DEXTEROUS MANIPULATION WITH A ROBOTIC HAND

A Thesis

by

Osman Kaya

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
June 2017

DEXTEROUS MANIPULATION WITH A ROBOTIC HAND

Approved by:

Associate Professor Erhan Öztop, Advisor
Department of Computer Science
Özyeğin University

Assistant Professor Özkan Bebek
Department of Mechanical Engineering
Özyeğin University

Assistant Professor Emre Uğur
Department of Computer Engineering
Boğaziçi University

Date Approved: 25 May 2017

ABSTRACT

From grasping to fine manipulation, the human hand with its diverse functionality is the result of a highly evolved anatomical model along with an elaborate control system governed by a large portion of the motor cortex. In the robotics context, the flexible and the dexterous manipulation are one of the most desired type of skills. Although simple manipulation tasks can often be performed by grippers of lesser kinematic complexity, their limitations are well-known. To this end, we investigate dexterous manipulation skills on an anthropomorphic robot hand.

The thesis can be considered as the investigation of two related manipulation skills. In the first part of the study, a sensorless grasping method is described. The human hand has a high-precision, fine-grained sensory feedback in the palm and the inside of the fingers. Although this feedback is highly relevant for precise grasps, precision is often not necessary to perform power grasps. Instead of complicated sensor setups, an alternative approach is proposed for robotic grasping tasks based on external force estimation using the dynamic model of the hand and disturbance observers. Estimation accuracy is confirmed using a force sensor and the estimations are found to be useful for creating soft/power grasp behavior.

In the second part, human-in-the-loop control with heterogeneous control for dexterous manipulation is investigated on a setup with a robotic hand and a robotic arm. The goal of the study is to experimentally verify that in tasks where the manual and explicit trajectory tuning is not possible, the autonomous movement can be learned by giving a basic policy to a robotic system, after which a human can learn and transfer an orthogonal complex part of the policy. The approach is particularly useful for dexterous manipulation tasks since the human can learn an otherwise difficult task

at a slower speed and the merged autonomous policy can further be improved using other techniques. The approach is shown on a ball swapping task in which a robotic arm is controlled by the human and a robotic hand is given an initial basic policy. After the human learns to adapt to the movement of the hand, a base autonomous policy is fused. As a result, we experimentally show that, in certain tasks, complex autonomous policies can be constructed by first destructuring the control policy into a simple and complex part, then delegating the complex learning part to a human, and finally creating an autonomous control policy by recombining the parts.

ÖZETÇE

İnsan eli obje kavramadan hassas manipulasyona oldukça farklı fonksiyonları yerine getirebilmektedir. Bu işlemler gelişmiş anatomik el yapısının yanısıra motor korteksin büyük bir kısmını kullanan kontrol mekanizmasını da kullanmaktadır. Robotik bağlamında, esnek ve cerbezi manipülasyon en çok ihtiyaç duyulan becerilerdendeir. Basit manipülasyon görevleri kinematiği karmaşık olmayan tutucularla yapılabilmesine rağmen, kısıtlamalar yaygın olarak bilinmektedir. Bu sebeple, bu tezde, insansı bir robot el üzerinde cerbezi manipülasyon becerileri araştırılmıştır.

Bu tezde birbiriyle alakalı iki manipülasyon becerisi araştırılmıştır. Çalışmanın ilk kısmında, sensörsüz bir kavrama yöntemi açıklanmıştır. İnsan elinin avuç ve parmak içi alanlarda yüksek hassasiyete sahip sensör geribeslemesi bulunmaktadır. Bu geribesleme hassas kavrama için gerekli olsa da, güçlü kavrama için yüksek hassasiyet çoğu zaman ihtiyaç duyulmamaktadır. Karmaşık sensör kurulumlarına alternatif olarak, robotik kavrama için elin dinamik modeline ve dışsal kuvvet tahminine dayalı bir sensörsüz kavrama yöntemi geliştirilmiştir. Tahmin doğruluğu bir kuvvet sensörü kullanılarak onaylanmıştır ve önerilen yöntemle yumuşak veya sıkı kavrama yeteneklerinin gerçekleştirilebildiği görülmüştür.

İkinci kısımda, cerbezi manipülasyon için heterojen kontrollü insanlı-döngü kontrolü, bir robot kol ve robot elden oluşan bir sistem üzerinde incelenmiştir. Çalışmanın amacı, elle veya otomatik yörünge ayarlamanın mümkün olmadığı durumlarda, robota temel bir idare vererek idarenin karmaşık kısmın insandan elde edilebileceğini, sonuç olarak karmaşık bir görev için gerekli otonom idarenin bulunabileceğini deneysel olarak göstermektir. İnsanlar zor bir görevi düşük hızda öğrenebileceği ve öğrenimden sonra diğer tekniklerle idare geliştirilebildiği için bu yaklaşım cerbezi manipülasyon

görevleri için özellikle uygundur. Yöntem, örnek olarak bir robot kolun insan tarafından kullanıldığı ve bir robot elin otonom çalışarak geröekleştirdiği avuç-içi top çevirme görevinde gösterilmiştir. Elde edilen sonuçlarda, karmaşık otonom idarelerin, karmaşık kısımlarını öğrenmeyi insana delege ederek, insan öğreniminden sonra otonom control idaresinin elde edilebilecği gösterilmiştir.

ACKNOWLEDGEMENTS

My sincere thanks to Erhan Öztop for his guidance and supervision. I thank Barkan Uğurlu for his encouragement and support. I also thank Özkan Bebek and Emre Uğur for their very helpful feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Among hand studies, it is generally agreed that, after the human hand was freed from the constraints of locomotion, it evolved primarily for tool manipulation abilities[1, 2]. Napier identifies two unique grasps, namely precision and power grasps[3] and these grasps are hypothesized to be specializations for aggressive throwing and clubbing[4]. Learning the plethora of manipulation skills and learning to use tools effectively took millions of years in the process and these skills are still at the heart of the modern human civilization.

The goal of the robotics is to develop and invent the necessary theories and technologies to create capable robots so that we do not have to do as much. Many of these desired capabilities directly correspond to manipulation. Therefore, in this thesis, we study dexterous manipulation for the purpose of improving our knowledge on the subject.

The thesis is organized as follows. In the rest of this chapter, a brief literature overview is given. In the second chapter, we propose a practical external force estimation method which can be utilized for common grasps. In the third chapter, technical setup for the human control of an arm robot is explained. In the fourth chapter, teaching a robot a dexterous manipulation task is studied on a ball swapping task. Finally, in the fifth chapter, a brief discussion is provided.

1.1 Manipulation Dexterity

In the robotics domain, the definition of the dexterity is not clear-cut and varies significantly[5]. To give two examples; Klein et al. define it as "the kinematic extent over which a manipulator can reach all orientations"[6]. Bicchi gives the definition

"capability of changing the position and orientation of the manipulated object from a given reference configuration to a different one, arbitrarily chosen within the hand workspace"[7]. The important difference in definitions is often where they fall between object-centric and manipulator-centric views. On the other hand, a dexterous manipulation is differentiated from a non-dexterous one as being performed by multiple cooperating manipulators to change an object's state[8]. Used in this sense, it is easy to understand why dexterous manipulation is commonly used in its relation to hand manipulators with complex kinematic structure, although it is not exclusive to them.

Motor synergy appears to be one of the key mechanisms of CNS in generating dexterous hand function [9]. With this inspiration, robotic hands are sometimes designed synergistically and/or controlled via synergies imposed over finger joints. For instance, Ajoudani et al. utilized tele-impedance control in simplifying hand complexity into distinct motor patterns [10]. Gabiccini et al. proposed a framework that incorporates the structural properties of a hand in a quasi-static setting so that it could be driven via synergistic actions [11]. Catalano et al. made use of adaptive synergies to demonstrate various grasping behavior on the UNIPI-hand [12].

In robotics, dexterous manipulation is sought, besides synergies, through learning-by-demonstration [13] and reinforcement learning (see [14],[15]) which have proven to be powerful techniques for generating complex robotic hand skills.

In general for grasping, statistical learning appears to be effective [16][17], and the contribution of tactile input seems important once the object is in contact with the hand.

1.2 Human in the Loop Learning

In human in the loop(HitL) control, the main goal is to increase task efficiency by involving the human to the robot's control loop. For complex tasks, the robot may

fail to learn the task efficiently and involving the human may improve the success. Similarly, leaving the easily automatable tasks to the robot will reduce the load on the human.

There are different variations of human in the loop learning. In [18] Leeper et al. categorizes the HitL control strategies to three types. We add the fourth one to it:

- Direct control: The human directly controls the movement of the robot. This is common in teleoperation tasks.
- Shared control: The human and the robot both controls the movement. In this mode of control, the robot controls some aspects of the movement and the human controls the motion. Driving assistance is a good example of this type. Another alternative can be the human and the robot both controls the same aspects of the movement at the same time.
- Supervisory control: The human decides the high-level behavior and the robot has the control over the movement. This is utilized in scenarios where the human decides on the waypoints and the motion is autonomous.
- Heterogeneous control: The human and the robot operates on different parts of the robot. Control domains of the robot and the human are orthogonal and their responsibility is strictly separated.

Several successful applications exist in the literature for the HitL control. DeDonato et al. apply the approach to control a limited bandwidth robot. Tsui et al. use a supervisory type of control to a wheel-mounted robotic arm to reduce cognitive load on the human. Leeper et al. discuss the HitL robotic grasping strategies.

1.3 Human-in-the-Loop Learning

Human in the loop learning methods are based on the human in the loop control. The goal of these methods is to create autonomous policies by having the robot learn

from the human in the control loop. The goal is to learn the task controls effectively in cases where the human cannot teach alone or the robot cannot feasibly learn alone. Although learning by demonstration is a very effective technique for tasks involving position control, there are few examples where it is used for dynamic tasks because the force policies can not be demonstrated. HitL learning, on the other hand, is significantly effective on dynamic skills and cooperative manipulation.

As an early example, Oztop et al. demonstrated synthesis of ball swapping task using human-in-the-loop robot skill generation[19]. In a follow-up work[20], they propose to integrate a hand robot to the human's body schema. The idea is that once the human internalizes the robot as a part of an extension to their body, controls for rich dexterous skills can be obtained.

Babič et al. proposes an interestimg approach in [21] to learn dynamic skills. Instead of using the cognitive functions of the human, they utilize sensorimotor learning skills of the human. They show the success of the task on a seesawing task. In [22], Peternel et al. teach a humanoid robot to balance by using the approach and considerably simplifies the learning process. These type of force based skills usually cannot be easily transferred by "learning by demonstration".

CHAPTER II

ENVIRONMENTAL FORCE ESTIMATION WITH A ROBOTIC HAND

It is well known that humans mostly rely on tactile feedback for manipulation tasks and infants heavily utilize tactile feedback when learning how to grasp[24]. Therefore, contact force information is the primary factor in dexterous manipulation task performance and is seen as a prerequisite[25].

