# Manipulators and Manipulation in high dimensional spaces

Vikash Kumar

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Emo Todorov, Chair

Sergey Levine

Dieter Fox

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

**Abstract**

Manipulators and Manipulation in high dimensional spaces

Vikash Kumar

Chair of the Supervisory Committee:
Associate Professor Emo Todorov
Computer Science & Engineering and Applied Math

Hand manipulation is one of the most complex form of biological movements. Despite its significance in multiple fields such as biomechanics, neuroscience, robotics and graphics, our understanding of dexterous manipulation is quite superficial and far from being reproducible. The unique capabilities of the human hand have long inspired roboticist in their pursuit to develop manipulators with similar dexterity. Simple and isolated tasks such as grasping can of course be accomplished by simpler devices. Nevertheless, if robots are to perform a wider range of tasks in less structured environments than what is currently possible, they are likely to need manipulators approaching human levels of dexterity.

The primary goal of this thesis is to realize dynamic dexterous manipulation on physical hardware. The thesis makes following contributions towards this goal

- Design and development of "*ADROIT* Manipulation Platform" – a 28 degrees of freedom arm-hand system capable of hosting dexterous dynamics manipulation, thanks to the custom designed fast, strong and compliant pneumatic actuation system.

- Behavior synthesis techniques – that are capable of synthesizing the details of dynamic dexterous manipulation in high dimensional manipulators – and scalable approaches that bridge the wide divide between what's possible in simulation and what works on the physical hardware under real world conditions.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I want to express my sincere gratitude to the department of Computer Science and Engineering, University of Washington for providing a platform and an opportunity to an inquisitive mind trying to pursue some wild ideas. I want to thank my advisors Prof. Emanuel Todorov, Prof. Sergey Levine for initiating a rogue mind to the basics of doing research, to help it comprehend asking the right questions in order to find directions in unknown territories, and for providing guidance all along the way. This thesis would not have been possible without their support and belief in my crazy ideas.

Last but not the least a hug goes out to everyone who supported me along the way to make it a memorable journey.

*Collaborators*: Tom Erez, Yuval Tassa, Abhishek, Tingfan, DJ, Igor, Svet, Arun, Kendall, Joseph, Harley, Alex, Akshay, Evangelos, Aravind, Ankit, Rahul, Prasang, Peter Henry, Evan, Paul, Dan and "you"

*CSE*: Lindsay, Elise, Lorraine, Rebecca, Jennifer, Air compressor, CSsupport, Mark, Frank -the ever smiling building guard, mentors and "you"

*Institutions*: UW, IIT, Nehru Hall, Dr. Piyush Kant, DAV, Mr. L.R. Saini, Children Convent

*Special ones*: Mona, Shalini, Arunima, Mahak and "you"

*Seattle*: An army of roommates, a swarm of best buddies, my extended family, the volleyball beehive, a squad of dance partners, badminton opponents, partners in crimes, hiking–traveling–road trip buddies and "you"

# DEDICATION

to my parents

Ms. Bindu Gupta (Mummy)

&

Mr. Suresh Prasad Gupta (Pops)

This thesis, and where I'm today, is a testimony of your dreams, dedication and perseverance.

# Chapter 1
# INTRODUCTION

## *1.1 Motivation*

Recent technological advancements are aggressively pushing the state of the art in robotics, especially in the field of legged locomotion. Not only robust agile quadrupedal locomotion, but dynamic bipedal locomotion in unstructured environment have been demonstrated. As processors are getting faster, real time planning with full state space over long horizon is becoming possible, which will further improve these results. Despite the similarity of the problem between legged locomotion and hand manipulation – both involve planing trajectories for kinematic trees interacting with the environment by means of contacts – these advancements have not yet had a comparable impact on dexterous dynamic manipulation. Robotic manipulation still involves simple quasi-static movements with simple low degree-of-freedom manipulators. Most real world solutions involve custom manipulators with task specific hand-tuned solutions. Given that the design goal of robotic devices is to interact with and manipulate the world so as to solve real tasks, having one custom manipulator with carefully tuned solution per task does not scale. If robotic devices are to perform tasks of daily necessities in home environments or dangerous tasks in hostile environments, robust and universal manipulators capable of handling a variety of tasks are required.

Hand manipulation is one of the most complex form of biological movements. Despite its significance in multiple fields such as biomechanics, neuroscience, robotics and graphics, our understanding of dexterous manipulation is quite superficial and far from being reproducible. While evolution has converged on multiple solutions for locomotion, the solution for manipulation seems quite standard across different species – prehensile behavior using

opposable thumb. This likely reflects the difficulty of the problem, which is further supported by the fact that only few animal special are capable of dexterous manipulation while most can locomote.

### *1.2 Traditional approaches*

Traditional approaches toward manipulation are task specific, and consists of quasi-static movements carefully reasoned and designed by human engineers. These approaches often limit dexterity and translate into rigid grasps with force closure. In contrast, human manipulation hardly involves rigid grasps. Instead it heavily relies on complex dynamic phenomena such as rolling, stiction, deformation, pushing, making and breaking of contacts. Any deficiencies in the ability to sense these interactions cause manipulation abilities to deteriorate. Which suggests that in addition to capable hardware, a capable manipulator will likely need a feedback controller that understands and exploits these kinematic constraints and dynamic phenomena in order to deliver comparable results.

### *1.3 Biological counterpart*

According to the cortical homunculus, hand functions require over one quarter of the brain power allocated for the whole body's motor/sensory activities. Not only profound task based morphological adaptations, neuromuscular adaptations restraining the hand movements to task based synergies are also observed across all species capable of manipulation. Special neural pathways for hand behaviours that bypass the spinal circuits have also been discovered [83]. Nature's solution towards manipulation involves complex hardware with heavy computational requirements, which further emphasises the scale and difficulty of the problem under consideration. Perhaps in the domain of robotics as well, the feasible solution for agile dexterous manipulation exists as a complex combination of the hardware and the controller.

### *1.4 Thesis: General philosophy*

This thesis investigates the challenges with dexterous manipulation from multiple prospectives – hardware, controllers, model-based approaches, model-free approaches, optimization based, data driven, demonstration based approaches to name a few – to further the state of the art in realizing dynamic dexterous manipulation on hardware under real world considerations. The general philosophy with regard to the hardware is to leverage off the shelf components to the extent possible, and to opt for redesign if absolutely necessary. Hardware functionally is prioritised over the form factor. General philosophy while designing the planners and the controllers is to prioritize scalability and generalization over carefully tuned task specific solutions.

Efforts in terms of hardware have been consolidated into a highly flexible and customizable platform, geared towards hand manipulation research – the "*ADROIT* manipulation platform". The investigations and the results presented in this thesis primarily revolves around *ADROIT* platform but bears no platform specific assumptions. All the results presented in this body of work should be applicable to related problems with necessary changes.

### *1.5 Thesis: Contributions & organization*

This thesis makes significant contributions along two broad thrust directions.

1. **Design and development of the *ADROIT* manipulation platform**: *ADROIT* is 28 degree of freedom arm-hand system capable of hosting dynamic dexterous manipulation. The mechanical superiority of the *ADROIT* hand manifests from the fast, strong and compliant custom designed actuation system that powers its muscles. Specific contributions toward this thrust direction include

    (a) A capable general purpose actuation system for tendons driven mechanisms have been realised. The novelty of this system lies in the unique combination of strength, speed and force it can deliver, which makes it ideal of hand manipu-

lation research. Chapter 3 presents the design details of the *ADROIT* platform, and evaluates it mechanical capability for supporting dexterous manipulation.

(b) Pneumatic actuation is ideal for the *ADROIT* platform for various reasons – compact form factor, high force to weight ratio, mechanical simplicity, inherent compliance, resemblance to the biological muscles – to name a few. These desirable properties of the pneumatic actuators have rarely been exploited due to the inherent complexity of controlling the third order system. In Chapter 4 and Chapter 5 we present our results on taming these complexities in order to expose the highly desired but rarely exploited benefits of the pneumatic actuators.

2. **Synthesis of dynamics and dexterous hand manipulation behaviors**: *ADROIT* manipulation platform is high dimensional system capable of exhibiting complex manipulation behaviors. Synthesizing manipulation behavior for *ADROIT* involves searching through this high dimensional space full of discontinuities (due to finger-object and finger-finger contact) while exploiting various dynamic phenomenon (like rolling, sliding, deformation, stiction etc), limits and constraints. Curse of dimensionality and ill-behaved search space have long plagued different approaches from effectively generating dexterous dynamic hand manipulation strategies that scale to physical hardware. Specific contribution towards synthesising manipulation behaviours include

(a) A model predictive control based behaviour synthesis framework is presented in Chapter 6. The framework is capable of generating dynamic dexterous manipulation behaviour in real time. The framework plans with the full dynamics of the high degree of freedom manipulators and scales well across tasks.

(b) Various data driven learning based techniques have been investigated in Chapter 8 to push the boundary of dexterous manipulation on physical hardware. Viable solutions have been proposed, and validated, to bridge the gap between what is possible in the simulations and what works on the physical hardware. In Chap-

ter 10 we further extend our methods to leverage expert demonstration in order to scale to more complicated manipulation tasks while still being sample efficient. In chapter 11, we develop techniques to generalize our approach across different task variations.

# Chapter 2
# RELATED WORKS

**"Dexterous Manipulation is controlled, and dynamic maneuvering of free objects by an end effector in order to achieve desired outcomes"**

Although simple robotic grippers can be used for a wide variety of useful manipulation tasks, they cannot match the flexibility and dexterity of the human hand. The past few years have seen extensive developments in the design of anthropomorphic finger-finger hands, accompanied by impressive developments in actuation, design, and control. However, dexterous manipulation still represents one of the most challenging control problems in robotics. These challenges involve both hardware and software. On the hardware side, the form factor and complexity of a human-scale 5-finger hand presents a considerable challenge in the design of actuators and joints. On the software side, the complexity of the physical system places a considerable burden on standard model-based control methods. This challenge is further exacerbated by the fact that many of the most interesting dexterous manipulation skills are inherently non-prehensile and involve complex and rapidly changing contact dynamics. A simple gripper can either hold or drop an object, while a complex hand can maneuver it into various orientations, rotate and reposition it, or even toss it into the air and catch it. This complexity and diversity of behaviors require a control with excellent generalization and sophisticated capability to adapt to a complex and unpredictable physical environment.

There is no concrete definition of *dexterous manipulation* in the literate, some refer pushing objects via end-effector as manipulation, while others refer to moving an object from one place to another (commonly referred as *pick and place*) as manipulation. Truly dexterous manipulation is hard to define, in this thesis we will use the term "dexterous manipulation" to refer to any behavior that involves dynamic maneuvering of free (greater than 6

independent degrees of freedom) object using an end-effector in order to achieve the desired outcome. Note that we will not use the work "manipulation" as a reference to *quasi-static pick and place* movements. "**Dexterous manipulation**" will be used for dynamics maneuvers while we will resort to the term "pick and place" when referring to quasi-static reorientation maneuvers.

While there is an extensive body of literature on synthesizing grasping and quasi-static pick and place behaviors, the literature on dexterous manipulation is a little sparse. In this chapter, we will focus on reviewing the field along different directions that are relevant to this thesis.

