment layer. The former can be translated to the later. This enable the robot to plan tasks with the higher level abstract primitives, while execute it by the embodiment specific task primitives. To facilitate manipulation motion planning, Barry et al. (2013) use a Rapidly exploring Random Tree (RRT) to sequence motion primitives. Detry et al. (2013) modularize a grasp planning task by two constraints: gripper constraints and task constraints. While the former module handle grasp stability, the later module select grasps by the task requirements.

Besides task-specific motion primitives, modular approaches are also used to tame the complex grasp planning problem. The concept of "hand synergies", for example, is a modu-

lar approach originating in the neurophysiological studies (Santello et al., 1998; Santello and Soechting, 2000). In this field of study, roboticists try to understand how does human central neural system (CNS) simplify the grasping strategy and how to mimic this mechanism in robot system. This concept is used in grip force control (Gabiccini et al., 2011) as well as grasp planning (Gioioso et al., 2013). Similar to this idea, robot "Eigen grasp" is proposed to study the modularity in robot embodiment. Instead of directly searching good grasps the high dimensional configuration space of robotic hands, this space can be reduced by generating a set of grasp starting positions, hand preshapes Miller et al. (2003) or eigengrasps Ciocarlie and Allen (2009) that can then be tested on the object model. Such approaches reduce the dimensionality of the hand configuration space, but doing so implies a corresponding reduction in the accessible hand postures.

Chapter 3

Modular approach in grasp planning

3.1 Introduction

Given an object and a multi-fingered robotic hand, generating a set of contacts on the object's surface which ensure grasp stability while being feasible for the hand kinematics is a common problem in grasp synthesis. Over the last few decades, robot grasping has been a popular topic and numerous approaches for grasp planning have been proposed Sahbani et al. (2011). Most of these approaches adopt iterative methods, which are usually able to find a solution within a finite number of iterations and the average computation time is usually in the scale of a few to tens of seconds. However the number of iterations required grows quadratically with the size of the problem and this creates an uncertainty of the time for the robot to plan a grasp. The upper bound of the computation time is barely analyzed in the literature.

Moving from the traditional engineering environment into a human dominated environment necessitates a fast grasp planning strategy to respond in real time. For example, when reaching out to grasp an object, a robust grasping strategy must be able to adapt rapidly to external perturbations that can modify the initial object position and orientation relative to the robot hand. In the case of catching a flying object Kim and Billard (2012), the robot has only a few milliseconds to plan a grasp before the object touches the floor.

Another application is receiving objects handed over by humans with a robot hand (Fig. 3-1). In many circumstance the object must be grabbed quickly: one such example is when the object is heavy or hot; other examples involve time-pressing situations, e.g. in surgery a robot assistant must react sufficiently quickly to doctors handing back implements to ensure smooth running of the surgery.

Besides human-robot interaction, real time planning for the pick-and-place task in the industrial environment may also be necessary: spare parts could be randomly placed on the conveyer belt. The conveyer belt runs constantly at a high pace and leaves no time for the

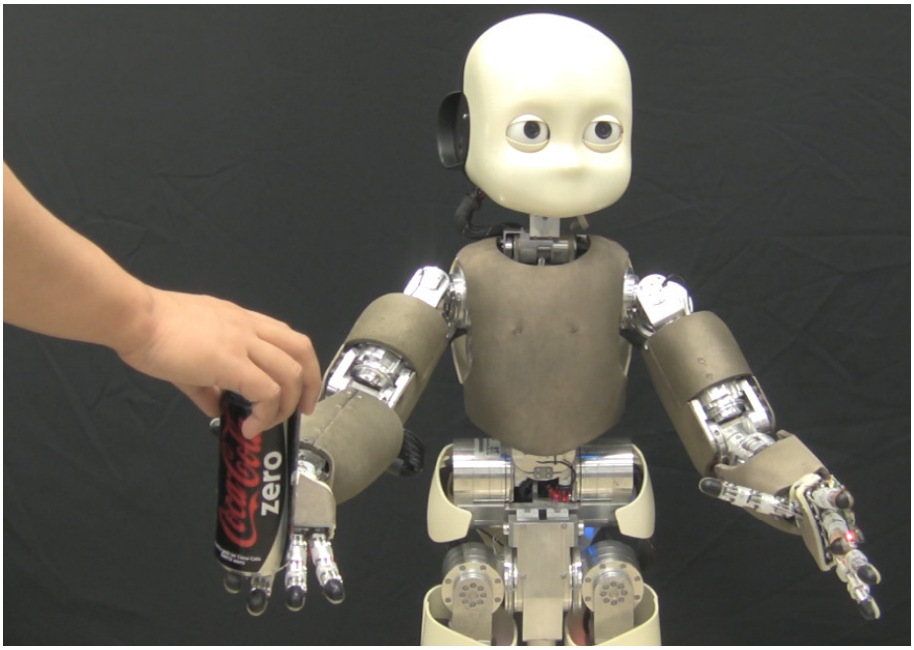


Figure 3-1: A human hands a can to an iCub

robot to stop its action and replan. The robot must therefore respond swiftly to avoid incurring delays in production. Given the limited computational power available on computers embedded in the robot, a computationally expensive algorithm would result in a prohibitively long decision time, leading to task failure in the above scenarios.

Because of the complexity of the problem, real time grasp planning has not been extensively studied. To tackle this problem, we propose a closed-form solution which requires at most three steps to compute a new grasp, and hence guarantee a short computation time and the uncertainty is reduced to the largest extent. In this chapter, we first present a real time grasp planning strategy for familiar objects (Section 3.2, 3.3). We then present an extension of this method to plan grasps for novel objects.

3.2 Fast grasp planning for familiar objects

Traditional manipulation planning strategy usually involves inverse kinematics and optimization, which are computationally expensive. The reported computation time varies from 0.1 seconds to a few minutes. Recently, there have been some attempts to tackle the problem with real time solutions. Richtsfeld et al. (2008) use a laser scanner to detect cylindrical shapes and plan grasps. This method is limited to cylindrical objects. Kanehiro et al. (Harada et al., 2008) use approximation models of the friction cone and roughly estimate the force closure criterion. However, this approximation may limit their solutions. In the planning step, they use random

sampling techniques to generate grasping postures and loop through the samples to find a grasp satisfying all the kinematic constraints. The reported computation time varies from 10*sec* to 25*sec* including path planning of the arm using a 2GHz core. Daoud et al. (Daoud et al., 2011) employ a genetic algorithm optimization approach to provide an initial grasp before online manipulation. This evolutionary approach relies on several iterations of optimization before reaching the solution. The reported time is 12.61*sec* for a spherical object with a 2.2GHz core. The latter two methods, due to their iterative approaches, do not guarantee fast computation in all cases. In contrast, with our closed-form solution the computation time is bounded within a few milliseconds.

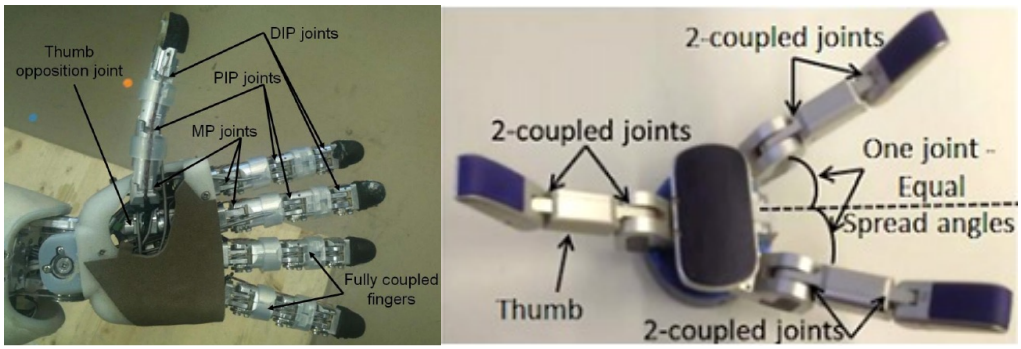
We avoid using these by adopting a learning approach. Our method for planning grasps for familiar objects starts by generating a training dataset of stable grasps for the objects. A *Gaussian Mixture Model* (GMM) (Cohn et al., 1996) is learned from the data, and the target pose is predicted via *Gaussian Mixture Regression* (GMR). Hence there is no inverse kinematics computation nor iterative optimization in our method. Generally speaking, our approach is to:

1. Generate a set of stable grasping demonstrations for a given object and a robot hand (Section 3.2.1).
2. Build a statistical model for the training dataset offline (Section 3.2.2).
3. Use the model to quickly generate a new grasp, given a starting object-hand configuration (Section 3.2.3).

3.2.1 Grasp generation given the hand kinematics

Two robot platforms available in our lab are chosen to perform the grasping tasks: the iCub and the Barrett hand. The iCub has an anthropomorphic hand with 9 degrees of freedom: 3 in the thumb, 2 in the index, 2 in the middle finger, 1 in the ring and little fingers and 1 for the adduction/abduction movement (Figure 3-2(a)). The Barrett hand is an industrial grasper with 3 fingers and 4 degrees of freedom: 1 for each finger and 1 for the separation between the second and the third finger (Figure., 3-2(b)). These two platforms differ drastically in the range of motion for each finger and provide very different grasp demonstrations. They will hence grasp objects in very different ways.

Starting from the geometry of an object and the kinematic property of a robot hand to compute a feasible grasp is time consuming. To achieve fast planning, we do this computation offline. There are numerous possible ways to grasp one object depending on the task's needs (**?**El-Khoury et al., 2013). To encapsulate all the possible ways, a large amount of training data is needed. Collecting this amount of data on a real robot is time consuming. Therefore, instead of using a real robot, we generate training data by synthesis.



(a) iCub hand (b) Barrett hand

Figure 3-2: Two robot platforms used in this work. This method is not bounded to these two robots. It can be applied to any other robot hands given their kinematics.

Two different approaches are used here: optimization and simulation. We use simulation method for the Barrett hand and use optimization method for the iCub hand. In simulation, we use a trial-and-error approach: in the state space we try to generate as many grasps as possible and select those feasible ones. In principle we can generate more variety of grasps by this method, as some of them might be hard to reached by optimization. The 4 d.o.f Barrett hand is particularly suitable for this approach. For the 14 d.o.f iCub hand, however, the state space is much larger and hence the trial-and-error approach is expensive. Instead, for the iCub hand we use an optimization method.

3.2.1.1 Optimization

We use the optimization algorithm proposed in the work of El-Khoury and etc. (El-Khoury et al., 2013) to generate grasps for the iCub. The iCub hand is modeled in 8 dimensions in this algorithm and the thumb, index and middle finger are taken into account.

This optimization algorithm formulates the problem as a constraint-based minimization for a set of hand configuration parameters (hand position $\boldsymbol{h}$, hand orientation $\boldsymbol{o}$ and finger joints $\boldsymbol{\theta}$). These parameters are subjected to a number of constraints to satisfy the following criteria:

1. The grasp is kinematically feasible for the robot hand;
2. The grasp is a force-closure grasp;
3. The robot hand is not penetrating the object;
4. The robot fingertips contact the object surface;
5. The force provided by the robot hand is able to raise the object.

The iCub's finger joints can only apply a limited amount of torque. The less joint torque required, the easier it is for the iCub to lift the object. For this reason, we choose the objective function to be the minimum joint torque required to balance the gravity wrench, formulated as:

$$J(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}) = \|\sum_{i,j} \tau_i^j\| \tag{3.1}$$

where τ_i^j is the ith joint torque of the jth fingers under the force feasibility constraints:

$$\tau_i^j \in [\bar{\tau}_i^j, \hat{\tau}_i^j] \tag{3.2}$$

where $\bar{\tau}_i^j$ and $\hat{\tau}_i^j$ are the lower and upper boundaries of τ_i^j. Minimizing this cost function is equivalent to minimizing the energy required in the joint space in order to accomplish the grasping task.

The optimization is solved by the Interior Point OPTimizer (IPOPT) method proposed by Wächter and Biegler (Wächter and Biegler, 2006), written in the AMPL Model Language for Mathematical Programming. To generate a variety of grasps, we exploit the fact that the IPOPT solver converges to local solutions. We provide the solver with a large number of initial conditions, varying from 1000 to 2000. From these initial conditions, which are located in different areas of the space, the IPOPT converges to their corresponding local optima. By this means 500 to 1000 optimized grasps for an object can be obtained. They will be used as the training data in the next phase. The average computation time for the IPOPT to converge to one solution is $2.65sec$, with a standard deviation of $1.82sec$. As additional information, the quality Q of each optimized grasp is calculated in the form described in (Ponce et al., 1997):

$$Q = \|\frac{1}{3}\sum_j \boldsymbol{c}^j\| \tag{3.3}$$

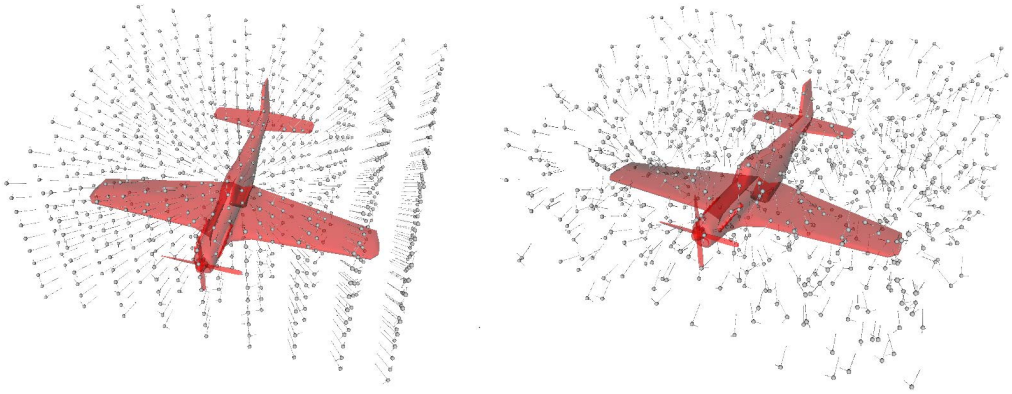
where $\boldsymbol{c}^j$ is the contact point (i.e. fingertip) position of the jth finger. Though it is not included in the optimization, the quality is used in the comparison between the training set and the result set shown in Section 3.3.

To ensure the robot fingertips contact the object surface, the object has to be expressed by an implicit equation. For example, a cylinder can be expressed as:

$$\left(x^2+y^2\right)^{10} + z^{20} = 1 \tag{3.4}$$

During optimization, this will be used as an hard constraint for the all the fingertip position. For more complex shapes, the implicit equation can be learn by Gaussian process (El-Khoury et al., 2013).

The algorithm above can generate a variety of high quality force-closure grasps for a given



(a) Initial distribution (b) Final distribution

Figure 3-3: An illustration of part of the grasp position lattice of an airplane model. Each grey dot in the lattice represents one robot hand position. The long arrows at each dot represent the hand normal directions and the short arrows represent the fix finger directions. The hand normals are initialized by pointing toward the center of the object, as shown in (a). Small random variance is then added to each grasp later to even the distribution and the final distribution is shown in (b).

robot hand kinematic structure and an object model. Since IPOPT is a continuous optimization solver, generating grasps on complex objects requires a continuous implicit representation of the whole object surface model.

3.2.1.2 Simulation

As the Barrett hand is modeled in the widely used simulator GraspIt! (Miller and Allen, 2004), we use simulation to generate its data. GraspIt! is designed for grasp analysis and it provides a library of robots and object models. Its quality measurement module computes the grasp quality according to all the contacts between the hand and the object, in the form described by Ferrari and Canny (Ferrari and Canny, 1992). A grasp planning module for primitive shapes, i.e cylinder, sphere, cuboid and cone, is available, allowing users to easily generate grasps (Miller et al., 2003). To sample grasps for objects with complex shapes, we alter the module and generate grasps as follows.

Firstly a robot hand position "lattice" is generated. Each vertex in the lattice represents one robot hand position, where the hand will be placed to grasp the object (Figure 3-3). The object is located in the center of the lattice surrounded by the grasping positions. All palm normals are initially pointing to the center of the object. Random finger separation angles are assigned to each point to form a list of grasp configurations for testing. According to the object size, 1000 to 20000[1] testing grasps can be generated to ensure that the entire object is surrounded by the lattice and the farthest point to grasp the object is included. The density of the hand position lattice depends on the object shape. Objects with sharp edges, where the normals on

[1] More complex and bigger shapes need more testing points.

the surface change sharply, should have a higher lattice density compared to those with smooth surfaces.

In the final step before testing, small random perturbations are added to each grasp so that the testing points are evenly and continuously distributed in all dimensions. To test these grasps, the hand is first placed at each position on the test list with the desired posture (hand orientations and finger joints). Next, the fingers clutch around the object until contacts or joint limits prevent further motion. We then use the quality measurement module to compute the quality of each grasp. The non-zero quality grasps, i.e. force-closure grasps, are recorded and used as training data. Note that not all the testing grasps result in feasible grasps. Points causing collisions are removed from the list and only the force-closure grasps are kept as the training data. The average generating rate for the feasible grasps is roughly one per five seconds.

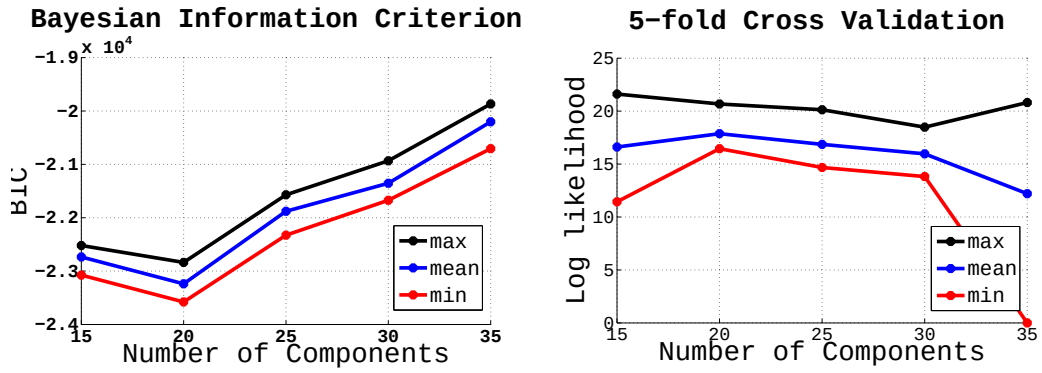
The Barrett hand has one joint in each finger. These three joints can only rotate in one direction and how much they rotate is determined by the object surface, given the hand position, orientation and the separation angle. Therefore we drop this redundant information and model a Barrett hand grasp only with the hand position, hand orientation and the finger separation angle. The robot kinematics is programmed into the simulator and all simulated robot movement is feasible.

The above two methods can be used to generate both simple shapes and complex shapes. The size of the generated training data varies from 500 to 1600 (Table 3.1). Each training dataset is split into 5 groups for the 5-fold cross validation in the later step.

3.2.2 Model learning

The second phase of the approach is to build a model Ω for the grasp demonstrations. A *Gaussian Mixture Model* (GMM) is used here to get a probabilistic encoding of the joint distribution $p(\boldsymbol{h},\boldsymbol{o},\theta \mid \Omega)$. We choose to use GMM because of its ability to effectively extrapolate the missing data, as has been exploited in many applications (Calinon et al., 2007; Sauser et al., 2011). It also has the advantage of capturing the non-linearity of the space, as well as determining how likely a point in the input space is under the model. The ability to estimate the likelihood of an input query point is crucial: an inference far away from the region covered by the training data can be unreliable, resulting potentially in an infeasible grasp. With GMM we are able to make sure that each input query point is located in or projected to a reliable region (this is explained in the next phase).

Therefore, in the grasp planing phase, we first make sure that a new query point locates in a reliable region by checking it likelihood. Given a set of sample grasps represented by the hand position $\boldsymbol{h}$, orientation $\boldsymbol{o}$ and the finger configuration $\boldsymbol{\theta}$, we model the distribution with a GMM as a sum of K Gaussian components:



(a) BIC

(b) 5-fold cross validation

Figure 3-4: The *Bayesian Information Criterion* and 5-fold cross validation test results of the training dataset of the Barrett hand and a joystick shaped object. For each number of Gaussians, the test is run 5 times. After empirical testing, the number of Gaussians is chosen to be 20. The corresponding experiment are shown in Section 3.3.

$$P(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}|\Omega) = \sum_{k=1}^{K} p_k p(\boldsymbol{h},\boldsymbol{o},\boldsymbol{\theta}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \tag{3.5}$$

where k is the number of Gaussian components, p_k the prior of the Gaussian component and the $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ the corresponding mean and covariance as:

$$\boldsymbol{\mu}_k = \begin{pmatrix} \boldsymbol{\mu}_{h,k} \\ \boldsymbol{\mu}_{o,k} \\ \boldsymbol{\mu}_{\theta,k} \end{pmatrix} \quad \boldsymbol{\Sigma}_k = \begin{pmatrix} \boldsymbol{\Sigma}_{hh,k} & \boldsymbol{\Sigma}_{ho,k} & \boldsymbol{\Sigma}_{h\theta,k} \\ \boldsymbol{\Sigma}_{oh,k} & \boldsymbol{\Sigma}_{oo,k} & \boldsymbol{\Sigma}_{o\theta,k} \\ \boldsymbol{\Sigma}_{\theta h,k} & \boldsymbol{\Sigma}_{\theta o,k} & \boldsymbol{\Sigma}_{\theta\theta,k} \end{pmatrix} \tag{3.6}$$

A GMM approach requires that the data space is locally convex. For a complex object shape, however, the grasp space of hand configuration — coupled with the finger joint space and constrained by the geometry of the object surface — may be a non-smooth manifold. In both of the data generation methods described above, we evenly distribute the testing points so as to reduce the possibility of missing small good grasp regions. By these means we obtain most of the possible grasps for the object and approximate a locally convex data distribution, which is suitable for a GMM.

Before training we 1) convert all data into the object reference frame and 2) normalize the data so that all dimensions have a zero mean and a unit variance. Initialized by the K-means, the *Expectation-Maximization algorithm* (EM) (Dempster et al., 1977) is used to find the value of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ that maximizes the probability of the training data under the GMM. The number of Gaussian K is selected by the *Bayesian Information Criterion* (BIC) and verified by 5-fold cross validation to make sure the model is not overfitting (Figure 3-4).

3.2.3 Grasp Planning

With the learned GMM model of the grasping demonstrations, we plan a feasible grasp given a current hand configuration $\boldsymbol{q}$=$\{\boldsymbol{h},\boldsymbol{o}\}$. As discussed above, we first need to determine whether the $\boldsymbol{q}$ is a valid query point. To do this we define a membership function $f(\boldsymbol{q})$ as:

$$f(\boldsymbol{q}) = \sum_{k=1}^{K} \bar{\boldsymbol{N}}(\boldsymbol{q}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.7}$$

where $\bar{\boldsymbol{N}}$ is the normal distribution with the output being normalized between 0 and 1:

$$\bar{\boldsymbol{N}}(\boldsymbol{q}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = exp\left(-\frac{1}{2}(\boldsymbol{q}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{q}-\boldsymbol{\mu}_k)\right) \tag{3.8}$$

We consider a point to belong to the model if its Mahalanobis distance to any Gaussian component is less than a given threshold σ. In our experiments, we find that within 1 standard deviation the success rate of finding a feasible grasp is constantly high. For example in the Barrett hand and the model plane grasping task, the rate of producing a feasible and stable grasp within 1 standard deviation is 85% (Table 3.1) while it is 64% within 3 standard deviations. On the other hand, it is possible that GMM encapsulates two different clusters of data within a single Gaussian, leaving the mean of the Gaussian at an infeasible point. This means getting closer to the means does not ensure a higher success rate. Taking this trade-off into account, we choose 1 standard deviation as our threshold, which gives us a cutoff criterion $\eta = exp(-\frac{1}{2}\sigma^2)$. If the membership function of a point has a higher value than η, we consider this point as a valid query point. Note that the finger configuration $\boldsymbol{\theta}$ is not part of this input query point as $\boldsymbol{\theta}$ will be inferred by GMR later.

This membership function differs from the marginal likelihood $p(\boldsymbol{h},\boldsymbol{o})$ in two aspects. Firstly, it gives each Gaussian component the same weight, regardless of their priors p_k. As the prior of each Gaussian is proportional to the number of data points that are explained by this Gaussian, using this information in our selection may bias our choice toward the "most popular" grasps, yielding less variety in the results. Secondly, $\bar{\boldsymbol{N}}$ is a normalized function bounded between 0 and 1. This ensures the points with the same Mahalanobis distance from a Gaussian will have the same membership value, regardless of the size of the covariance (Sauser et al., 2011).