For precise grips, precise force sensing in three axes becomes a necessity. This is an important consideration in hand design process and adds complications to the overall design because of the required sensor and wiring accommodation. The sensors also come with a financial cost and calibration processes take more time.

On the other hand, for power grips, these restrictions do not apply and coarse force estimations can be found control methods. With this in mind, we aimed at utilizing disturbance observer (DoB) technique, which is extensively used in industrial automation for robust control [26, 27], so that we can infer environmental forces acting on the robot fingers.

In general, a DoB can estimate the overall disturbance acting on the system due to robot dynamics (i.e. inertial forces, Coriolis effect, and gravity), friction, and external forces. In [28], Ugurlu et al. hypothesized that if the robot dynamics and frictional loads are sufficiently compensated, DoBs would solely output environmental interaction forces thereby eliminating the need for special sensors. Sharing the

This chapter is published in [23] and used in accordance with IEEE and Ozyegin University reuse policy.

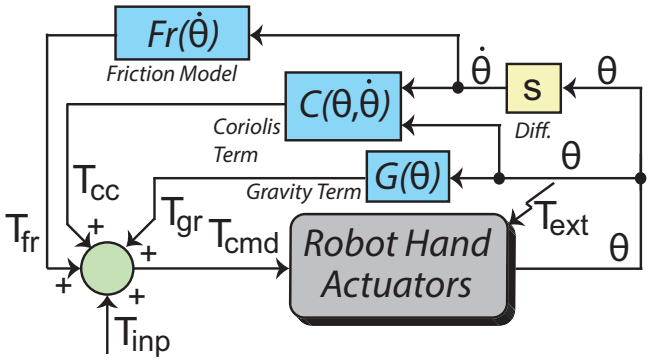

Figure 1: The compensation scheme when the robot is in torque control mode.

common objective, a framework is presented to estimate contact force information, in an attempt to eliminate the need for tactile/force sensors to reduce complexity in dexterous robotic manipulators, e.g. a multi-fingered anthropomorphic robot hand. The framework incorporates model-based compensation loops to account for disturbances based on robot dynamics and joint friction.

2.1 Compensation Schemes

The general compensation scheme is given in Fig 1. It includes dynamics (Coriolis&centrifugal, gravity) and friction compensation loops.

2.1.1 Dynamic Load Compensation

Coriolis and gravity compensation torques are calculated using the dynamics model of the robot and CAD data. Angular velocities for joints, $\dot{\theta}$, are low-pass filtered to reduce the noise. Inertia term is omitted in equations, as the joint acceleration values were observed to be negligible during the nominal operation.

2.1.2 Friction Compensation

With the condition of gravity and Coriolis forces are compensated, friction becomes the only effective disturbance force observed on the system while no external force

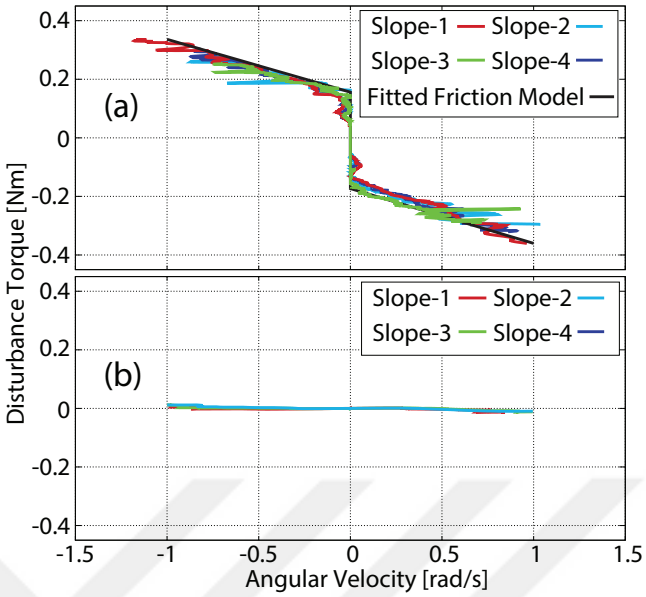

Figure 2: a) Frictional torque - angular velocity variations for 4 distinct ramp inputs. The piece-wise fitted friction model is acquired by using these data. b) Having compensated all dynamics and frictional loads, the DoB outputs near zero variation, adequately confirming the accuracy of the proposed compensation scheme. In other words, the robot is approximately free of any dynamics and frictional load.

is applied. Therefore, DoB outputs frictional torque. With this in mind, we implemented ramp inputs with various slope values and observed the frictional torque-angular velocity variations for a single joint is given in Fig. 2(a) [26]. In this figure, red, cyan, green, and blue curves indicate actual frictional torque-angular velocity variations while black lines stand for the fitted model. A similar procedure was carried out for all other joints, thus not plotted.

As may be seen in Fig. 2(a), joint friction can be modeled as a piecewise linear equation, composed of stiction and viscous friction (see the black curve). These two components constitute an accurate representation of the friction model as Stribeck effect is observed to be negligible in our case. The model for the i^{th} joint can be formulated in terms of its angular velocity ($\dot{\theta}_i$) as follows,

$$T_{fr}(\dot{\theta}_i) \quad = \quad a_1(\lambda_1 + \lambda_2\dot{\theta}_i) - a_2(\lambda_3 - \lambda_4\dot{\theta}_i) \tag{1}$$

where λ_1 and λ_3 are static friction coefficients for positive and negative direction, and λ_2 and λ_4 are viscous friction coefficients when $\dot{\theta}_i > 0$, $a_1 = 1$ and $a_2 = 0$. Likewise, when $\dot{\theta}_i < 0$, $a_1 = 0$ and $a_2 = 1$. Following the model parameter identification, friction compensation is realized by feeding the actuator with T_{fr} to account for loads due to joint friction.

2.1.3 Compensation Procedure

General dynamic model of a torque controlled robot is defined by the following equation,

$$T_{cmd} = J_r\ddot{\theta} + T_l + T_{fr}(\dot{\theta}), \tag{2}$$

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) = T_{ext} + T_l, \tag{3}$$

where θ is the joint angle vector, J_r is the rotor inertia, T_{cmd} is motor command torque vector, T_{ext} is external force vector, T_l is the joint torque vector on the links, T_{fr} is the frictional torque vector, $M(\theta)$, $G(\theta)$, $C(\theta,\dot{\theta})$ are inertia, gravity, and coriolis terms, respectively. Combination of (2) and (3) results as follows.

$$\begin{aligned} T_{cmd} &= M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) \\ &+ T_{fr}(\dot{\theta}) + J_r\ddot{\theta} - T_{ext} \end{aligned} \tag{4}$$

To eliminate the effects of these dynamic forces acting on the system, overall compensation scheme is designed as shown in Fig. 1 [28]. In this scheme, actuators are supplied with additional torques T_{fr}, T_{gr}, T_{cc} to respectively compensate for $Fr(\dot{\theta})$, $G(\theta)$, $C(\theta,\dot{\theta})$ in a feedforward manner. Incorporating the compensation terms, T_{cmd} can be rewritten as:

$$T_{cmd} = T_{inp} + T_{fr} + T_{gr} + T_{cc} \tag{5}$$

In (5), T_{inp} is the task-specific input. Provided the compensation torques sufficiently account for Coriolis, gravity and frictional loads ($T_{cc} + T_{gr} + T_{fr} \cong C(\theta, \dot{\theta})\dot{\theta} + G(\theta) + Fr(\dot{\theta})$), these terms may be canceled out; therefore, the dynamic model of the robot can be expressed as follows.

$$T_{inp} + T_{ext} = J_r \ddot{\theta} \tag{6}$$

Having compensated torques based on Coriolis effect, gravity and friction terms, we re-ran disturbance estimation to experimentally verify the accuracy of the compensation schemes. To this end, the robot joints are actuated via torque inputs and we observed the DoB outputs for every joint. No external force was implemented. An exemplary data is provided in Fig 2(b). As may be observed, DoB output is simply varied near zero; adequately validating the fact that the robot is free of any dynamics and frictional load. Furthermore, this figure also validates our approach in omitting the inertia terms, as there is almost zero DoB output. Though using DoB for friction identification may be infeasible for more complex joint designs and less accurate under model uncertainties [29], we found it to be accurate where the explicit dynamic model exists.

2.1.4 Force Estimation

The simple DoB architecture for a single joint is displayed in Fig. 3 [26]. Motor model, rotor inertia, actual disturbance torque, estimated disturbance torque (DoB output) and command torque input are symbolized with $P(s)$, J_r, T_d, T_d^* and T_i, respectively. In this scheme, an approximation for the inverse plant model is utilized to estimate the disturbance acting on the joint.

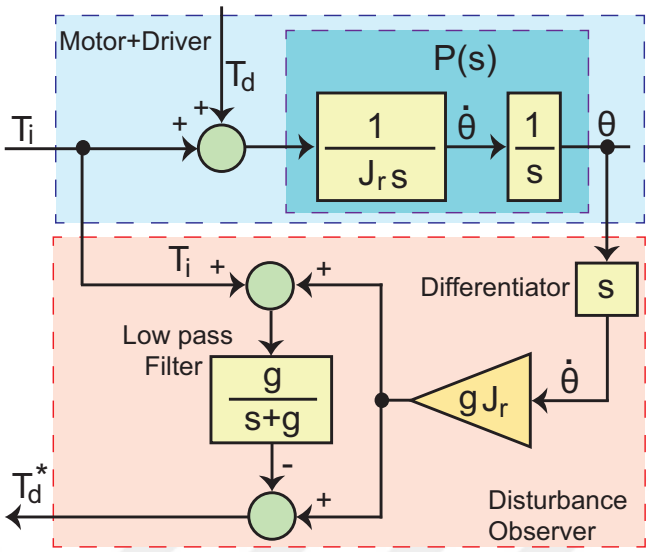

Figure 3: Basic disturbance observer scheme.

$$T_d^* = gJ_r s\theta - \frac{g}{s+g}(T_i + gJ_r s\theta) \tag{7}$$

$$T_d^* = \frac{s^2 J_r \theta}{\frac{1}{g}s+1} - \frac{1}{\frac{1}{g}s+1}T_i \tag{8}$$

$$T_d^* \cong \left(J_r\ddot{\theta} - T_i\right) \cong T_d \tag{9}$$

For a multi-joint robotic system, T_d^* term consist of the resultant disturbance torque that arises due to robot dynamics (inertial, Coriolis, and gravity) friction, environmental interaction. Given the fact that inertial effect is negligible, and Coriolis, gravity and friction terms are compensated, DoB simply outputs the environmental force. At this stage, please note the analogy between the final DoB equation, (9) and our final derivation after the compensation, namely, Eq. (6). Input torque and external force (T_{inp}, T_{ext}) in Eq. (6) corresponds to T_i and T_d in Eq. (9).