## *2.1 Hand Designs*


Figure 2.1: End effectors designs prevalent in the industry

There is an abundance of hand design prevalent in the industry (Figure 6.1). The common

theme that emerges across all use cases is that almost all these designs are customised to their specific applications. Moving forward, the curse of custom manipulators has to break, if robots has to enter the domain of home-care, elderly care, disaster response, space and marine exploration etc. Its not that capable manipulators does exists, we haven't been successful in extracting performance out of these machines.

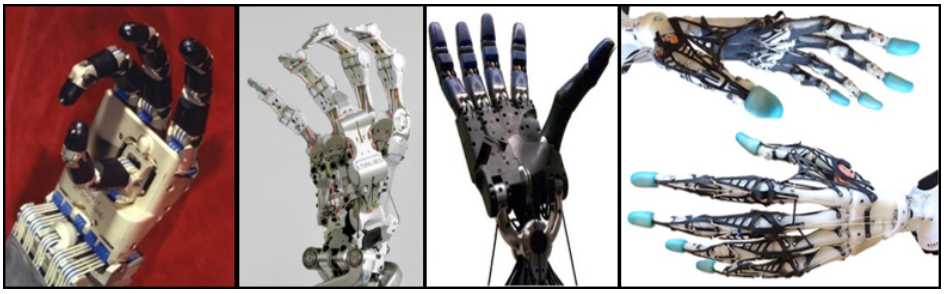


Figure 2.2: Dexterous hands. From left to right: Utah-MIT hand, DLR hand, ShadowHand, UW Hand

There exist multiple dexterous tendon-driven systems Figure 2.2 (including the Utah-MIT hand[38], the ACT hand[27], Gifu hand [69], as well as cadaver hands [21]) that are comparable to human hands kinematically, yet the lack of suitable actuation has hindered control applications. Furthermore, 3D printing technology [59] [118] [117] [28] [123] [116] [24] has made it surprisingly easy to design and build new hands (and possibly other mechanisms) with large numbers of joints and tendons attached to them. Extracting performance from these dexterous system requires a actuation system that is capable of achieving a unique rare combination of speed strength and compliance. This thesis offers a unique solution which we believe is quite universal and reflects the state-of-the-art in actuation technology. This actuation system is detailed in Chapter 3. In the next section, we present literature review for pneumatic actuation technology.

### *2.2 Pneumatic Actuation*

The requirement for high dexterity naturally leads to the choice of pneumatic actuation. Indeed this may be the only available technology that combines speed, strength and compliance on the mechanism level with small and lightweight actuators. Two types of valves have successfully been implemented in controlling pneumatic actuators, proportional valves and digital on-off Pulse Width Modulated(PWM) valves. Eric et al [10], studied the use of PWM valves and highlighted some of the advantages and disadvantages of each. PWM valves, while cheap, excludes the use of any previously well studied analytic control approaches because of the switching action of the valves. Carducci et al [18] et al implemented non linear optimization techniques to characterize PWM valves.

Although proportional valves have no such limitation, they are expensive and require extensive experimentation to characterize the flow from the valves. Pneumatic dynamics modelling for proportional valves has received significant attention in the past – parametric [122] as well as physical models [85] have been developed. Physics based mathematical model of a pneumatic system is developed from three equations, the ideal gas law, conservation of mass and the energy equation. The most popular assumption among researchers has been to neglect the temperature dynamics and consider the expansion and compression of air as a thermodynamic process with the specific heat coefficient $n$ varying from $1-1.4$, derivation of these pneumatic models is described in [11]. Researchers have successfully developed models based on assumption that the process is either isothermal ($n$=1), isentropic ($n$=1.4) or a polytropic process ($1 < n < 1.4$). Carneiro et al. [19] reviews these different approaches to model the thermodynamic process.

Richer et al. [84], reported slightly different approach to modelling the pressure dynamics by considering the expansion process as adiabatic and the compression as isothermal, hence having different specific heat coefficients for mass flow into and out of the chamber. In Gulati et al, [33] a force error based Lyapunov pressure observer was designed considering both the charging and discharging process as isothermal, In Gulati et al [34], the pressure observer

was used alongside a sliding mode controller. In both these works, the pressure estimates was tested for simple trajectories like sinusoidal and square waves. Pandian et al. [77] also presented a pressure observer and sliding mode controller, but again the observer was only tested for simple reference pressure trajectories.

Xue et al.[114] developed a controlled auto-regressive moving average (CARMA) pneumatics model based on theoretical analysis of the pressure dynamics. Tassa et al.[100] developed a non linear parametric model for the pressure dynamics based on experimental work on a humanoid robot. In most of the previous work on modelling pneumatic actuators, researchers have refrained from commenting on the model's performance with factors such as cylinder volume, cylinder leakage and fast piston movements. In this work, we investigate the effect of these parameters and focus on developing a model that can predict pressure changes for rapid command inputs and volume changes, something that has not been reported before and is important for robotic applications. in Chapter 4 and Chapter 5 we build and improve on these pressure dynamics models as we exploit ideas from the field of model based trajectory optimization to design a high performance pneumatic controller for our system. This work is a natural continuation of the previous works in [104] and [122].

### *2.3 Grasping and Manipulation*

Grasping has received more attention from the robotics community than dexterous manipulation. Researchers have approached grasping primarily from two different directions. One approach focuses on improving the design of the manipulator so as to improve its capabilities in terms of producing stable grasps. The other approach exploits optimization based techniques to evaluate the stability of the grasp using grasp metrics [1]. Various grasp metrics has been proposed with no clear advantage of using one over the other. These methods are less likely to generalize for dexterous manipulation but are based on important ideas. One such idea is force closure which can be treated as the ZMP equivalent in case of grasping. Another idea is position the manipulator in pre-grasp configuration and close the finger while relying on the compliance of the joints to grasp the object. Graspit [62] is a tool for simulating

and evaluating grasp for different shapes which is quite popular in the community. While grasping has always been the center of hand research, one common approach towards reactive manipulation is tele-operation using data gloves. In tele-operation the hard part of planning and decision making is left to the intelligent user while the controllers blindly follows the joint trajectories. Contact invariant trajectory optimization [68] is an offline method capable of synthesising hand manipulation but its slow and trades physics for visually expressive manipulation behavior. Such approaches are hard to generalizable and are less likely to scale in real applications. Also, see review paper [13] [74]

### *2.4 Dimensionality Reduction and Synergy Based Control*

Although rich and diverse, animal forms exhibit characteristic movements that can be attributed to its morphology, neural system and habitat. Researchers have ascribed these to bio-muscular and neural factors [49] [110]. Low dimensional embedding has been found at the level of kinematics [88], instantaneous muscle activity[111], spatio-temporal muscle activity [23] and feedback control law[58]. These low dimensional embeddings have long been exploited in the graphics community to reduce the dimensionality of search spaces to synthesize full body movements. It has also been demonstrated that such embeddings also exists in hand movements. Approximately 95% of the postural variance associated with hand grasping can be explained using four principal components [107]. Such low dimensional embedding and synergy spaces have been exploited by the robotics community to accelerate the pace of grasping research [62]. However, it has restricted the capabilities of present robotic devices to simple grasps. Similar to biological systems, present day robots have many dofs. While synergies and low dimensional spaces have helped us control and emulate some of the functionalities; they have restricted the behaviours to simple movements that conceal the expressiveness and dexterity of these robots. In Chapter 6 we develop techniques to leverage synergy based techniques that preserved the expressiveness and dexterity of the system.

### *2.5 Capturing Hand Manipulation*

Human hand function has been recorded and studied extensively in the past, mostly in the context of static grasping as opposed to more dexterous manipulation. Investigations have lead to different shape-based grasp taxonomies [124] and grasp evaluation metrics [1] [63]. These metrics have been widely leveraged by data driven [14] and optimization based techniques [62] for static grasp synthesis. However they have limited utility for more dexterous manipulation involving dynamic contact phenomena.

To the best of our knowledge there has not been prior work that successfully captured rich and physically-consistent dynamic interactions for hand manipulation. This is not surprising, given that the full suite of sensor technologies needed to do this is not readily available. Researchers [113] have however extensively looked at using human motion data for robot programming and robot teaching. Human demonstrations were recorded using a combination of vision, motion tracking and hand tracking systems. Recorded demonstrations were then segmented, classified into predetermined primitive (reaching, pregrasp, grasp types etc), and then appropriately sequenced to generate a robot program equivalent to the human demonstration. These approaches (including references 2-10 in [37]) propelled the advancement in grasp planning towards manipulation, but have been limited to basic reaching and grasping, and pick-place operations with simple grippers.

Scaling attempts towards manipulation using dexterous manipulators were challenged by the low fidelity recordings of human demonstration. Occlusions due to compact workspace co-inhabited by the object being manipulated significantly impact the ability of motion tracking and vision-based techniques to observe and record real-life hand manipulation demonstration. Extensive manual work [41] was required to clean-up the recorded datasets with specific attention towards individual hand-object interaction. Furthermore, phenomena such as rolling, sliding, compliance, deformations which heavily dominate manipulation, and geometric inconsistencies are extremely difficult to fix in datasets recorded using such techniques. Technological limitations and physically-inconsistent datasets significantly impact

the pursuit of understanding manipulation from emprical data.

The need for realistic grasping and manipulation of virtual objects has always been clear in Virtual Reality applications. [16] summarizes prevalent reaching and grasping techniques being used. As the focus of these applications are speed and ease of use, various nonphysical tricks (space warping, nonphysical forces) are leveraged to achieve desired visual effects. Physical consistency is often compromised for enhanced visual experience. [37] presents a VR system that captures and analyses human demonstration for motion intentions to explore multi-fingertip haptic interface for programming dual arm multi-finger robots.

The graphics community has long utilized motion capture system and dimensionality reduction techniques to synthesize movements for animated characters. While the focus has always been on full body movements, motion capture techniques have been used for hand movement synthesis [119] [87]. These approaches have not been widely successful in the robotics research owing to the real world physics constraints (that can be violated in animation), its inability to generalize and computationally expensive post processing step. An up-to-date discussion on motion capture based hand manipulation can be found in [120]

Recent years have seen significant developments in 3D visualization devices including the Oculus Rift[73], Google Cardboard[32], Microsoft Hololens[60], Sony HMZ-T3W head-mounted display[95]. These devices have wide viewing angles for immersive experience, and some of them have integrated head tracking. In the context of hand manipulation, however, it is not clear that immersion is necessary or even desirable. The alternative – which we adopt here – is to use a stereoscopic monitor (BenQ) and combine it with head-tracking to create the impression of a virtual workspace that is glued to the monitor and can be viewed from different angles. The ZSpace system [126] is a commercial product based on this approach. Here we achieve the same effect using LCD shutter glasses (NVidia 3D Vision 2) tracked by an infrared motion capture system (OptiTrack V120:Trio.) The advantage of keeping the projection surface fixed in space is that image correction for head rotation becomes unnecessary. In contrast, such correction is essential when using head-mounted displays, and is difficult to implement at low-enough latencies to fool the human visual system into

perceiving stable images. Furthermore, fixed monitors avoid any optical distortions and provide high resolution over the relevant workspace.