In the case that $\boldsymbol{q}$ is not a valid query point, we need to project it to a new point $\boldsymbol{q}^*$ that has a membership function value higher than η. Here we use a closed-form solution by considering each individual Gaussian component. We first compare the Mahalanobis distances between the query point $\boldsymbol{q}$ and each Gaussian to find the nearest Gaussian component. The projection point $\boldsymbol{q}^*$ is found by projecting $\boldsymbol{q}$ to this nearest component (Figure 3-5). In the Mahalanobis space the Gaussian is in a uniform shape. As a result, the projection point $\boldsymbol{q}^*$ lays on the direction

from the $\boldsymbol{q}$ to the center of the Gaussian. Therefore the projection point $\boldsymbol{q}_k^*$ of the k^{th} Gaussian can be written as:

$$\boldsymbol{q}_k^* = \boldsymbol{q} + \alpha_k(\boldsymbol{q} - \boldsymbol{\mu}_k) \tag{3.9}$$

where α_k is a scalar. With $\sigma = 1$ and the equation

$$\bar{\boldsymbol{N}}_k(\boldsymbol{q}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = exp(-\frac{1}{2}\sigma^2) \tag{3.10}$$

we have the equation to easily compute $\boldsymbol{q}_k^*$:

$$-\frac{1}{2}(\boldsymbol{q}_k^* - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{q}_k^* - \boldsymbol{\mu}_k) = -\frac{1}{2} \cdot 1^2 \tag{3.11}$$

Once the projection point $\boldsymbol{q}^*$ is found, the *Gaussian Mixture Regression* (GMR) is used to predict a feasible finger configuration $\boldsymbol{\theta}^*$ for it. First we define:

$$\boldsymbol{\mu}_{q,k} = \begin{pmatrix} \boldsymbol{\mu}_{h,k} \\ \boldsymbol{\mu}_{o,k} \end{pmatrix} \qquad \boldsymbol{\Sigma}_{qq,k} = \begin{pmatrix} \boldsymbol{\Sigma}_{hh,k} & \boldsymbol{\Sigma}_{ho,k} \\ \boldsymbol{\Sigma}_{oh,k} & \boldsymbol{\Sigma}_{oo,k} \end{pmatrix} \tag{3.12}$$

and GMR then uses:

$$\hat{\boldsymbol{\mu}}_{\theta,k} = \boldsymbol{\mu}_{\theta,k} + \boldsymbol{\Sigma}_{\theta q,k}(\boldsymbol{\Sigma}_{qq,k})^{-1}(\boldsymbol{q} - \boldsymbol{\mu}_{q,k}) \tag{3.13}$$

$$\hat{\boldsymbol{\Sigma}}_{\theta\theta,k} = \boldsymbol{\Sigma}_{\theta\theta,k} - \boldsymbol{\Sigma}_{\theta q,k}(\boldsymbol{\Sigma}_{qq,k})^{-1}\boldsymbol{\Sigma}_{q\theta,k} \tag{3.14}$$

Finally, all the K components are taken into account and the target finger configuration $\boldsymbol{\theta}^*$ is predicted as the mean $\hat{\boldsymbol{\mu}}_\theta$ with the covariance $\hat{\boldsymbol{\Sigma}}_{\theta\theta}$ according to:

$$\hat{\boldsymbol{\mu}}_\theta = \sum_{k=1}^{K} \beta_k(\boldsymbol{q}^*)\hat{\boldsymbol{\mu}}_{\theta,k} \tag{3.15}$$

$$\hat{\boldsymbol{\Sigma}}_{\theta\theta} = \sum_{k=1}^{K} \beta_k(\boldsymbol{q}^*)^2\hat{\boldsymbol{\Sigma}}_{\theta\theta,k} \tag{3.16}$$

where

$$\beta_k(\boldsymbol{q}^*) = \frac{p_k p(\boldsymbol{q}^*|\boldsymbol{\mu}_{q,k}, \boldsymbol{\Sigma}_{qq,k})}{\sum_{k=1}^{K} p_k p(\boldsymbol{q}^*|\boldsymbol{\mu}_{q,k}, \boldsymbol{\Sigma}_{qq,k})} \tag{3.17}$$

Due to the probabilistic nature of the GMR, the inferred result $\boldsymbol{\theta}^*$ is not a unique value but a mean value with variance. Though this mean does not guarantee a feasible solution, it provides a good estimation of a feasible one.

To find the closest Gaussian component we used the Mahalanobis distance rather than the

Euclidean distance. The advantage of this is that it takes into account the correlations among each dimension of the hand configuration. In a space of different types of measurements, i.e. length and angle, Mahalanobis space is a better representation than the Euclidean space. Indeed, humans do not always use the Euclidean distance to select their grasps. We may move our hand further than needed to grasp an object, in order to avoid flipping our hand to another orientation. The performance of this method is discussed in the next section.

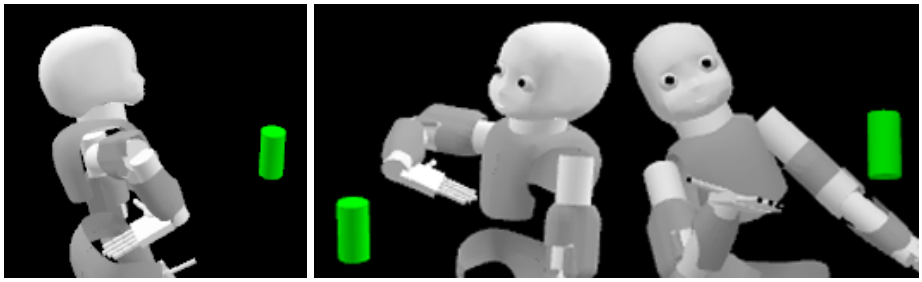
3.3 Experiments of planning grasps for familiar objects

This section presents a few results of our method (Figure 3-5, 3-6[2], 3-8). As mentioned above, grasps of the iCub hand are described in 14 dimensions: hand position (3D), hand orientation (3D in Euler angle), finger joint angles (8D), and grasps of the Barrett hand are described in 8 dimensions: hand position (3D), hand orientation (4D in axis-angle representation), finger separation angle (1D). Six different objects are presented here: cylinder, cuboid, ashtray, shoe, joystick and airplane model. For each object, three different initial postures and their final grasps are shown. Figure 3-5 shows the results of iCub grasping a cylinder, and the corresponding projections from the initial query points to the model. The results of the cylinder and cuboid show that a variety of grasps can be obtained for simple shapes to satisfy different task requirements. The ashtray, airplane model and joystick shapes are chosen from the GraspIt! object library, showing the method indeed works with complex shapes. In some figures the wrist may seem to rotate over 360 degrees to reach the final grasps from the initial pose. This is because the path planning of the arm is not taken into account in our approach. In terms of the hand orientation solely, a much smaller rotation is needed to go from the initial pose to the final grasp.

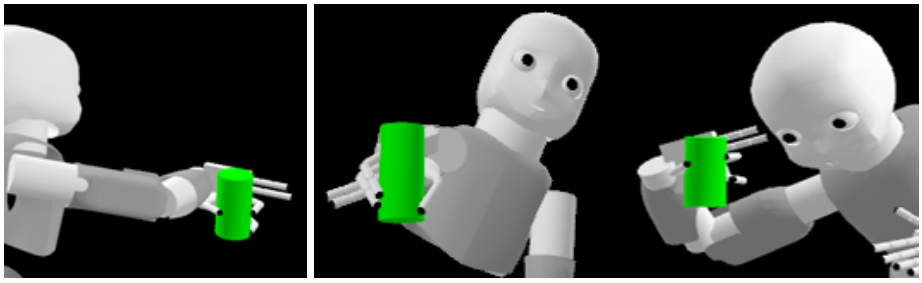
To test the computation time we generated 3000 random initial query points for each grasping task. The initial query points are placed at different distances away from the object surface, varying between $3cm$ to $50cm$, and the hand orientation is random. The initial finger configuration is not taken into account in finding the feasible region and hence it is set to the robot hand starting values. The computation time and experimental details are shown in Table 3.1.

Table 3.1 also shows the success rate of generated grasps with the iCub and the Barrett hands. A grasp is considered to be successful if it satisfies the force-closure criterion, is feasible for the hand kinematics and is not in collision with the object (see Section 3.2.1). When executing the obtained grasp, the fingers will continue to clutch until contact is made; if they contact the object surface before reaching the expected finger configuration, they will halt to avoid penetration. All the results shown in Figure. 3-5, 3-6, 3-8 are good grasps.

[2]The small penetrations and gaps between the fingers and the object are caused by two factors, (1) that the width of the fingers are not taken into account in the optimization and (2) the variance of the results. A supplemental implementation will be applied on the real scenario to handle the variances.



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3



(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

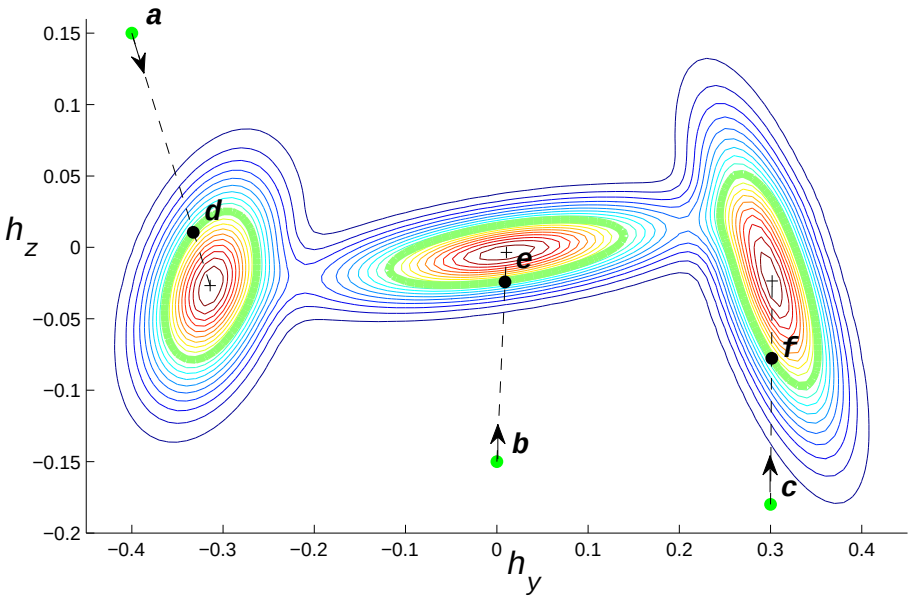
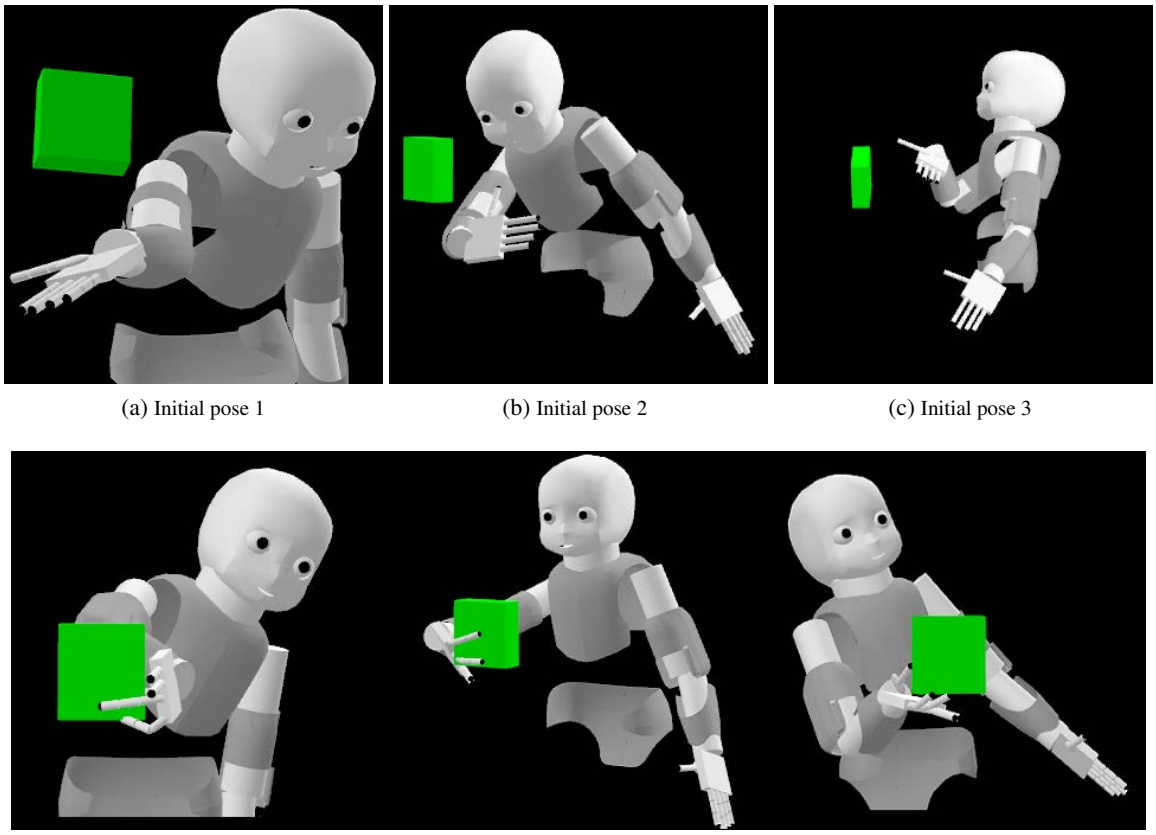


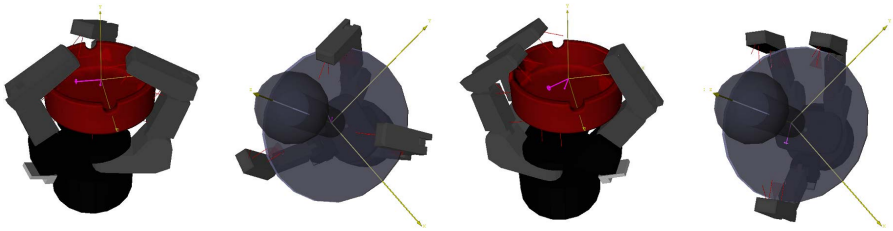
Figure 3-5: Two-dimensional illustration of the learned model. h_y and h_z correspond to the hand position along the y and z axis of the object reference frame. a, b and c are the initial query points, while d, e and f are their corresponding computed grasps. Green dots correspond to initial query inputs $\boldsymbol{q}$, black dots to found feasible query inputs $\boldsymbol{q}^*$, contours to parts of the space with constant likelihood, and the thick green contour to threshold value $\eta = \exp(-\frac{1}{2}\sigma^2)$ of each Gaussian, where $\sigma = 1$ standard deviations. The initial finger joint angles in a,b,c are all set to zero. After each feasible query point is found, GMR is used to predict the finger configuration to get the final grasp d,e,f.



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3

(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

Figure 3-6: Examples of iCub hand grasping of a cuboid. The first row (a,b,c) shows the initial postures and the second row (d,e,f) shows the corresponding final grasps.



(a) Best grasp found. Quality is 0.16. (b) Best grasp found. Quality is 0.19. (c) Neighbor grasp of (a). Quality is 0.027. (d) Neighbor grasp of (b). Quality is 0.03.

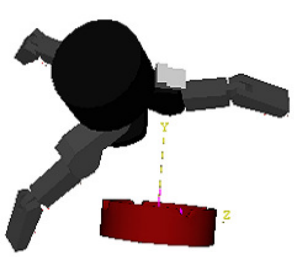
Figure 3-7: (a) The best grasp found for the Barret hand and the ashtray. (b) The best grasp found for the Barret hand and the joystick. (c) The nearest grasp of (a) in the training set. Note the gap between the finger and the object. (d) The nearest grasp of (b) in the training set.

As can be seen from Table 3.1, the success rate depends on the dimensions of the grasp space and the surface geometry of the target objects. Grasps in lower degrees of freedom (the Barrett hand) have higher success rates than those in higher degrees of freedom (the iCub hand). This suggests that the higher dimension grasp space is more complex than the lower dimension grasp space and needs more data to represent the full complexity. On the other hand, objects with smooth surfaces have a success rate around 90%. Objects with a couple of prominences have success rates over 85% as the configuration space of grasping is discontinuous. In the Barrett hand and airplane model task, the failed grasps are concentrated on two places: the thin edges of the wings and the propeller. Grasping these places requires high accuracy and more training data on these parts would be needed.

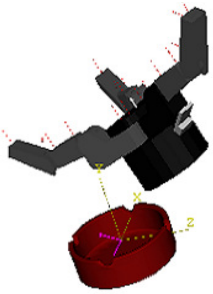
To compare with the training data, we compute the grasp quality of the results with the same metrics we used in data generation. The mean of the grasp quality of the training set and the result set are similar, though the result set has slightly higher value in most of cases. We are able to find some grasps of higher quality than all grasps in the training set (Figure 3-7). This shows that GMM is able to model and generalize the high dimensional grasp space, especially for objects with smooth surfaces.

This approach provides a good estimation of a stable and feasible grasp for the given object and robot hand. In contrast to the common approach of learning from human demonstrations, these grasps are generated solely according to the mechanics of the robot hand. Some resulting grasps are markedly different from human grasps, especially for the Barrett hand which is very different from the human hand. Our method may therefore out perform human demonstrations in some contexts by better exploiting differences between human and robot “physiology”.

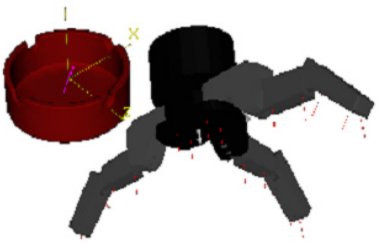
To model the actual contact points between the robot hand and the object is difficult in real time because of the high dimension of the solution space and the non-linearity of the kinematic constraints. In our method, instead of computing the actual contact point position, we compute the most likely solution using a GMM. Though a certain amount of accuracy is traded off to



(a) Initial pose 1

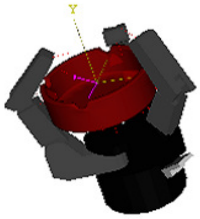


(b) Initial pose 2

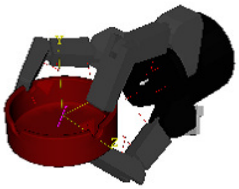


(c) Initial pose 3

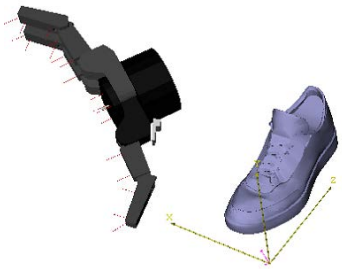
(d) Final grasp 1



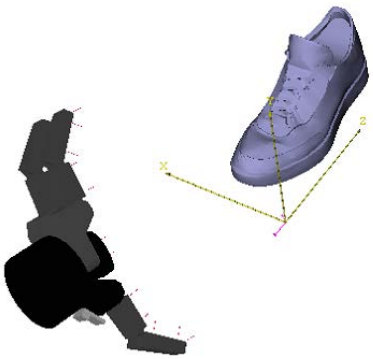
(e) Final grasp 2



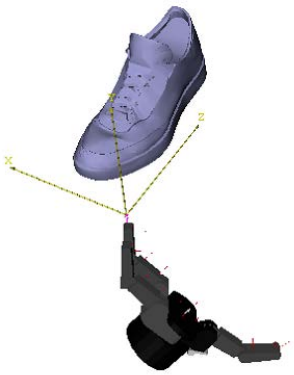
(f) Final grasp 3



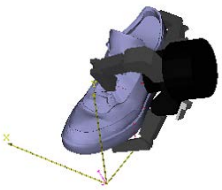
(g) Initial pose 4



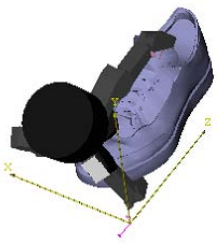
(h) Initial pose 5



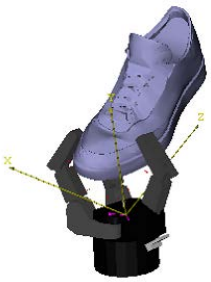
(i) Initial pose 6



(j) Final grasp 4

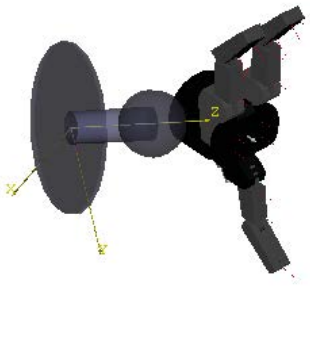


(k) Final grasp 5

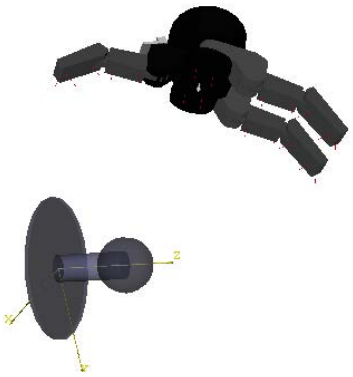


(l) Final grasp 6

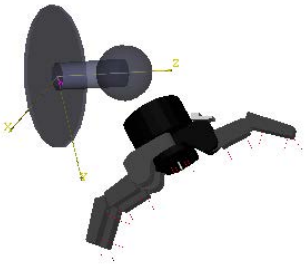
Figure 3-8: Examples of Barrett hand grasping different objects (ashtray, shoe, joystick, airplane model). The first and third rows (a,b,c,d,e,f and m,n,o,p,q,r) show the initial postures and the second and forth rows (g,h,i,j,k,l and s,t,u,v,w,x) show the corresponding final grasps.



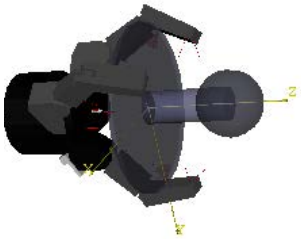
(a) Initial pose 7



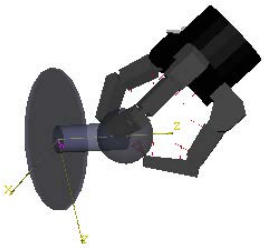
(b) Initial pose 8



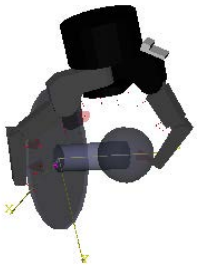
(c) Initial pose 9



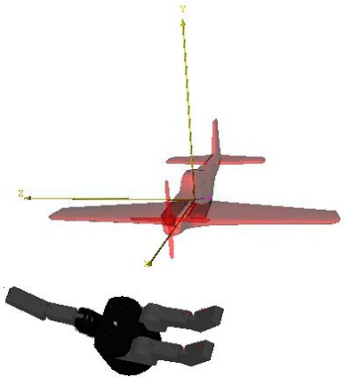
(d) Final grasp 7



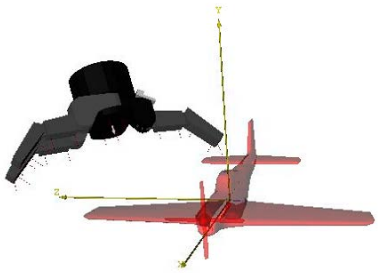
(e) Final grasp 8



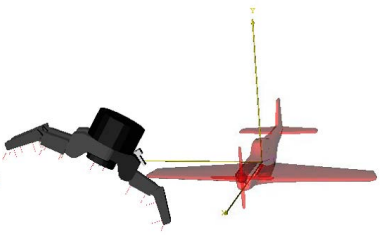
(f) Final grasp 9



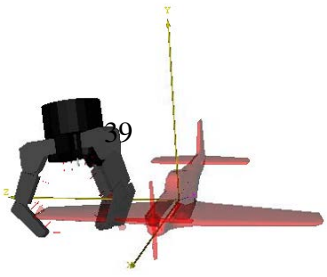
(g) Initial pose 10



(h) Initial pose 11



(i) Initial pose 12



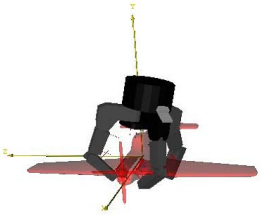


Table 3.1: Average computation time for generating new grasps for the iCub hand and the Barrett hand.

Robot/Object	Number of training data	Average Grasp Quality(train)	Number of Gaussians	Force-Closure Grasp Found	Average Grasp Quality(result)	Mean of Computation Time(*msec*)	Variance (*msec*)
iCub/Cylinder	621	0.0965	40	90%	0.1008	9.1	0.0001
iCub/Cuboid	532	0.1317	40	89%	0.1224	9.4	0.0007
Barrett/Ashtray	1560	0.0975	15	100%	0.1644	4.3	0.0001
Barrett/Shoe	629	0.0019	25	99%	0.0034	6.9	0.0001
Barrett/Joystick	1500	0.0061	20	98%	0.0064	5.9	0.0002
Barrett/Plane	1374	0.0002	55	85%	0.0003	15.1	0.0003

achieve the real time goal, the overall performance is satisfying. In the experiments listed above, over 90 percent of the testing points find good grasps within a few milliseconds. This method is most efficient for objects with a smooth surface. For complex objects this method can achieve a high success rate of over 85%. When grasping the parts requiring high precision, additional feedback from visual or tactile sensors is needed for further refinement of the grasp.

This approach requires the object model to be pre-trained. This is to say, we can only plan grasps for familiar objects with this method. It is useful for robot working in a control environment with limited number of objects, such as operation rooms. For domestic service robot, however, this is not enough. New objects will continuously come to the house and hence the robot has to be able to grasp novel object shape. An extension of this method to work on novel objects is discussed in the next two sections.

3.4 Grasping novel objects based on familiar parts

A method to compute grasps for novel objects is in need for domestic service robots. In this chapter we study the problem of generating grasps for un-trained objects in real time.

Objects used in daily life have a variety of shapes. Very often they share similar shape parts, such as sphere, cylinder and box. These shapes repeatedly appear in our daily life, being the object shape or the part of the object shape. Hence we call them“shape primitives”.

To work with un-trained objects, we take the grasping by shape primitives approach (Miller et al., 2003). This approach makes the assumption that all object shapes can be decomposed into a set of primitive shapes where grasps can be planned easily. Based on this assumption,

we firstly build a set of GMMs to model the grasp distribution $(\Omega_i, i = 1,2,...N)$ for a set of N chosen shape primitives. When a unseen object is presented, of which the shape can be approximated as a combination of known shape primitives, it's grasp distribution is built by combining the primitives' models. The combined model is then used to quickly generate new grasps.

3.4.1 Combine grasp distribution of shape primitives

3.4.1.1 Primitive grasp distribution

Here we define our shape primitives to be a set of superquadrics. We learn the grasp distributions for a set of superquadrics and use them as "primitive grasp distribution".

Superquadrics

Superquadrics are a family of geometric shapes that includes a large variety of shapes we use in daily life, such as cuboid, sphere and cylinder. We choose superquadrics as our shape primitives for three reasons. Firstly, superquadrics and their combinations can be used to represent most of the daily life objects. The widely use of superquadrics in computer graphic and game industry for modeling object shapes and that show its versatility. Secondly, all superquadrics are symmetry to their x, y, z axis. This can reduce the number of testing grasps to 1/8. Thirdly, it's implicit expression is convenient for combining their grasp density, which will be explained in detail in the Section 3.4.1.2.