Since the torque that occurs due to environmental interaction, T_{ext}, can be obtained via DoBs, fingertip forces (F_{tip}) can be calculated via the related Jacobian for each finger.

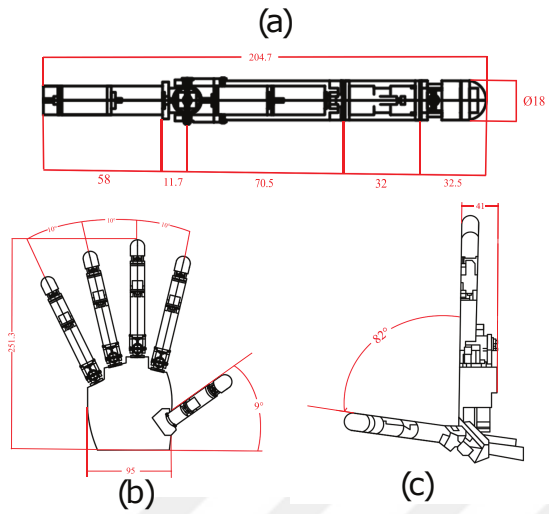

Figure 4: Dimensions of the Gifu Hand-III.

$$F_{tip} = (J^{-T})T_{ext} \tag{10}$$

If the Jacobian matrix, J, is not square, pseudo-inverse can be used.

2.2 Target Tasks and Experiment Results

2.2.1 Hardware Testbed: Gifu Hand-III

The robotic hand used in this study is the Gifu Hand III (Dainichi Co. Ltd., Japan) which consists of a thumb and four fingers (see Fig. 4) with total 16 degrees of freedom. The robot is connected to a Windows machine with several PCI boards, namely, A/D, D/A, Counter and Timer Boards. The A/D PCI cards allow the PC to read the motor currents. The Counter PCI cards are used to obtain a number of encoder clicks, i.e. joint angle changes. The D/A cards convert the PC's digital outputs to analog voltages that drive the hand motors. The controller of the robot hand runs at 500 Hz. Dimensions of the hand and a single finger is given in Fig. 4. Fig. 5 displays the actual robot while grasping two distinct objects.

Even though the robot hand is equipped with force-sensing resistors, they could

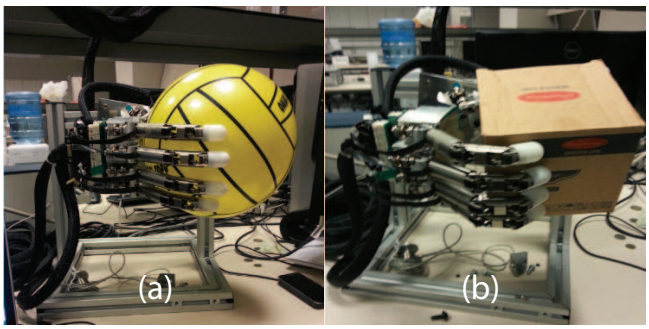

Figure 5: Gifu Hand-III graps two distinct objects. a) Plastic ball. b) Cartoon box.

not provide reliable sensory information. Therefore, we stripped off force sensing resistors from the hand and investigated an approach to estimate environmental contact force.

2.2.2 Contact Force Estimation

While running the proposed contact force estimation method, the fingers pushed a force sensor to see whether the estimation matches well with actual measurements. Fig. 6 depicts results for index and little fingers, where solid green and purple lines respectively indicate actual (sensor output) and estimated force outputs.

Scrutinizing Fig. 6, one can see that estimation results are in good agreement with actual measurements. The data sets shown in Fig. 6 are strongly correlated. Specifically, the Pearson correlation coefficient for index and little fingers are $r = .940$ and $r = .905$, respectively.

Similar results were acquired for other fingers. Although time responses differ in a sense, it is approved that the estimation algorithm can eliminate the need for tactile sensors. This greatly simplifies the hardware design and potentially reduces the costs.

2.2.3 Compliant Contact Detection

When performing a grasp in the presence of positioning inaccuracies, acquiring tactile sensing is of importance to detect early and/or unexpected contacts with objects. For

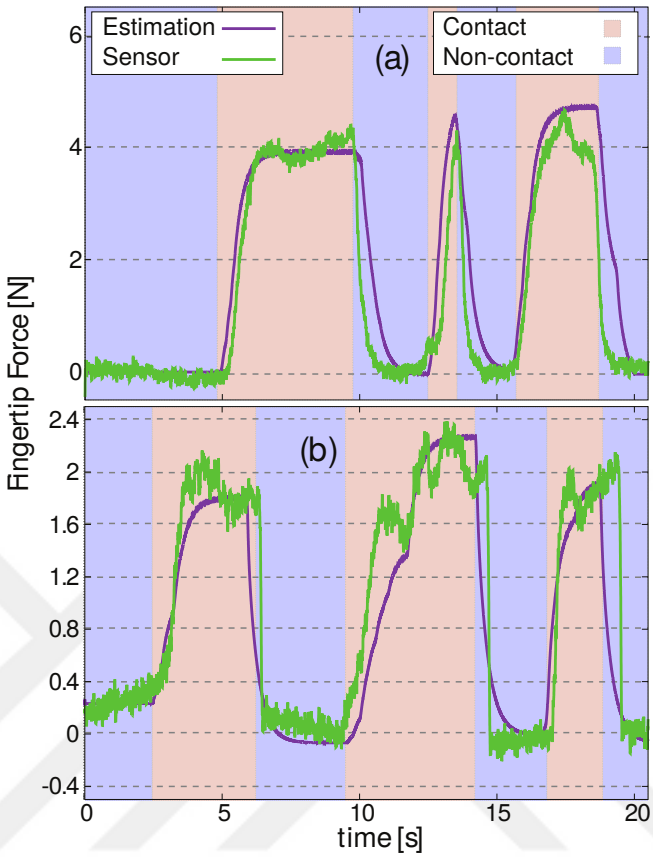

Figure 6: Fingertip force estimation is in good agreement with actual measurements. Misalignment in the figure is caused by synchronization problems between PC connected to the force sensor and the robot controller PC. a) Index fingertip force. b) Little fingertip force.

instance, Chen et al. utilized integrated joint torque sensors to detect early contacts for a compliant grasp with position uncertainties [30].

To see whether a similar contact detection task is technically possible with no tactile or torque sensors, we designated a task in which the robot hand fingers unexpectedly contacts with a rigid object. To assure compliant contact, the admittance controller in Fig. 7 was implemented. This controller uses a PD controller for the position control loop. On top of this controller, a force control loop is constructed using an admittance control scheme. In this controller, force estimator is used by means of fingertip force measurement.

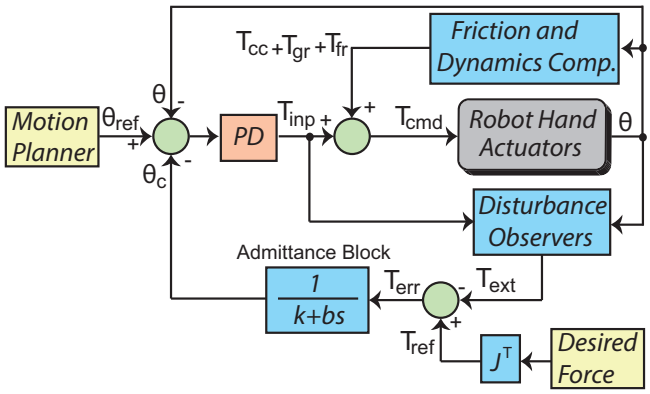

Figure 7: Admittance control was implemented to assure compliant contact. In this control scheme, both position and force control loops act on the actuator in a synergistic manner. Force control loops produces θ_c and updates the position reference so as to comply with the force constraint compliantly.

The force control loop processes the joint torque error. It is then inputted to an admittance block to compute the corresponding joint displacement θ_c. In other words, θ_c is the joint displacement that corresponds to the force error. When the system experiences force errors due to unexpected contact with the environment, θ_c updates the position reference $\theta_{ref} := \theta_{ref} - \theta_c$. The sensitivity to force error can be adjusted by the admittance parameters, namely, k and b.

In case there is no force error, θ_c simply becomes zero. Then the system becomes equivalent to a classical PD-based position controlled robot. Note that compensation scheme in Fig. 1 is enabled at all times to refine the controller performance.

In order to validate this approach, experiments were conducted when the proposed admittance control was activated. Only-PD control experiments are also conducted to provide a frame of reference for our results. Exemplary experiment results can be observed in Fig. 8 and Fig. 9 from index finger middle joint, and little finger base joint. Similar results were acquired from other joints, hence, not included.

Around $t = 0.4$, the robot hand fingers unexpectedly contacted with the object in such a way that the robot hand motion was restrained, i.e., it could not move.

In Fig. 8(a) and Fig. 9(a), dotted blue and solid purple lines indicate reference

and actual joint angles when the robot was controlled using the PD controller. As the robot hand motion was restrained after the contact, the actual angles stayed constant, and therefore, could not track the reference signals.

Dotted cyan and solid green lines stand for reference and actual joint angles when the robot was controlled using the proposed admittance controller. In this case, reference angles were updated so as to comply with the force constraints; the joints maintained their positions after the contact.

The current measurements are displayed in Fig. 8(b) and Fig. 9(b). Orange and red signals indicate current sensor readings for the proposed admittance and classical PD controllers, respectively. It should be noted that desired grasp behavior is achieved by proper admittance parameter selection. While Fig. 8(b) shows a soft and damped touch for an end-joint, a base joint in Fig. 9(b) has a less sensitive behavior which overshoots and oscillates before settling. Since the classical PD controller does not incorporate force constraint, the motors consume maximum current in an attempt to pierce through the object to follow the reference signals. As a result, the motor drives halted due to overcurrent protection. Grasp stability can further be improved by tuning admittance parameters.

The proposed controller simultaneously process force and position constraints in a compliant way; thus, the reference signal is automatically updated to cope with unexpected contact with the object. By the virtue of this approach, current consumption was well contained and the robot hand maintained its position. Compared to the case with PD controller, current consumption was decreased 3 to 5 times.

In this task, our force estimation algorithm played a major role as it facilitated the contact force information. As mentioned, the main contribution presented in this task is not utilizing an admittance controller, but rather showing that a task such as compliant contact detection or compliant grasping can be realized with the proposed force estimation method removing the need for extra sensors.