Another recent technological development are consumer depth cameras such as the Kinect[61]. They have enabled rapid progress in the area of activity recognition. While Kinect-style cameras focus on medium range sensing for full body tracking, other devices such as PrimeSense and LeapMotion[50] can be deployed for close-range tracking of hands [90]. This technology however suffers from occlusions, and appears to be better suited for recognizing a small set of predefined gestures than unconstrained finger tracking. This is why we have adopted an older but more functional approach in Chapter 9, which is to combine an infrared motion capture system for head and forearm tracking, with a CyberGlove [22] for wrist and finger tracking. The resulting system is considerably more expensive compared to recent consumer devices, but if we are to build an interactive simulation environment where users can indeed perform manipulation tasks, we do not see a viable alternative for the time being.

### *2.6 Model based Reinforcement Learning*

Depending on one's preference of terminology, the methods we will detail in Chapter 8 and Chapter 10 can be classified as model-based Reinforcement Learning (RL), or as adaptive optimal control [12]. While RL aims to solve the same general problem as optimal control, its uniqueness comes from the emphasis on model-free learning in stochastic domains [97]. The idea of learning policies without having models still dominates RL, and forms the basis of the most remarkable success stories, both old [102] and new [65]. However RL with learned models has also been considered. Adaptive control on the other hand mostly focuses on learning the parameters of a model with predefined structure, essentially interleaving system identification with control [7].

Our approach here lies somewhere in between (to fix terminology, we call it RL in subsequent sections). We rely on a model, but that model does not have any informative predefined structure. Instead, it is a time-varying linear model learned from data, using a generic prior for regularization. Related ideas have been pursued previously [51, 53, 64]. Nevertheless, as

with most approaches to automatic control and computational intelligence in general, the challenge is not only in formulating ideas but also in getting them to scale to hard problems – which is our main contribution here. In particular, we demonstrate scaling from a 14-dimensional state space in [53] to a 100-dimensional state space here. This is important in light of the curse of dimensionality. Indeed RL has been successfully applied to a range of robotic tasks [26, 43, 78, 101], however dimensionality and sample complexity have presented major challenges [25, 42].

### *2.7 Learning From Demonstration, and Generalization*

Although robotic reinforcement learning has experienced considerable progress in recent years, with successful results in domains ranging from flight [3] to locomotion [101] to manipulation [81][103][80], comparatively few methods have been applied to control dexterous hands. [112] report results for simple in-hand manipulation with a 3-finger hand, and our work reports learning of simple in-hand manipulation skills, such as rotating a cylinder, using time-varying linear-Gaussian controllers [45]. However, neither of these prior methods demonstrate generalization to conditions not seen during training. Our experiments demonstrate that our approach can learn policies for a complex precision grasping task, and can generalize to variation in the initial position of the target object. In contrast to in-hand manipulation, this task exhibits complex discontinuities at the point of contact. We overcome this challenge by combining learning from experience with imitation learning from human demonstrations, provided through a data glove teleoperation interface.

Initialization of controllers from demonstration is a widely employed technique in robotic reinforcement learning [81] [103]. However, most prior robotic reinforcement learning methods still use a hand-specified reward or cost function to provide the goal of the task during learning. Specifying suitable cost functions for complex dexterous manipulation can be exceedingly challenging, since simple costs can lead to poor local optima, while complex shaped costs require extensive intuition about the task. In Chapter 10, we define the cost in terms of the example demonstrations. This approach resembles the work of [35], which used an EM-

style algorithm to associate demonstrations with initial states in a reinforcement learning scenario. However, this prior work showed results on a simple inflatable hand with limited actuation, and did not demonstrate dexterous manipulation for complex tasks.

Researchers have addressed the issue of generalization by combining multiple linear-Gaussian controllers into a single nonlinear policy, using methods such as guided policy search [53] and trajectory-based dynamic programming [8]. Applying these methods to manipulation tasks could allow training more generalizable manipulation skills, and also bring in additional sensory modalities, such as haptics and vision, as described in recent work on guided policy search [54]. In Chapter 11, we will explore two interpolation methods, deep learning, and nearest neighbors, for generalizing to a wider range of tasks.

# Chapter 3

# *ADROIT* MANIPULATION PLATFORM: DESIGN AND PERFORMANCE EVALUATION

The unique capabilities of the human hand have long inspired roboticist in their pursuit to develop manipulators with similar "dexterity". We use this term here to refer to a combination of features: many independently-controlled degrees of freedom (dofs), speed, strength and compliance. Simple and isolated tasks such as grasping can of course be accomplished by simpler devices. Nevertheless if robots are to perform a wider range of tasks in less structured environments than what is currently possible, they are likely to need manipulators approaching human levels of dexterity.

In this chapter we describe a pneumatic actuation system for dexterous robotic hands. It was motivated by our desire to improve the ShadowHand system, yet it is quite universal and indeed being used with a second robotic hand [116] developed int the Movement control lab, CSE, UW. The actuation system allows us to move the ShadowHand skeleton faster than a human hand (70 msec limit-to-limit movement, 30 msec overall reflex latency), generate sufficient forces (40N at each finger tendon, 125N at each wrist tendon), and achieve high compliance on the mechanism level (6 grams of external force at the fingertip displaces the finger when the system is powered.) This combination of speed, force and compliance is a prerequisite for dexterous manipulation, yet it has never before been achieved with a tendon-driven system, let alone a system with 24 degrees of freedom and 40 tendons.

## *3.1 Motivation*

The specific motivation behind the work described in this chapter is somewhat accidental. We purchased a ShadowHand [92] with the goal of developing and testing advanced control

schemes for object manipulation. After experimenting with it briefly we concluded that, at least for the unit we received, the actuation needs to be substantially faster and more compliant in order to support dexterous object manipulation. We then disconnected the actuators (air muscles) and observed that, when we pulled on the tendons manually, the resulting finger motion was very fast and compliant. Thus we decided to return the built-in actuation system and develop an alternative. Upon further reflection it became clear that such a development is very much needed in the field. Indeed there exist multiple tendon-driven hands (including the Utah-MIT hand[38], the ACT hand[27], as well as cadaver hands [21]) that are comparable to human hands kinematically, yet the lack of suitable actuation has hindered control applications. Furthermore, 3D printing technology has made it surprisingly easy to design and build new hands (and possibly other mechanisms) with large numbers of joints and tendons attached to them – see below. The question then is, how does one pull on all the tendons. Here we offer a solution which we believe is quite universal and reflects the state-of-the-art in actuation technology.

The rest of the chapter is organized as follows. In the next section we outline the design considerations and the choices we made. We then describe the design of the new actuation system in detail, followed by experimental results characterizing speed, strength and compliance of the improved ShadowHand. Finally we consider future simplifications which could make the system considerably less expensive while preserving its advantages. We also summarize the application of our new actuation system to an independently developed 20 dof UW hand [116].

### *3.2 Outline of design considerations and choices*

The requirement for high dexterity naturally leads to the choice of pneumatic actuation. Indeed this may be the only available technology that combines speed, strength and compliance on the mechanism level with small and lightweight actuators – in turn allowing portable drives with as many as 40 units to be built (the ShadowHand has 40 tendons). On the other hand, the built-in actuation system in the ShadowHand was pneumatic and did not meet

our expectations. This however can be attributed to factors that can be avoided, as follows.

First, in order to obtain sufficient force from small air muscles, one has to mount them so that they are pre-stretched – meaning that even when the system is inactive there is a lot of passive "co-contraction". Since the tendons unavoidably slide over multiple surfaces, putting them under tension causes so much friction that the compliance advantage of pneumatic actuation is lost. Another example where passive tensioning increases friction is the ACT hand, where the electric motors need to be augmented with springs so as to prevent the tendons from slipping off pulleys (the ShadowHand does not use pulleys). The solution then is simple: replace the air muscles with air cylinders which do not need tensioning. One caveat here is that most cylinders have pneumatic seals with unacceptable friction levels. However we found a cylinder (AirPel [4]) that is effectively frictionless, weighs 46 grams, and generates 42 N of linear force at 100 PSI.

Second, the ShadowHand uses small inexpensive valves mounted on-board. Such valves have insufficient flow rate – resulting in sluggishness that matches the reputation pneumatic systems have in robotics. Yet nowadays one can find qualitatively better valves: fast, proportional, and with high flow rate. We selected the FESTO MPYE series, although there may be other comparable choices. To be clear, there is no perfect valve at the moment. Apart from the higher price, high quality valves are large and must be mounted off-board. This alone is not an issue (given that the compressor is also off-board), but off-board mounting means that there is an air line between the valve and the cylinder, and the longer the line is the slower the actuator becomes. However, for the combination of line lengths, flow rates, pressures and cylinder volumes used here, we were able to achieve maximum force in the cylinder around 10 msec after sending a command to the valve – which is faster than the response of human hand muscles to neural input.

Another deviation from the ShadowHand design is that we attached a linear magnetic sensor to each cylinder. Even though the ShadowHand has a joint angle sensor in each dof, we reasoned that sensing the cylinder positions directly is useful for avoiding tendon slack and also calibrating the tendon moment arms; and that other hands may not have

joint angle sensors – indeed we have already built an alternative hand (UW Hand [116]) which falls in the latter category. Overall, our philosophy was to design and build the best-performing actuation system we could afford (on a budget of $60,000). This includes a National Instruments PXI systems allowing us to sample 48 pressure sensors at 32KHz and 48 linear position sensors at 9KHz and average within batches, resulting in very low noise measurements. Later in this chapter we discuss options for building a less expensive system with similar performance.

### *3.3 Design Objectives*

We envisioned our system to be a general purpose actuation module capable of actuating most pneumatically driven robots, with an added emphasis towards tendon driven systems. The intended use is limited to research settings at the present level of development. Design choices listed below (in decreasing order of priority) were made while designing the system.

1. Compatibility with requirements of most pneumatically driven robots with focus towards tendon driven systems

2. Minimum hardware bottlenecks

3. Maximum computational capability

4. Use of off-the-shelf parts

5. Modularity in design at all levels

6. Accommodate extensions and facilitate modifications at all levels

7. Intended use under research settings

8. Safety

9. Weight

10. Cost

### *3.4 Design*

The hardware (Fig 3.1) consists of the following modules:

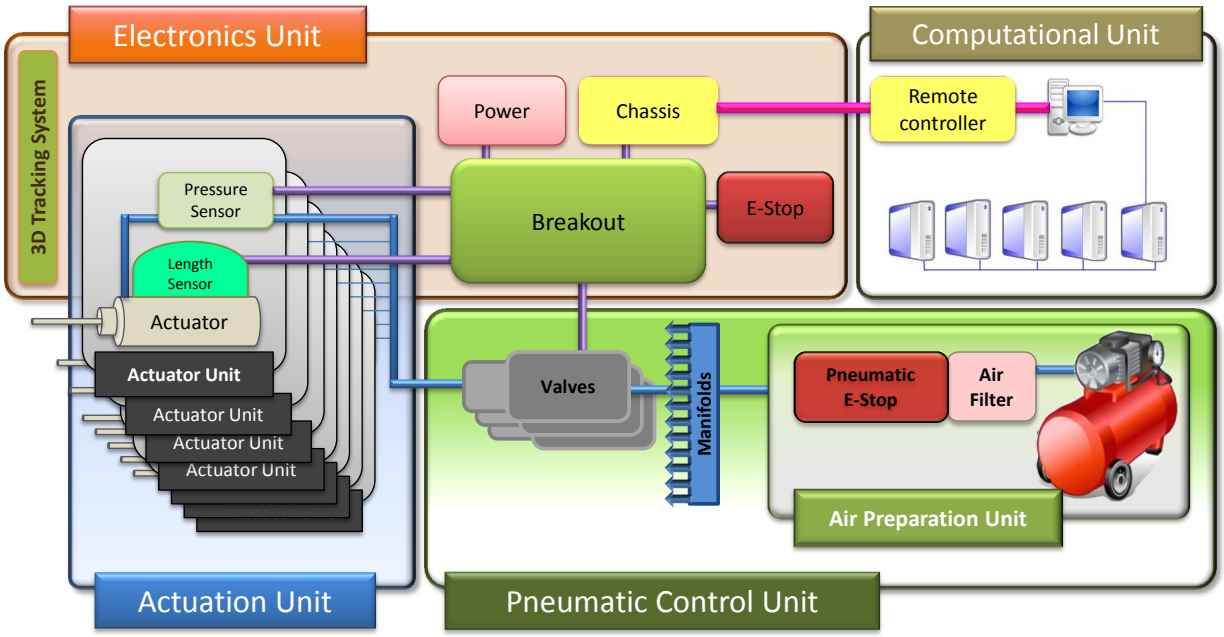



Figure 3.1: Hardware design schematics. Air pathway is represented using blue lines. Electrical pathway is represented using purple lines. Digital data pathway is represented using pink lines.