To represent a superquadric we have:

$$r(x,y,z) = \left(\left(\frac{x-x_0}{a_1} \right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y-y_0}{a_2} \right)^{\frac{2}{\varepsilon_2}} \right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z-z_0}{a_3} \right)^{\frac{2}{\varepsilon_1}} \tag{3.18}$$

where (x_0, y_0, z_0) is the center, a_1, a_2, a_3 define the scale in the x, y, z axis respectively, and $\varepsilon_1, \varepsilon_2$ define the shape of the superquadric. We use the value of r to measure the relative position of a point x, y, z to the superquadric shape:

$$r \begin{cases} < 1, & \text{inside the superquadric} \\ = 1, & \text{on the surface of the superquadric} \\ > 1, & \text{outside the superquadric} \end{cases} \tag{3.19}$$

For sphere, cylinder and box primitives, the shape parameters are:

1. Sphere: $\varepsilon_1 = 1, \varepsilon_2 = 1$
2. Cylinder: $\varepsilon_1 = 1, \varepsilon_2 = 0.1$

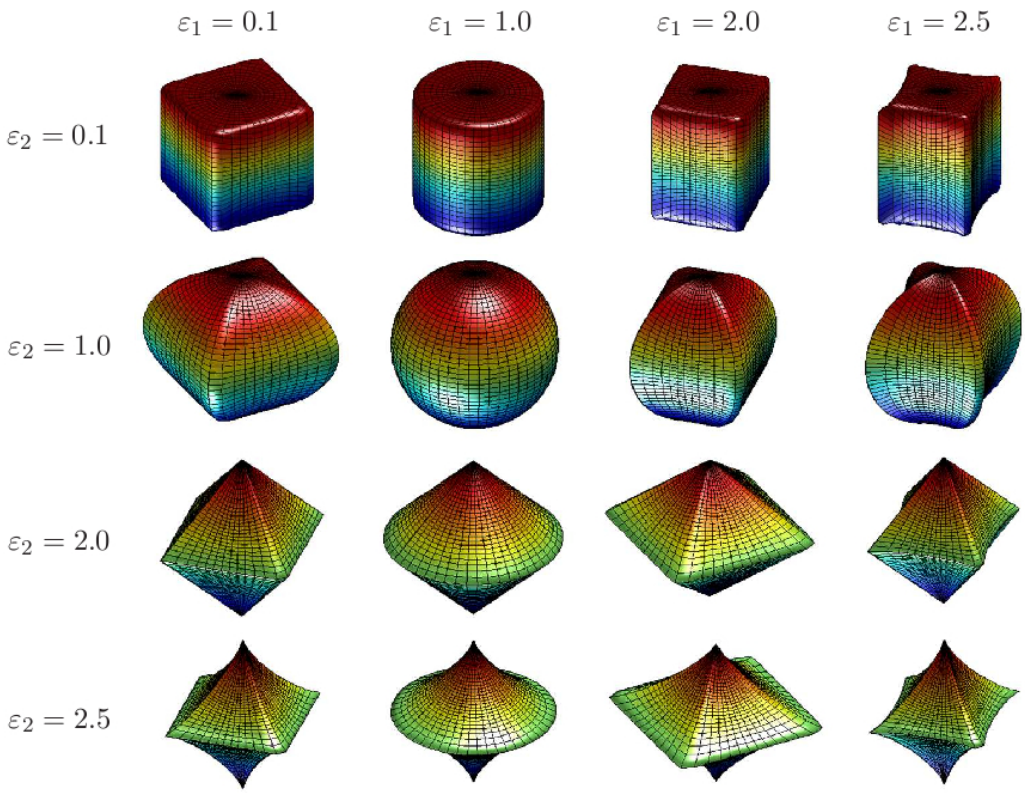


Figure 3-10: Illustration of 3D superquadric shapes with varying rounding parameters[4].

3. Box: $\varepsilon_1 = 0.1, \varepsilon_2 = 0.1$

Figure 3-10 shows how does the shape varies by these two factors [3].

Learning grasp distribution for shape primitives

With the method described in Section 3.2, we build GMMs for the feasible grasp distribution for a set of shape primitives, i.e. superquadrics. Again, we model the distribution with a GMM as a sum of K Gaussian components:

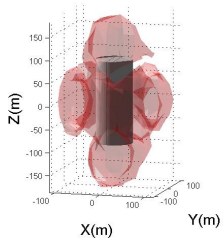
$$P(\boldsymbol{h}, \boldsymbol{o}, \boldsymbol{\theta} | \Omega) = \sum_{k=1}^{K} p_k p(\boldsymbol{h}, \boldsymbol{o}, \boldsymbol{\theta} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.20}$$

where k is the number of Gaussian components, p_k the prior of the Gaussian component and the $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ the corresponding mean and covariance.

Figure 3-11 visualizes the grasp distributions encoded by GMMs for three shape primitives: a box, a sphere and a cylinder. The robot hand we use here is the Barrett hand.

[3]Figure from internet source http://www.vincent-morio.com/content/en/gallery.html

TODO TODO



(a) Box (b) Sphere (c) Cylinder

Figure 3-11: A 3D visualization of the feasible grasp distribution for three shape primitives and the Barrett hand. The red contours are the isosurfaces of the grasp distribution. The "reder" the area is, the denser the distribution is.

3.4.1.2 Combining grasp distribution

For the an object composed by a few shape primitives, its grasp distribution is the combination of the the grasp distributions of its primitive components. However, this "combination" is not a summation of the GMMs: we need to exclude the grasps causing collision.

Before combining the grasp distributions, which is encoded by GMMs, of different shape primitive components of an object, we need to reshape each GMM by the object geometry. This is because when primitives combine to a more complex shape object, the grasp space of one part might be blocked by other part of the object.

For example, an object combined by a cylinder and a sphere is shown in Fig. 3-12(a) and it's primitives with their grasp distributions are shown in (b) and (c). As can be seen in (b) and (c), the top part of the grasp distribution of the cylinder will be inside the sphere and the bottom part of the grasp distribution of the sphere will be inside the cylinder. Grasps generated from these parts will cause collision between the hand and the object and hence they have to be excluded from the model. Further, even when the robot hand is placed in a feasible location for all components, when the fingers clutch to grasp one component of the object, the moving trajectory of the finger might be blocked by other components. This kind of grasps should also be excluded from the grasp density of the whole object.

Therefore, we take into account two kinds of collisions: collisions between the palm and the object, collisions between the finger trajectories and the object. To avoid collisions, we use the "object sigmoid" function to remove the collision parts.

Object sigmoid

We define a shape descriptor "object sigmoid" for objects modeled by superquadric. The

object sigmoid is a 3 dimensional sigmoid function defined as:

$$s(x,y,z) = \frac{1}{1+e^{-100(r(x,y,z)-1)}} \tag{3.21}$$

where r(x,y,z) is a function of the location defined in the form of a superquadrics. When we model an object shape with a superquadric, the object sigmoid has different values inside, on and outside the object:

$$s \begin{cases} \to 0, & \text{inside the object} \\ = 0.5 & \text{on the surface of the object} \\ \to 1 & \text{outside the object} \end{cases} \tag{3.22}$$

In equation 3.21 we choose a large coefficient, i.e. 100, for r to make a sharp transition between 0 and 1 and hence a sharp cut on the object surface. Hence the object sigmoid gives a description of the shape of the object in the space: zero inside the object and one outside the object.

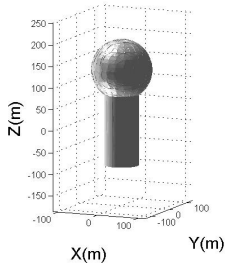
Each primitive component has its own object sigmoid. Before combining the individual to form the whole distribution, each individual distribution is multiplied by all other component's object sigmoid. In this way, the likelihood inside the other parts of the object is reduced to zero, while the likelihood outside the object remain. The grasp distribution is hence "trimmed" by the other components of the object. The grasp distribution of the whole object is the summation of all the trimmed grasp distribution:

$$\mathbf{\Omega}(x,y,z) = \sum_{i=1,2..}^{N} \left(\Omega_i \prod_{j=1,2..(j\neq i)}^{N} s_j \right) \tag{3.23}$$

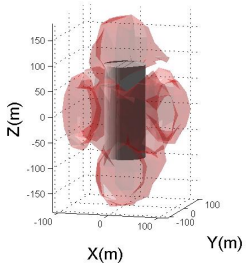
where Ω_i, s_j, N is the grasp distribution of the $i-th$ primitive component, the object sigmoid of the $j-th$ primitive component and the total number of primitive components. The total number of Gaussians in the GMM of the whole object is the sum of the number of each primitives.

Figure. 3-12(d) shows the resulting grasp distribution of the whole object, which is the combination of the trimmed grasp GMM of the cylinder (Figure. 3-12(d)) and the sphere (Figure. 3-12(f)). Strictly speaking the combined grasp distributio is not a density function as the integral of the probability of the whole space is not normalized to one. Despite this it does not effect the computation of a new grasp as we consider each Gaussian component individually.

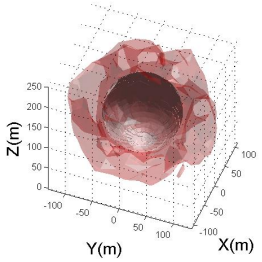
The equation above removes the grasps inside the object and hence avoids the collision between the robot palm and the object. However, grasps causing collision between finger trajectory and the object still remains.



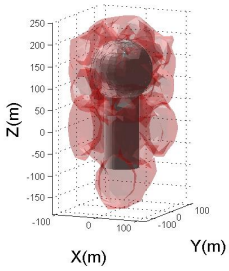
(a) Novel object



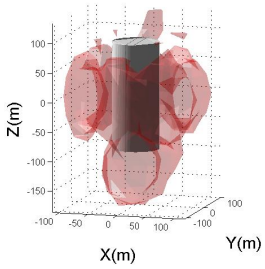
(b) Cylinder grasp distribution



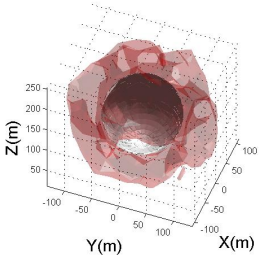
(c) Sphere grasp distribution



(d) Combined object grasp distribution



(e) Trimmed grasp distribution



(f) Trimmed grasp distribution

Figure 3-12: (a) A combination of a cylinder and a sphere. (b) A 3D illustration of the grasp GMM of the cylinder. The red patch is the isosurface of the grasp GMM. (c) The grasp GMM of the sphere. (d) The combined grasp GMM of the whole object (d=e,f)). (e) The trimmed grasp GMM of the cylinder. The top part of the GMM is removed. (f) The trimmed grasp GMM of the sphere. Part of the bottom of the GMM is removed.

TODO: Trim grasp distribution by finger trajectory

3.4.2 Plan Grasp by combined grasp distribution

With the combined grasp distribution for the whole object, we can fast plan a grasp with the method described in Section 3.2. Starting from an initial hand position and orientation $q = \{h, o\}$, the first step to compute a new grasp is to project q to the feasible region of the GMM, where the probability of finding a stable grasp is high. This is done by finding the minimum Mahalanobis distance between q and its projection point $q_k^\star$ in each Gaussian component of the GMM.

The projection point $q_k^\star$ of the k-th Gaussian component is computed as:

$$q_k^\star = q + \alpha_k (q - \mu_k) \tag{3.24}$$

where μ_k is the mean of the k-th Gaussian and α_k is a scalar determined by the boundary of the feasible region.

For $q_k^\star$ we have

$$-\frac{1}{2}(q_k^\star - \mu_k)^T \Sigma_k^{-1} (q_k^\star - \mu_k) = -\frac{1}{2} \cdot 1^2 \tag{3.25}$$

with equation 3.24 and 3.25, α_k can be computed and hence $q_k^\star$.

The feasibility of each projection point is checked by the object sigmoids:

$$l_k = \prod_{j=1,2,\ldots(j\neq k)}^{N} s_j \tag{3.26}$$

If l_k is smaller than 1, indicating that this point is inside or on the surface of the object, the likelihood at point $q_k^\star$ is zero.

We find the nearest projection point from the $q_k^\star$ with non-zero density. The nearest $q_k^\star$ is chosen to be the final grasp hand position and orientation $q^\star$. The grasp distribution $\Omega^\star$ which $q^\star$ locates in is used to compute the corresponding joint configure though GMR. This allows us to compute the expected value of the finger joints from the conditional $p(\theta \mid q^\star, \Omega^\star)$ (Section 3.2).

3.5 Experiments of planning grasps for non-familiar objects

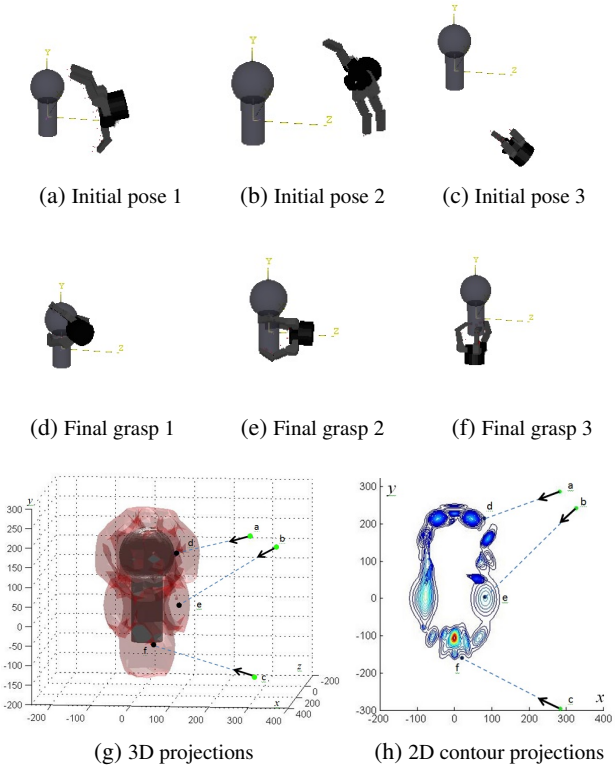
We test our approach initially on a novel object that is a combination of a sphere and a cylinder (Figure. 3-12(a) and Table 3.3). We choose to use the Barrett hand for the implementation as it is available in our lab. As explained above, the grasp of the Barrett hand is formulated as

Approach and object	Force-Closure Grasp Found	Mean of Computation Time(*msec*)	Variance (*msec*)
Pre-trained grasp GMM for the novel object	98.1%	13.8	0.015
Combined grasp GMM for novel object	92.1%	21.9	0.011
Combined grasp GMM for spray flask	91.0%	16.0	0.004
Combined grasp GMM for bedside table	82.2%	21.1	0.006

Table 3.2: Success rate and computation time of different methods and objects

Shape primitives	Object	Size(*cm*)	Number of training data	Number of Gaussians in GMM
Sphere 1	Novel object (Fig. 3-12)	radius 7	12096	60
Cylinder 1	Novel object (Fig. 3-12)	height 15 and radius 4	15608	60
Box 1	Spray flask (Fig. 3-14)	$6 \times 9.5 \times 8$	9256	40
Box 2	Spray flask (Fig. 3-14)	$4 \times 11 \times 4.5$	7544	40
Box 3	Spray flask (Fig. 3-14)	$2.5 \times 4 \times 7$	3400	30
Box 4	Bedside table (Fig. 3-16)	$52.5 \times 3 \times 52.5$	8668	20
Box 5	Bedside table (Fig. 3-16)	$2.8 \times 57.6 \times 2.7$	4392	20

Table 3.3: Shape primitives used in experiments



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3

(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

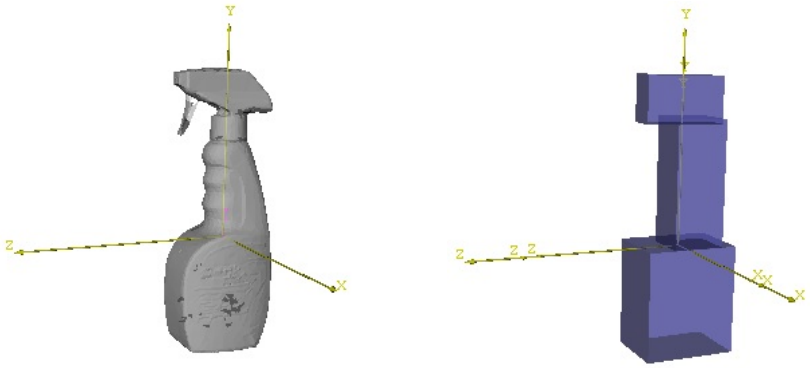
(g) 3D projections (h) 2D contour projections

Figure 3-13: Examples of Barrett hand grasping of a novel object. (a-d) Initial hand postures and final grasps. (g) A 3D illustration of the projection between the initial hand postures and the final grasps. (h) a 2D illustration of the interaction of GMM at $z = 0$.

the combination of the hand position (h), orientation(o) and finger joint angles(θ). The grasp GMMs of the sphere and cylinder are pre-trained with randomly generated stable grasps from the simulator GraspIt!.

We compare this new approach with the previous approach that directly train a grasp GMM for the whole object, by generating grasps for the object from 1000 starting points. Figure. 3-13 shows a few resulting grasps. As shown in the Table 3.2, the success rate and the computation time of the new approach is in the same scale of the previous approach. The computation time is computed by Matlab on a machine with 4GM RAM.

Further, we train 5 different boxes as our primitives (Table 3.3) and use them to approximate two daily life object: a spray flask and a bedside table. The spray flask is approximated as the combination of box 1, 2 and 3 (Figure. 3-14) and the bedside table is approximated as the combination of box 4 and 5 (three copies of box 4 as the surfaces and 4 copies of box 5 as the legs). The result is shown in Table 3.2. A few initial hand postures and their resulting grasps are shown in Fig 3-15 and 3-17.



(a) (b)

Figure 3-14: (a) A spray flask. (b) A spray flask approximated by 3 boxes.

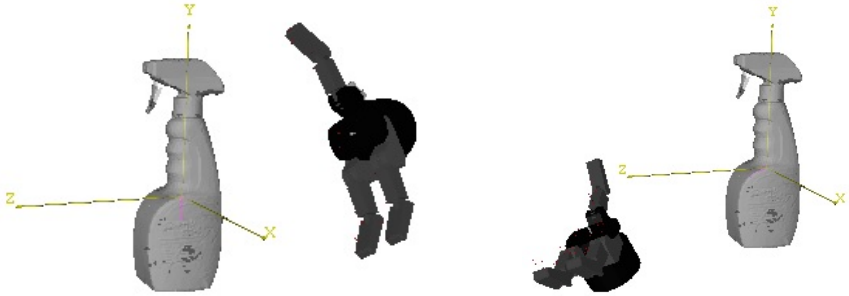
3.6 Discussion

We presented a method for computing grasps in real time. This is demonstrated on two very different robot platforms: the iCub hand and the Barrett hand. The result shows that the method can capture the versatility of grasps that are typical of grasps performed by an industrial gripper, and those that can be performed by a humanoid hand. With the computation time in the millisecond scale, this method would enable the robot to react quickly in robot-human interaction, such as picking up heavy objects from a person's hand, as well as adopting to fast perturbations in a dynamic environment.

We achieve this goal by using a closed-form solution. A GMM model is learned from the grasping demonstration generated offline for a given object. During the online execution no iterative method is used, we only need to solve a few equations with basic arithmetic operations. Hence the computation time is significantly shorter than the conventional optimization methods. Though we need to pre-train the robot to grasp different objects, in many scenarios such as surgery assistance, robots and humans must work with a predefined set of objects. This allows us to build the grasping model for each object beforehand.

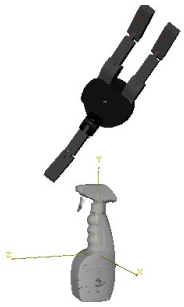
To find the closest Gaussian component we used the Mahalanobis distance rather than the Euclidean distance. The advantage of this is that it takes into account the correlations among each dimension of the hand configuration. In a space of different types of measurements, i.e. length and angle, Mahalanobis space is a better representation than the Euclidean space. Indeed, humans do not always use the Euclidean distance to select their grasps. We may move our hand further than needed to grasp an object, in order to avoid flipping our hand to another orientation.

Our approach provides a good estimation of a stable and feasible grasp for the given object and robot hand. To model the actual contact points between the robot hand and the object is

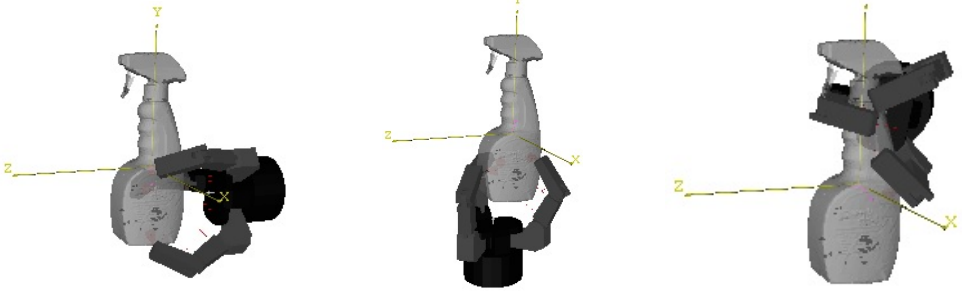


(a) Initial pose 1

(b) Initial pose 2



(c) Initial pose 3

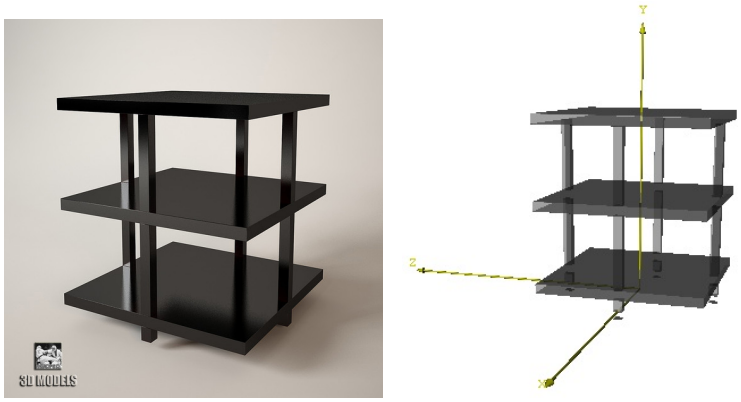


(d) Final grasp 1

(e) Final grasp 2

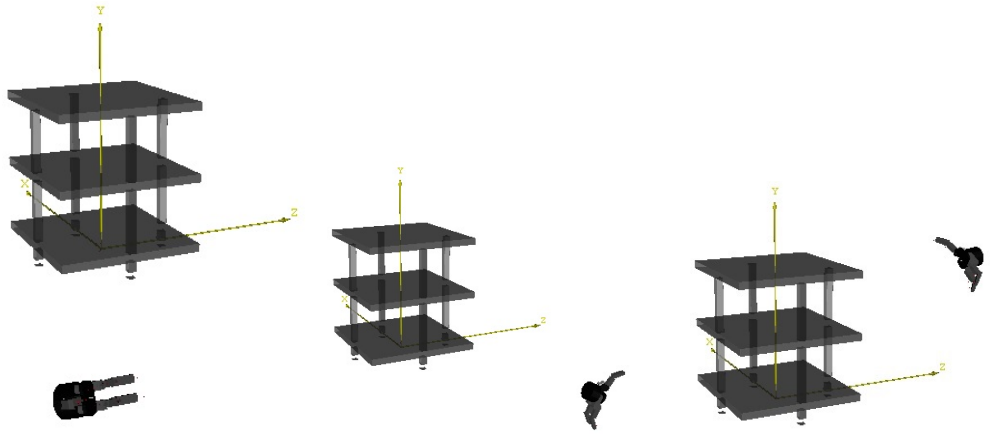
(f) Final grasp 3

Figure 3-15: Examples of Barrett hand grasping of a spray flask.

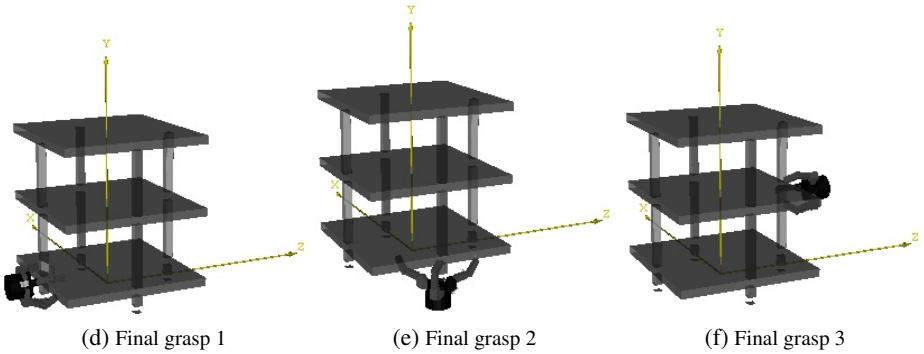


(a) (b)

Figure 3-16: (a) A bedside table. (b) A bedside table approximated by 7 boxes.



(a) Initial pose 1 (b) Initial pose 2 (c) Initial pose 3



(d) Final grasp 1 (e) Final grasp 2 (f) Final grasp 3

Figure 3-17: Examples of Barrett hand grasping of a complex shape bedside table (a-d) Initial hand postures and final grasps.

difficult in real time because of the high dimension of the solution space and the non-linearity of the kinematic constraints. In our method, instead of computing the actual contact point position, we compute the most likely solution using a GMM. Though a certain amount of accuracy is traded off to achieve the real time goal, the overall performance is satisfying. In our experiments, over 90 percent of the testing points find good grasps within a few milliseconds. This method is most efficient for objects with a smooth surface. For complex objects this method can achieve a high success rate of over 85%. When grasping the parts requiring high precision, additional feedback from visual or tactile sensors is needed for further refinement of the grasp.

When extending this approach to grasp non-familiar objects, we exploit an modular approach: grasp by shape primitives. The grasp distribution is learnt for each shape primitive and is combined to form the distribution for the whole object. Experiments show that this method enable the robot to grasp complex objects with high successful rate (over 80%) while maintaining the computation time with in millisecond scale.