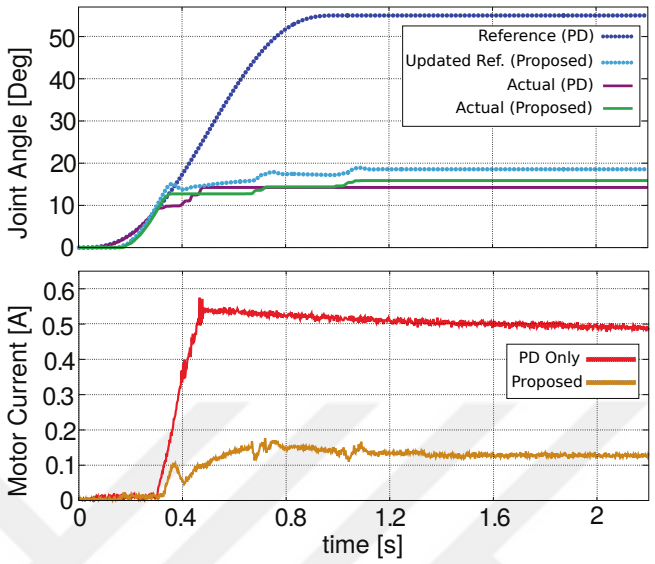

Figure 8: Compliant contact detection experiment results for the index finger, middle joint. a) Joint angle references and measurements. b) Current measurements.

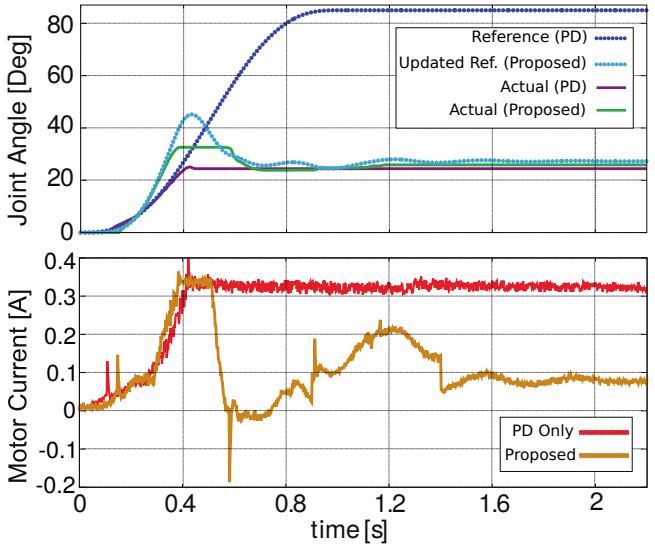

Figure 9: Compliant contact detection experiment results for the little finger, base joint. a) Joint angle references and measurements. b) Current measurements.

CHAPTER III

HUMAN CONTROL OF ROBOT ARM

Providing an initial movement to a robot directly often becomes a time consuming menial hand-tuning with a lot of trial and error. It becomes especially difficult when the robot is multi-DoF and the task is dynamic. In such setups, movement imitation often becomes the preferred way of transferring the human skill to a robot. For compliant robots, kinesthetic teach-in is used in numerous works such as in [31, 32, 33]. In our setup, the robot arm is not compliant, therefore we used teleoperation to guide the robot movement. Human control is tested on both joint space and end effector space. Control in end effector found to be more suitable.

3.1 Hardware

For the teleoperation task, we used an OptiTrack optical tracker system and a Kuka robotic arm and GifuHand III robotic hand. Two computers are also used, one for each. The robot hand is not teleoperated and integrated later. Relevant technical information for the tracker and the arm are given in following subsections.

3.1.1 OptiTrack

For human control, an Optitrack S250:E model optical tracker system is used to track the poses of tracked objects[34]. It is a low-latency infrared camera system with 5 cameras which captures and streams the frames to a PC for processing. As the standard practice goes, rounded silver markers are placed on objects of interests to capture their position. Three or more such markers are arranged planarly so that orientation can also be calculated. In the rest of this thesis, the term 'marker' is used as a shorthand for 3-marker rigid body, 'single marker' is used otherwise.

As software, a program named Motive and an SDK named NatNetSDK is provided. Motive software and the custom client -which uses the SDK- works in a server-client model. Motive is used for calibration, marker tracking, and data broadcasting. Camera parameters such as FPS, exposure and threshold reflectivity for binarization are also configured through the program. The image resolution is 832x832. The SDK is a C library, with available bindings for C, C++, Matlab and some .NET languages. In this work, we used the maximum 250 FPS, hand-tuned the exposure and threshold values according to the illumination of the environment. We used a native C++ client to process the data.

3.1.2 Kuka R900 Agilus

An Industrial type 6-DoF robotics arm, Kuka R900 Agilus sixx[35], is used as the target plant for teleoperation. Its joint configuration sequence from the base is RPPRPR (R meaning roll, P meaning pitch). As in most industrial type robots, by default, movements are preprogrammed and run in an open loop. Its internal controller logic and trajectory planner are not accessible or reprogrammable. It is controllable in position control and velocity control modes, torque-control is hardware locked and not available. The robot is controllable in joint space, configuration space of the end-effector and configuration space of a predefined tool.

To create online behavior an extra software package called RSI(remote sensor interface) is provided by the vendor, which enables users to execute online trajectories as long as torque, velocity and position are within limits. In this interface, external inputs are expected as either target position or target velocity.

Joint axes of the robot are shown in Figure 10 for reference.

3.1.3 Computers

Three computers are used for the system. The first computer is used as the robot arm controller which is connected to the arm with a cross-ethernet cable and connected to

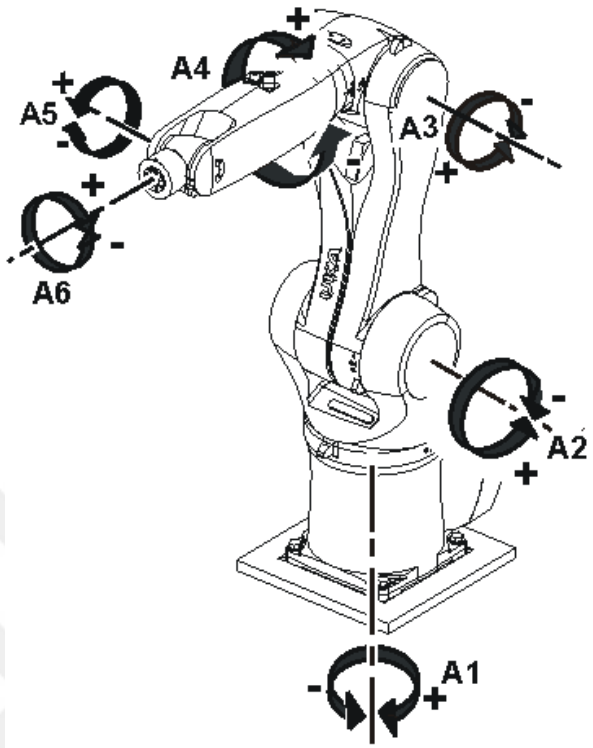

Figure 10: Joint axes of KUKA R900-sixx AGILUS robot.

the optical tracker computer and robot hand controller via the local network. It has 2.5 GHz Intel Core i5-3210 CPU, 3.5 GB memory, and runs on Linux. The second computer runs the optical tracker program Motive and the custom client performing the coordinate transformation. It has 2.53 GHz Intel Xeon-5649 CPU, 12 GB memory, and runs on Windows 7. The third computer controls the robot hand. Integration of the hand is explained later in chapter 4 It has 3.07 GHz Intel Xeon-X5675 CPU, 3.48 GB usable memory, and runs on a 32-bit Windows 7. OS choice (and therefore the memory limit) was dictated by robot hand encoder drivers.

3.2 Optical Tracker Setup

The global frame of the cameras is initialized to the ground plane in calibration stage. Axes are shown in Figure 10. To track hand pose, two markers are used which are called base marker and hand marker in the rest of the thesis. Base marker is placed

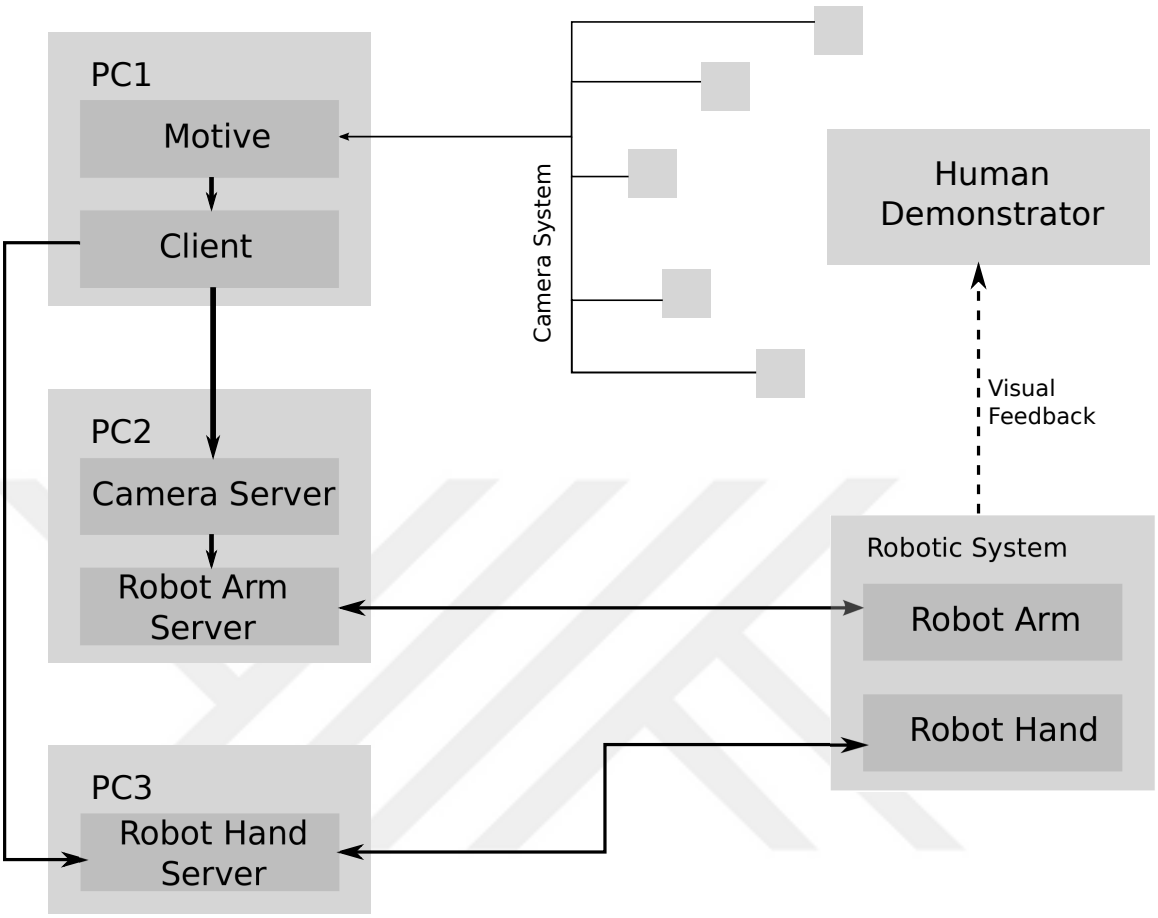

Figure 11: General communication diagram for the experiments. The robot hand is augmented to the system for the tasks explaine in chapter 4.

on an immobile table and used as base reference and hand marker is fastened to the back of the demonstrator's hand.