1. Actuation unit

2. Pneumatic control unit

3. Electronics unit

4. Computational unit

Out of the four units, only the actuator unit is specific to the robot being driven by the system. Presently this unit is configured for tendon driven hands. However, it should be noted that the actuator units bear no restriction towards tendon driven systems. All actuation mechanisms (tendon driven/ direct actuation) under all configurations (single/double acting, bundled/ distributed) are supported. The last three modules can be used for any pneumatically actuated system with appropriate configuration of the actuator unit.

## *3.5 Actuation unit*

Presently, the actuation unit is configured as a tendon driven hand, thus we call it 'Muscle Actuation Unit'. It consists of the following sub-modules

1. Hand muscles actuation unit

2. Arm muscles actuation unit

In addition to the above, the muscle actuation unit houses two components of electronic units - the pressure and length sensor. These are discussed in detail in section 3.7.

Muscles acting over the finger and wrist tendons form 'hand muscle actuation unit' . It consists of Actuator units, and Housing assembly. Muscles acting over the lower arm, elbow, upper arm and shoulder form 'Arm muscles actuation unit'.

### *3.5.1 Actuator unit*

**Design parameters**: The selection of actuators was based on the design parameters listed below in decreasing order of priority.

1. Minimum friction and stiction

2. Maximum force output

3. Fast with minimum response time

4. Minimum weight of actuators

5. Compactness of actuator unit

6. Compatibility with sensory devices

7. Availability in different configurations for different use cases

8. Durability

It should be noted that these principles focus on generic actuation mechanisms. No restrictions specific to tendon-driven systems were made.

**Hardware details**: The actuator unit (Figure:3.2) consists of double-acting Airpel series cylinders commercially manufactured by Airport Corporation. Single-acting cylinders are often air-return or spring-return and do not allow control over the return force. Double-acting cylinders were selected for complete control over the actuation force in both directions (although this feature is not yet utilized). The finger tendon's actuator unit has stroke length of 37.5mm, can produce up to 42N of force and weights 45.7grams. The wrist tendon's actuator unit has stroke length of 50mm, can produce up to 125N of force and weights 95.7grams. Detailed specifications can be found at [4].

Double-acting modules were used in single-acting mode in the Muscle Actuation Unit. Finger tendons were actuated using M9 cylinders and wrist tendons were actuated using M16 cylinders. In tendon driven systems, it seems preferable to have a small force on the actuators to avoid tendon slack when the muscles are in passive non-pulling state. However, the non-zero force from the passive tendon creates co-contraction and adds undesirable stiffness to the joints. The return force of single acting cylinders are fixed and non observable and cannot be compensated using pneumatic forces since both forces act in same direction. Double-acting cylinders with no return were employed to simulate virtual variable stiffness springs

to intelligently handle tendon slack and minimize joint co-contraction. Variable stiffness simulated springs facilitated control over the return force which would not have been possible if single acting cylinders were used. For compatibility with length sensors, magnetic cylinder pistons were used.

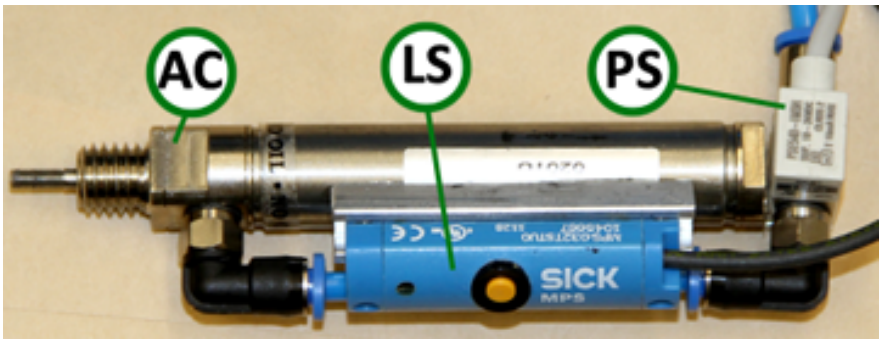



Figure 3.2: Cylinder unit [AC: Actuator, LS: Length Sensor, PS: Pressure Sensor]

**Design evaluation and experience**: The actuator selection was the most critical and challenging selection of the entire design. More than 16 models from 6 different manufacturers were rigorously tested over the listed design parameters. Though all the considered models performed well on most deign parameters, requirements 1 and 3 were exceptionally severe.

The models from Airpel significantly outperformed others on criteria 1 and 3. The stiction and friction values for these models were exceptionally small - the piston fell under its own weight when the cylinder was not horizontal. This is possible because the traditional pneumatic seals have been replaced with "air seals" with precision-fit graphite pistons that slide freely (without lubrication) inside a Pyrex glass cylinder providing unique ability to impart smooth motion at very low pressures, slow speeds and short strokes. There is a small air leak of about 2SL/min [5], however this is not an issue when using high flow rate valves.

To sum up, the Airpel cylinder we selected rates well on all parameters except 4 and 5, on which it rates moderately.

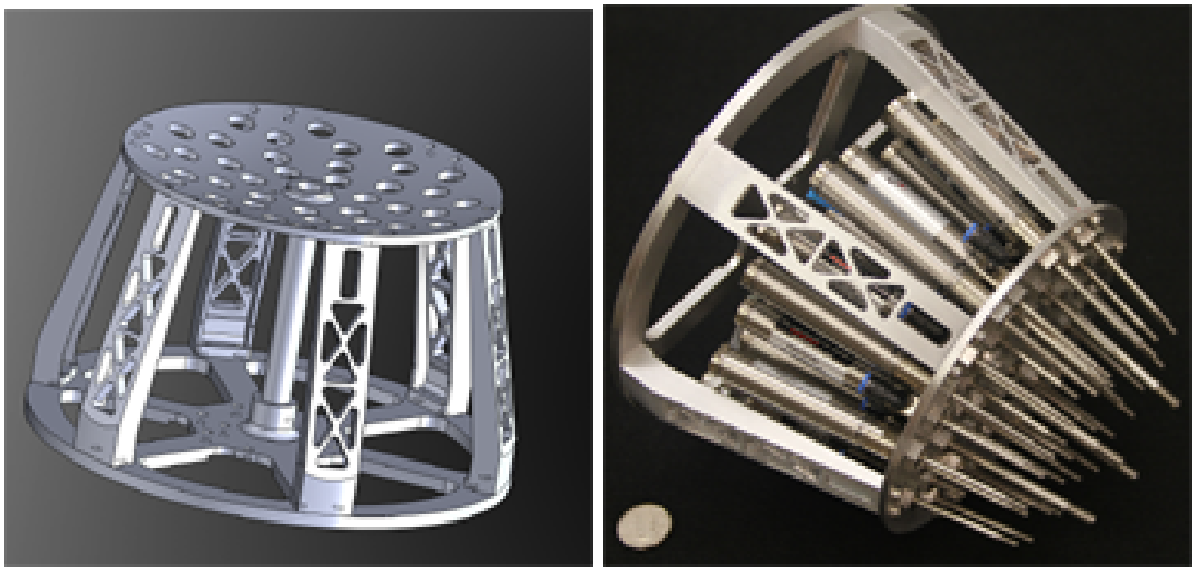
### *3.5.2 Housing assembly*

**Design parameters**: The design parameters for the housing assembly are listed below in decreasing order of priority:

1. Strength, load bearing and stability
2. Minimum off-axis actuator loads
3. Compactness of Hand muscles actuation unit
4. Weight
5. Ventilation
6. Compatibility with different hands
7. Ease of component assembly
8. Machinability

**Hardware details**: Figure: 3.3 shows the final housing assembly without the actuator unit. The assembly contains 36 of the M9 Airpel actuator units for finger tendons, and 4 of the M16 Airpel actuator units for wrist tendons.

Tendon-driven robotic hands usually route finger tendons via the center of wrist joint in order to minimize the moment arms of finger tendons on the wrist joint. As a result, all finger tendons come out of the hand via an opening at the wrist. To reduce off-axis actuator loads, it is desirable to mount the cylinders so that they all point to this opening – suggesting a concave mounting plate. The back plate is free of cables and connectors and has mounting holes for attachment to a robot arm. It is compatible with the Shadow Arm robot – which we have also redesigned with air cylinders, but this will be described elsewhere. Figure: 3.4 shows the complete Muscle actuation unit without the back plate. Note that if we did not



(a) CAD model (b) Final machined assembly

Figure 3.3: Housing assembly

attach a length sensor to each cylinder (which doubles the cylinder diameter) the diameter of the assembly could be reduced roughly by half.

**Design evaluation**: Machining the housing assembly (and in particular the curved plate) turned out to be harder than it first appeared, but was eventually successful. It weighs 660grams, and can sustain about 75N from each actuator with a factor of safety 3. When the actuators and the ShadowHand are mounted, the entire system weighs 4.5kg. When attached to a robot arm, most of this mass is near the base (elbow), thus we do not expect it to be problematic.

### *3.6 Pneumatic control unit*

**Design Parameters**: For the pneumatic control unit, the following design parameters (in decreasing order of priority) were established.