In contrast to the common approach of learning from human demonstrations, these grasps are generated solely according to the mechanics of the robot hand. Some resulting grasps are markedly different from human grasps, especially for the Barrett hand which is very different from the human hand. Our method may therefore out perform human demonstrations in some contexts by better exploiting differences between human and robot "physiology".

Chapter 4

Learning human manipulation skill

4.1 Introduction

In our daily 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 like writing and using chopsticks. Service robot won't be able to really "serve" human without these manipulation abilities. Enabling robot to do manipulation tasks can alleviate human workload and free human from many chores. However, a robot with human level manipulation skill still only live in science fiction.

Generally, manipulation tasks are very difficult for robot. Different from pure motion planning, manipulation planning aim to not only move the robot to a desire state, but also change the environment to a desire state. Therefore in addition to robot motion planning, the impact robot put on the environment, i.e. robot environment interaction, has to be planned. The object interactions are usually complex and hard to predict as they involve complicated contact situations, and the changing kinematic and dynamic property of the environment. The complicated physics in object interactions makes manipulation tasks difficult. The multi-body interaction and the effect of friction can cause abrupt changes in the environment. This makes the environment nonlinear and non-stationary.

Control methods depending on invariable environmental parameters is not efficient for most manipulation tasks. Adaptive control method, which focus on handling varying parameters and initial uncertainly, is required for manipulation. Adaptive control strategy is usually hard to design, especially when it involves complex environment. This demands deep inside to the task and the kinematic and dynamic property of the environment.

To this end, we conduct a learning from human demonstration approach to gain adaptive control strategy. This approach has benefits in two-fold. First, we do not need to analytically derive the the kinematic and dynamic property of the environment in order to design the con-

troller. Second, it provides a framework to easily program robot with a task skill. With an increasing use of robot in daily life, more and more tasks will need to be programmed. Non-linear control methods are usually limited to narrow categories of tasks and hence needs to be done task by task. It is impractical to pre-program all such object manipulation tasks manually. Learning from human demonstration enable even non-programmers to program robots to do various of tasks quickly.

Humans can perform these skilled tasks and adapt to the changes without difficulty. At the heart of this skill is prediction Flanagan et al. (2006). Studies from neuroscience suggest that human develop internal models for motor control, 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 the tasks and launch the corresponding motor correction and motor reaction to adapt.

Inspired by this concept, we propose an approach to learn human adaptive control strategy. This adaptive control strategy is encoded with a modular model, which allows fast adaption to nonlinear and non-stationary system. 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. By this method, we modularize human adaptive control strategy. Internal models (forward model and inverse model) are learnt within each modules with a representation that can be easily transfer to robot. When a robot executes a similar task, the forward models estimate the context of the task and "contextized" the inverse models to generate proper command that drives the object. Each learnt control strategy is weighted by the similarity of the current task context and its corresponding context. The optimal strategy is computed as the linear combination of the weighted commands of each strategy.

The approach does not require any prior knowledge of the kinematics nor dynamics of the operation system, nor is it restricted to a specific robot platform. The control strategy is learnt on the object level and hence can be transfer from human to robot directly. This work contributes a framework to modularize human adaptive control strategy of manipulation tasks and to transfer the learnt internal models to robot. To verify our approach, we use a *Opening Bottle Caps* as an experimental. An adaptive control strategy is required here, as the friction between the bottle and the cap surfaces has multiple phases. We modularize the human control strategy in this task and implementing it on a robot to open both familiar and novel bottles.

In the next few sections, I will present our approach of learning a multiple module model of a human manipulation strategy 4.1, detail the experimental setups 4.2 and discuss the results 4.3.

4.2 Modular approach in manipulation

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 4.2.1).
2. Extracting human control strategies for different contexts and build multiple internal models(Section 4.2.2).
3. Use the multiple models to compute motor commands for a robot(Section 4.2.3).

Figure 4-1 shows an overview of our framework.

4.2.1 Human demonstrating tasks with direct contact to objects

The first step is to demonstrate a task to a robot. Two major forms of demonstration are used in teaching manipulation tasks: kinematics teaching and tele-operation.

Kinematics teaching

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. (2014b). 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. Hence kinesthetic teaching is not feasible for demonstrating multifinger task.

Tele-operation teaching

To control multi-finger hands, some researchers use tele-operation Bernardino et al. (2013); Kondo et al. (2008); **?**. This usually relies on data gloves to sense hand posture, 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 the above methods provide no direct force feedback to the human demonstrator during manipulation.

Human direct demonstration

In some studies, the human demonstrate manipulation tasks with their own bodies Asfour et al.

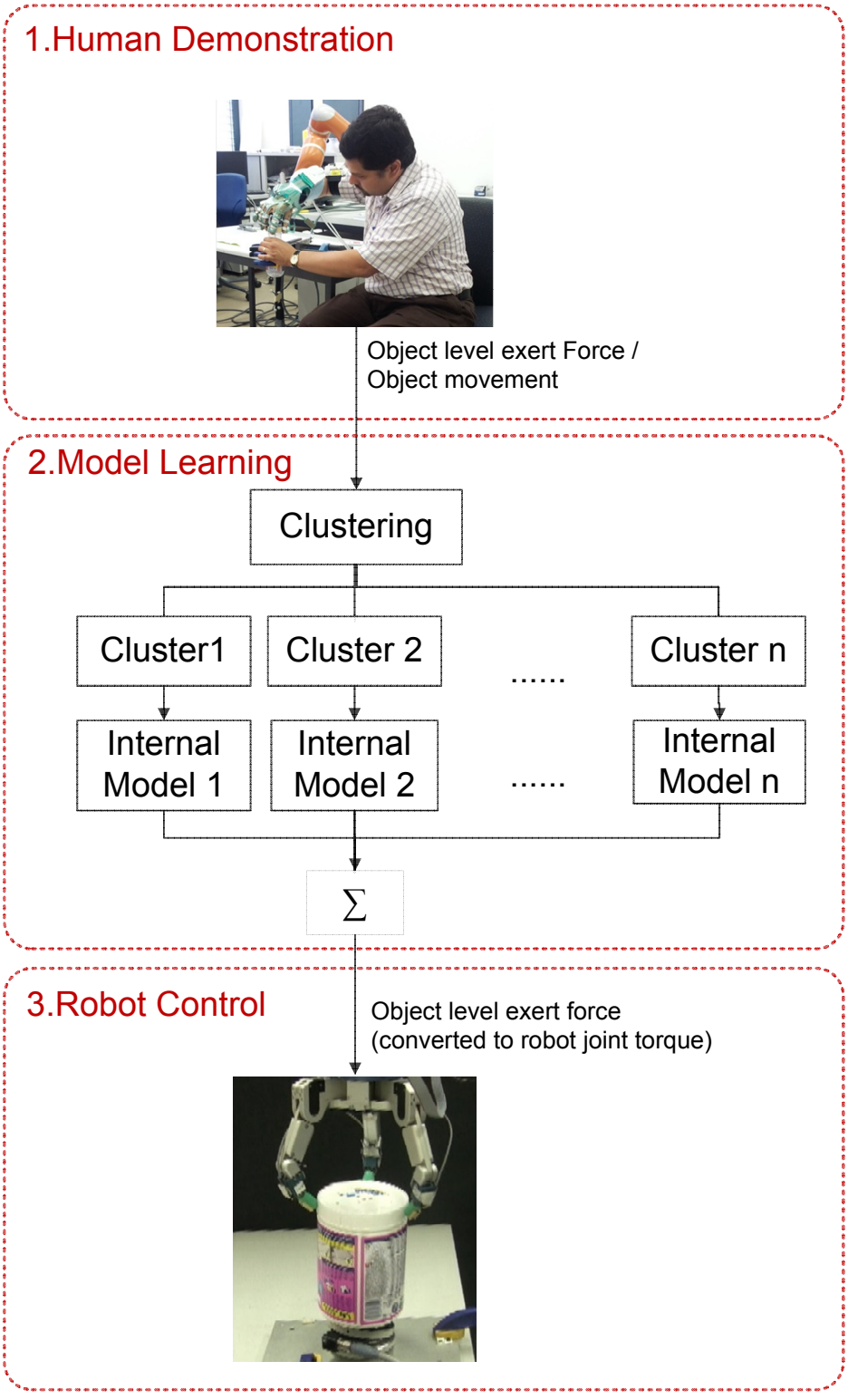


Figure 4-1: System overview.

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

Figure 4-2: Sensors used in the human demonstration of opening a bottle cap task.

(2008). With direct interaction with the object, human 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. Because the difference of the embodiment between human and robot, a mapping between them has to been built. Various mapping methods have been proposed Do et al. (2011); Asfour et al. (2008); Hueser et al. (2006), 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. These methods are quite robot specific or task specific. In general, how to effectively transfer human skills to robots skill remains a open problem.

Here We use 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 demonstrations, a human wears tactile sensors and directly interacts with objects. In this way, human demonstrator have direct cutaneous and kinaesthetic feedback, which is desirable for good manipulation demonstration. To make the information embedded in the demonstration can be easily transfer to robot, the demonstration is expressed from an "object centric viewpoint".

Object centric representation

The object centric viewpoint considers a manipulation task from the object's perspective, which suggests the goal of learning a manipulation task is to reproduce the same object behavior, rather than reproduce the demonstrator's movement. This makes sense as human may use different ways to accomplish a manipulation task: the motion or posture might be different but the effect on the object is the same. What the robot needs to imitate is the effect on the object but not the movements. Hence our approach takes this principle and learns a control strategy to produce a desired object behavior, rather than to produce a human or robot behavior. This strategy expressed from the object perspective can be transfer to a robot platform by covering

the exert force to robot joints' torque with Jacobian matrix. From the object centric viewpoint a manipulation problem is simplified: we transfer from a problem of controlling multifinger (multi-endeffector) and its impact of the environment to controlling an object.

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. 4-2 shows a few sensors we used in the opening bottle caps task. The representation of data will be explained in Section 4.2.2.1

Demonstrations in different task contexts

During the demonstrations, the teacher perform a task a number of times to generate enough data for capturing the key features. Further, the teacher perform the task under different system kinematics and dynamics configurations, e.g. friction conditions, in order to explore how human adapt to different task contexts. These different configurations are chosen to cover a wide range of different contexts. These wide range demonstrations are then used to learn a multiple module model.

4.2.2 Learning a Multiple-Module Model

In this section we detail our modeling method, explaining why do we use modular approach and how do we model human manipulation strategy.

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); Sugimoto et al. (2012) presents promising modular based approaches.

The excellent ability of human to manipulate different objects in different contexts and to quickly adapt to change of contexts suggested that our central nervous system (CNS) maintains multiple internal models of outside environments, rather than a single internal model that adapts to new environment Neilson et al. (1985). Indeed, neuroscience evidence suggests that animal brain sum modular stimulates of muscles to generate different motions Mussa-Ivaldi et al. (1994). Inspired by this, we take a modular approach to model human adaptive control strategy.

Our modular strategy

The modular approach we take is inspired from MOSAIC (Modular selection and identification for control) Haruno et al. (2001), one of the pervasive multiple module models for human motor control mechanism.

The system of MOSAIC is constituted by multiple parallel modules. Each module has three components: a forward model, an inverse model and a responsibility factor (RF). The forward model is responsible for estimating the task context in real time, and the inverse model is used to generate appropriate motor command for the context. These two models are connected by the RF. The task context estimated by the forward model is compared with the actual current task context. The RF of each module is computed according to the accuracy of the estimation: the more accurate the forward model predicts, the higher the RF is (detailed in Section 4.2.3.2). The RF of all modules are computed and then normalized. The inverse models are weighed by its normalized RF. The final motor command is the linear combination of the commands factored by their weighs. With this mechanism, the module predicting the current task context better takes more responsibility in the final motor command. Figure 4-5 sketches the work flow of this system. Though the MOSAIC has gained a large concerns in neuroscience community, this approach is not widely used in robot control.

We take this paradigm and model our internal models by GMM. Training GMM with the Expectation Maximization algorithm (EM), we estimate the optimal values of the model parameters. Compare to the early work Wolpert and Kawato (1998) which use Neural Network and has to manually tune the variance of each forward model, GMM has the advantage of automatically computing the all the model parameters. Later work Haruno et al. (2001) fixes the hand tuning problem by modeling with Hidden Markov Model (HMM) and optimize the model with EM. With this method the forward models are assumed to be linear. In our approach, GMM allows non-linear system to be modeled. Fig. 4-5 illustrates the workflow of our approach. Compare to the switching modular method Narendra and Balakrishnan (1997), 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 some tasks 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. In our approach we build the forward and inverse model separately.

Despite the many applications and discussions of the modular approach, how to systematically modularize the control strategy presented during the human demonstration, i.e. determine the number of modules and build appropriate model for each module, still remains a open problem. We tackle this problem by a data driven approach. We cluster the demonstration data with

an hierarchical approach and model each cluster as a pair of forward and inverse model. This solution can be applied to modularize many manipulation tasks. A similar clustering method has been applied to group and build tree structure of human motion pattern primitives Kulić et al. (2008). To cluster the motion primitives, a high value and a low value of the cut off parameter are tested to evaluate the trade off effect between facilitating quick tree formation and introducing misclassification. In our approach, the cut off parameter is determined by the variance of the data and hence avoid this step. This provide us with a proper grouping of the data and which can generate proper motor command for control. 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.

4.2.2.1 Object centric manipulation strategy

As mentioned in Section 4.2.1, one of the challenges in imitation learning is the mapping problem, i.e. how to map the teacher's motions to the robot's motions so that they produce the same effects. This mapping becomes more difficult for object manipulation tasks, the goal of which is to deliver the object from the current state to a desired state. During this process the movement of the manipulator is bounded not only by its own kinematic constraints but also bounded by 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 we take an "object centric approach" Okamura et al. (2000), of which the control policy is taken from the object point of view.

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 Okamura et al. (2000). 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.

4.2.2.2 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 different strategies from demonstration and build one module for each of the strategies. During implementation, the system can quickly estimate the context by the sensory inputs and weight the modules according to their reliability and then generate a contextilize motor command.

Different tasks may need different number of modules. This number may not be easy to find. In previous studies Sugimoto et al. (2012); Haruno et al. (2001), the number of modules is defined as number of different target objects or different phases in the task, which can be clearly distinct such as no load or on load. However this is not always the case. In a task involve more phases, human may regard different phases as the same task context and handle them with the same control strategies. A recent study suggests that modularize a control strategy by the number of objects can cause redundancy of modules and hence is inefficient Stphane Lalle and Rousset. (2009).

Here we propose a data-driven approach to properly define the number of modules for a given task. To this end, we should differentiate different types of strategies. The differences can be reflected from the different patterns of the force-torque-displacement tuple. We differentiate the patterns by clustering. 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, such that the data in the same group are more similar to each other than those in different groups. This technique has very important applications in computer vision and language processing Warren Liao (2005). Numerous clustering algorithms have been proposed for different purposes.

Before clustering, we need to measure the similarities, i.e. the distances, between the data points we want to cluster. The similarity metric is user defined according to the purpose of clustering. One of the mostly used metric is the Euclidean distance. Different from an usual clustering problem, however, the data we need to cluster are not a set of single data points but a set of time series. Each time series include a sequence of data point. 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

Dynamic time warping is a technique for measuring the similarity between two time series, which may have different speeds or durations. It has an important application in speech recognition. For example, two person may speak the word “hello”in different speeds. By the DTW, one is able to tell they are speaking the same word. This is achieved by warping the single in the time dimension. The two time series are “aligned” by finding their optimal match. The

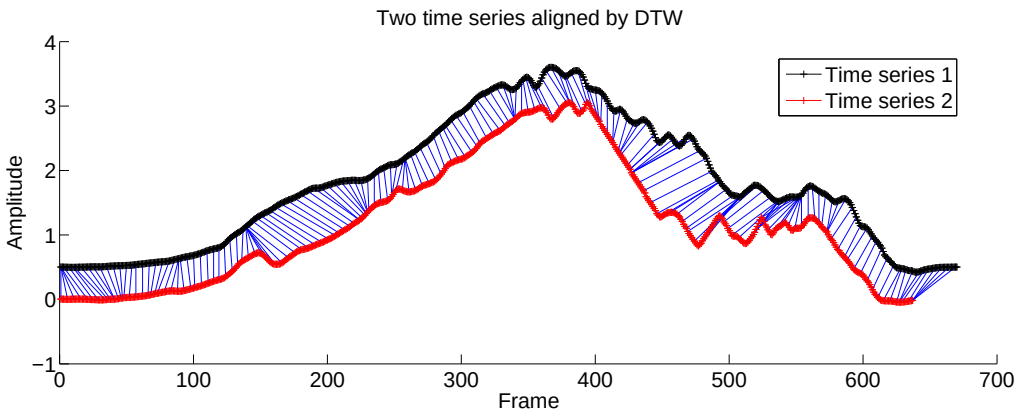


Figure 4-3: Two time series aligned by DTW. Red and black lines are the raw time single. The blue lines connect the matching points between them. DTW wrap the two time series nonlinearly so that the time independence similarity can be measured. The time series 1 is moved up by 0.5 for display reason.

similarity is computed as the average distance between the corresponding points of the two time series. By this method, the similarity computed by DTW is independent to the variance in the time dimension. Figure 4-3 shows an example of alignment of two time series by DTW.

Here we make an assumption that our manipulation task is time independent, i.e. in the time scale, whenever we apply the same force we will achieve the same state. This assumption is feasible for a large range of tasks.

Grouping data

Two of the most dominate clustering methods are k-means clustering and hierarchical clustering. K-means is a centroid based grouping method. Given the number of clusters k, it find a way to group the data such that the sum of distance from each point to its belonging group center is minimized. Hierarchical clustering is a connectivity based method. There are two types of hierarchical clustering method: agglomerative and divisive. Here we focus on the agglomerative method as it is more widely used. The hierarchical clustering method groups similar data iteratively. At the beginning each data point is a single cluster. In each iteration, two most similar clusters are merged to one. This step repeats until a stop criteria is satisfied or all data is merged to one cluster. Usually the merges are done in a greedy manner and hence no optimization is needed. Figure 4-4 illustrates the principle of hierarchical clustering. This clustering algorithm does not require to know the number of clusters beforehand.

In our case, the number of clusters is an unknown variable. Therefore we use the hierarchical (agglomerative) clustering method Willett (1988) to group our data. The similarity (distance) between each pair of time series is computed by DTW. This produces a distance matrix. Each element in the matrix is the distance between two time series a and b, where a and b are the row and column index of the element. At the beginning of clustering, each time

series is a single cluster. The distance between each cluster is read from the distance matrix. After one iteration, clusters are merged and most new clusters contain more than one times series. The distance between the new clusters is computed by the average distance between a member of one cluster to a member of the other cluster (average linkage).

Our hierarchical clustering method has one more constraint: the threshold of distance. Two clusters can be merged to one only when their distance is smaller than the threshold. This threshold is set by the variance of the data from the same setup. As mentioned above (Section 4.2.1), a task is demonstrated a few times under the same setup. These demonstrations are presumed to be handled with the same strategy and hence belong to the same cluster. The variance of these demonstrations gives a reference of the variance of a cluster. The largest variance, across the variance of all setups, is used as the threshold for the clustering. 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.

When the clusters can not be merged further, i.e. all clusters are further away from each other than the threshold, we define the number of modules for this task to be the number of the remaining clusters. Each cluster is then modeled as a module. Each module encodes a strategy of handling a specific task context.

4.2.2.3 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.

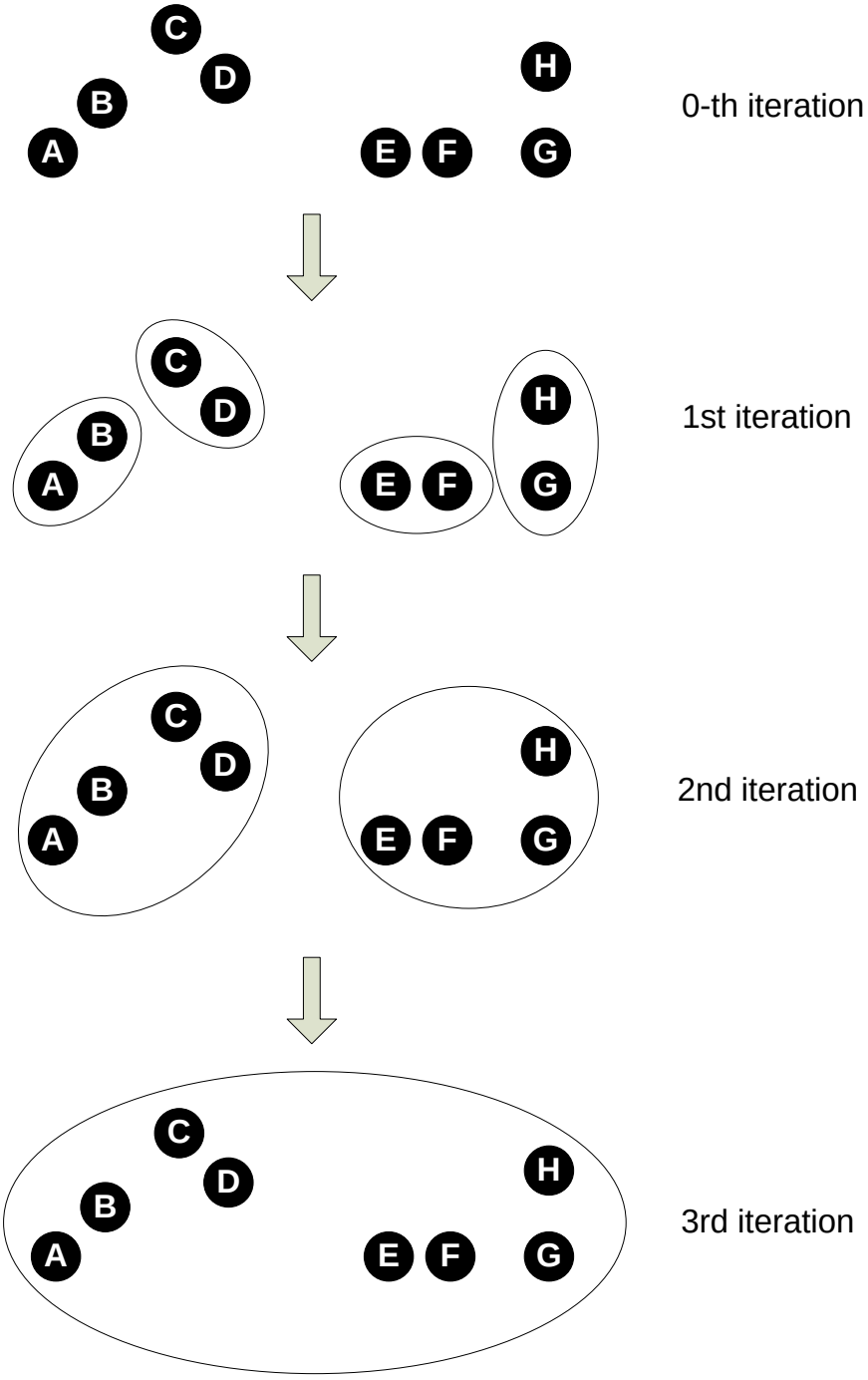


Figure 4-4: A sketch of the hierarchical agglomerative clustering method. The nearest two clusters are grouped into one at each iteration until a single cluster is formed.