Although the hand marker alone is sufficient to track the pose of the demonstrator's hand, the base marker is still necessary to have continuous Euler angles in a well-defined workspace which roughly corresponds to camera-visible poses of the hand marker.

As a side note, base marker was initially planned to be used as torso frame and second marker to be attached to human hand. The scenario was that when demonstrator moves or rotates while keeping relative pose of their hand the same, the robot

should stand still. Although it works as desired when the markers are visible to cameras, allowing the demonstrator to move, often makes tracking harder due to markers getting occluded. It is also comparatively better to fix base marker to a stationary base since human torso slightly moves and creates noticeable noise. The setup explained below and the calculations reflect this case.

3.3 Pose Tracking and Coordinate Mapping

The demonstration is performed as follows. The demonstrator is instructed to initialize their upper arm parallel to the body, elbow angle making a straight angle and back of the hand -where the marker is attached- facing cameras. In robot's home configuration, orientation is matched to demonstrator's orientation to make the human control feel natural, the position is based on the reachable positions of the human hand. For example, if the demonstrator can reach 50 cm in a certain direction, the robot also needs to be able to reach. Positional configuration is not very strict since the demonstrator observes the robot during operation, however orientational disparity between the robot end-effector and demonstrator hand may create a slight sense of disorientation. This is not studied as it is out of scope, however, anecdotally the demonstrators are observed to be not aware of this effect to some extent and felt disoriented when the disparity is above a certain threshold. Setting the robot's initial joint state to a similar state as of demonstrator's also prevents the robot getting close to singular configurations. Technically this may not be a very sound solution, but it could be argued that singular poses of the robot and the human arm mostly overlaps due to them having almost same joint sequences and similar proportions of link lengths. A scaling factor for target position would be necessary for this to work if the robot had vastly different link lengths. In practice, no significant problem was encountered with such a direct mapping approach.

In equations below, Q_b and Q_h denotes ground-to-base marker rotation and ground-to-hand marker transformation; P_b and P_h denotes the positions of base and hand markers. Where used, subscript t_0 denotes the value at the start of demonstration which is initial configuration. These values are directly streamed from the server program in quaternions, therefore they are denoted with Q rather than R. The value ${}^{b}_{h}Q_t$ is calculated with the following equations.

At $t = 0$, $Q_{b(t_0)}$, $Q_{h_(t_0)}$, $P_{b(t_0)}$ and $P_{h(t_0)}$ are captured. Positional offset between base and hand markers is simply calculated as $P_{offset} = P_{h(t_0)} - P_{b(t_0)}$. Rotational offset is calculated as $Q_{offset} = Q_{h(t_0)}/Q_{b(t_0)}$, where division is the right division. In the following frames, the target position P_{target} and target rotation Q_{target} is calculated by

$$P_{target} = P_h - P_b - P_{offset}$$

$$Q_{target} = (Q_h/Q_b)/Q_{offset}$$

Q_{target} is then converted to Euler angles with ZYX ordering with right-handed coordinates. Since mapping between quaternions and Euler angles is not one-to-one, X-Y-Z angles are restricted to the conventional intervals $[-\pi, \pi]$, $[-\pi/2, \pi/2]$, $[-\pi, \pi]$, respectively. To map the target reference from camera frame to robot frame, following transformations are also performed where E_{target} is the Euler angle representation of Q_{target}, P and E are 3 element column matrices.

$$R = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$P_{target} = R \cdot P_{target}$$

$$E_{target} = R \cdot E_{target}$$

3.4 Configuration of the Arm

The mechanical and electronic switches of the robot along with the computer running its software reside in a cabinet called KRC(Kuka Robot Controller). The computer runs two operating systems in parallel, a Windows XP Embedded OS for user interface, and a real-time VxWorks OS for robot control. RSI package, in specific, allows direct communication with the controller. Following steps were Configuration of communication with a custom external controller goes through these steps.

- An XML message configuration file is created. In this file input message fields and output message fields are declared. Protocol, server IP, and port are also declared in this file.
- A configuration diagram for the controller is created. The diagram is designed through a program named RSIVisual and resembles circuit diagrams. In execution, values from input messages are passed to special blocks such as positional correction blocks or the correction blocks. Outputs that can be received are joint positions, end-effector pose, and motor currents of joints. Other parameters -which includes workspace limits, output units and precision, port number, of dropped packets before timeout- are also configured in this file.
- These two files are copied to a specified folder in KRC Windows machine.
- A KRL(Kuka Robot Language) script is written. It facilitates communication loop with an external UDP server, which is the custom controller. Whether to run in absolute or relative mode is declared in this script. Initialization of robot's home position is also performed here for practical reasons.
- KRC cabinet is connected to a server PC through a crossover cable.
- A custom controller, server program, is implemented.

3.5 Control of the Arm

In this section, implemented custom controllers are explained. As mentioned in the previous section, the controller runs as a server program. It listens to the robot's output through a UDP connection and sends back a target position/velocity as input in a control loop.

Before implementing the controllers, we experimented with simple point-to-point trajectories which use the robot's internal controller. With internal controller, running smooth trajectories, such as composition of sine trajectories and minimum jerk trajectories, work exceptionally well with near-zero tracking error in both joint space and cartesian space. In each cycle, a new target position is given to the robot, and the built-in controller guarantees that it the by next cycle. In a sense, it can be thought of as robot's built-in trajectory planner is overridden by a custom planner. However, when the trajectory is not very smooth, it creates very jerky motion and often causes the robot to halt.

If the server program fails to send a response, the robot will produce a timeout error and will halt. The target must be realizable by the robot's internal controller in one control cycle, otherwise, it will result in a torque limit error or velocity limit error or workspace limit error. Except for the workspace limit error(which is set in the configuration), other limits are dynamic and undocumented. For instance, velocity limits for going up and down for the third joint is different, since the required torque is dependent on the state of other joints.

We experimented with different PD controllers; in joint space and in cartesian space, crossed with absolute position control and relative position control. In absolute control mode targets are passed as absolute targets to the robot, in relative mode relative updates with respect to the previous timestep is passed, which can also be considered as discrete velocity. For this reason, absolute position control required extra acceleration and velocity limiters to run without jerks and errors. Relative position

control, on the other hand, proved to be more useful for all use cases. Therefore, relative controller is used in both joint space and cartesian space. Also to note, cartesian space is used for teleoperation and joint space is used for autonomous execution and replaying recorded session. An exponential filtering is also applied during teleoperation to eliminate the effects of human-caused sudden movements and tracking-caused noise. This filtering is not needed and used in the autonomous execution.

The final controller used is a straightforward one. General control loop logic including gain values is given below.

Algorithm 1 Control logic.

$sock\_robot \leftarrow$ init_robot_connection()
$sock\_camera \leftarrow$ init_camera_connection()
$s \leftarrow$ init_home_state()
$s_{ref} \leftarrow s$
$g \leftarrow s$
if *mode* **is** $JOINT$ **then**
    $K_p \leftarrow 0.03 \cdot [1, 1, 1, 1, 1, 1]$
    $K_d \leftarrow 0.0005 \cdot [1, 1, 2, 2, 2, 2]$
if *mode* **is** $CARTESIAN$ **then**
    $K_p \leftarrow 0.04 \cdot [1, 1, 1, 0.8, 0.8, 0.8]$
    $K_d \leftarrow 0.005 \cdot [1, 1, 2, 1.5, 1.5, 1.5]$
while listen($[sock\_kuka, sock\_camera]$) **do**
    if data_available($sock\_camera$) **then**
        $g \leftarrow$ read_goal_state($sock\_camera$)
    if data_available($sock\_robot$) **then**
        $s \leftarrow$ read_state($sock\_robot$)
        $e_t \leftarrow g - s$
        $s_{ref} \leftarrow e_t \cdot K_p - (e_t - e_{t-1}) \cdot K_d$
        if $filtering$ **is** *enabled* **then**
            $s_{ref} \leftarrow s_{ref} * 0.7 + s_{ref(t-1)} * 0.3$
        send_reference_state($sock\_robot, s_{ref}$)

Two simple sine reference are sent to all joints to observe tracking delay and tracking error in joint space control. The results are shown in Figure 12. For clarification, in both graphs, A refers to joints $A1$ to $A6$ but they may not be visible since same reference is used for all of them. In the first reference amplitude is modulated and in

the second frequency is modulated. The system has about 120ms of delay. Therefor error with respect to 120ms-shifted position is also shown(red lines in the graphs).

The simple sine references cannot be tested directly in cartesian-space control due to the robot's singular configurations. Therefore, a sample execution of the controller is shown in Figure 13. It is also more representaative of the actual use. In Figure 13, references and actual values for position and orientation is given on the left and tracking errors are given on the right. The input is filtered aggressively in order to ensure hardware safety. In effect, this causes flatter velocity and acceleration profiles at the expense of some delay.

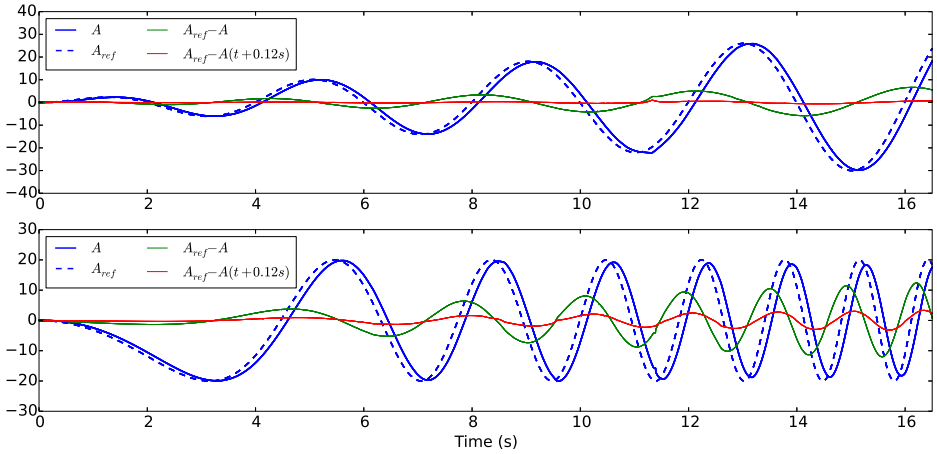

Figure 12: Two reference signals are executed for all 6 joints. Green line is the actual tracking error, red line is the tracking error shifted by 120 ms delay. (top) An increasing amplitude sine reference. (bottom) An increasing frequency sine reference.