1. High flow rate

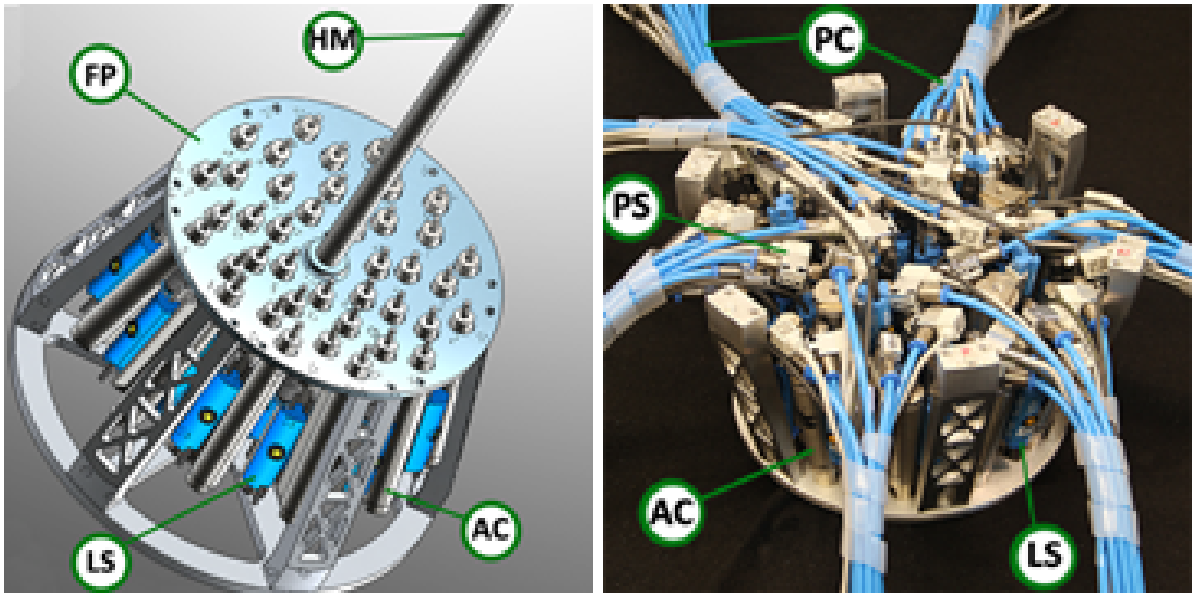



Figure 3.4: Muscle Actuation Unit (a) CAD model (b) Final assembly (without back plate). [FP: Front Plate, HM: Hand Mount, LS: Length Sensor, AC: Actuator, PS: Pressure Sensor, PC: Pneumatic Connectors]

2. High update frequency
3. Minimum pneumatic latency
4. Minimum pneumatic bottlenecks and avoid air pockets
5. Independent of type of actuation unit
6. Modularity
7. Facilitate modification and accommodate extension
8. Compact and light weight
9. Cost

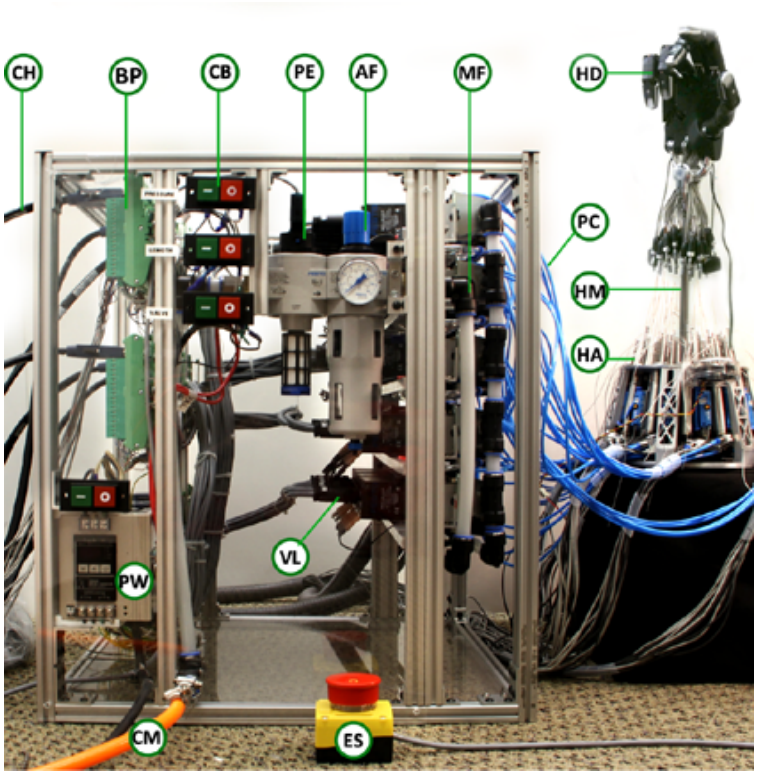



Figure 3.5: Pneumatic Control Unit (2ft x 2ft x 2ft) [CH: To chassis, BP: Breakout Panel, CB: Circuit Breaker, PE: Pneumatic E-stop, AF: Air Filter, MF: Air Manifold, HD: Hand, PC: Pneumatic Connections, HM: Hand Mount, HA: Hand muscle actuation unit, VL: Valve, ES: E-Stop, CM: To Compressor, PW: Power]

**Hardware details**: The pneumatic control unit (Fig 3.5) consists of the following sub-units:

1. *Air preparation module* consists of air compressor and LFR-3/4-D-5M-MAXI-A filter regulator with a flow rate of 7,600L/min at 6 bar. Detailed Specifications can be found in [31]

2. *Pneumatic control Valves*: Pneumatic control unit contains 40 MPYE 3/5 proportional valves from FESTO. Pneumatic valves (MPYE-5-M5-010-B) used for 'Hand muscle

actuation unit' have a flow rate of 100l/min at 6bar, bandwidth of 125Hz and weight 290g. Detailed specifications can be found at [31].

3. *Air supply manifolds*: MPYE series proportional valves were designed by FESTO as stand alone, high end devices to be used as individual units. Thus no manifold solution have been made available for these valves. After careful examination, PAL manifolds of TIGER-2000 series[31] valves were modified for compatibility with MPYE series valves. Pneumatic control unit contains 5 PAL-10-B manifolds[31], each serving 10 pneumatic control valves.

**Evaluation and Experience**: Proportional valves were preferred over binary switching valves despite their moderate performance on parameters 8 and 9. This was driven primarily by our application requirement that called for fine-grained control over flow rate for fast and precise actuator control.

Binary switching valves achieve variable flow rates using pulse width modulation (PWM) which restricts smooth and precise control over flow rates. Moreover, PWM frequency is limited by the bandwidth of the device, further constraining smooth behaviours. A bank of binary valves can get very loud when switching at a fast rate.

PAL manifolds are capable of serving valves with flow rates up to $2600l/min$ i.e. our manifold solution is capable of facilitating upgrades to all MPYE series models without any modification. High flow rate (20X the requirement of the presently used MPYE-5-M5-010-B model) manifolds are preferred since they act as pneumatic buffers, thus avoiding air pockets under heavy flow requirements. Our air supply manifold presently contains 2 expansion slots. Addition of manifolds for future expansion is facilitated by the high level of modularity in the design, within reasonable efforts.

Pneumatic latencies are a function of tube lengths between the actuators and the valves. The air path between them was minimized to the extent possible. Pneumatic manifolds and valves were configured to minimize the pneumatic interface (face where valves expose their pneumatic connections to actuators) surface area, thus minimising the tube length between

valves and respective actuators.

### *3.7 Electronics unit*

The electronics unit consists of the followings components:

1. Sensors: Cylinder pressure sensors, and piston length sensors (housed by 'Muscle actuation unit' Figure:3.2).

2. NI PXI Chassis with multiple A/D and D/A boards.

3. Power Supply.

4. Emergency Stop.

**Design parameters**:

1. Sensors

   (a) Observability of entire state of the system

   (b) Resolution, bandwidth and range

   (c) Compatibility with generic pneumatic actuators

   (d) Stand alone, compact and light weight

   (e) Flexibility and ease of replacement

2. Chassis

   (a) Minimum latency

   (b) Sensing resolution, bandwidth and range

   (c) Data bandwidth for communication with computational unit.

   (d) Support for multiple communication protocols

(e) Expandable and reconfigurable

(f) Support for different operating systems

(g) Compact, enclosed and safe

3. Power Supply

(a) Compatible with sensor and actuator requirements

(b) Clean and reliable

(c) Peak load capacity

(d) Electromagnetically decoupled output channels

(e) Overload protection, over voltage protection and short-circuit protection

(f) Indication monitor

4. Emergency Stop

(a) Minimum latency

(b) User and hardware safety

**Hardware details**:

1. Sensors

(a) *Pressure* Electronics unit contains 48 solid state SMC pressure sensor. 40 sensors are housed by 'Hand muscle actuation unit' and rest 8 by 'Arm muscle actuation unit'.

(b) *Length* Electronics unit contains 48 Sick magnetic piston length sensor. 40 sensors are housed by 'Hand muscle actuation unit' and rest 8 by 'Arm muscle actuation unit'.

(c) *3D-tracking system* consists of active marker motion tracking system from PhaseSpace. [82]

2. Chassis: National Instrument's *9-Slot 3U PXI Express: PXIe-1078* module with 1GB/s is configured as Chassis. Each slot has a bandwidth of 250MB/s. Present configuration of modules sample 40 hand length sensors at 9kHz and 48 pressure and 8 arm length sensors at 32kHz. Chassis is also equipped with 1mbps CAN module which is used to communicate with ShadowHand sensors. Computational unit uses one 798MB/s bandwidth PXIe-PCIe data channel for complete control over the chassis. Detailed specifications can be found at [71]

3. Power Supply: 24 V DC 10Amps *S8VS-24024A* switching power supply from OMRON [75] is used as power source. One electromagnetically separated channel powers the sensors while the other powers the Pneumatic control valves.

4. Emergency Stop: A hybrid combination of software and pneumatic e-stop is used as an Emergency unit. Pneumatic stop has a flow rate of 6,500 l/min at 6 bar and weighs 600gms.

**Design evaluation and experiences**:

Since the intended applications of our system was towards research applications, there was never a preference towards onboard electronics.

Designing onboard electronics can definitely make the overall system compact and independent. These advantages from design of onboard electronics would have come at the cost of reconfigurability and extendability of the hardware. Furthermore, it would have conflicted with our overall design parameter-4 of using off-the shelf components to the extent possible. All connections were made using TBX-68, a DIN rail mount screw terminal connector block from NI, for accessability, reconfigurability and debugging purposes.

1. Sensors

Table 3.1: Sensor Specifications

| Specifications | PSE540(A) [94] | MPS [93] |
|---|---|---|
| Measuring Range | 0-1Mpa | 32-256mm |
| Operational Voltage | 12-24 VDC, ripple(P-P) ±10% | 15-30 VDC, ripple(P-P) ±10% |
| Analog output | 1-5 VDC | 0-10 VDC |
| Resolution | <2% | 0.05mm |
| Linearity | <±0.7% | 0.3mm |
| Repeatability | <±0.2% | 0.1mm |
| Sample Time | N/A | 1 ms |
| Current consumption | <15mA | 25mA |
| Output impedance | 1kΩ | 2kΩ |
| Weight | 4.6g(without wires) | N/A |
| Max Speed | N/A | 3m/s |

(a) *Pressure*: Air path adds latency to any pneumatic system. Since our actuators are driven using off-site high flow valves, pressure sensors were placed closest possible to the actuators to account for the pneumatic latency.

(b) *Length*: Length sensing capability was a difficult choice, as it posed several challenges at multiple levels. Piston length sensing capabilities were added to cater to some of our recent efforts in the direction of bio-mimetic tendon driven systems [28][123]. In such systems, it is particularity difficult to have joint angle sensors. Access to tendon excursions helps with kinematic modeling [123].

*Size of Actuator unit*: The size of the actuation unit had to be increased to accommodate for length sensing without which the diameter of the hand muscle actuation unit would have been comparable to the typical dimensions of a human forearm.