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
```

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 displacement and a is the one dimensional exert torque and grip force. 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 4.2.3.2). 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,

$$p(s,a \mid \Omega) = \sum_{n=1}^{N} p_n p(s,a \mid \mu_n, \Sigma_n) \tag{4.1}$$

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

$$\mu_n = \begin{pmatrix} \mu_{s,n} \\ \mu_{a,n} \end{pmatrix} \qquad \Sigma_n = \begin{pmatrix} \Sigma_{ss,n} & \Sigma_{sa,n} \\ \Sigma_{as,n} & \Sigma_{aa,n} \end{pmatrix} \tag{4.2}$$

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 4.2.3.2). Figure 4-5 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. $p(s_t, s_{t-1}, a_{t-1} \mid \Omega_I)$. 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. $E(s_t|s_{t-1}, a_{t-1}, \Omega_I)$. 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 4.2.2.2, 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.

4.2.3 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

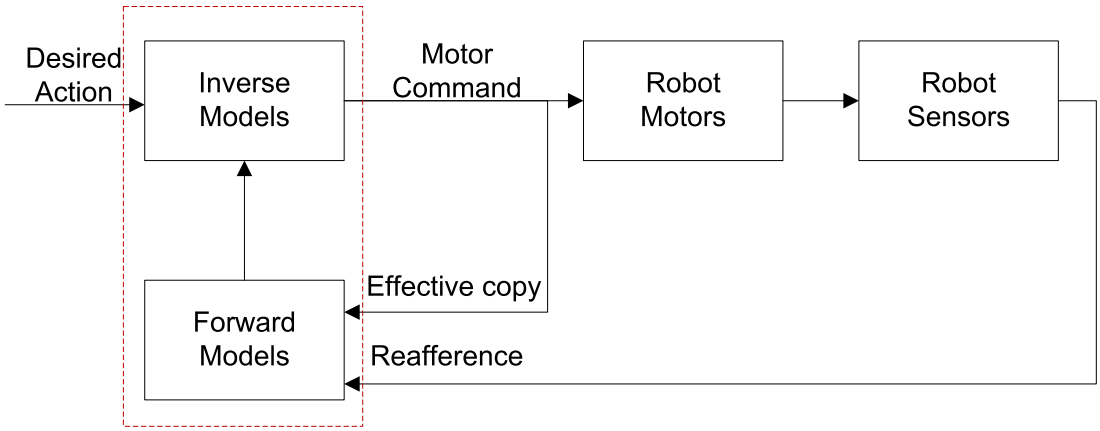
$$s_{t+1} = f(s_t, a_t) \tag{4.3}$$

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}:

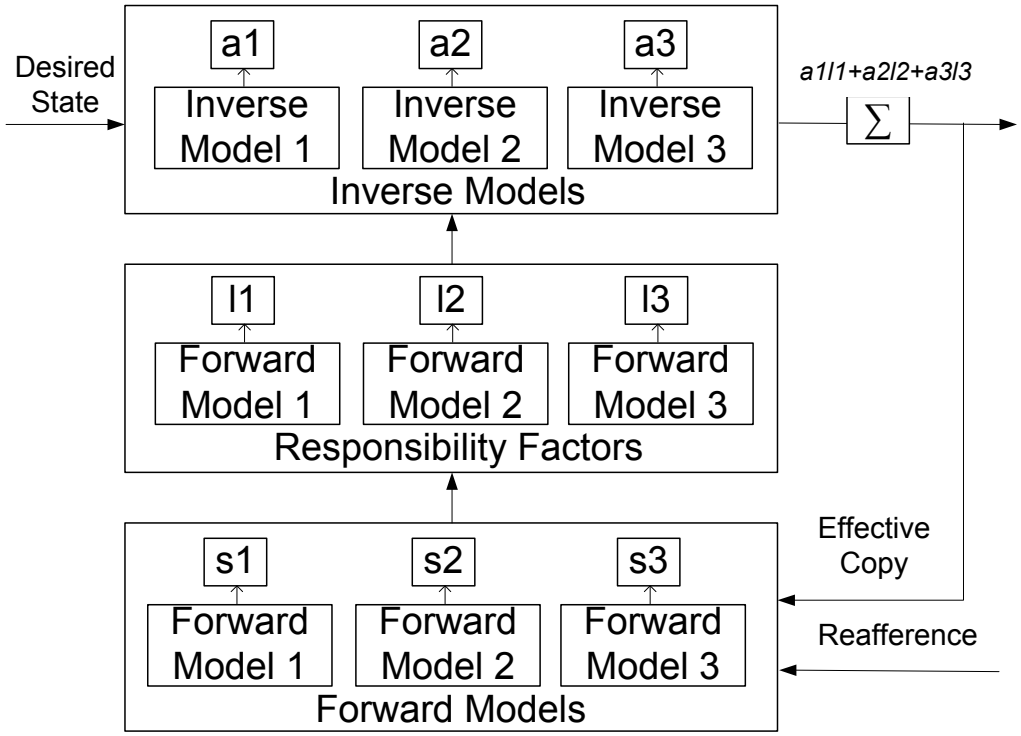
$$a_t = g\left(s^*_{t+1}, s_t\right) \tag{4.4}$$

According to the discussion in Section 4.2.2.3, equation 4.3 represents the forward model and equation 4.4 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



(a)



(b)

Figure 4-5: 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).

4.2.3.1 Anticipate sensory output by forward model

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

$$\hat{s}_t^k = E\left(s_{t-1}, a_{t-1} \mid \Omega_I^k\right) \tag{4.5}$$

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:

$$\mu_{q,n}^k = \begin{pmatrix} \mu_{s_{t-1},n}^k \\ \mu_{a_{t-1},n}^k \end{pmatrix} \tag{4.6}$$

$$\Sigma_{qq,n}^k = \begin{pmatrix} \Sigma_{s_{t-1}s_{t-1},n}^k & \Sigma_{s_{t-1}a_{t-1},n}^k \\ \Sigma_{a_{t-1}s_{t-1},n}^k & \Sigma_{a_{t-1}a_{t-1},n}^k \end{pmatrix} \tag{4.7}$$

and GMR then uses:

$$\hat{\mu}_{s_t,n}^k = \mu_{s_t,n}^k + \Sigma_{s_tq,n}^k (\Sigma_{qq,n}^k)^{-1}(q - \mu_{q,n}^k) \tag{4.8}$$

$$\hat{\Sigma}_{s_ts_t,n}^k = \Sigma_{s_ts_t,n}^k - \Sigma_{s_tq,n}^k (\Sigma_{qq,n}^k)^{-1}\Sigma_{qs_t,n}^k \tag{4.9}$$

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:

$$\hat{\mu}_{s_t} = \sum_{n=1}^{N} \beta_n(q)\hat{\mu}_{s_t,n}^k \tag{4.10}$$

$$\hat{\Sigma}_{s_ts_t,n} = \sum_{n=1}^{N} \beta_n(q)^2 \hat{\Sigma}_{s_ts_t,n}^k \tag{4.11}$$

where

$$\beta_n(q) = \frac{p_n p(q|\mu_{q,n}^k, \Sigma_{qq,n}^k)}{\sum_{n=1}^{N} p_n p(q|\mu_{q,n}^k, \Sigma_{qq,n}^k)} \tag{4.12}$$

4.2.3.2 Weight modules by 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

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

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 context.

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:

$$\lambda_t^{\ j} = \frac{p(s_t, s_{t-1}, a_{t-1} \mid \Omega_I^j)}{\sum_{k=1}^{K} p(s_t, s_{t-1}, a_{t-1} \mid \Omega_I^k)} \tag{4.13}$$

where K is the number of modules.

4.2.3.3 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 4.2.3.1. 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

$$s_t = \sum_{k=1}^{K} \lambda^k a_t^k = \sum_{k=1}^{K} \lambda^k E\left(s_{t+1}^*, s_t, a_{t-1}^k \mid \Omega_I^k\right) \tag{4.14}$$

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.

4.3 Experiment on a opening bottle cap task

The proposed multiple module 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 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.

4.3.1 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 4-6 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.

4.3.1.1 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 ($c1-b4$) (Fig. 4-7). 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

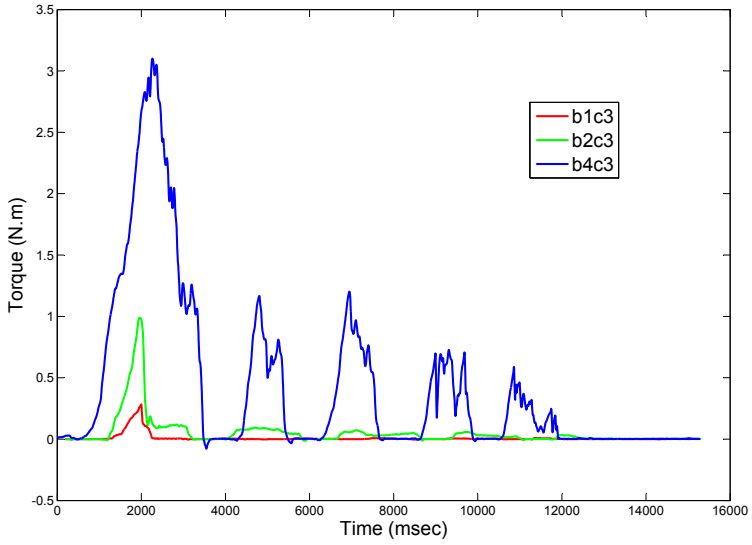


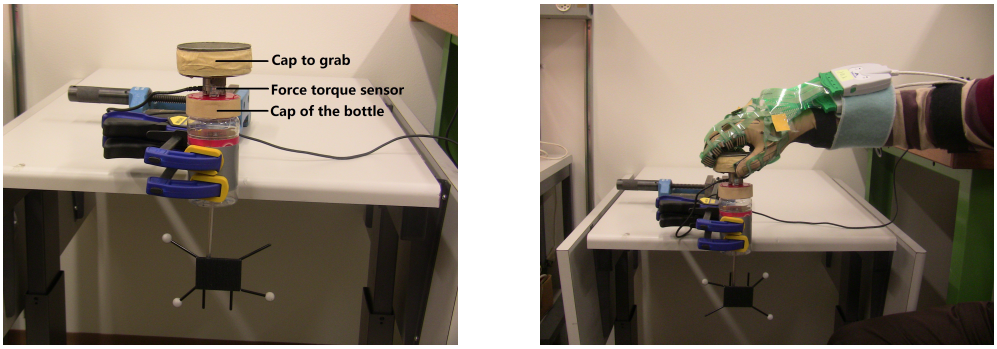
Figure 4-6: Exert torque for opening three different bottles.

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. Precise value of friction coefficient between plastics varies by type of the plastic. According to an internet resource [2], 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. 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 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. 4-8). 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 4.1).

[2] http://www.tribology-abc.com/abc/cof.htm

Figure 4-7: Bottles and caps for human demonstration. From left to right: b1 c1, b2 c2, b3 c3, b4 c4



(a) (b)

Figure 4-8: 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.

	 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 4.1: 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.

4.3.1.2 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 4.2.2.1, 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 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 (Fig. 4-8). 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 4.3.2).

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 (Fig. 4-8). 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 4.3.2).

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

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). 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 $N{\cdot}m$.

4.3.2 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.

4.3.2.1 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.

4.3.2.2 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.

4.3.2.3 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. 4-9 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 Fig. 4-9, 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 decided by the static FCO. Once the cap starts to move, the FCO between bottle and cap transits to kinetic FCO, which 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.

4.3.3 Learning Modules

In this section, we explain how do we encode the training data into a few different modules. As mentioned in Section 4.2.2, the first step is to cluster the data and find out the number of modules required in this task (Section 4.3.3.1). After that, a forward model and an inverse model is built for each module (Section 4.3.3.2) and we use these modules to generate motor commands.

[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.

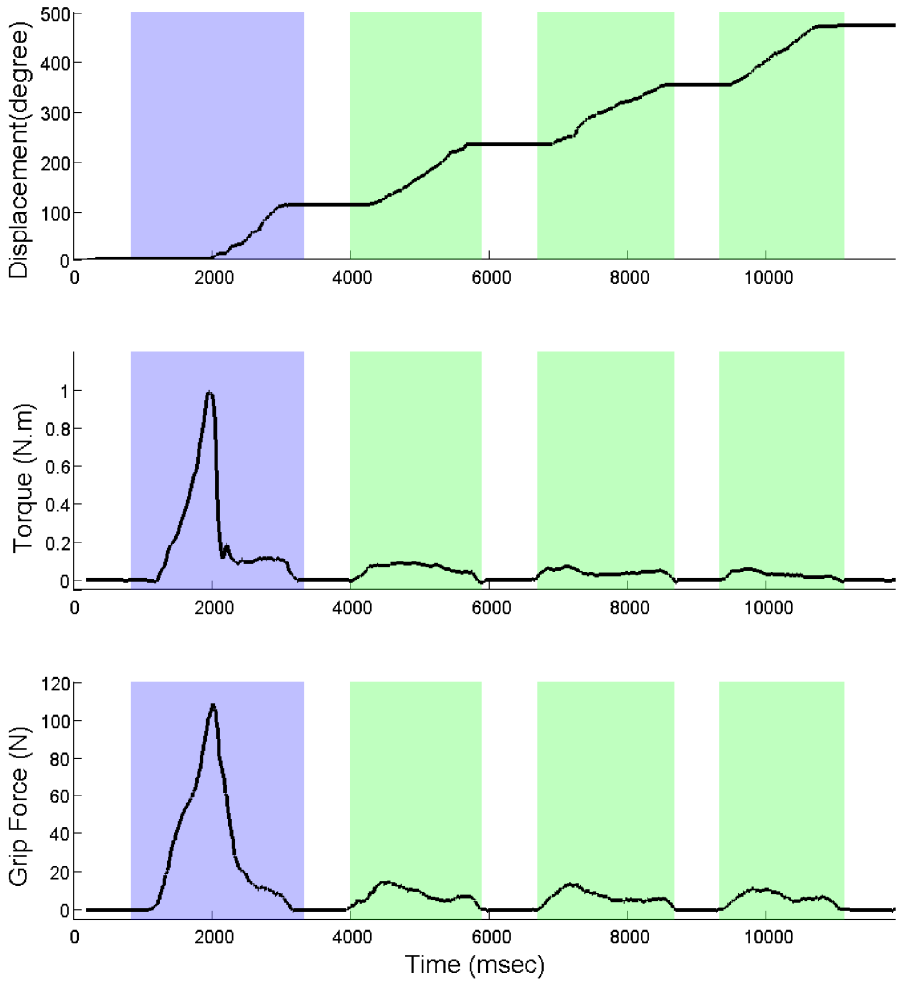


Figure 4-9: Aligned data of all three channels. Highlighted parts mark the turning process: blue blocks denote the first cycle, i.e. the phase I, and green blocks denote the later cycles, i.e the phase II. Phase I is significantly different from the phase II.

4.3.3.1 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. 4-10). 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 4-11 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 4.2.2.2) 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;
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 4.2. 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.

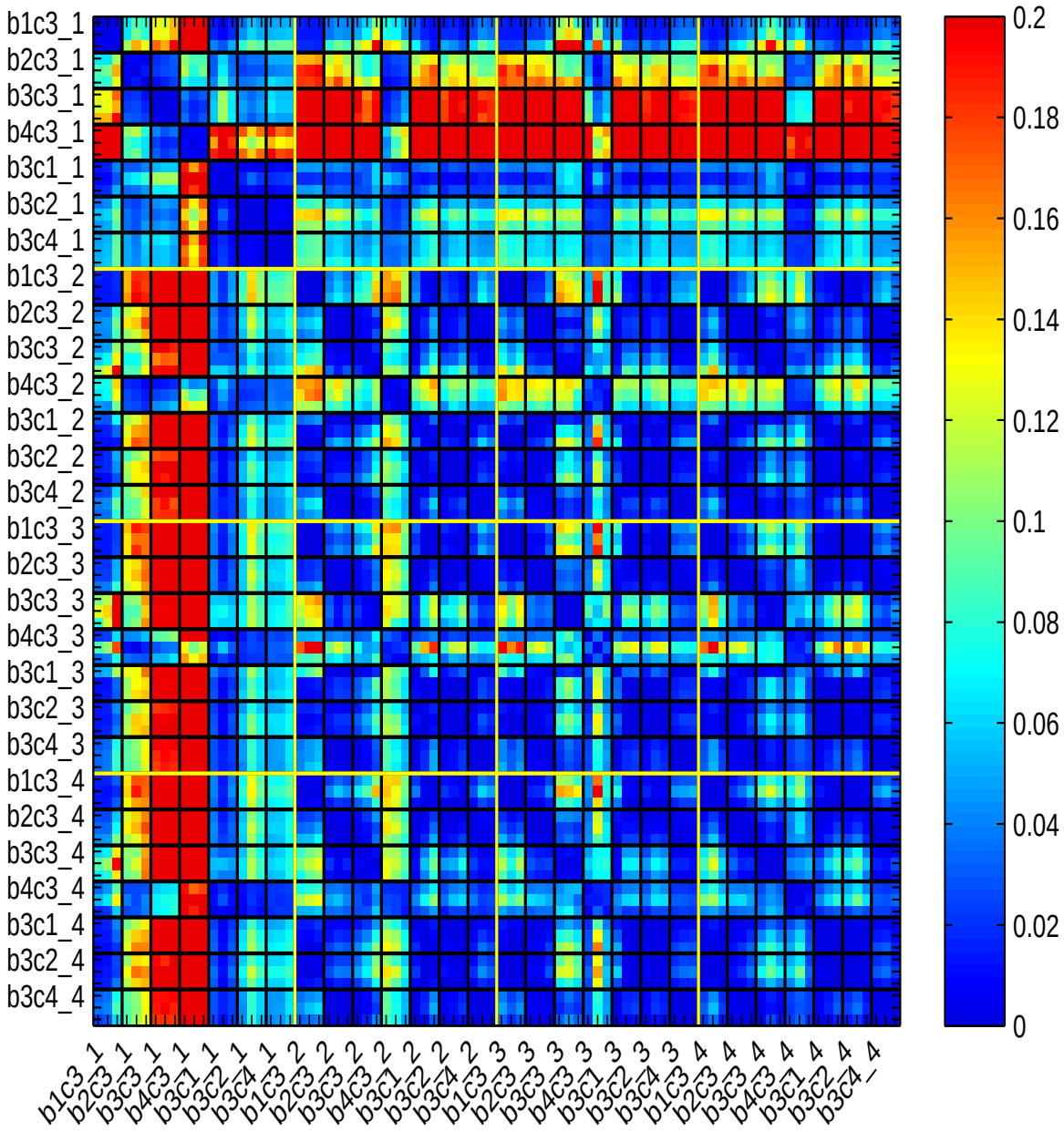


Figure 4-10: A heatmap representation of the distance matrix of 84 time series (7 setups $\times$ 4 cycles $\times$ 3 trials). The labels are in the format of "setup_cycle". For example, "b1c2_1" represents the first cycle of the b1c2 setup. The yellow lines divide the x and y axis by the 4 cycles and 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.

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

Table 4.2: Clustering result

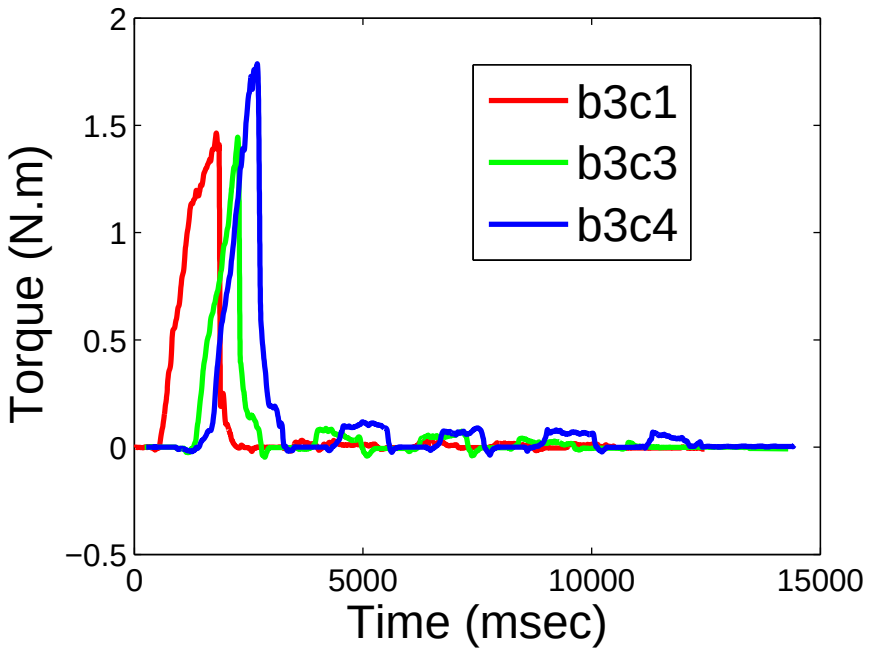


Figure 4-11: Exert torque for opening bottle b3 with three different cap sizes.

4.3.3.2 Learning Modules

We encode the data in each of the module by GMM. As explained in Section 4.2.2, 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 4-12.

4.3.4 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*[7] tactile sensor, which is calibrated to provide contact force information, for the grip force feedback. The cap displacement is

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

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

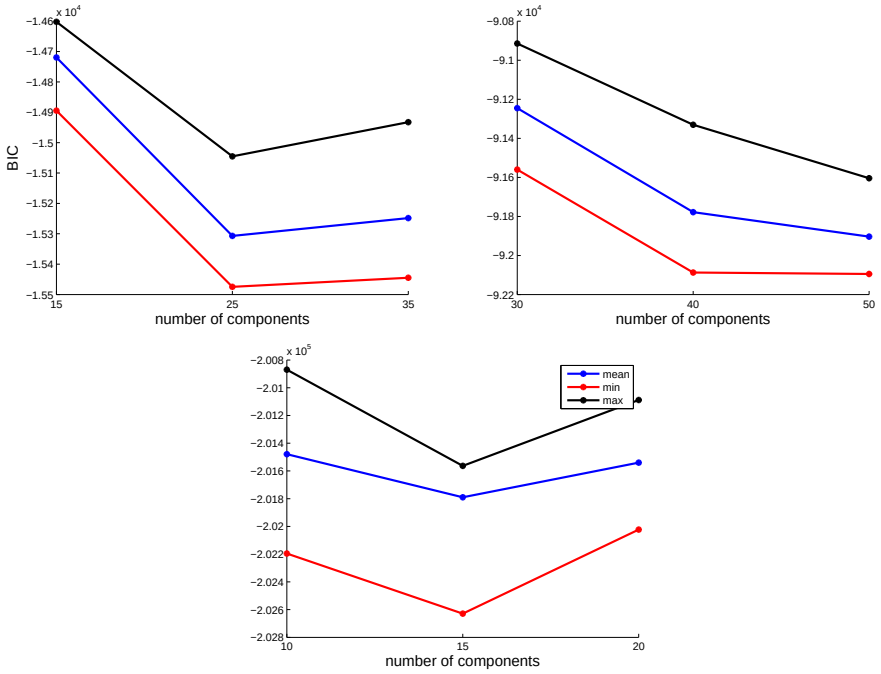


Figure 4-12: BIC test result for clusters. (a) Cluster 1, (b) Cluster 2, (c) Cluster 3.

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.

4.3.5 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 in figures 4-13- 4-16. Figure 4-17 is a similar plot to figure 4-6, 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 (Fig. 4-13).

Bottle b4 is the most difficult bottle to open in the training set and it's phase *I* requires more than 3 Nm (Fig. 4-6). 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 4-14, 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: Fig. 4-6, b3: Fig. 4-11, b5: Fig. 4-15). 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). 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. 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.

4.4 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 weighted 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

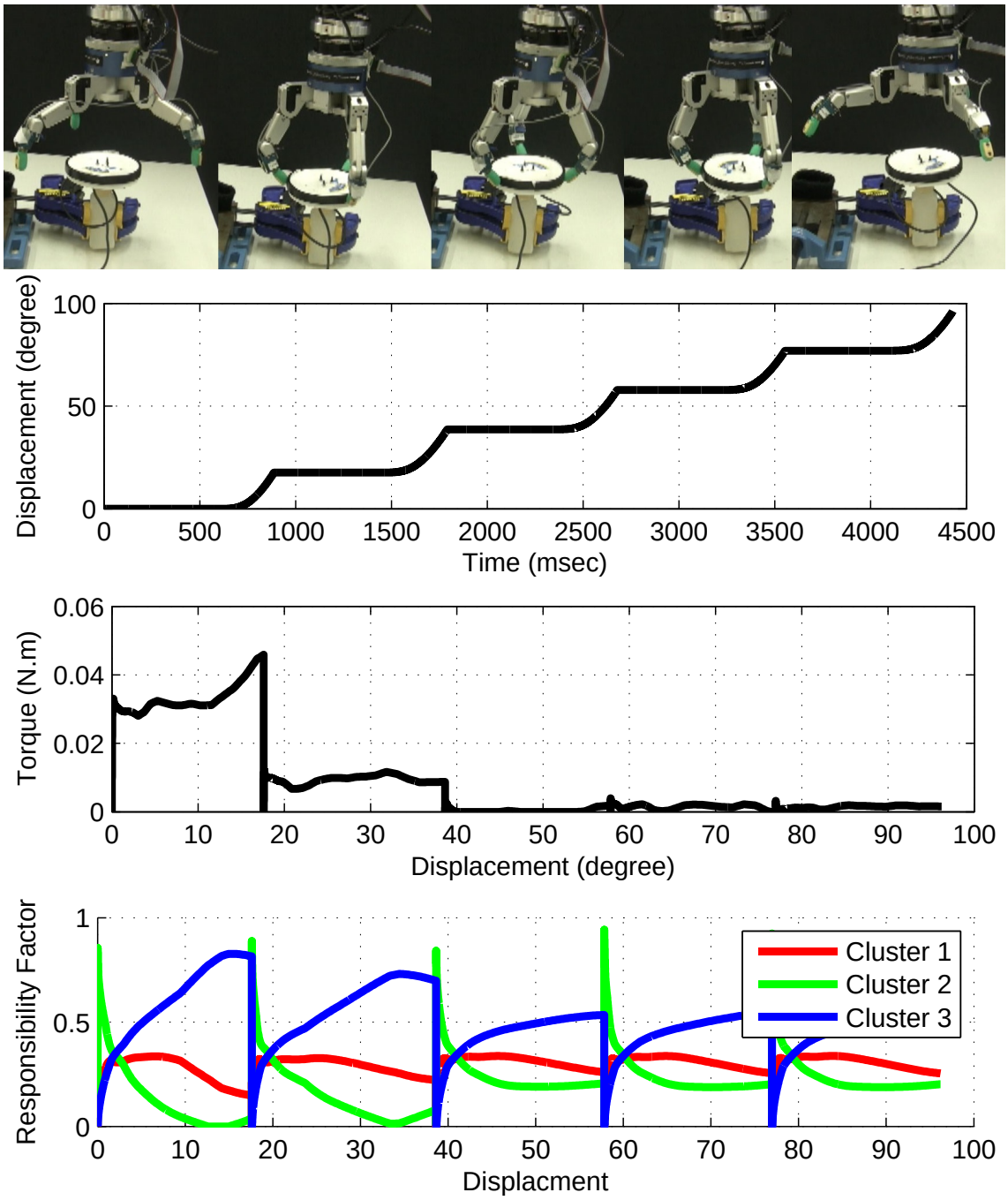


Figure 4-13: Robot demonstration on opening b1.

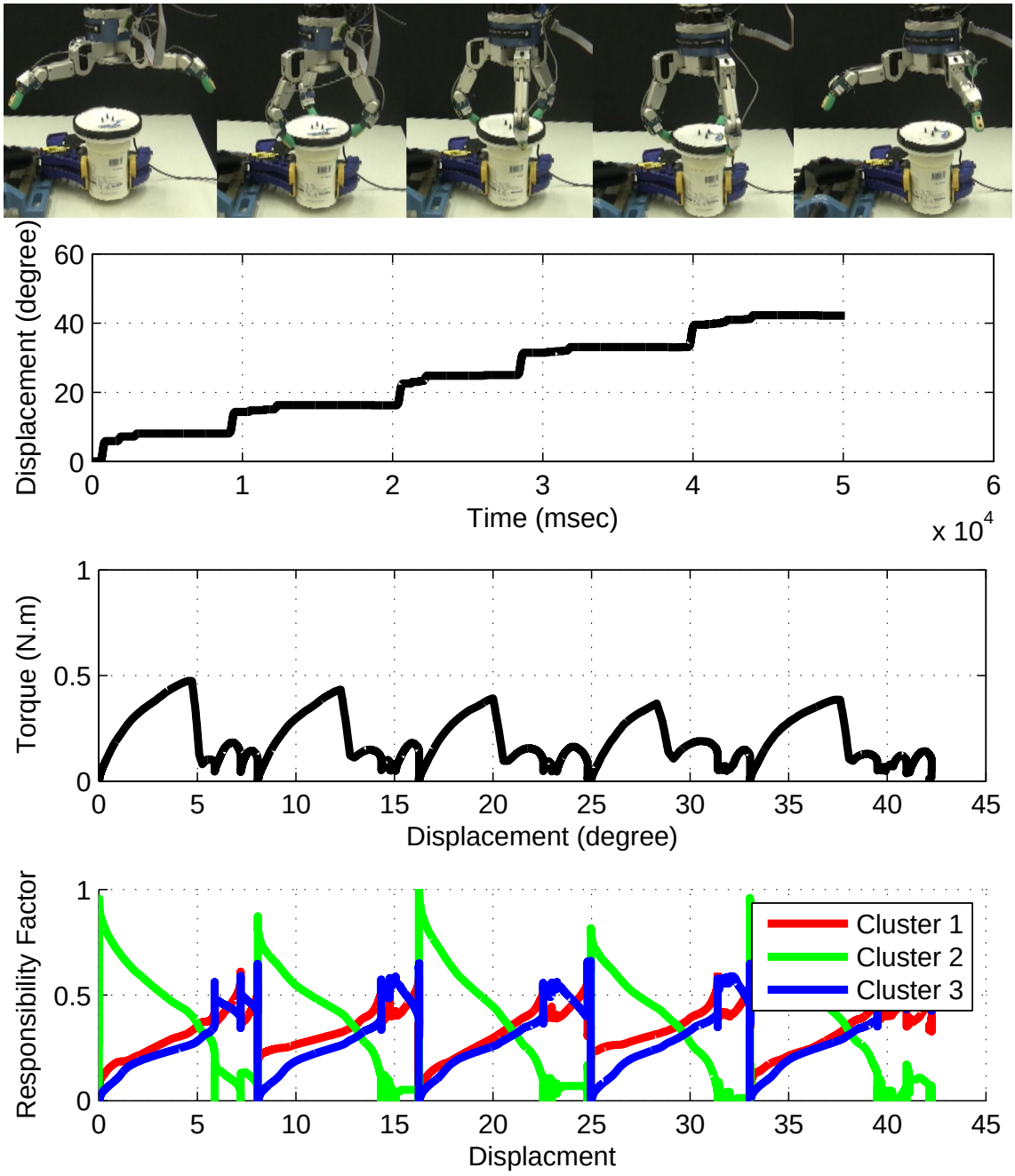


Figure 4-14: Robot demonstration on opening b4.

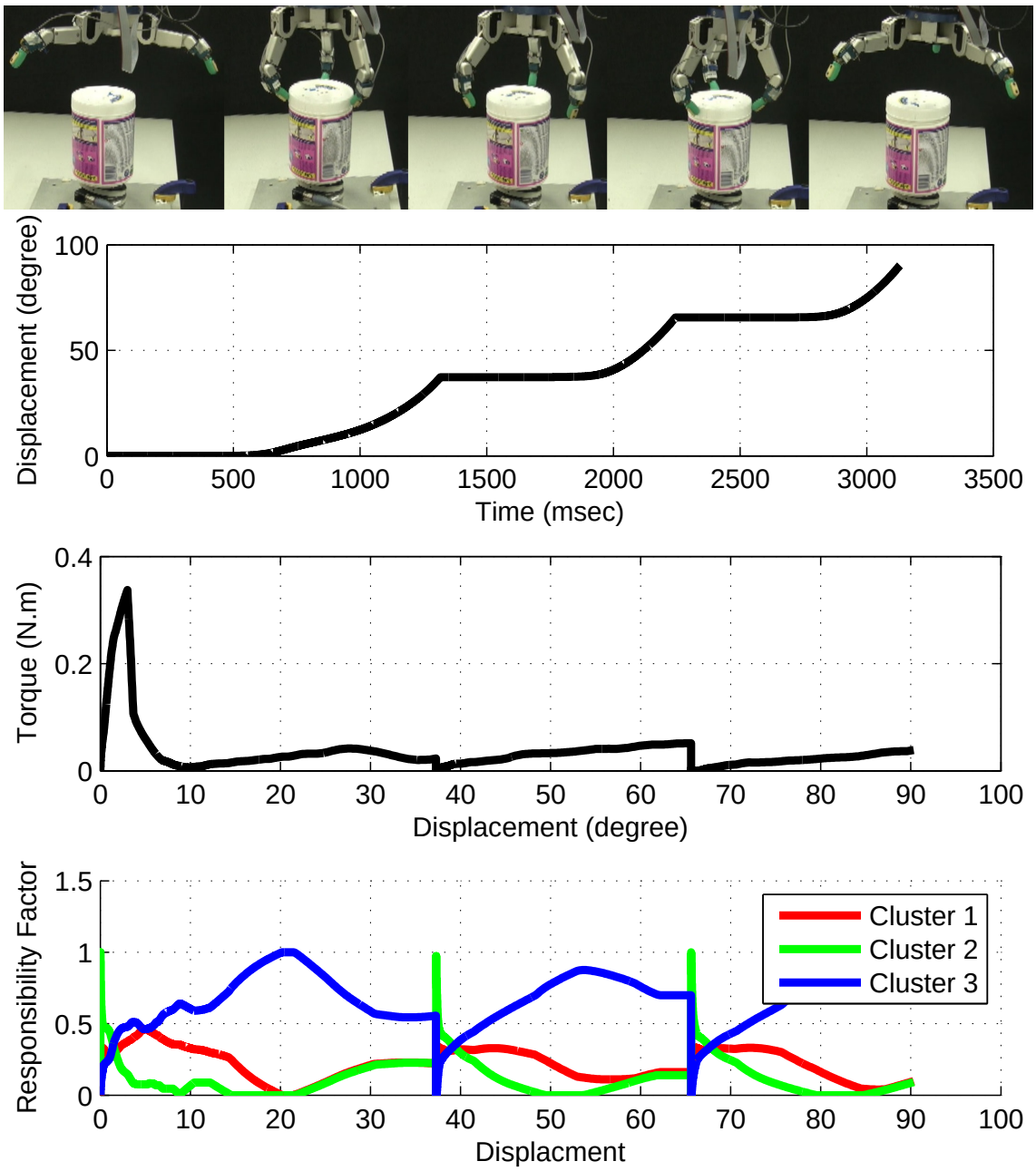


Figure 4-15: Robot demonstration on opening a new bottle (b5).

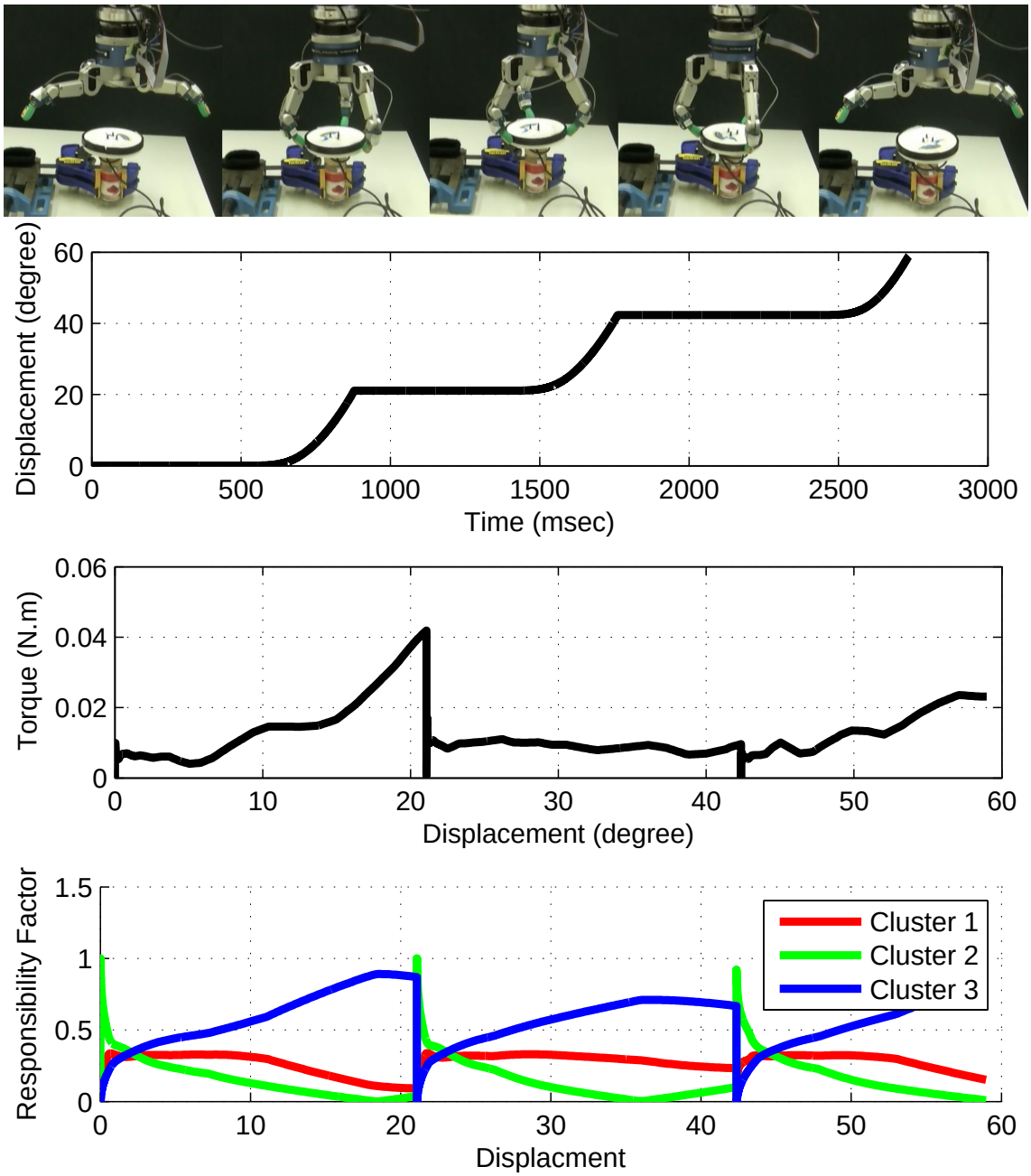


Figure 4-16: Robot demonstration on opening a new bottle (b6).

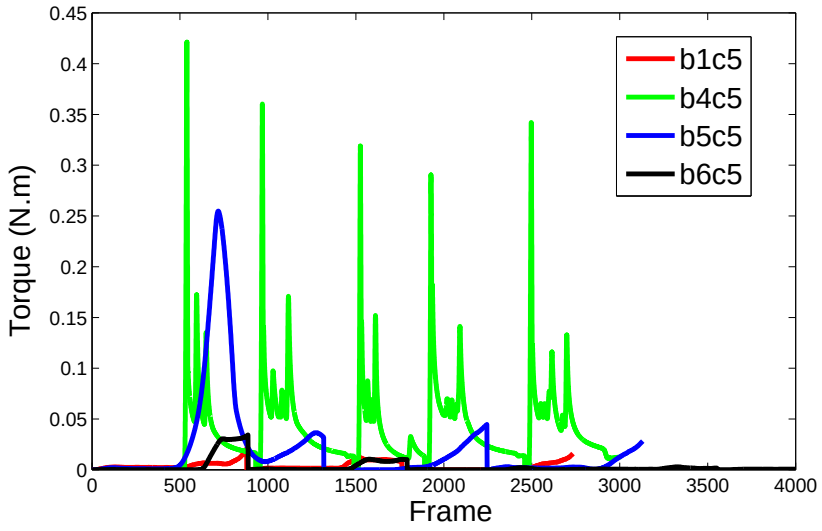


Figure 4-17: Robot exert torque for opening four bottles: b1 b4 b5 b6. Time is warped and shifted for displace purpose.

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 muscle 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 can be combined with the grasp planning study discussed in the Chapter 3 to develop task specific grasping strategy (El-Khoury et al., 2013; Dang and Allen, 2014a). This analysis will be progressed in the future work.

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.

Chapter 5

Learning Motion Primitive for manipulation tasks

5.1 Introduction

This chapter focus on learning motion primitives for manipulation tasks. In previous chapters, we focus on learning the strategy of dexterous manipulation including multi-finger grasp planing and adaptive force control strategy. These are done on the "end effector level" and rely on the robot limb to deliver the end effector to a proper position. In this chapter, we look into the whole body movement and study how to programm robot to achieve a proper posture for a desire task.

Motion primitive

To accomplish a more complex task, a sequence of motion is needed. As discussed in the introduction chapter, the high dimensional searching space makes this sequence of motion difficult to generate. To reduce the searching space, the concept of motion primitives in neuroscience (Bizzi et al., 2008) has been introduced to planning. The basic principle is to discretize a manipulation task into a set of motion primitives, that each serves for an elementary manipulation function. After modeling each primitive, the whole task then can be achieved by coordinating them properly.

Modeling motion primitives remains an open problem. Many literatures discuss how to design motion primitives that accomplish specific tasks (Michelman and Allen, 1994; Felip et al., 2012; Ijspeert et al., 2013). In those works motion primitives are modeled as a set of differential equations or control rules. New motions are generated by tuning the parameters in the models. Deriving these equations and control policies is not an easy work, as well as fine tuning the parameters to generate new motions. These are in need of deep understanding of the

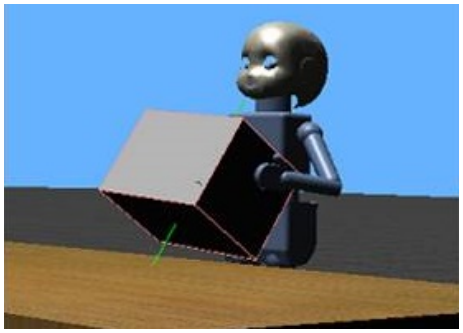


Figure 5-1: iCub grasping a box by both arms

task and the dynamic model.

These difficulties can be alleviated by using the learning by demonstration approach and modeling the motion in the state space.. In this chapter we propose an easy to use system for learning manipulation motion primitives from human demonstration. To achieve this goal, we exploit the application of the mimesis model (Inamura et al., 2004) in learning motion primitives for object manipulation.

Mirror neurons and Mimesis Model

Mimesis model is a mathematical realization of the function of the mirror neurons. Mirror neurons is a kind of neurons found in primates and birds, which fires both when the animal observe and execute a motion. In human brain, mirror neurons has show track in the area of the premotor cortex, the supplementary motor area, the primary somatosensory cortex and the inferior parietal cortex. These areas contribute to human control of motion and sensory reception. It is generally believed that mirror neurons is associated with animal's ability of learning by imitation and understanding action of others (Rizzolatti and Craighero, 2004). Motivated by this idea, many researchers try to understand the function of mirror neurons and hence implement it on robots to equip them with human level imitation and learning ability. We are also inspired by this idea and hence try to mimic the mechanism of mirror neuron to learn manipulation motion primitives.

Inamura and etc. develop the mimesis model that realize the functions of the mirror neurons: observe motion, recognize motion and generate motion. This mimesis model has been shown to be effective in motion recognition, generation and robot coaching (Inamura and Shibata, 2008; Okuno and Inamura, 2011). It is built based on the Hidden Markov Model (HMM). In the mimesis model, all demonstrated motion patterns are first encoded by a HMM. These HMMs are then projected to a topological space called "proto-symbol space". In this space, each HMM is projected as a point called "proto-symbol", and is labeled by the character of its representing motion, such as "grasp low box" or "grasp high box". The similarity between two motions (HMMs) are represented as the Euclidean distance between their proto-symbol.

Recognition of a unknown motion is done by projecting the unknown motion to the proto-symbol space. This give us a new proto-symbol. If the new proto-symbol is very close to a known proto-symbol, then it is very likely the unknown motion is the motion represented by the closest proto-symbol. At the other hand, new motion generation is done by exploring new proto-symbols, i.e. interpolate between the known proto-symbols. The generated new motions will be similar to but different from the motions encoded by the surrounding proto-symbols.

As we label each proto-symbol, the mimesis model provides a base of understanding of human and robot behavior. This even allow the robot user to adjust robot motion by natural language. For example, starting from the "gasp low box" motion, we can instruct the robot to raise its arms higher to grasp a box on the top of a cabinet by the command "not high enough, go higher to grasp". This command will generate a motion closer to the motion labeled by "grasp high box".

Most of previous work of the mimesis model focus on learning whole body movement. Our work extend the mimesis model to learn motions of manipulation that involve interaction with objects. The work of Kunori et al. (Kunori et al., 2009) using hidden Markov models to encode motion primitives for object manipulation has a similar concept to our work. While they focused on extracting key features and reshaping movements for good performance, we focus on combining known manipulation motion primitives to generate new motions that can achieve the desired effects. Although interpolation of known motions is not new in motion synthesis (Hoshino, 2004; Glardon et al., 2004), most of the existing work focus on free body motion. The application on object manipulation is rarely discussed.

The goal of this work is to develop an easy to use system for the robot to learn manipulation motion primitives and generate new motions to adapt to unseen scenarios. The system is implemented for a bimanual grasping task. Different from the static fingertip grasping synthesis 3, in this task we focus on the grasp reaching motion.