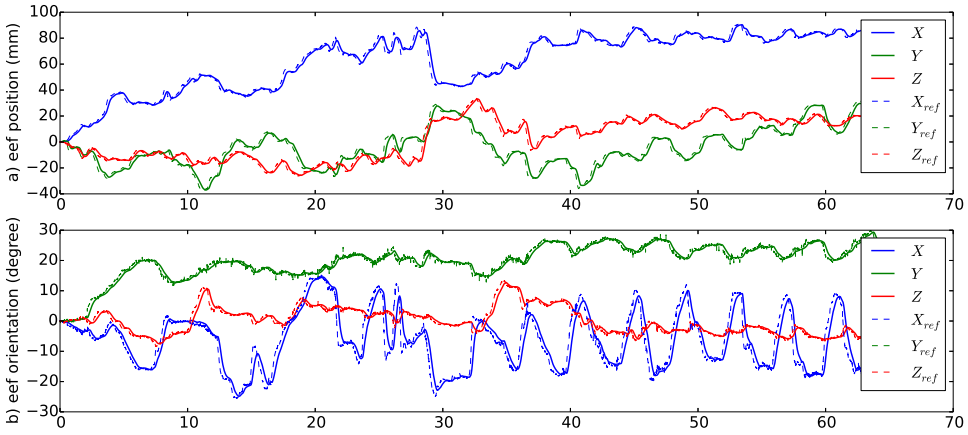

Figure 13: Human control of the arm.

CHAPTER IV

HUMAN-IN-THE-LOOP LEARNING OF A DEXTEROUS TASK

In this chapter, human-in-the-loop control with heterogeneous control is examined for dexterous manipulation. Heterogeneous control is a type of shared control in which the control channels of the human and the robot are orthogonal. The robot is initially given a basic control policy for a subset of parameters, which is insufficient to perform a given task but close enough to a successful policy. To complement the policy, the human demonstrator is expected to learn the remaining parameters to achieve the task. After the human learning is completed, initial robot policy parameters and the parameters learned by the human are aggregated to achieve full autonomy.

For demonstration, a ball swapping task[36] is chosen, which requires a robot hand to swap the positions of a pair of balls. A still from the task is shown in Figure 14. The experimental setup consists of the human controlled robot arm and an autonomous robot hand attached to it. Joints of robot fingers are given sine trajectories with a 4 second period and constant phase differences between consecutive fingers. The hand alone completely fails to perform the task and needs additional arm movement to perform it. Although initially not clear, operator can learn how to control the arm to achieve the task. Combining the successful performance of the human for the robot arm and the autonomous robot hand resulted in a fully autonomous ball swapping.

The human control part is executed the same as in chapter 3. The only addition is, to synchronize the start times of the hand and arm, the hand is also sent a start signal from the optical tracker cameras.

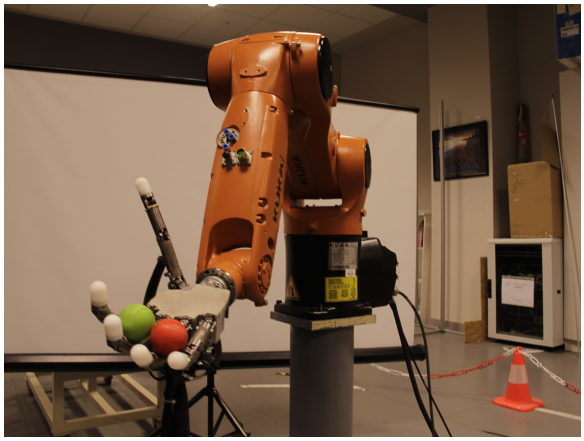

Figure 14: Ball swapping task setup.

4.1 Robot Hand Joint Trajectories

The main goal in the selection the finger joint trajectories was to pick as simple trajectories as possible. This is sensible since the goal is to not having to fine-tune the parameters. Therefore we chose a basic convex position for the hand's home position and chose sine waves as trajectories. Yaw joints of the fingers are disabled and phases of two pitch joints in fingers are set to the same value for further simplification. The thumb is also not used since it was not needed for the task. The trajectories $f(t)$ along with their phases ϕ and amplitudes A are given below. Indices the as defined in chapter 2.

$$f_i(t) = A_i sin(2\pi\omega t + \phi_i)$$

$$\phi_{0,3,6,9} = 0, \quad \phi_{1,2} = 0.2\pi, \quad \phi_{4,5} = 0.6\pi, \quad \phi_{7,8} = \pi, \quad \phi_{10,11} = 1.4\pi$$

$$A_{0,3,6,9} = 0, \quad A_{1,4,7,10} = 30, \quad A_{2,5,8,11} = 20$$

$$\omega = 0.25$$

4.2 Human Learning of Arm Movement

A sample learning session of a human is shown in Fig. 15. The control is performed in cartesian space and the data shown in the figure are the recorded joint positions (not reference data). The user in this case is not a naive user and have tried the task a few times priorly. These previous sessions were mostly used for gain tuning and finding a suitable home position. As can be noticed from the rhythmic cycles, the human learns how to perform the task around $t = 150s$. Once learned, the human can consecutively repeat it a number of times until dropping the balls around $t = 195s$. The balls are placed back again around $t = 210s$ and the human continues to execute the task successfully.

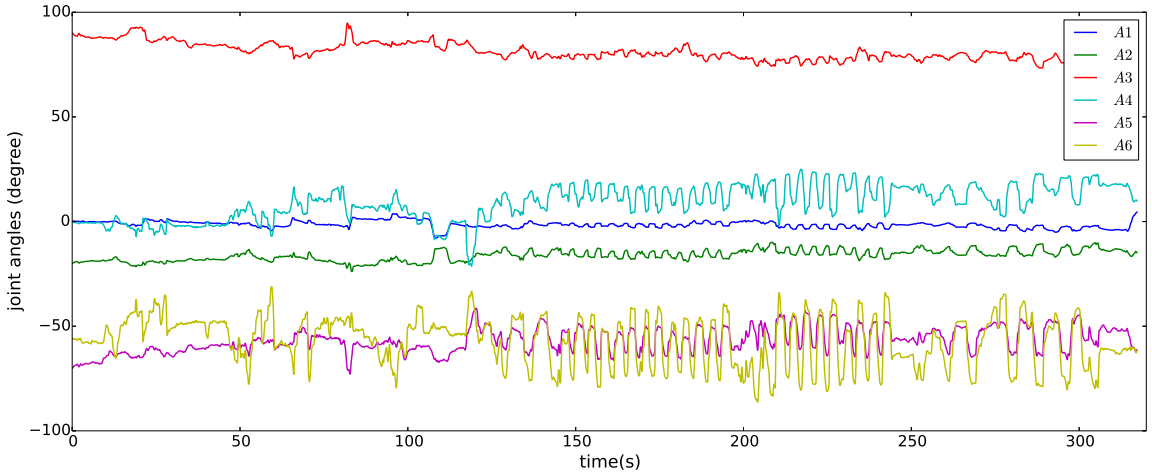

Figure 15: Recorded joint angle data while the demonstrator learns and performs the tasks.

During human learning, the state of the robot is recorded. After that, the movement is broken into 4-second cycles and each one is replayed in a loop to find the successful trajectories. Since the start state and the end state are not continuous an extra second is added to reset from end state to start state for looping. The hand is paused for this duration and arm movement is linearly interpolated between end to start. Most successful cycles are decided with a simple procedure. First, successful

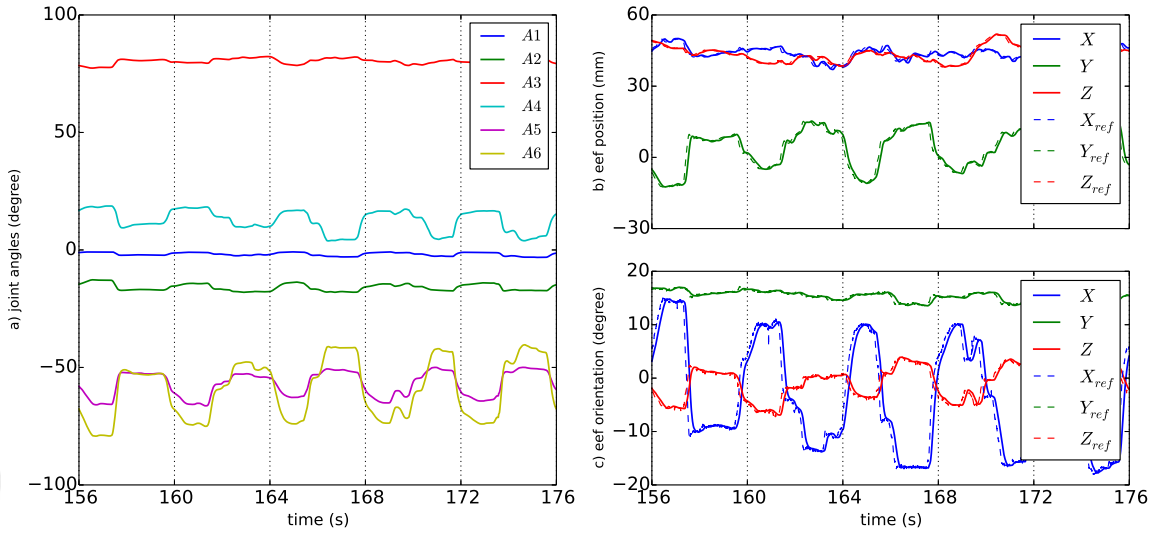

Figure 16: a) Joint angles of the robot during demonstration between t=[156, 176]. This is the same data shown in Figure 15, b) Position of end-effector in millimeters. c) Orientation of end-effector in degrees with Euler-XYZ ordering.

cycles are found by executing each cycle 10 times, failing cycles are eliminated. Then, the cycle with the flattest joint space acceleration is selected as the best one.

For clarity, a zoomed version of Figure 15 is shown for $t = [156, 176]$ in Figure 16. The best cycle in joint space according to the criteria was $t = [164, 168]$. Further processes are performed on this specific trajectory.

4.3 Reduction of Dimensionality with PCA

In human learning experiments, one common observation was that the learns the task by reducing their movements to a few specific movements. Although all 6 DOFs are available, it is logical to expect that the task is easier to perform by mostly changing the orientation with little change in position and the experimenter behavior demonstrates this. In this light, it is plausible to think that the main intention is to move within a few principal axes and the changes in other axes are mostly noise. With this in mind, removing the redundant axes can greatly improve the task performance. Another motivation is that our main goal is finding a simple successful

movement which is hard to fine manually but easy to extract from a human. Also, fewer parameters are always a plus for any kind of movement.

Principal component analysis is a very standard process for this purpose and we used linear PCA. The method is applied in joint space on extracted data and surprisingly first component alone was able to correspond to 99.2% of the variance and execute the task successfully. In Figure 17, some PC's are barely used in the movement, therefore found redundant.