*Number of Analog Input channels*: Due to high impedance of length sensors we observed *ghosting* [72][70]. In case of ghosting, every sensor at $i^{th}$ channel of DAQ gets coupled with the one connected to every $(i+1)^{th}$ channel. Most acceptable and efficient way to eradicate ghosting is to interleave null/ground channels between each sensor channels. Null channel interleaving completely eliminated ghosting but resulted in 2X number of DAQ channels.

2. *Chassis*: 2X AI channels were used for length sensors to mitigate the effects of ghosting. This requirement was met by replacing one PXIe6363 module with PXIe6255 module that has more number of channels but a lower sampling frequency. This reduced the rate at which Hand actuation unit's length sensor can be sampled from 32KHz to 9Khz.

3. *Power supply*: Power is distributed via modular power strips. Individual sensors/actuation banks can be turned on/off to minimize noise floor and system load when not in use. Modular power distribution facilitates addition of different power supply units for individual sensor/actuator submodules, thus facilitating upgrades at all levels.

4. *Emergency Stop*: Control valves provide unreliable flow rates when pneumatic input is fed without operating power. Unreliable flow rates can damage the robot by pushing it outside its stability regime or setting joints into oscillations. Emergency module was designed to allows pneumatic flow input only when control valves are powered up.

   Initially, we considered a complete shutdown in case of operational emergency e.g. robot performing an undesirable movement. However, careful observations (listed below) revealed that powering down the pneumatics is a more reliable and faster option.

   - Valve charging up its actuator's pressure when emergency was triggered: Due to pneumatic buffers feeding the control valves, the actuator will continue charging up till the pressure drops in the buffers and then actuator's residual pressure flushes out.

- Actuator is pressurized and valve is closed to maintain the chamber pressure when emergency is triggered: This pressure gets trapped in the cylinder and emergency not avoided.

The correct way to process an operational emergency is to exhaust the input pressure by flushing out the pneumatic buffers (using a high flow exhaust port) and opening the valves to exhaust out the cylinder pressure. This requires a pneumatic shutdown while valves maintain their input operating power. For a non operational emergency, circuit breakers are provided at all levels from top, which powers down the entire system, to bottom level, which powers down individual subcomponents.

### *3.8 Computational Unit*

**Design parameters**: Design parameters established for the computational unit have been listed below in decreasing order of priority

1. Reliable communication
2. Minimum communication latency
3. Maximum computational capacity
4. Maximum data bandwidth

**Hardware details**: NI PXIe-PCIe8371, x4 MXI-Express is used to communicate with the chassis using a high bandwidth PCI Express link. Any normal desktop or server computer with PCI express slot can serve as a Computational unit. 3D motion tracking system communicates using a standard ethernet port. PCIe link is used to retrieve pressure sensor and length sensor data, and to command the Pneumatic control assembly.

**Design evaluation and experiences**: Data communication using PCIe link and standard ethernet link ensured that our system is compatible with any standard computer with-

out any special requirements. Presently a 3.0Ghz AMD Phenom(tm) II X6 1075T, 8.00 GB machine with Win7 OS is used to control the system.

### *3.9 Design Evaluation*

Final hardware was evaluated on various design parameters using two tendon driven hands. Actuation system was perfectly compatible with both hand designs without modification.

1. 24-dof ShadowHand, developed by ShadowHand company [92]

2. 20-dof UW hand, being developed at our lab independent of the actuation system [116]

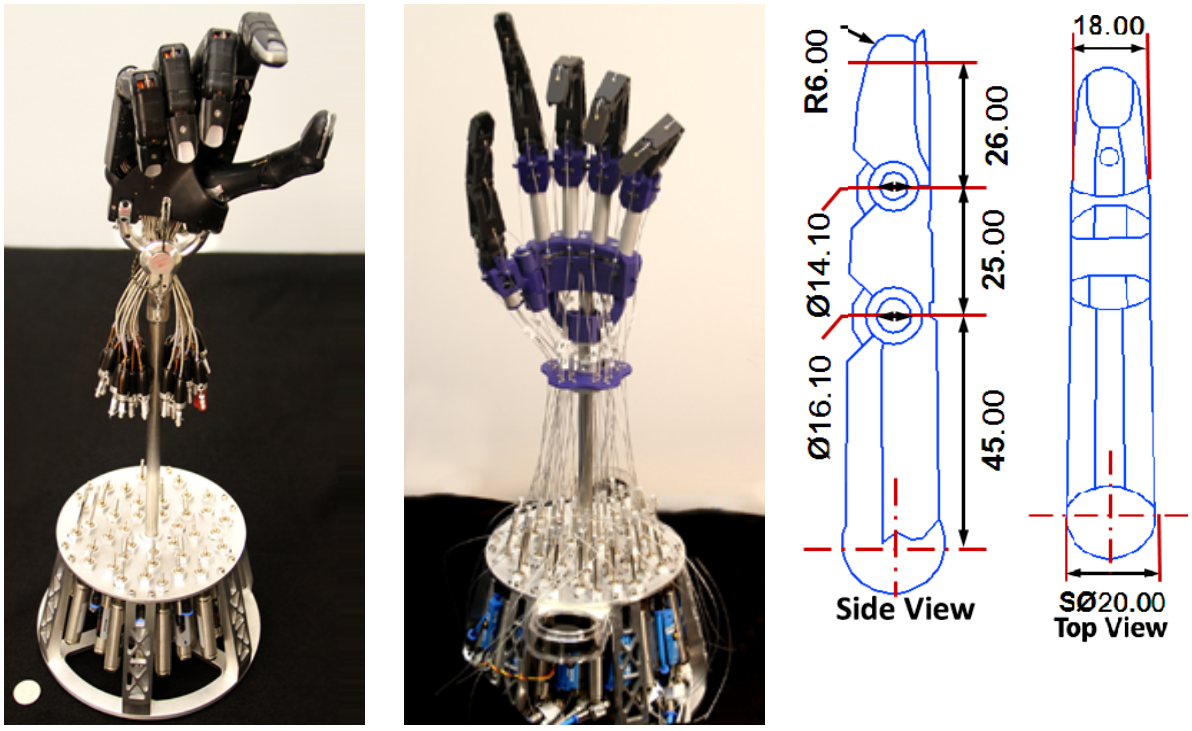



Figure 3.6: (a) ShadowHand mounted on the actuation unit. (b) UW hand[116] mounted on the actuation unit (c) ShadowHand finger dimensions

### *3.9.1 Force and compliance*

System force and compliance characteristics were studied using the ShadowHand and UW Hand. An external force of 6 grams for ShadowHand (and 8gms for UW Hand) at the index finger tip was enough to flex the MCP joint thus confirming the exceptional compliance of the final system. Typical characteristic force behaviours are summarized in Table 3.2 & 3.3

Table 3.2: Actuator force characteristics

| Specification: No load connected to piston | Orientation | Force |
|---|---|---|
| Minimum external force to break stiction and friction | Horizontal | 2.5g |
| Minimum external force to break stiction and friction | Vertical | Piston falls under its own weight |

Table 3.3: Hand force characteristics

| Specification: Finger and actuators oriented vertically | Shadow Hand | UW Hand |
|---|---|---|
| Minimum actuation force at finger tip to move MCP joint (at atm pressure) | 4.0g | 2.0g |
| Minimum actuation force at finger tip to move MCP joint (at min slack correction pressure) | 6.0g | 8.0g |
| Maximum flexing force at Index finger tip | 300.5g | 705g |
| Maximum extension force at Index finger tip | 439.4g | 700g |

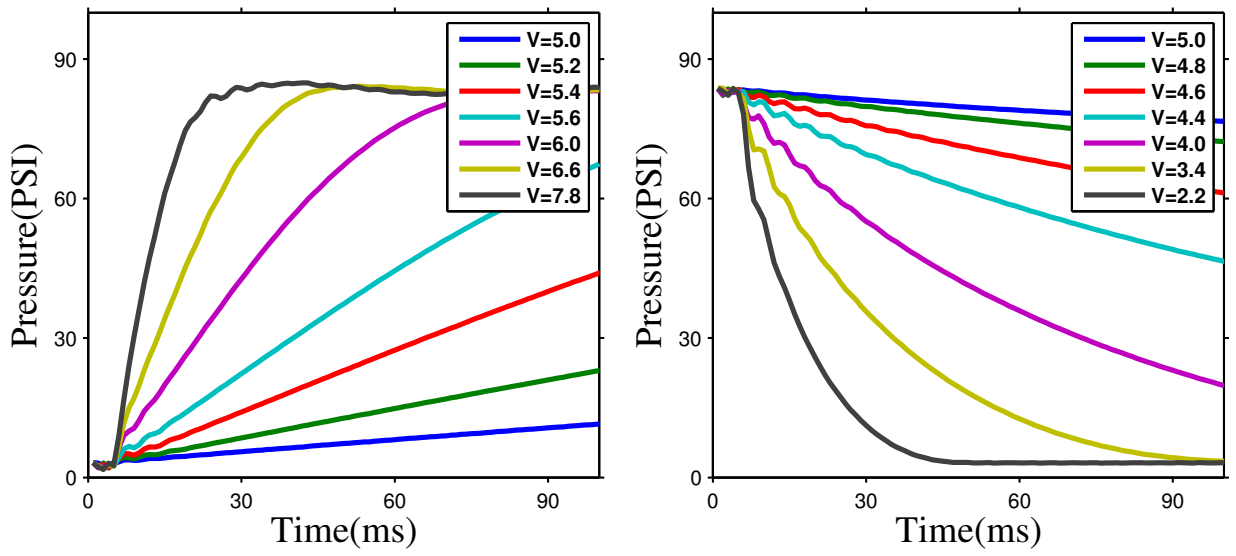



Figure 3.7: (a) Pressure behaviours while pressuring cylinder from zero using valve command(V) (b)Pressure behaviours while exhausting cylinder from zero using valve command(V)

### *3.9.2 Actuation Speed*

Our prime motivation for the development of this actuation system was to use tendon-driven hands and perform dexterous hand manipulation experiments. Any dexterous hand manipulation demands agility and responsiveness from its actuation hardware. These capabilities are evaluated in the present and the following section. The actuation system's speed capabilities were evaluated using a simple open loop bang-bang control strategy over the index finger MCP joint. The goal was to achieve full stroke movements (joint limit to joint limit) at maximum frequency. Control switching frequency was gradually increased until finger started making incomplete strokes, i.e. reversed before hitting the joint limits. Using this simple strategy, a frequency of about $7Hz$ was achieved. We are working towards a more principled way to further improve actuation speed by carefully modelling valve and pneumatics of our

system. Further details are provided in [122]

### *3.9.3 System latency and event timings*

System responsiveness was evaluated using a reflex experiment. During the experiment, small external disturbances were applied at the finger tip of the middle finger. The system was programmed to detect the disturbance (using the joint angle sensors in the ShadowHand) and react by extending the index finger. The system was found capable of detecting minute disturbances and reacting very quickly. One can consider multiple definitions of response latency. The first change in pressure is observed 8 msec after the disturbance (this corresponds to reflex latencies defined in terms of muscle activity in the biological motor control literature). See Fig 3.8. When measured in terms of the resulting motion, the latency is around 29 msec.