5.2 Learning by mimesis model

We adopt the mimesis model to learn motion primitives of manipulation from human demonstrations. In this approach, the demonstrated motions are firstly encoded by the Hidden Markov Model (HMM) (Rabiner, 1989). A topological space, i.e. proto symbol space, is then constructed to represent the similarities between the HMMs. In this space, each HMM is abstracted to a labeled point: proto symbol. New motions are generated by new proto symbols. The correlation between the location of the new proto symbols and their physical effects is learnt by regression. This correlation allows us to directly query a new motion by a high level task requirement.

In short, the general approach has 5 steps as listed underneath:

1. **Human demonstration of motion primitives**: A human teacher demonstrates manipulation motion primitives (Section 5.2.1).
2. **Motion symbolization**: Abstract the motion primitives by HMM and create the proto-symbol space (Section 5.2.2).
3. **Motion interpolation**: Interpolate the proto-symbol space and construct new HMM. (Section 5.2.3).
4. **New motion generation**: Generate motion using proto-symbols (Section 5.2.4).
5. **Learning motion effects**: Robot reproduces the motion and learn the correlation between the location of the proto-symbols and the effect of the generated motions (Section 5.2.5).

5.2.1 Human demonstration of motion primitives

The motion primitives of manipulation are first demonstrated by human. The same primitives were demonstrated a few time so that the HMM is able to encode the general features of the movement. Each primitive has its own purpose and distinct pattern. To enable the robot to work in different task context, different primitives needs to be demonstrated. For example, to design a motion primitive for fetching boxes in different sizes (size is the task context), we need to demonstrate at least two primitives: grasping a small box and grasping a big box (Figure. 5-2). Grasping a box with the size between the big one and the small one is achieved by interpolation between these primitives. For more complex motion, more than two primitives may be required to achieve the desired motions.

In this approach, the demonstrated motions do not only provide the dynamics of the motion primitives, but also define the feasibilities of the motion primitives. In the example given

(a) (b)

Figure 5-2: Human bi-manual grasps. (a) Human grasping a small box. (b) Human grasping a big box.

above of grasping different sizes of boxes, one should demonstrate the motion primitive for grasping the smallest feasible box and the motion primitive for grasping the biggest feasible box. Here the "feasibility" is defined according to the limitation of joints. As the new motions are interpolations of the demonstrations, joint limits or singularity can be avoided in the new motions by well chosen demonstrations.

5.2.2 Motion symbolization

In order to store and label the observed motions, we construct the "proto-symbol space". This is done by two steps: first encode the motion pattern by HMM, and second project them to be a set of "proto-symbols" in the proto-symbol space, where the similarity between different motion patterns are represented as Euclidean distance.

Hidden Markov Model

Hidden Markov Model is a stochastic mathematical model for sequential data. It describes a data sequence as a Markov process, with which the data is described as transitions between a set of states. In Markov process, the future state of the process depends solely on its current state. It can be thought as a 'memoryless' process: it only need to 'remember' the current state but not the whole process's history in order to predict the future. This is a strong condition. Temporal pattern with this characteristic, e.g. speech, handwriting and sequence of body movements, can be approximated by a Markov model. In a simple Markov process, transition between states is directly visible. Different from this, the states of hidden Markov Model is not directly visible,

i.e. hidden.

HMM has two layers: the hidden states and the outputs. The outputs are directly visible but their pattern is controlled by the hidden states. How do the hidden states transit to each other and how do they emit to the outputs determine the output pattern. Though they are not directly visible, the hidden states can be inferred by the observable outputs. A well known application of HMM is speech recognition. Speech recognition systems use HMM to analyze the signal of speech in order to discover the meaning. Here the sound of the speech is the observable output and the meaning of the speech is the hidden state.

A HMM can be fully described by a triple $(\pi, \boldsymbol{A}, \boldsymbol{B})$:

1. π: the vector of the initial probability of each state.

2. $\boldsymbol{A} = a_{ij}$: the state transition matrix. This is the probability that one state (q_i) transits to another (q_j): $Pr(q_i|q_j)$.

3. $\boldsymbol{B} = b_{ij}$: the output probability (confusion matrix). This is the probability that one state (q_j) produce an output (o_i): $Pr(o_i|q_j)$. If the outputs are discrete, i.e. have a countable number of states, this can be simply represented by a matrix. If the outputs are continuous, the output probability is usually described by a GMM.

Figure 5-3 illustrates the mechanism of HMM encoding a motion pattern. This is a left-to-right continuous Hidden Markov Model (CHMM). In a general HMM, the hidden states can transit to any other states. In a left-to-right HMM the transition has more constraints. The states are placed in a "left to right" order, each state can only transits to the state at its right or transits to back to itself, the possibility of it transiting to any other states is zero. A left-to-right HMM restricts the complexity of the data patter it can model and is adequate for modeling motion primitives.

To encode the motion primitives by HMM, the pattern is described by a set of variables: $\lambda = \{\boldsymbol{Q}, \boldsymbol{A}, \boldsymbol{B}, \pi\}$, where $\boldsymbol{Q} = \{q_1, ..., q_N\}$ is a finite set of states, $\boldsymbol{A} = \{a_{ij}\}$ is the state transition probability matrix denoting the probability that node q_i transits to q_j, $\boldsymbol{B} = \{b_i\}$ is the continuous output probabilities denoting the probability distribution that the output vector $o[t]$ is given by q_i, and π is the the initial distribution. The π is the same as for each CHMM as we use left-to-right CHMM model. Therefore, the parameter set $\mathscr{P} = \{a_{ij}, b_i\}$ characterize the behavior of the motion primitive. We call $\mathscr{P}$ a proto-symbol.

The CHMM is learned using the Baum-Welch algorithm (Rabiner, 1989)[1]. For simplicity, we use a single Gaussian model for the output of each node in the CHMM. This allows us to synthesis new motion simply by interpolating the means and covariances of the Gaussians (Section 5.2.3).

[1] We use the Hidden Markov Model Toolkit (HTK) to learn the HMM

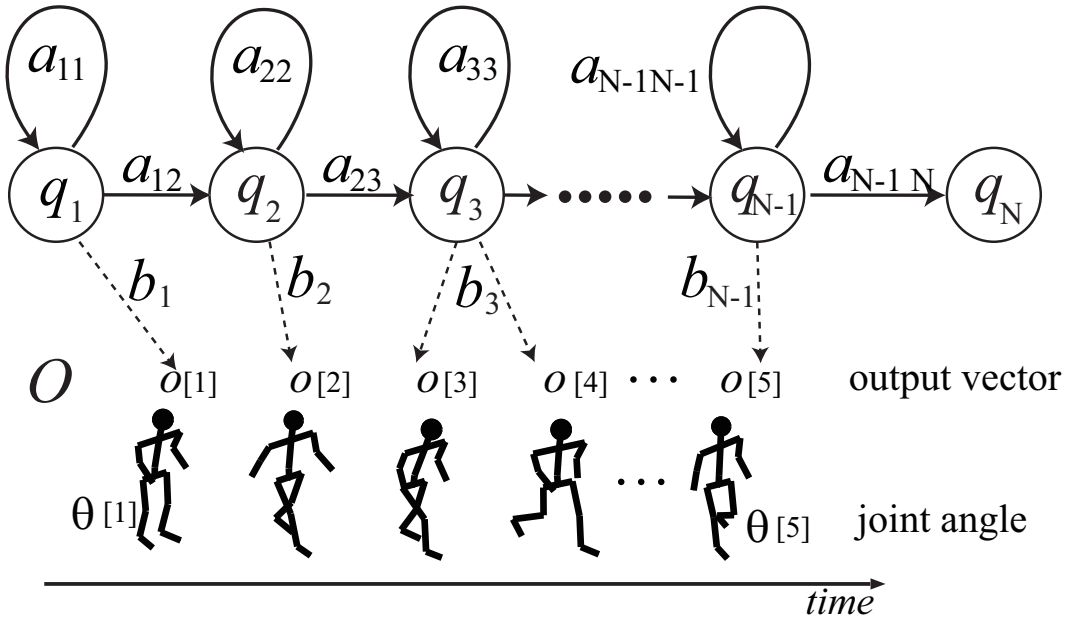


Figure 5-3: A illustration of encoding a motion by Continuous Hidden Markov Model[2].

The proto-symbol space is constructed to represent the similarity between CHMMs. This requires to compute the similarity between each pair of CHMMs. In this work, we use the Bhattacharyya distance (Kailath, 1967) as our similarity metric, as it is a symmetric metric with respect to two probability variables. The Bhattacharyya distance $BD(p,q)$ between two Gaussian distributions $p(x;\mu_p,\Sigma_p)$ and $q(x;\mu_q,\Sigma_q)$ is defined as follows:

$$\begin{aligned} BD(p,q) &= -log\int_{-\infty}^{\infty}\sqrt{p(x)q(x)}dx \\ &= \frac{1}{8}\mu_{pq}\left(\frac{\Sigma_p+\Sigma_q}{2}\right)^{-1}\mu_{pq}^T+\frac{1}{2}log\frac{\mid\frac{\Sigma_p+\Sigma_q}{2}\mid}{\mid\Sigma_p\mid^{\frac{1}{2}}\mid\Sigma_q\mid^{\frac{1}{2}}} \end{aligned} \tag{5.1}$$

where

$$\mu_{pq}=\mu_p-\mu_q \tag{5.2}$$

The Bhattacharyya distance $\mathrm{DB}(\lambda_1,\lambda_2)$ between two HMMs is computed by summing the distances between the Gaussian distributions, i.e. the output probability distributions for the nodes:

[2]Reprinted with permission.

$$DB(\lambda_1,\lambda_2) = \sum_i \sqrt{BD\left(\mathcal{N}_{1i}\left(\mu_{1i},\Sigma_{1i}\right),\mathcal{N}_{2i}\left(\mu_{2i},\Sigma_{2i}\right)\right)} \tag{5.3}$$

where $\mathcal{N}_{ji}(\mu_{ji},\Sigma_{ji})$ is the output probability at the i-th node q_i of the HMM λ_i.

The proto-symbol space is constructed by the multi-dimensional scaling technique (MDS) (Schiffman et al., 1981). This technique computes the locations of the CHMMs in the proto-symbol space by minimizing the criterion:

$$S^2 = \sum_{i,j}(DB_{ij} - d_{ij}) \tag{5.4}$$

where DB_{ij} is the Bhattacharyya distance between the ith and jth CHMMs and d_{ij} is their Euclidean distance between their proto-symbols.

5.2.3 Motion interpolation

To generate new motions, new location in the proto-symbol space is explored. This is done by interpolation between different proto-symbols. In the left-to-right model, the expected duration s_i of the state q_i can be computed as

$$s_i = \sum_{n=1}^{\infty} n(1-a_{ii})a_{ii}^{n-1} = \frac{1}{1-a_{ii}}, \tag{5.5}$$

where a_{ii} is the probability of self-transition at the state q_i.

A new proto-symbol $\hat{\mathscr{P}}$ is expressed by the linear combination of m proto-symbols $(\mathscr{P}_1,...,\mathscr{P}_m)$. The weights of different proto-symbols are expressed by the mix coefficient c_j. The expected duration $\hat{s}_i$ for the new motion in the state q_i is computed as

$$\hat{s}_i = \sum_j^m c_j s_i^{(j)} \tag{5.6}$$

with this we can compute the new state transition probability $\hat{a}_{ii}$ as

$$\hat{a}_{ii} = \frac{\hat{s}_i - 1}{\hat{s}_i} \tag{5.7}$$

Note that according to the Eq. 5.6, $s_i \geq 1$ and hence the following constraint must be satisfied

$$\sum_j^m \frac{c_j}{1-a_{ii}^j} \geq 1 \tag{5.8}$$

To compute the new output probability b_i, since there are only one Gaussian in each state,

we simply sum the means and variances of the Gaussians of the same state as

$$\hat{b}_i(O) = \mathcal{N}(O; \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\sigma}}_i^2) = \sum_j^m c_j b_i^{(j)}(O) \tag{5.9}$$

where

$$\hat{\boldsymbol{\mu}}_i = \sum_j^m c_j \boldsymbol{\mu}_i^{(j)} \tag{5.10}$$

$$\hat{\boldsymbol{\sigma}}_i^2 = \sum_j^m c_j^2 \boldsymbol{\sigma}_i^{(j)^2} \tag{5.11}$$

$\mu_i^{(j)}$ and $\sigma_i^{(j)}$ are the mean and variance of the Gaussian representing the i-th state.

In theory this method can also be used to extrapolate the proto-symbols with negative mixing coefficient, which allows us to explore outside the feasible region defined by the demonstrations. This can generate motions that out of our experience however the feasibility can not be guaranteed, i.e. this may gives joint angles over the robot's limit.

5.2.4 Motion generation

New motion sequence is generated from the new proto-symbols by using an averaging method (Inamura et al., 2004). The steps of generation are as follow :

1. : Starting from a node q_1, let the motion element sequence be $O = \phi$.
2. : Using the transition probability $\{a_{ij}\}$ to generate the states q_j.
3. : Using the output probabilities $\{b_i\}$ to decide the output label o_k.
4. : Adding the output label o_k to the motion elements sequence O.
5. : Stop when the generation process reaches the end node q_N.

Due to the stochastic nature of this method, motions generated by the same HMM are not identical at each time. Nevertheless, they have the same dynamics as they are generated from the same parameters $\boldsymbol{A}$ and $\boldsymbol{B}$. We repeat the above steps and average the generated motions to produce the final motion. As the duration of each generated motion is different, before averaging we uniform the time in each motion by:

$$\bar{\theta}(t) = \theta\left(T\frac{t}{T_u}\right) \tag{5.12}$$

where T is the time duration of each motion, and T_u is the uniformed time duration. After this joint angles are averaged amount all generated motions.

5.2.5 Learning motion effects

Different from free body motions, motion of object manipulation needs to achieve ceratin outcome, such as grasping a given size of box. However the physical effects of the demonstrated and generated motions are unknown, as the robot have different embodiment to the human demonstrator. For example, the motion for a human to grasp a $30cm$ length box may only allow the robot to grasp a $15cm$ length box. Therefore, a learning process is needed to quantify the correlation between the location of the new proto-symbol and its physical effect.

To do this, we first interpolate the proto-symbol space with a few different mixing coefficients. We then generate the corresponding motions and perform them with a robot. The platform we used is the iCub in the Webots simulator. As the iCub has the same joint configuration of arm as the one provided by Kinect, we directly apply the generated motions to the iCub. The outcome of the motion, for example the size of the box the robot can grasp with the motion, are recorded with their corresponding mixing coefficients.

The correlation between the sizes and the mixing coefficients is then found by regression analysis. Figure 5-6 shows an example of the result of the regression. With this result, we are able to infer the mixing coefficient for generating a proper motion. By using the method detailed in section 5.2.4, the motion with a desired effect can be generated. Our experiments verify that this method can generate new grasping motions and the result will be discussed in the next section in details.

5.3 Experiment of learning motion primitives

This section presents the implementation of the system in learning bi-manual grasping motion primitives. Bi-manual grasping is regularly used in daily life. One of the most commonly used strategies is putting two hands at the opposite sides of a bulky object to apply antipodal grasps (Figure 5-5). This motion primitives, including an approaching motion and a lifting motion, can be used to grasp many different objects. In our experiment, we focus on learning this strategy and verify that it can be generalized to grasp objects in unseen scenarios.

The strategy is demonstrated in two different scenarios: grasping boxes with different sizes and grasping boxes placed on different heights. As explained in section 5.2.1, the demonstrations are chosen to define the boundary situations of the grasping motions. In this experiment, four different motion primitives are demonstrated: grasping the biggest feasible box, grasping the smallest feasible box, grasping the lowest feasible box and grasping the highest lowest box. Objects with size or height outside the feasible area might be able to be grasped by the same strategy, but the motion may be very close to infeasible joint angles or is not natural for human behavior. In our case, bi-manual grasp of a box longer than the distance between the left and right elbow is very difficult for the iCub; bi-manual grasp of a very small size box is possible but human would normally use single hand grasp.

All the demonstrated motion sequences are recorded by Kinect, a skeleton tracking device widely used both in gaming industry and academic research (Ren and O'Neill, 2012). It is a marker-less stereo camera which can automatically detect and track human joint configuration. The output data from Kinect is converted to joint angle space.

In this experiment, the grasping motions only involve the arms. The objects are placed in the working space of the human demonstrator so that the human do not need to change the location to grasp the objects. Due to the technical limitation of Kinect, it can not record the wrist joint and hence wrist is omitted in our current experiment. In total, 8 degrees of freedom is recorded in the human motion: left shoulder (3D), right shoulder (3D), left elbow (1D) and right elbow (1D). When the hands get contact with the objects, the wrist joints will change due to the force applied by the arm. This adds extract uncertainties to the grasping motion, as well as a certain amount of compliance. As a result the box may rotate a certain angle after lifting (Figure 5-10).