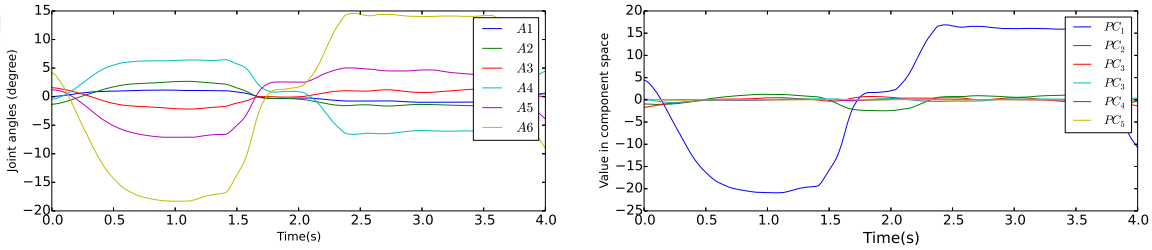

Figure 17: (left) Mean-shifted joint trajectories. (right) Trajectory encoded in the component space.

Additionally, we applied the method also on the reference data recorded from the user. Although it was fairly good it has two problems. One problem is that there need to be at least two PCs (for position and orientation) or an arbitrary scaling parameter is needed to normalize the data. Second, the Euler angles are not linear and this creates small artifacts in reconstructed data. Therefore, the method is used in joint space. Lastly, kernel PCA with RBF kernels is also evaluated and we found that linear PCA performs better for this specific data unless a great number of kernels used.

4.4 Representation of the Trajectory with Movement Primitives

In most learning-by-demonstration setups one common step is the parametrization of the movement. The main reason for parametrization is that generally not repeating

the movement strictly but capturing a general model of a baseline behavior. When a modification is desired, the behavior is later modified using only these parameters, hence reduces the complexity to a great degree.

The are various existing methods to learn a demonstrated trajectory. One very common approach is using splines, where the trajectory is fit to a combination of a number of splines. The number of splines and their viapoints are decided by different heuristics[37, 38]. The success is heavily dependent on these heuristics which are often difficult to find. Another problem is that it is usually a poor fit for nonlinear functions [39]. A similar method to overcome the nonlinearity problem is using Gaussian kernel regression. However, the main problem with this approach is its time dependency, which makes temporal perturbations difficult. One successful method global stability guarantees by Khansari-Zadeh et al. learns the attractor patterns in state-space from demonstrations with Gaussian mixture models [39, 40]. This method could have been used although we did not.

We have used dynamic movement primitives(DMPs) because it allows encoding rhythmic movements and probably the most established method for rhythmic trajectories. The method is briefly presented here. For a thoroughly detailed exposition please refer to [41, 42, 43].

DMP method consists of three main parts, which are, canonical system, transformation system, and a nonlinear function approximator(also called forcing term). Main transformation system is defined as in Eq. 11. The model is a damped spring model where y and $\dot{y}$ are desired position and velocity, g is the goal state, f is a nonlinear forcing term, τ is time constant, α_z and β_z are positive constants. With the proper selection of α_z and β_z and by setting the forcing term $f = 0$, the model is a stable linear dynamical system with the attractor state g.

$$\tau\ddot{y} = \alpha_z(\beta_z(g - y) - \dot{y}) + f \tag{11}$$

The main idea is to introduce a nonlinear forcing term to exhibit arbitrarily complex trajectories until reaching the goal state. However, direct modification of the Eq. 11 makes the system nonlinear. To solve the problem a canonical equation is introduced and f is defined as a function of phase. The addition of the canonical system removes the need for explicit time coupling in the control policy and allows for adaptive and reactive movements to emerge. Compared to discrete version, in rhythmic version canonical equation is simple and linear. It is usually chosen as in Eq. 13 where ϕ is the phase. Forcing term is defined as the normalized weighted sum of N activation function ψ, weighted by w so that it becomes linear. Gaussian basis functions are one of the most common general function approximator and activation functions ψ are defined as Gaussians with centers c_i and bandwidth h_i. In rhythmic primitives, cosine of the distance to the center is used to activate the function in every cycle. The centers are equally spaced in phase because the phase function is linear they are also equally spaced in time which is not the case in discrete form.

$$\tau\dot{\phi} = 1 \tag{12}$$

$$f = \frac{\sum_{i=1}^{N} \psi_i w_i}{\sum_{i=1}^{N} \psi_i}, \text{ where } \psi_i = \exp(-h_i cos(\phi - c_i)) \tag{13}$$

Weights of the activations are often simply called parameters and their values are found with a regression method. Locally weighted regression(LWR) is often used with dynamic motor primitives although any other suitable regression method can be used. LWR is a common used local nonparametric memory-based modeling method[44, 45]. In parametric models, data is fit to a function with predefined number of parameters and these parameters are found using all data points, hence global. In comparison, nonparametric models build local models which fit a variable number of parameters to a subset of data points. Therefore, LWR and its variations are often used in

robotics for its efficiency to estimate local linear models and its suitability for real-time control[46, 47].

We used the method with the open parameters: $\alpha_z = 50$, $\beta_z = 12.5$, $c_i = 2\pi i/N$, $h_i = 1$. τ is used as $4/2\pi$ in fitting phase as it is the movement period. Trajectory used here is the first principal component found in the previous section, the method would be applied to other components in the same manner independently had they been used.

In Figure 18, input trajectory and learned trajectory are shown with the velocity profile. Two consecutive cycles are shown to display the discontinuity at cycle ends, which is removed in the learned trajectory due to its guarantees. In Figure 19 phase, activation functions and learned weights are shown. In Figure 20 learned dynamical system is executed with different time constants. Although the graph gives the wrong impression that it is a linear timescaling, it is not. Good selection of open parameters provides convergence guarantee and critical damping no matter how small τ is.

As the final step learned trajectory is mapped back to joint space by applying inverse-PCA and used as it is. The resulting movement was simple and performed the task successfully. The execution could be sped up by a factor of 2.1 above which the robustness starts to degrade.

The simple movement is difficult to create by hand as the axis is quite arbitrary and the trajectory is hard to guess. In a dynamic task such as this, finding a hand-coded trajectory would take a significant amount of time because the balls can drop often. Therefore, the benefits of the human control are clear. On the other hand, in the human learning sessions, humans failed to learn the task at higher speeds. The procedure was able to achieve this improvement by finding a better version of the human provided policy.

One glaring fact is that aside from temporal scaling additional benefits of the DMP method -such as spatial scaling, spatial or temporal disturbance handling- are not

employed. Still, having the critical damping is exteremely important and cannot be satisfied with more direct methods, such as splines. Another reason for using DMPs is that no other method creates parameters in phase space Since the phase(rather than state) is the most important aspect of the the task, it will be highly useful in follow-up works.

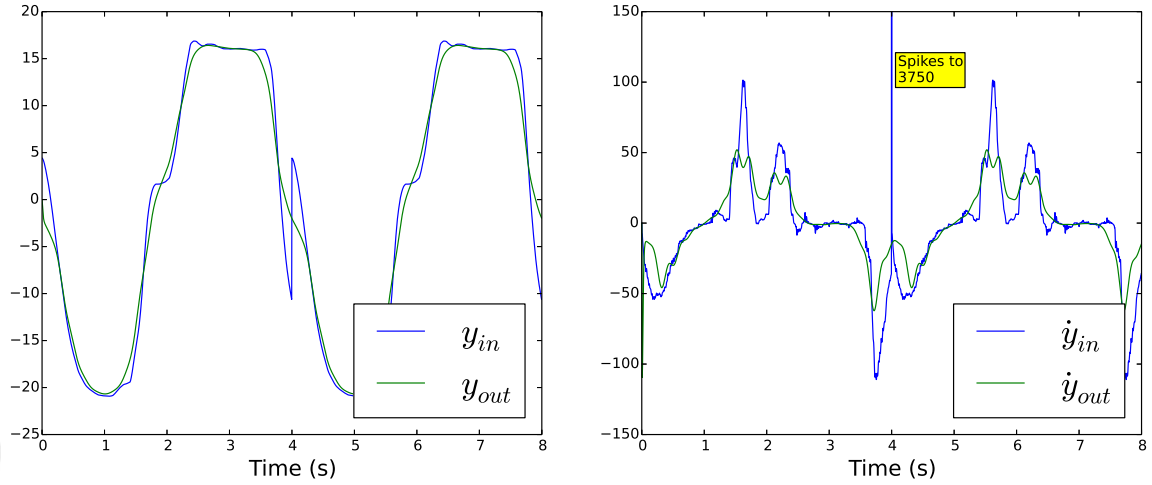

Figure 18: Reference and learned trajectories and their derivatives.

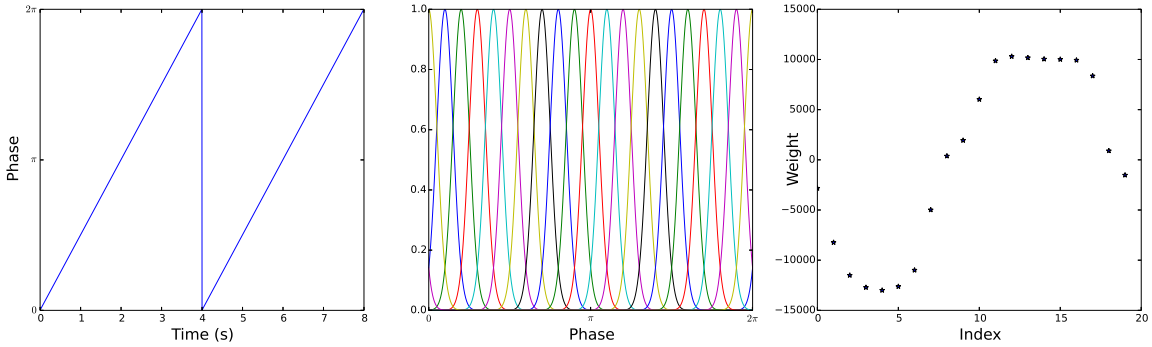

Figure 19: Phase, activation functions and weights

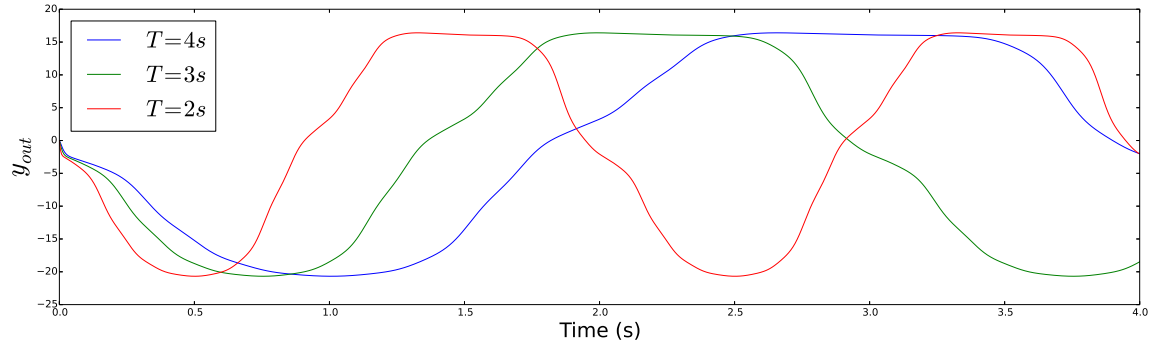

Figure 20: Learned dynamical system run with different periods to perform.