### *3.9.4 video attachment*

A supplementary video can be viewed here https://youtu.be/5DPrU76FbGk. The video demonstrates speed, reflex and compliance properties of the actuation system with ShadowHand skeleton. Speed behaviour is demonstrated using a sequence of flection and extension of joints (limit to limit), one at a time. Entire sequence of flexing and extension for all 24 joints merely takes 2.44 seconds. Each movement is roughly 70 milliseconds. Hand reflex is demonstrated using the experiment mentioned in subsection above. System compliance is demonstrated using 3 experiments. First, we demonstrate that actuation yields away to air blow from an average adult. Second, dead weights of 1g, 2g and 5g were dropped on the finger from a height of 5 cms to show compliance. Third, we demonstrate the effects of gentle interactions from an average adult.

## ***3.10 How to make the system less expensive***

The approximate cost breakdown of our present system is as follows:

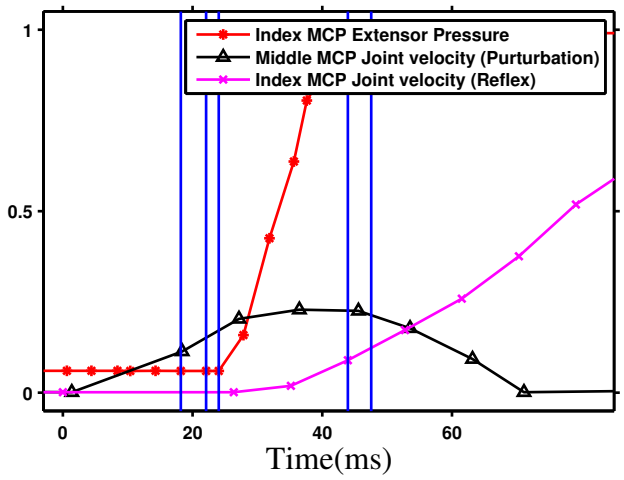



Figure 3.8: Time stamps(from left to right): $T1$(Event Trigger, Middle finger MCP movement detection) = 18.179ms, $T2$(Actuation voltage written to valve) = 22.084ms, $T3$(Pressure wave arrival) = 24.044ms, $T4$(Index finger MCP movement detected) = 46.943ms, $T5$(maximum pressure at the actuator) = 47.55.

Table 3.4: Cost breakdown of *ADROIT* actuation system

| Component | Unit price | Total |
|---|---|---|
| FESTO valves | $800 | $32,000 |
| SICK sensors | $250 | $10,000 |
| AirPel cylinders | $50 | $2,000 |
| Pressure sensors | $50 | $2,000 |
| NI PXI system | | $10,000 |
| Custom machining | | $4,000 |

The most expensive components (and thus primary candidates for simplification) are the valves, linear sensors, and electronics. The PXI system we configured is an overkill,

considering that the sensors have low noise to start with (so the high sampling rates and mini-batch averaging are not essential) and the bandwidth of the valves is only 125 Hz so there is no point in having a fast control loop (indeed we are only using 200 Hz). The National Instruments system has very mature drivers and is overall a great choice, but one could build a considerably less expensive replacement for the present purposes, perhaps using multi-channel A/D chips on custom circuit boards mounted in the forearm.

The biggest potential for savings are in the valves and position sensors. Presently we use one valve and one sensor per cylinder. This results in a universal pneumatic drive which can be used to actuate any mechanism with up to 40 tendons. Note however that 40 is actually quite a lot, and is only needed when using so-called 2N designs where tendons act on individual joints and are arranged in agonist-antagonist pairs (as opposed to the more distributed action found in the human hand and in the ACT hand). If we are willing to assume that most or all tendons will always operate in agonist-antagonist pairs, we could use one valve and one position sensor per tendon/cylinder pair (note that we still need all cylinders because even in this organization the two tendons in a pair will typically have different and possibly variable moment arms). The FESTO MPYE valves are 5/3 valves and are in fact designed to power pairs of cylinders. With these simplifications, the cost can be reduced by half. The ShadowHand skeleton (without any actuation) still costs around $60,000. However, as shown in [116], one can 3D-print a hand with comparable dexterity with cost of materials around $100 and a couple of days of assembly work. Combining these two advances, it should be possible to make dexterous robotic hands whose performance exceeds any product available on the market to today, for less than $40,000. These hands rely on off-the-shelf components and 3D printing, and could be built in academic labs.

Finally, we are not certain that proportional valves are actually needed. We clearly need valves that respond quickly and have high flow rates, but what if they were binary (and thus presumably a lot cheaper)? In principle proportional valves provide smoother movement, but given that air dynamics introduce low-pass filtering, it remains to be seen how much the performance of the (to-be-developed) control schemes will degrade in the presence of binary

valves. If the degradation turns out to be negligible, this will result in substantial further reduction in the cost of the robotic hands we envision.

### *3.11 Full hand arm system*

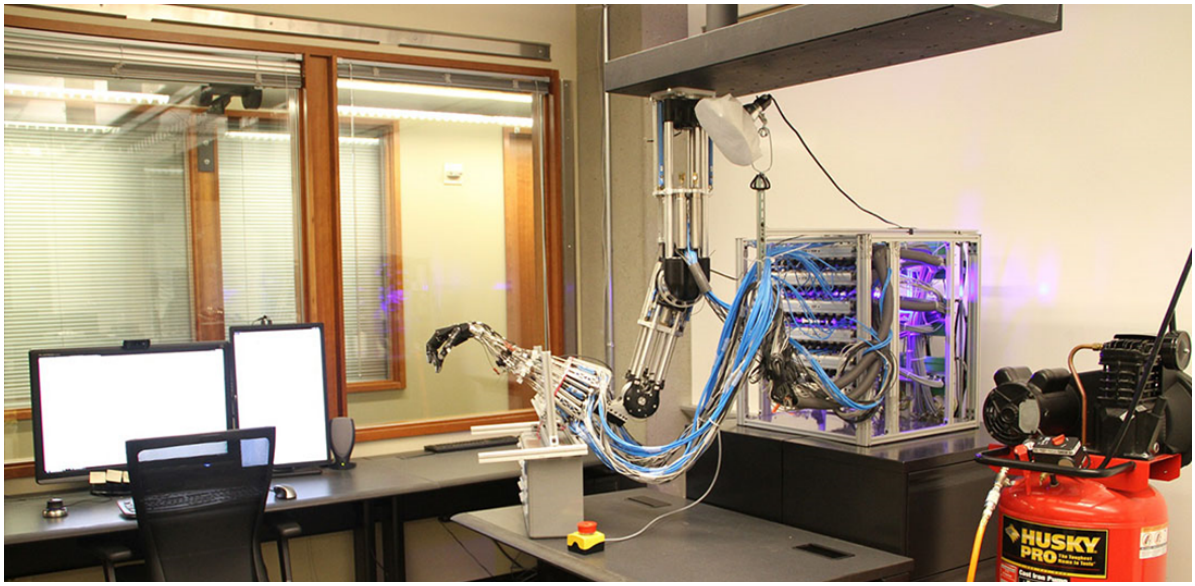



Figure 3.9: Complete *ADROIT* system

Following similar design choices, a full arm hand assembly has been achieved. Note that due to high torque requirements of the arm multiple units of Airpel M24's were used for the shoulder flex joint. Stroke lengths are different for different arm joints as well. Shoulders cylinders have smaller stroke lengths than that of the torso flex. Torso roll has the lowest stroke length. The final hand arm system is presented in Figure 3.9.

### *3.12 Summary*

The development of the present system was motivated by our desire to solve complex dexterous manipulation problems. We emphasize that we are not building hardware for the sake of building hardware. Indeed the primary focus of our research group is control; see [67] [98]

[86] for examples of recently-developed control schemes applicable to complex robots. If the robotic hardware we need already existed, we would be more than happy to focus on using it and making progress in terms of control. Unfortunately suitable hardware in terms of robotic hands does not appear to exist (with the possible exception of the DLR hand which is not available commercially), and so we were forced to develop the system described here. This development is now complete and we are ready to make a transition to control experiments. We are also finalizing the re-design of the ShadowArm robot with similar actuation, and will soon be able to mount the hand assembly on the new arm. Once the entire system is functional and used in specific manipulation experiments, we will be able to characterize its capabilities in context. But the basic tests performed here already illustrate that the system is very capable.

We hope that other research groups, as well as commercial entities, will be interested in building similar actuation systems. This chapter contains a lot of technical details that should help in replication efforts, and we are happy to provide additional details upon request.

### *3.13 Next steps*

As mentioned before, in order to achieve a general purpose manipulator a unique combination of dexterity, speed, strength, and compliance are required. In this chapter, we presented how to solve some of the design challenges towards the ambitious goal. The flexibility and dexterity of the *ADROIT* system come for a price. The dexterity of the hand leaves us with a high dimensions space to plan manipulation behaviors. Furthermore, the space in which the manipulation behavior evolves is quite compact and full of discontinuities, which makes the job of planners extremely hard to deliver good manipulation policies. In chapter 6, we discuss different techniques for dealing with the dimensionality of this space in order to produce reactive, rich and dexterous manipulation behavior in real time.

Pneumatic actuators are clean, compact and have high very high force density. In addition, their close resemblance to biological muscles makes them ideal for hand actuation. While the choice of pneumatics is very desirable for our system, it complicates the controls

system. Pneumatic actuators are 3rd order system with sensitive dynamics. They are inherently smooth but the compressibility of the air limits the overall bandwidth of the system. Pneumatic actuation makes the resultant movements smooth and human-like but introduced significant latency in the dynamics system. In following chapters, we will address challenges posed by the pneumatic actuation system. Tools from model-based optimal control and trajectory optimization will be leveraged to extract performance from pneumatic actuation. First, modeling of pneumatics actuators is presented in the next chapter (4). The following chapter (5) will go into the details of the low-level controller design that abstracts low latency high bandwidth force actuators from 3rd order pneumatic actuators.

# Chapter 4

# *ADROIT* MANIPULATION PLATFORM: ACTUATOR DYNAMICS MODELLING

ADROIT Manipulation platform is a pneumatically actuated, tendon driven 28 degree of freedom platform being developed for investigating complex hand manipulation behaviors. ADROIT derives its unique capabilities, necessary to support dynamic and dexterous manipulation, from a custom designed high performance pneumatic actuation system for tendon driven hands. The custom pneumatic actuation system is fast, strong, low friction-stiction, compliant and is capable to actuating a shadow hand skeleton faster that human capabilities – at a unique combination of speed, force and compliance that has never been achieved before. In this chapter, we develop models for the pneumatic muscles of ADROIT and perform a thorough investigation of the various parameters that affect pressure dynamics in a pneumatic system such as, different cylinder types, leakage from valves and cylinders, valve deadzone, input pressure fluctuations etc to improve the model's accuracy.

## *4.1 Introduction*

Adroit manipulation platform [48] is a 28 dof robotic system being developed with an aim to achieve complex human-like object manipulation. It consists of a 24 dof bio mimetic hand and a 4 dof robotic arm. The robot is pneumatically actuated and it owes its unique capabilities such as speed, strength and compliance to the custom actuation structure of the robot, each joint is antagonistically actuated through tendon transmission, similar to antagonistic muscles in the human body.