Each grasping motion is demonstrated five times. In all demonstrations, the starting postures are the starting posture used by Kinect: the Ψ pose that with two arms raising over the head, both palms facing inside.

The raw data are noisy due to the limitation of the motion capture device. To denoise the motion signal, we used second order low pass filters to smooth the motion outputs and remove high frequency noise caused by vibration of the machinery. Each motion is low-pass filtered

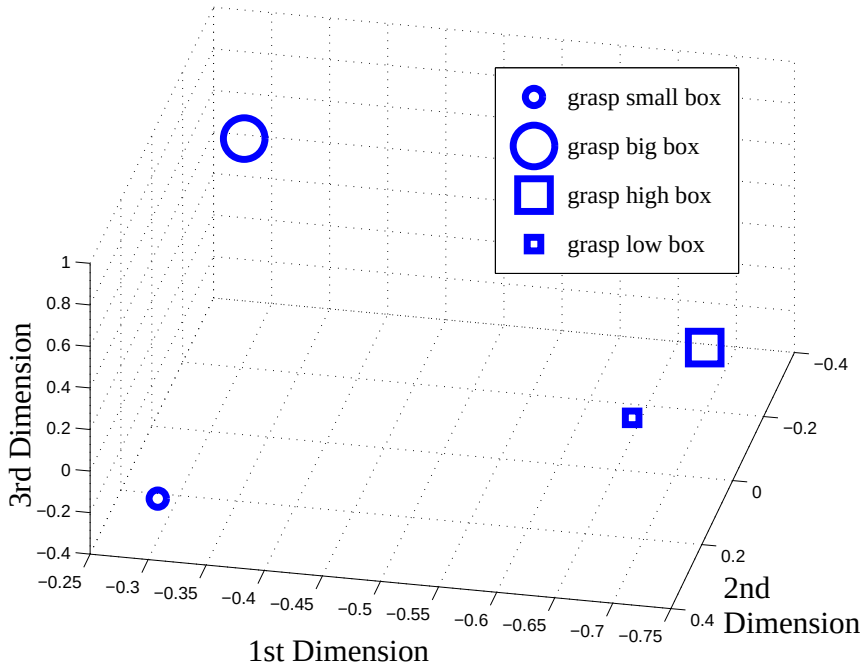


Figure 5-4: Proto-symbol space constructed by four motion primitives

by 1Hz, 5Hz and 10Hz and all the filtered results are supplied as the training data for the Mimesis Model.The demonstrated motions are performed by the Webots iCub to find out their outcomes.

In the symbolization step (Section 5.2.2), the four motion primitives are encoded by four CHMMs. To completely distinguish between four points we need at lease a three dimensional space. Hence we construct a three dimensional proto-symbol space by using the MDS with these CHMMs (Figure 5-4). To generate new grasping motions, we interpolate (Section 5.2.3) the proto-symbol space with different mixing coefficients. New motions are then generated at each of the interpolation points as detailed in the Section 5.2.4. These generated motion are then perform by the Webots iCub to examine their effects.

All motions are modeled in ten states, determined by five-fold cross validation, and each state is represented by a single Gaussian to keep the simplicity.

5.3.1 Grasping different sizes boxes

In this scenario we demonstrate the strategies of grasping different sizes of boxes. The boxes are placed on a cylindrical stand with 84*cm* height. The human demonstrator stands 20*cm* in front of the cylindrical stand (Figure 5-5).

(a) Motion 1 (b) Motion 2 (c) Motion 3 (d) Motion 4

(e) Motion 1 (f) Motion 2 (g) Motion 3 (h) Motion 4

Figure 5-5: (a)-(d): Human demonstrating bi-manual grasp of a small box (size 20cm(length) $\times$ 15cm(width) $\times$ 10cm(height)). (e)-(h): Human demonstrating bi-manual grasp of a big box (size 40cm(length) $\times$ 20cm(width) $\times$ 15cm(height))

Mixing Coefficient of the Interpolation Points	Box size of successful grasps (*cm*)
0(small box) 1(big box)	43
0.2(small box) 0.8(big box)	39
0.5(small box) 0.5(big box)	35
0.8(small box) 0.2(big box)	28
1(small box) 0(big box)	25

Table 5.1: Mixing coefficient of the interpolation points and the box Size of Successful grasps (training)

Given Box Size	Predicted Mixing Coefficient
27	0.89(small box) 0.11(big box)
30	0.72(small box) 0.27(big box)
36	0.44(small box) 0.56(big box)
40	0.16(small box) 0.74(big box)

Table 5.2: Given Box Sizes (*cm*) and the Predicted Mixing Coefficient (testing)

Figure 5-5 shows the motion sequences. As can be seen from the figure, for grasping the small box, the hands move directly to it, while for grasping the big box, the arms first open to create a certain distance between hands and then close to reduce the distance until contact with the box. This is to avoid unwanted collision with the box during reaching.

New motions are then generated by mixing the demonstrations. To learn the effect of the motions, all demonstrated and generated motions are performed by the robot. The sizes of the boxes are initially estimated by forward kinematics, and then verified by robot executing the motion to grasp a box. The motion that can hold a box and lift it vertically without any slipping is consider to be a successful grasp. Table 5.1 shows the mixing coefficients of the motions and the corresponding size of boxes of successful grasps. Note that mixing coefficients of them always sum to 1. When we make the mixing coefficient to be 1 for one motion and 0 for the other, the generated motion simply corresponds to the motion with mixing coefficient 1. Linear regression is the applied to find out the correlation between the mixing coefficients and the box sizes.

Figure 5-6 shows the linear regression result of the mixing coefficients and the size of successful grasped boxes. With the regression result, given a size of box, the mixing coefficient of generating a corresponding grasping motion can be deduced. To test this method, we apply this method to grasp four un-demonstrated boxes with different sizes. All of them can be

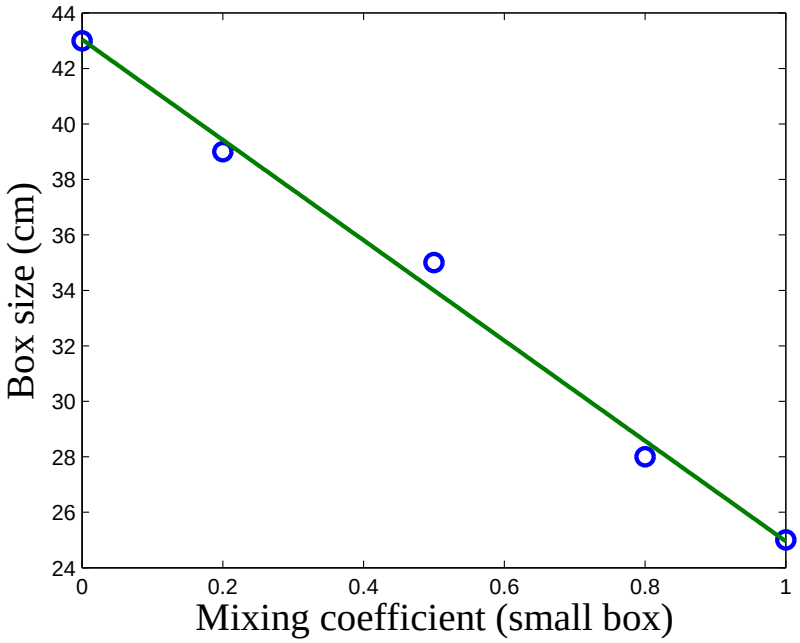


Figure 5-6: Linear regression of the interpolation points

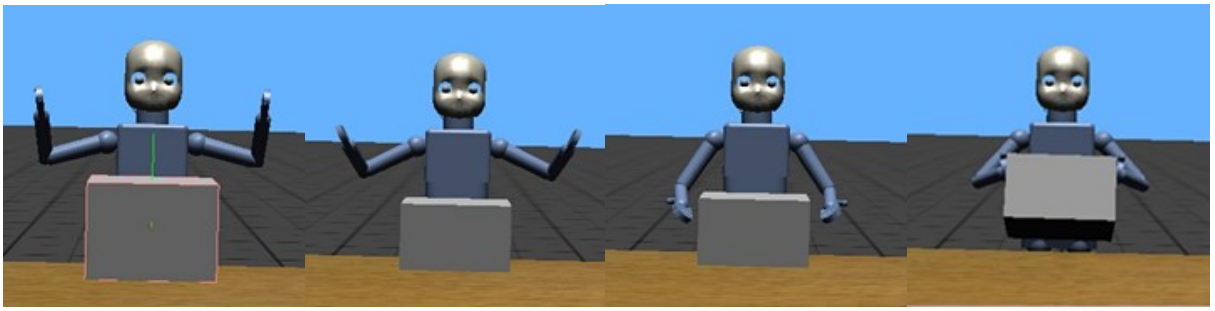
successfully lifted by the synthesis grasping motions. Table 5.2 lists the given boxes size and the computed mixing coefficient and Figure 5-10 shows the corresponding motions.

5.3.2 Grasping boxes from different positions

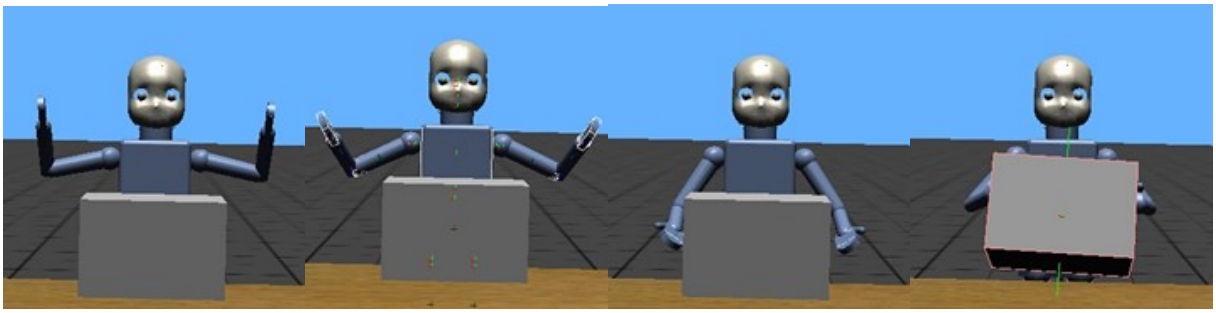
In this scenario the goal is to grasp boxes from different heights. Two motions are demonstrated to grasp a high and a low box. In the demonstrations the high box is placed at the height of 150*cm* and the low box is placed at 70*cm*. In this case, the two demonstrations are not only different in the arm trajectories but also different in the time duration (Figure 5-8). The motion of grasping the high box lasts shorter than grasping the low box as the initial hand position is closer to the box position. At the same time the lifting parts of the motions are different: for the high box the lifting distance is smaller than the low box because of the joint limits of the arms.

Following the same process as described above, we interpolate between the motions for grasping low box and high box (Table 5.3). We apply linear regression and hence find out the correlation between the mixing coefficients and the heights of the box (Figure 5-9).

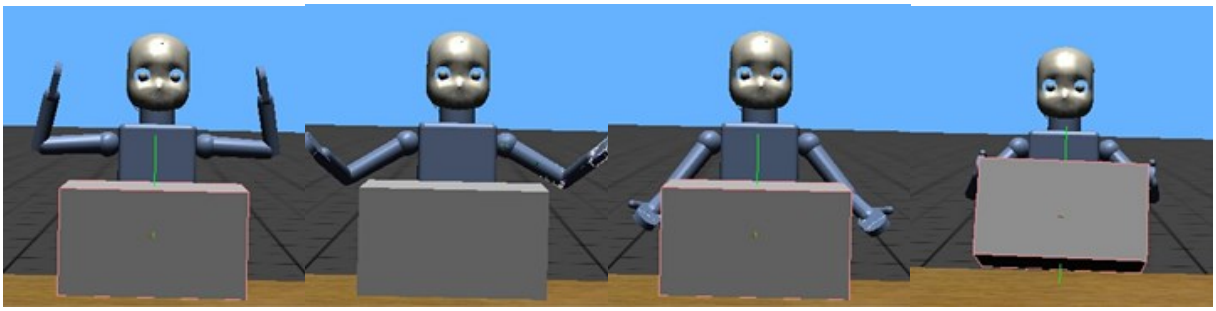
With the learned correlation, we query the mixing coefficients for four different un-demonstrated heights (Table 5.4). The generated motions are tested with the Webots iCub, which successfully lifted all the boxes.



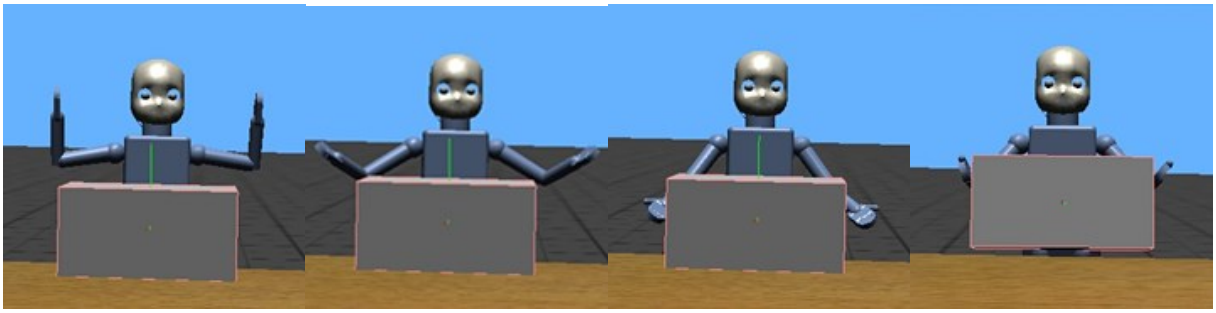
(a) Motion 1 (b) Motion 2 (c) Motion 3 (d) Motion 4



(e) Motion 1 (f) Motion 2 (g) Motion 3 (h) Motion 4

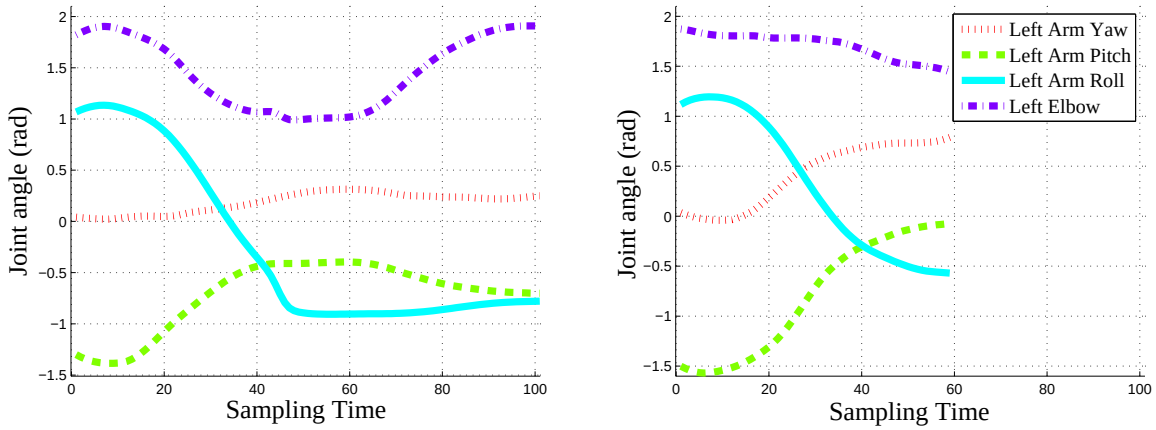


(i) Motion 1 (j) Motion 2 (k) Motion 3 (l) Motion 4



(m) Motion 1 (n) Motion 2 (o) Motion 3 (p) Motion 4

Figure 5-7: Robot grasping different boxes with the generated motions. (a)-(d) Box size 0.27cm. (e)-(h) Box size 0.3cm. (i)-(l) Box size 0.35cm. (m)-(p) Box size 0.4cm.



(a) Grasping box in a low position

(b) Grasping box in a high position

Figure 5-8: (a) Left arm motion of a human demonstration of grasping a low box. (b) Left arm motion of a human demonstration of grasping a high box.

Mixing Coefficient	Box height(*cm*)
0(high box) 1(low box)	49
0.2(high box) 0.8(low box)	53
0.5(high box) 0.5(low box)	58
0.8(high box) 0.2(low box)	62
1(high box) 0(low box)	64

Table 5.3: Mixing coefficient of the interpolation points and the box heights (center of mass from the ground) of successful grasps (training).

Given Box Height	Predicted Mixing Coefficient
50	0.02(high box) 0.98(low box)
55	0.34(high box) 0.66(low box)
60	0.66(high box) 0.34(low box)
63	0.86(high box) 0.14(low box)

Table 5.4: Given Box Height (*cm*) and the Predicted Mixing Coefficient (testing)

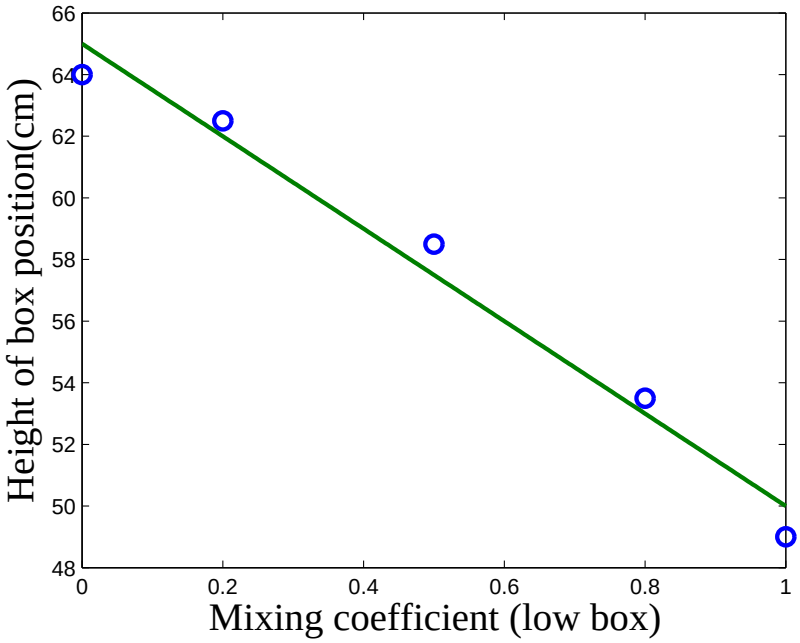
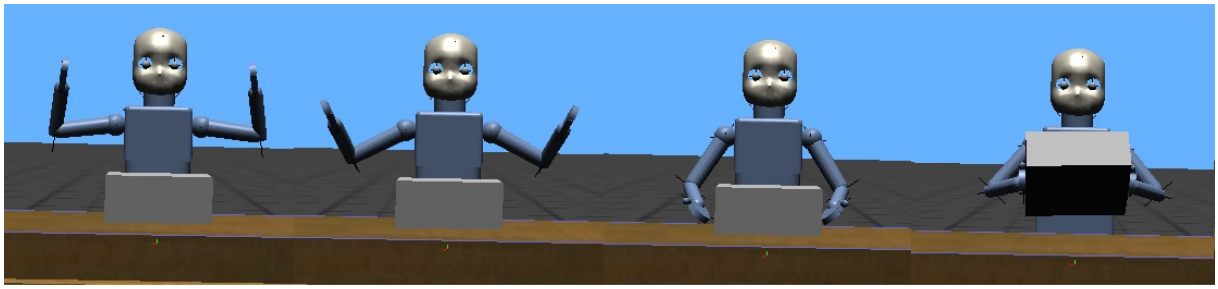
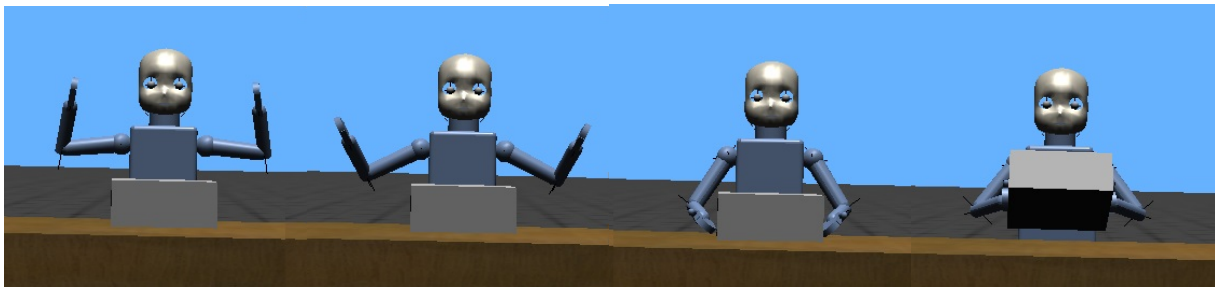


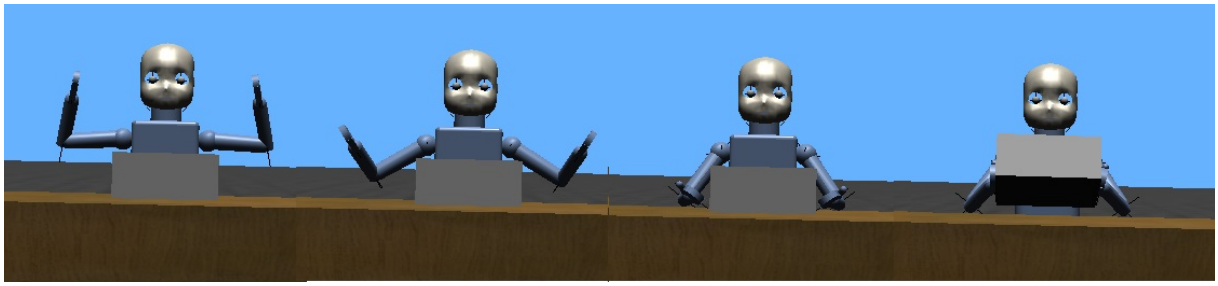
Figure 5-9: Linear regression of the interpolation points



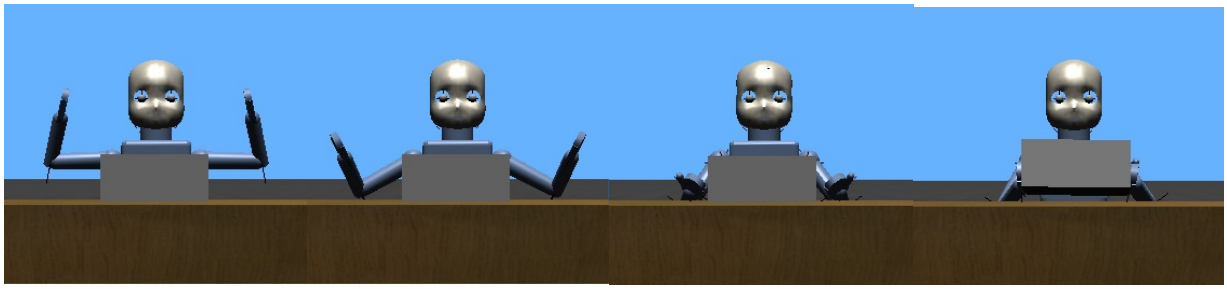
(a) Motion 1 (b) Motion 2 (c) Motion 3 (d) Motion 4



(e) Motion 1 (f) Motion 2 (g) Motion 3 (h) Motion 4



(i) Motion 1 (j) Motion 2 (k) Motion 3 (l) Motion 4



(m) Motion 1 (n) Motion 2 (o) Motion 3 (p) Motion 4

Figure 5-10: Robot grasping boxes from different heights with the generated motions. (a)-(d) Box at height 50*cm*. (e)-(h) Box at height 55*cm*. (i)-(l) Box at height 60*cm*. (m)-(p) Box at height 63*cm*.

5.4 Discussion

The system we present in this chapter uses the mimesis model to learn motion primitives for object manipulation. It provides an easy to use interface for robot motion generation. Motion primitives are the elementary motions that accomplish basic functions. Recent brain research Bizzi et al. (2008) provide more evidences to support the hypothesis that, in order to reduce the degree of freedom, the vertebrate motor system generate motions by combining a small number of motor primitives. Our experiment shows that by combining two motion primitives (8 d.o.f) indeed we can generate many different motion patterns. This framework largely simplifies the modeling of motion primitives.

From the function point of view, the motion primitives form the vocabulary of motions. This naturally allows us to label motion primitives by words. In our system, each motion primitive is symbolized in the proto-symbol space and labeled by its effects, i.e. "grasp high box", "grasp low box" and etc. This provides a linguistic interface for the human to instruct robot motion, by giving language instructions like "go lower to grasp the box". By matching the words in human instruction and the labels of the motion primitives, the robot will be able to adjust the mixing coefficient and generate new motion to execute the command.

We implement the system in the Webots simulator with the iCub robot. The interpolation of the proto-symbols produces new motion primitives. The correlation between the new motions and their effects are learnt using first order linear regression. In our future work of learning more complex motion primitives, higher order or nonlinear regression may need to be employed.

The presented experiment provides a good starting point for our future study in learning more motion primitives for object manipulation. It shows that it is possible to directly control the outcome of the motion pattern, without fine tuning different variables in the model. This has the advantage, from the user point of view, of planning the motion primitives more intuitively. In this chapter, we show that with the mimesis model control in one dimension (object size, object height) works effectively. In the future work, we will further study the control in multi-dimension, i.e. interpolation between multi-proto-symbols.

Bibliography

Anderson, B. D., Brinsmead, T. S., De Bruyne, F., Hespanha, J., Liberzon, D., and Morse, A. S. (2000). Multiple model adaptive control. part 1: Finite controller coverings. *International Journal of Robust and Nonlinear Control*, 10(11-12):909–929.

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

Athans, M., Castanon, D., Dunn, K.-P., Greene, C., Lee, W., Sandell Jr, N., and Willsky, A. S. (1977). The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method–part i: Equilibrium flight. *Automatic Control, IEEE Transactions on*, 22(5):768–780.

Barrett, H. C. and Kurzban, R. (2006). Modularity in cognition: framing the debate. *Psychological review*, 113(3):628.

Barry, J., Hsiao, K., Kaelbling, L. P., and Lozano-Pérez, T. (2013). Manipulation with multiple action types. In *Experimental Robotics*, pages 531–545. Springer.

Bekiroglu, Y., Laaksonen, J., Jorgensen, J. A., Kyrki, V., and Kragic, D. (2011). Assessing grasp stability based on learning and haptic data. *Robotics, IEEE Transactions on*, 27(3):616–629.

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

Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA.

Billard, A. G., Calinon, S., and Guenter, F. (2006). Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robotics and Autonomous Systems*, 54(5):370–384.

Bizzi, E., Cheung, V., d'Avella, A., Saltiel, P., and Tresch, M. (2008). Combining modules for movement. *Brain Research Reviews*, 57(1):125–133.

Brooks, R. A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1):139–159.

Brost, R. C. (1988). Automatic grasp planning in the presence of uncertainty. *The International Journal of Robotics Research*, 7(1):3–17.

Bryson, J. J. (2005). Modular representations of cognitive phenomena in AI, psychology and neuroscience. In Davis, D. N., editor, *Visions of Mind: Architectures for Cognition and Affect*, pages 66–89. Idea Group.

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

Calinon, S. (2008). Robot programming by demonstration. In *Springer Handbook of Robotics*, pages 1371–1394. Springer.

Calinon, S. and 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*, pages 255–262. ACM.

Calinon, S., Guenter, F., and 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.

Chang, L. Y., Srinivasa, S. S., and Pollard, N. S. (2010). Planning pre-grasp manipulation for transport tasks. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2697–2704. IEEE.

Ciocarlie, M. and Allen, P. (2009). Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7):851–867.

Cohn, D., Ghahramani, Z., and Jordan, M. (1996). Active learning with statistical models. *Arxiv preprint cs/9603104*.

Dang, H. and Allen, P. K. (2012). Learning grasp stability. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2392–2397. IEEE.

Dang, H. and Allen, P. K. (2014a). Semantic grasping: planning task-specific stable robotic grasps. *Autonomous Robots*, pages 1–16.

Dang, H. and Allen, P. K. (2014b). Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 36(4):309–330.

Daniel, C., Neumann, G., Kroemer, O., and Peters, J. (2013). Learning sequential motor tasks. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2626–2632. IEEE.

Daoud, N., Gazeau, J., Zeghloul, S., and Arsicault, M. (2011). A fast grasp synthesis method for online manipulation. *Robotics and Autonomous Systems*.

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

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

Detry, R., Baseski, E., Popovic, M., Touati, Y., Kruger, N., Kroemer, O., Peters, J., and Piater, J. (2009). Learning object-specific grasp affordance densities. In *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, pages 1–7. IEEE.

Detry, R., Hjelm, M., Ek, C. H., and Kragic, D. (2013). Generalizing task parameters through modularization. In *Autonomous Learning Workshop (Workshop at ICRA 2013)*.

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

Dizioğlu, B. and Lakshiminarayana, K. (1984). Mechanics of form closure. *Acta mechanica*, 52(1-2):107–118.

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

Do, M., Romero, J., Kjellstrom, H., Azad, P., Asfour, T., Kragic, D., and Dillmann, R. (2009). Grasp recognition and mapping on humanoid robots. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 465–471. IEEE.

Ekvall, S. and Kragic, D. (2007). Learning and evaluation of the approach vector for automatic grasp generation and planning. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4715–4720. IEEE.

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

El-Khoury, S. and Sahbani, A. (2010). A new strategy combining empirical and analytical approaches for grasping unknown 3d objects. *Robotics and Autonomous Systems*, 58(5):497–507.

El-Khoury, S., Sahbani, A., and Perdereau, V. (2007). Learning the natural grasping component of an unknown object. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2957–2962. IEEE.

Faria, D. R., Martins, R., Lobo, J., and Dias, J. (2012). Extracting data from human manipulation of objects towards improving autonomous robotic grasping. *Robotics and Autonomous Systems*, 60(3):396–410.

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

Felip, J., Laaksonen, J., Morales, A., and Kyrki, V. (2012). Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks. *Robotics and Autonomous Systems*.

Felip, J., Laaksonen, J., Morales, A., and Kyrki, V. (2013). Manipulation primitives: A paradigm for abstraction and execution of grasping and manipulation tasks. *Robotics and Autonomous Systems*, 61(3):283–296.