CHAPTER V

CONCLUSION

To conclude, in this work, we investigated manipulation strategies on two experimental setups. In the first experiment, we proposed that with a good dynamic model of a robot hand, stable grasps with variable stiffness can be executed by estimating environmental contact forces on fingertips. In the method, forces exerted on the fingertips are measured with a force sensor and found consistent with the estimations. To ensure compliant contacts, a variable stiffness admittance controller is implemented. The controller performance was found suitable for cylindrical grasp tasks for stiff and soft power grasps.

The limitations and the problems of the approach are also clarified. The most important limitation is that the approach is not very suitable for precision grasps as expected, due to low sensory precision. Another limitation is that dynamic model accuracy is naturally affected by mechanism wear. Mechanical backlashes on the fingers have a negative effect specifically on stiffness variations.

We conclude that the proposed simple method is a good alternative to using pressure sensors in certain scenarios for the reasons of lower complexity and reduced cost.

In ball swapping task, we also found many interesting problems specific to dynamic and dexterous tasks. One problem is that many policy search reinforcement learning methods used in robotics, (e.g. REPS[48], PI^2[49], PoWER[50]) aim to optimize weight parameters without changing phase, even if the task might be more suitable with a modified phase. In not very dynamic tasks, such as commonly benchmarked reaching task, optimal parameters are independent of execution speed and it does not

create any problems. However, in dynamic tasks the assumption becomes counter-intuitive. For example, increasing the cycle duration of the ball swapping task without modifying other parameters would fail the task if the human learned the policy makes use of the inertia of the balls. The opposite problem occurs for making the task faster. How the human adapts to the task is often cannot be qualitatively answered even by the person. Although, the selected task might seem irrelevant for real-world use cases, similar tasks such as bipedal walking or carrying a liquid-filled glass exhibit the same behavior. A naive approach for learning the faster movement would also require the human to learn the faster movement, which is not a solution since the learning phase would be considerably more difficult for humans. In our experiment, this was the case and the human failed to learn the task at lower periods. In reinforcement learning scenarios for this type of dexterous tasks, we believe the main improvement would come with learning the phase modulation and the weight parameters would be of secondary importance. In its most simple form, a representative example task would be learning the optimal policy for rotating a cart-pole as fast as possible or as slow as possible.

To the best of our knowledge, this type of temporal task improvement is not well-studied and we intend to investigate the problem in a future work. To this end, we have done an amount of preliminary work which is not reported in this work due to lack of results. In this work, we created a reinforcement learning setup. Initially, we planned to use the force estimation method devised in chapter 2 to detect the ball positions, unfortunately as mentioned in its limitations we were not able to have precise estimation due to the dexterous movement and low weight of the balls. In order to have a reward signal, a regular camera is used to track the phases of the balls and a few reward functions are devised. We found that it is difficult to find a successful policy for very fast movements by only optimizing the weights of the arm movement. The problem, as mentioned previously, is that the phases and amplitudes

of the fingers start to become more important as the period of the movement changes. Therefore we decided to use the same canonical system of the arm movement with the finger movement and intend to find significant improvements by learning the phase function. The results are yet to be had and will be presented in a future work.

Another interesting observation in human learning was that after the human gets used to the controls, they focus on coming up with a task decomposition plan which is often considered a difficult problem in robotics domain. The subtasks in the ball swapping task were moving the ball from some space between two fingers to another and moving from lower finger joints to the upper joints between some two fingers. This is clearly observable from the replays of the recorded data. This can also be observed from slight pauses in the movement where the joints are not moved and the proper phase waits for the next subtask. It would be fair to say that actual learning time was mostly spent on finding a repeatable decomposed subtasks and various spontaneous plans were discovered spontaneously by the human. It is an interesting question whether there is a general principle in people governing how we evolve our task decomposition plans and if there is one could this form a better understanding for the robotics counterpart.

Bibliography

[1] R. L. Susman, "Fossil evidence for early hominid tool use," *Science*, vol. 265, no. 5178, pp. 1570–1574, 1994.

[2] T. L. Kivell, J. M. Kibii, S. E. Churchill, P. Schmid, and L. R. Berger, "Australopithecus sediba hand demonstrates mosaic evolution of locomotor and manipulative abilities," *Science*, vol. 333, no. 6048, pp. 1411–1417, 2011.

[3] J. R. Napier and R. H. Tuttle, *Hands*. Princeton University Press, 1993.

[4] R. W. Young, "Evolution of the human hand: the role of throwing and clubbing," *Journal of Anatomy*, vol. 202, no. 1, pp. 165–174, 2003.

[5] R. R. Ma and A. M. Dollar, "On dexterity and dexterous manipulation," in *Advanced Robotics (ICAR), 2011 15th International Conference on*. IEEE, 2011, pp. 1–7.

[6] C. A. Klein and B. E. Blaho, "Dexterity measures for the design and control of kinematically redundant manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 72–83, 1987.

[7] A. Bicchi, "Hand for dexterous manipulation and robust grasping: A difficult road towards simplicity," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, 2000.

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

[9] A. D'Ausilio, E. Bartoli, and L. Maffongelli, "Grasping synergies: A motor-control approach to the mirror neuron mechanism," *Physics of Life Reviews*, 2015, in press.

[10] A. Ajoudani, S. B. Godfrey, M. Catalano, G. Grioli, N. G. Tsagarakis, and A. Bicchi, "Teleimpedance control of a synergy-driven anthropomorphic hand," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, November 2013, pp. 1985–1991.

[11] M. Gabiccini, E. Farnioli, and A. Bicchi, "Grasp and manipulation analysis for synergistic underactuated hands under general loading conditions," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, St. Paul, US, May 2012, pp. 2836–2842.

[12] M. Catalano, G. Grioli, A. Serio, E. Farnioli, C. Piazza, and A. Bicchi, "Adaptive synergies for a humanoid robot hand," in *Proc. IEEE Intl. Conf. on Humanoid Robots*, Osaka, Japan, November 2012, pp. 7–14.

[13] S. Schaal, A. Ijspeert, and A.Billard, "Computational approaches to motor learning by imitation," *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, vol. 358, 1431, pp. 537–547, 2003.

[14] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[15] S. Schaal and C. G. Atkeson, "Learning control in robotics trajectory-based optimal control techniques," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[16] B. Huang, S. El-Khoury, M. Li, J. J. Bryson, and A. Billard, "Learning a real time grasping strategy," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013, pp. 593–600.

[17] M. Fischer, P. van der Smagt, and G. Hirzinger, "Learning techniques in a data-glove based telemanipulation system for the dlr hand," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Leuven, Belgium, May 1998, pp. 1603–1608.

[18] A. E. Leeper, K. Hsiao, M. Ciocarlie, L. Takayama, and D. Gossow, "Strategies for human-in-the-loop robotic grasping," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*. ACM, 2012, pp. 1–8.

[19] E. Oztop, L.-H. Lin, M. Kawato, and G. Cheng, "Dexterous skills transfer by extending human body schema to a robotic hand," in *Proc. IEEE Intl. Conf. on Humanoid Robots*, Genova, Italy, December 2006, pp. 7–12.

[20] ——, "Extensive human training for robot skill synthesis: Validation on a robotic hand," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1788–1793.

[21] J. Babič, J. G. Hale, and E. Oztop, "Human sensorimotor learning for humanoid robot skill synthesis," *Adaptive Behavior*, vol. 19, no. 4, pp. 250–263, 2011.

[22] L. Peternel and J. Babič, "Humanoid robot posture-control learning in real-time based on human sensorimotor learning ability," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5329–5334.

[23] O. Kaya, M. C. Yildirim, N. Kuzuluk, E. Cicek, O. Bebek, E. Oztop, and B. Ugurlu, "Environmental force estimation for a robotic hand: compliant contact detection," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on.* IEEE, 2015, pp. 791–796.

[24] E. Oztop, N. S. Bradley, and M. A. Arbib, "Infant grasp learning: A computational model," *Experimental Brain Research*, vol. 158, no. 4, pp. 480–503, 2004.

[25] A. Schmitz, M. Maggiali, M. Randazzo, L. Natale, and G. Metta, "A prototype fingertip with high spatial resolution pressure sensing for the robot icub," in *Proc. IEEE Intl. Conf. on Humanoid Robots*, Daejeon, Korea, December 2008, pp. 423–428.

[26] T. Murakami, F. Yu, and K. Ohnishi, "Torque sensorless control in multidegree-of-freedom manipulator," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 2, pp. 259–265, 1993.

[27] E. Sariyildiz and K. Ohnishi, "On the explicit robust force control via disturbance observer," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1581–1589, 2015.

[28] B. Ugurlu, M. Nishimura, K. Hyodo, M. Kawanishi, and T. Narikiyo, "Proof of concept for robot-aided upper limb rehabilitation using disturbance observers," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 1, pp. 110–118, 2015.

[29] L. Le Tien, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, "Friction observer and compensation for control of robots with joint torque measurement," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, 2008, pp. 3789–3795.

[30] Z. Chen, T. Wimboeck, M. A. Roa, B. Pleintinger, M. Neves, C. Ott, C. Burst, and N. Y. Lii, "An adaptive compliant multi-finger approach-to-grasp strategy for objects with position uncertainties," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Seattle, US, May 2015, pp. 4911–4918.

[31] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[32] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[33] S. Bitzer, M. Howard, and S. Vijayakumar, "Using dimensionality reduction to exploit constraints in reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 3219–3225.

[34] NaturalPoint, Inc., "Optitrack s250:e product webpage," accessed 25-April-2017. [Online]. Available: http://www.optitrack.com/support/hardware/s250e.html

[35] KUKA AG, "Kuka r900 agilus," accessed 25-April-2017. [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/kr-agilus-sixx

[36] B. Moore and E. Oztop, "Robotic grasping and manipulation through human visuomotor learning," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 441–451, 2012.

[37] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Intelligent*

Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2. IEEE, 2003, pp. 1444–1449.

[38] J. Aleotti and S. Caselli, "Robust trajectory learning and approximation for robot programming by demonstration," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 409–413, 2006.

[39] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[40] ——, "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*. IEEE, 2010, pp. 2676–2683.

[41] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, pp. 1547–1554, 2003.

[42] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," *Robotics Research*, pp. 561–572, 2005.

[43] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[44] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," *Journal of the American statistical association*, vol. 83, no. 403, pp. 596–610, 1988.

[45] W. S. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *Journal of the American statistical association*, vol. 74, no. 368, pp. 829–836, 1979.

[46] S. Schaal and C. G. Atkeson, "Robot juggling: implementation of memory-based learning," *IEEE Control Systems*, vol. 14, no. 1, pp. 57–71, 1994.

[47] C. G. Atkeson, "Using locally weighted regression for robot learning," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 958–963.

[48] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[49] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[50] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2112–2118.