Ferrari, C. and Canny, J. (1992). Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE.

Fischer, M., van der Smagt, P., and 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*, volume 2, pages 1603–1608. IEEE.

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

Fod, A., Matarić, M. J., and Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous robots*, 12(1):39–54.

Fodor, J. A. (1983). *The modularity of mind: An essay on faculty psychology*. MIT press.

Friedrich, H., Münch, S., Dillmann, R., Bocionek, S., and Sassin, M. (1996). Robot programming by demonstration (rpd): Supporting the induction by human interaction. *Machine Learning*, 23(2-3):163–189.

Fu, M. and Barmish, B. (1986). Adaptive stabilization of linear systems via switching control. *Automatic Control, IEEE Transactions on*, 31(12):1097–1103.

Gabiccini, M., Bicchi, A., Prattichizzo, D., and Malvezzi, M. (2011). On the role of hand synergies in the optimal choice of grasping forces. *Autonomous Robots*, 31(2-3):235–252.

Gallardo, L. and Kyrki, V. (2011). Detection of parametrized 3-d primitives from stereo for robotic grasping. In *Advanced Robotics (ICAR), 2011 15th International Conference on*, pages 55–60. IEEE.

Garcia, S. (2009). Fitting primitive shapes to point clouds for robotic grasping. *Master of Science Thesis. School of Computer Science and Communication, Royal Institute of Technology, Stockholm, Sweden*.

Gioioso, G., Salvietti, G., Malvezzi, M., and Prattichizzo, D. (2013). Mapping synergies from human to robotic hands with dissimilar kinematics: an approach in the object domain. *Robotics, IEEE Transactions on*, 29(4):825–837.

Glardon, P., Boulic, R., and Thalmann, D. (2004). Pca-based walking engine using motion capture data. In *Computer Graphics International, 2004. Proceedings*, pages 292–298. IEEE.

Gribovskaya, E., Khansari-Zadeh, S. M., and Billard, A. (2010). Learning non-linear multivariate dynamics of motion in robotic manipulators. *The International Journal of Robotics Research*, pages 80–117.

Grillner, S. (2011). Control of locomotion in bipeds, tetrapods, and fish. *Comprehensive Physiology*, 2: Motor control:1179–1236.

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

Harada, K., Kaneko, K., and Kanehiro, F. (2008). Fast grasp planning for hand/arm systems based on convex model. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1162–1168. IEEE.

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

Hauser, K., Bretl, T., Harada, K., and Latombe, J.-C. (2008). Using motion primitives in probabilistic sample-based planning for humanoid robots. In *Algorithmic foundation of robotics VII*, pages 507–522. Springer.

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Bohg, J., Asfour, T., and Schaal, S. (2014). Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2):51–65.

Hoshino, K. (2004). Interpolation and extrapolation of repeated motions obtained with magnetic motion capture. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 87(9):2401–2407.

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

Hsiao, K., Chitta, S., Ciocarlie, M., and Jones, E. G. (2010). Contact-reactive grasping of objects with partial shape information. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1228–1235. IEEE.

Hsiao, K., Ciocarlie, M., and Brook, P. (2011a). Bayesian grasp planning. In *ICRA 2011 Workshop on Mobile Manipulation: Integrating Perception and Manipulation*.

Hsiao, K., Kaelbling, L. P., and Lozano-Pérez, T. (2011b). Robust grasping under object pose uncertainty. *Autonomous Robots*, 31(2-3):253–268.

Huang, B., Bryson, J., and 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, J. J., and Billard, A. (2013b). Learning a real time grasping strategy. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 593–600.

Huebner, K., Ruthotto, S., and Kragic, D. (2008). Minimum volume bounding box decomposition for shape approximation in robot grasping. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1628–1633. IEEE.

Hueser, M., Baier, T., and 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*, pages 2795–2800. IEEE.

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

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1398–1403. IEEE.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1547–1554. MIT Press.

Inamura, T. and Shibata, T. (2008). Geometric proto-symbol manipulation towards language-based motion pattern synthesis and recognition. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 334–339. IEEE.

Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *The International Journal of Robotics Research*, 23(4-5):363–377.

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

Kailath, T. (1967). The divergence and bhattacharyya distance measures in signal selection. *Communication Technology, IEEE Transactions on*, 15(1):52–60.

Kazemi, M., Valois, J.-S., Bagnell, J. A., and Pollard, N. (2012). Robust object grasping using force compliant motion primitives. presented at the Robotics: Science and Systems Conference, Sydney, Australia.

Kehoe, B., Berenson, D., and Goldberg, K. (2012). Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 576–583. IEEE.

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

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

Khoury, S. E., Li, M., and Billard., A. (2012). Bridging the gap: One shot grasp synthesis apporach. In *Intelligent Robots and Systems, IEEE/RSJ International Conference on*. IEEE.

Kim, S. and Billard, A. (2012). Estimating the non-linear dynamics of free-flying objects. *Robotics and Autonomous Systems*.

Kirkpatrick, D., Mishra, B., and Yap, C.-K. (1992). Quantitative steinitz's theorems with applications to multifingered grasping. *Discrete & Computational Geometry*, 7(1):295–318.

Kondo, M., Ueda, J., and 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. and Demiris, Y. (2013). Online quantum mixture regression for trajectory learning by demonstration. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3222–3229. IEEE.

Kroemer, O., Detry, R., Piater, J., and Peters, J. (2010). Grasping with vision descriptors and motor primitives.

Kroemer, O., Detry, R., Piater, J., and Peters, J. (2011). Grasping with vision descriptors and motor primitives. In *Informatics in Control, Automation and Robotics*, pages 211–223. Springer.

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

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

Kulić, D., Ott, C., Lee, D., Ishikawa, J., and 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.

Kulić, D., Takano, W., and 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.

Kulic, D., Takano, W., and Nakamura, Y. (2009). Online segmentation and clustering from continuous observation of whole body motions. *Robotics, IEEE Transactions on*, 25(5):1158–1166.

Kunori, H., Lee, D., and Nakamura, Y. (2009). Associating and reshaping of whole body motions for object manipulation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5240–5247. IEEE.

Kyota, F., Watabe, T., Saito, S., and Nakajima, M. (2005). Detection and evaluation of grasping positions for autonomous agents. In *Cyberworlds, 2005. International Conference on*, pages 8–pp. IEEE.

Laaksonen, J., Felip, J., Morales, A., and Kyrki, V. (2010). Embodiment independent manipulation through action abstraction. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2113–2118. IEEE.

Lee, D. and Ott, C. (2010). Incremental motion primitive learning by physical coaching using impedance control. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4133–4140. IEEE.

Li, M., Bekiroglu, Y., Kragic, D., and Billard, A. (2014a). Learning of grasp adaptation through experience and tactile sensing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, number EPFL-CONF-199937.

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

Li, Z. and Sastry, S. S. (1988). Task-oriented optimal grasping by multifingered robot hands. *Robotics and Automation, IEEE Journal of*, 4(1):32–44.

Meier, F., Theodorou, E., Stulp, F., and Schaal, S. (2011). Movement segmentation using a primitive library. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3407–3412. IEEE.

Michelman, P. and Allen, P. (1994). Forming complex dextrous manipulations from task primitives. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3383–3388. IEEE.

Miller, A. T. and Allen, P. K. (1999). Examples of 3d grasp quality computations. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1240–1246. IEEE.

Miller, A. T. and Allen, P. K. (2004). Graspit! a versatile simulator for robotic grasping. *Robotics & Automation Magazine, IEEE*, 11(4):110–122.

Miller, A. T., Knoop, S., Christensen, H. I., and Allen, P. K. (2003). Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE.

Morrow, J. D. and Khosla, P. K. (1997). Manipulation task primitives for composing robot skills. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 3354–3359. IEEE.

Mussa-Ivaldi, F. A. (1999). Modular features of motor control and learning. *Current opinion in neurobiology*, 9(6):713–717.

Mussa-Ivaldi, F. A., Giszter, S. F., and Bizzi, E. (1994). Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of Sciences*, 91(16):7534–7538.

Nakanishi, J., Radulescu, A., and Vijayakumar, S. (2013). Spatio-temporal optimization of multi-phase movements: Dealing with contacts and switching dynamics. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5100–5107. IEEE.

Narendra, K. S. and Balakrishnan, J. (1994). Improving transient response of adaptive control systems using multiple models and switching. *Automatic Control, IEEE Transactions on*, 39(9):1861–1866.

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

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

Neilson, P., Neilson, M., and O'Dwyer, N. (1985). Acquisition of motor skills in tracking tasks: Learning internal models. *Motor memory and control. Human Performance Associates, Dunedin*, pages 25–36.

Nguyen, V. (1987). Constructing stable grasps in 3d. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 234–239. IEEE.

Okamura, A. M., Smaby, N., and Cutkosky, M. R. (2000). An overview of dexterous manipulation. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 255–262. IEEE.

Okuno, K. and Inamura, T. (2011). Motion coaching with emphatic motions and adverbial expressions for human beings by robotic system-method for controlling motions and expressions with sole parameter. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3381–3386. IEEE.

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

Pais, A. L., Umezawa, K., Nakamura, Y., and Billard, A. (2013). Learning robot skills through motion segmentation and constraints extraction. HRI Workshop on Collaborative Manipulation.

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

Pelossof, R., Miller, A., Allen, P., and Jebara, T. (2004). An svm learning approach to robotic grasping. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3512–3518. IEEE.

Peretz, I. and Coltheart, M. (2003). Modularity of music processing. *Nature neuroscience*, 6(7):688–691.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

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

Pokorny, F. T., Hang, K., and Kragic, D. (2013). Grasp moduli spaces. In *Robotics: Science and Systems*.

Ponce, J., Sullivan, S., Sudsang, A., Boissonnat, J., and Merlet, J. (1997). On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *The International Journal of Robotics Research*, 16(1):11.

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

Ren, G. and O'Neill, E. (2012). 3d marking menu selection with freehand gestures. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 61–68. IEEE.

Richtsfeld, M., Ponweiser, W., and Vincze, M. (2008). Real time grasping of freely placed cylindrical objects. In *ICINCO-RA (1)'08*, pages 165–170.

Rizzolatti, G. and Craighero, L. (2004). The mirror-neuron system. *Annu. Rev. Neurosci.*, 27:169–192.

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

Romero, J., Kjellstrm, H., and Kragic, D. (2008). Human-to-robot mapping of grasps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, WS on Grasp and Task Learning by Imitation*.

Sahbani, A., El-Khoury, S., and Bidaud, P. (2011). An overview of 3d object grasp synthesis algorithms. *Robotics and Autonomous Systems*.

Salisbury Jr, J. (1985). *Kinematic and force analysis of articulated hands*. John Wiley & Sons, Inc.

Santello, M., Flanders, M., and Soechting, J. F. (1998). Postural hand synergies for tool use. *The Journal of Neuroscience*, 18(23):10105–10115.

Santello, M. and Soechting, J. F. (2000). Force synergies for multifingered grasping. *Experimental Brain Research*, 133(4):457–467.

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

Saxena, A., Driemeyer, J., and Ng, A. Y. (2008). Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173.

Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547.

Schaal, S., Peters, J., Nakanishi, J., and Ijspeert, A. (2005). Learning movement primitives. In *Robotics Research*, pages 561–572. Springer.

Schiffman, S. S., Reynolds, M. L., and Young, F. W. (1981). *Introduction to multidimensional scaling: Theory, methods, and applications*. Academic Press New York.

Shukla, A. and Billard, A. (2011). Coupled dynamical system based arm-hand grasping model for learning fast adaptation strategies under real-time perturbations. page 313. MIT Press.

Stphane Lalle, J. D. and Rousset., S. (2009). Multiple object manipulation: is structural modularity necessary? a study of the mosaic and carma models. In *Proceedings of the International Conference on Cognitive Modeling (ICCM 09)*, pages 306–311.

Stulp, F., Theodorou, E., Buchli, J., and Schaal, S. (2011). Learning to grasp under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5703–5708. IEEE.

Stulp, F., Theodorou, E. A., and Schaal, S. (2012). Reinforcement learning with sequences of motion primitives for robust manipulation. *Robotics, IEEE Transactions on*, 28(6):1360–1370.

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

Sztarker, J. and Tomsic, D. (2011). Brain modularity in arthropods: individual neurons that support what but not where memories. *The Journal of Neuroscience*, 31(22):8175–8180.

Takano, W. and Nakamura, Y. (2006). Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 425–431. IEEE.

Takano, W. and Nakamura, Y. (2008). Integrating whole body motion primitives and natural language for humanoid robots. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 708–713. IEEE.

Takano, W., Yamane, K., Sugihara, T., Yamamoto, K., and Nakamura, Y. (2006). Primitive communication based on motion recognition and generation with hierarchical mimesis model. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3602–3609. IEEE.

Tegin, J., Ekvall, S., Kragic, D., Wikander, J., and Iliev, B. (2009). Demonstration-based learning and control for automatic grasping. *Intelligent Service Robotics*, 2(1):23–30.

Wächter, A. and Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Warren Liao, T. (2005). Clustering of time series data: a survey. *Pattern recognition*, 38(11):1857–1874.

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., and Hirzinger, G. (2012). Comparison of object-level grasp controllers for dynamic dexterous manipulation. *The International Journal of Robotics Research*, 31(1):3–23.

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

Ying, L., Fu, J., and Pollard, N. (2007). Data-driven grasp synthesis using shape matching and task-based pruning. *Visualization and Computer Graphics, IEEE Transactions on*, 13(4):732–747.

Zheng, Y. and Qian, W.-H. (2005). Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research*, 24(4):311–327.

Zhu, X. and Ding, H. (2004). Planning force-closure grasps on 3-D objects. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1258–1263. IEEE.

Zhu, X. and Wang, J. (2003). Synthesis of force-closure grasps on 3-D objects based on the Q distance. *Robotics and Automation, IEEE Transactions on*, 19(4):669–679.