Pneumatically actuated robots are desirable yet hard to control. The compressibility of air makes the pressure dynamics highly non linear and reduces the bandwidth of the over all

system. Pneumatic actuators are controlled using a pneumatic valve which controls the rate of change of pressure (by controlling the orifice area) inside the cylinder chamber, thereby making the whole system follow third order dynamics. Furthermore, the pressure dynamics is also affected by undesirable factors such as valve dead-zone, air leakage from the valves and cylinders, delays from the connecting tube lengths and input pressure variations. The difficulty in accounting for these factors have led to a limited use of pneumatic actuators in robotic applications. Despite these drawbacks, pneumatic actuators are still desirable because they are clean, have a lower specific weight and a higher power rate than an equivalent electromechanical actuator. They are easy to maintain and handle. These factors make it worthwhile to explore pneumatic actuators as a viable actuation system for robotics.

As processors are getting faster, model based trajectory optimization techniques are increasingly being deployed to handle nonlinear systems [100]. These techniques leverage the model of the system to see through the planning horizon and deliver a locally optimal policy. The strength of trajectory optimization techniques lies in the predictive capability of the model of the system. Fast update of the policy enables it to handle non-linearities and modelling discrepancies. In order to tame the non-linearity and large timescale of pneumatic system, we plan to leverage the strengths of online trajectory optimization to build an effective low level controller that hides the complications of pneumatics and abstracts out a simple force actuator to the user. The goal of this chapter is to perform a thorough investigation across relevant parameters to develop models for ADROIT's pneumatic actuators that is robust to fast control signals and aggressive volume changes. This chapter strictly focuses on developing pneumatic models our of system. In next chapter, we leverage the models developed here to realize a model based high-performance pneumatic controller.

In this chapter, we use the thin-port pneumatic model, which is a popular method of modelling pneumatic drives derived from the principles of thermodynamics. We present the pressure modelling results on different pneumatic cylinders and valves, compare the performance of each of them to validate the method adopted. Since the actuation system of Adroit was inspired by Kokoro's DiegoSan [108] humanoid robot, we use simple 2-dof

shoulder joint of the DiegoSan robot as our second testbed for model validation.

In section 2 we discuss some of the previous research that has been done with pneumatic actuators. Modelling of the pressure dynamics is discussed in section 3. In section 4, a brief description of all the factors that affects pressure dynamics is given. In section 5, an overview of the hardware used and the control architecture for Adroit is provided. In section 6, the experimental method is described. The pressure modelling results with different hardware combinations is presented as a validation of the method adopted. Finally, we discuss the performance of the model and highlight the factors that affect the accuracy of the model.

## *4.2 Pressure Dynamics*

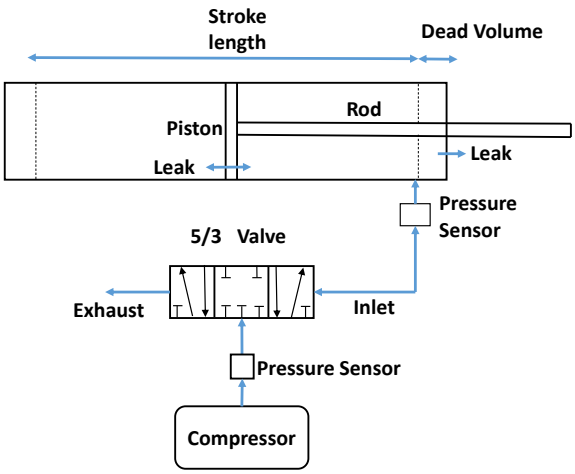



Figure 4.1: A schematic diagram of a single actuated pneumatic cylinder controlled using a 5/3 proportional valve.

A schematic diagram of a pneumatic system is shown in Figure 4.1.

### *4.2.1 Thin-Port model*

In a pneumatic system, the valve essentially controls the orifice area through which compressed air flows into the chamber of the actuator. In the thin port model, area of the orifice

is assumed to be small and also the plate through which the air flows from higher pressure region to lower pressure is considered to be 'thin'. A port is an orifice connecting two chambers as shown in Figure 4.2. The transfer of air mass is derived using thermodynamic properties of air considering the compression and the expansion stages of actuation as isentropic, hence the heat transfer coefficient for both the processes are assumed to be equal to specific heat ratio of air $n = 1.4$ [11].

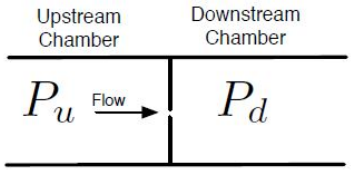



Figure 4.2: The thin port model assumption

The pressure dynamics inside a chamber of a pneumatic cylinder can be described by the equations 4.3 - 4.7, these equations model the rate of change of pressure in the chamber as a function of the air mass flow $\dot{m}$, inlet and exhaust port area $a_c$ and $a_a$ respectively, volume of the chamber $v$ and the time derivative of the volume $\dot{v}$. The mass flow from the compressor is controlled using a proportional valve and is described by the equation 4.3. In this equation, the air mass flow $\dot{m}$ depends on the port area and $f(p_u, p_d)$ where $p_u$ is the upstream pressure and $p_d$ is the downstream pressure. Equation 4.4 and 4.5 describes the $f(p_u, p_d)$. The behaviour of the pressure dynamics is dependent on the ratio of $P_u/P_d$. Above a certain value, the pressure dynamics becomes highly non linear and the air flow is referred to as choked flow. Equation describes the total mass flow into the system. $P_c$ is the compressor pressure, $P_r$ is the atmospheric pressure and $p$ is current pressure inside the chamber.

The terms $n,\ Rs,\ \alpha,\ \beta,\ k,\ \theta$ are well defined physical constants and are given by

$$\alpha = C\sqrt{\frac{2M}{Z\ R\ T}\frac{\kappa}{\kappa-1}} \qquad \theta = \left(\frac{\kappa+1}{2}\right)^{\frac{\kappa}{\kappa-1}} \tag{4.1}$$

$$\beta = C\sqrt{\frac{\kappa M}{Z\ R\ T}\left(\frac{2}{\kappa+1}\right)^{\frac{\kappa+1}{\kappa-1}}} \tag{4.2}$$

| | | |
|---|---|---|
| Gas Molecular Mass | $M$ | 0.029 for air, $Kg/mol$ |
| Temperature | $T$ | $K^\circ$ |
| Universal Gas Constant | $R$ | 8.31 $(Pa \cdot m^3)/(mol\ K^\circ)$ |
| Discharge coefficient | $C$ | 0.72, dimensionless |
| Compressibility Factor | $Z$ | 0.99 for air, dimensionless |
| Specific Heat Ratio | $\kappa$ | 1.4 for air, dimensionless |
| Mass Flow | $\dot{m}$ | $Kg/s$ |
| Pressure | $p$ | $Pascals$ |
| Area | $a$ | $m^2$ |

Table 4.1: Parameters and units of the thin-plate port model.

$$\dot{m} = af(p_u, p_d) \tag{4.3}$$

$$\phi(p_u, p_d) = \begin{cases} z(p_u, p_d) & p_u \geq p_d \\ -z(p_u, p_d) & p_u < p_d \end{cases} \tag{4.4}$$

$$z(p_u, p_d) = \begin{cases} \alpha p_u \sqrt{(\frac{p_d}{p_u})^{\frac{2}{k}} - (\frac{p_d}{p_u})^{(\frac{k+1}{k})}} & p_u/p_d \leq \theta \\ \beta p_u & p_u/p_d > \theta \end{cases} \tag{4.5}$$

$$\dot{m} = a_c f_p(P_c, p) - a_a f_p(p, P_r) \tag{4.6}$$

$$\dot{p} = \frac{n}{v}(R_s T\dot{m} - p\dot{v}) \tag{4.7}$$

### *4.2.2 Thin port Model With Leak*

The thin-port model presented above does not take leakage from the cylinders into account. To account for this, we consider that the leak is through an equivalent orifice area from the cylinder. We use the same thin port model to predict the mass flow from the chamber. This additional air flow out of the cylinder chamber can then be incorporated into the model. In these equations 4.8 - 4.10. It is important to note that $p_u$ is actually the chamber pressure and $p_d$ is atmospheric pressure. $a_l$ represents the leakage area. The rest of the constants are the defined in the appendix and they are the same values as the ones used in the thin port pressure dynamics model.

$$m_{Leak} = a_l f(p_u, p_d) \tag{4.8}$$

$$\phi(p_u, p_d) = -z(p_u, p_d) \; when \; p_u < p_d \tag{4.9}$$

$$z(p_u, p_d) = \begin{cases} \alpha p_u \sqrt{(\frac{p_d}{p_u})^{\frac{2}{k}} - (\frac{p_d}{p_u})^{(\frac{k+1}{k})}} & p_u/p_d \leq \theta \\ \beta p_u & p_u/p_d > \theta \end{cases} \tag{4.10}$$

The effective mass flow in the cylinder described in equation 4.6 can be modified to equation 4.11.

$$\dot{m} = a_c f_p(P_c, p) - a_a f_p(p, p_r) - a_l f(p, P_r) \tag{4.11}$$

## *4.3 Factors that affect pressure dynamics*

Pressure dynamics of air is highly non linear and has a large time delay due to the slow propagation of air pressure waves through the system. So, it is important to study the factors that might aggravate these effects. For example, valve deadzone adds more non-linearity to the system, length of the connecting tubes introduces proportional amount of delay into the system and leakage in the cylinders directly affects the air mass flow into the cylinder chamber. In this section we discuss these important factors.

### *4.3.1 Inlet and Exhaust port areas*

These are controlled by the valve and have a direct effect on the mass flow into and out of the cylinder chamber. The rate of change in pressure inside the cylinder depends on these areas. The change in the area values with respect to the control is non-linear. This non-linearity has to be understood to predict the air flow from the valves.

### *4.3.2 Valve leakage and deadzone*

Leakage in valves is the undesirable air flow through the inlet and exhaust ports. This has a significant effect in the pressure dynamics inside the cylinder. In an ideal valve, the air flow through the valve at zero command signal should be zero, but the flow rate we measured using a flow meter was 1 $l/min$. The valves also have a control range around midpoint in which no change in the air flow takes place, this is called the valve deadzone. For a small range in the control signal, the valve essentially does not control the air flow. This gives rise to a non linear behaviour.

### *4.3.3 Volume of cylinder*

The over all volume of the cylinder has an important influence on the pressure response inside the chamber. One one hand where the pressure dynamics is known to be notoriously slow, the pressure dynamics of a fully retracted cylinder (approx. volume $0.1cm^3$), exhibits pressure change on the time scale of 10 microseconds. A cylinder with smaller volume and volume fluctuations have a much different pressure response than the one with large volumes and volume fluctuations.

### *4.3.4 Leakage from cylinders*

Air leakage from the pneumatic cylinders also has a significant effect on pressure dynamics inside the cylinder. Some types of pneumatic cylinders tend to leak more than others, this depends on the type of seal that is used between the cylinder piston and bore. Cylinders

that have a rubber seal have lesser leakage, however these cylinders have larger friction. Anti-stiction/friction cylinders overcome the effect of friction by increasing the gap between the piston and the cylinder bore (air-seal), this results in higher leakage.

#### *4.3.5 Delay from connecting tubes*

Delay in the pressure dynamics is affected by the length of the connecting tubes. The longer the tube, more time it takes for the air pressure waves to travel the entire distance of the tube and hence a delay is introduced to the system. The connecting tube also introduce pressure loss because of the friction factor that increases tube resistance.

#### *4.3.6 Input Pressure fluctuations*

The source of the compressed air is not always at constant pressure. While actuating the adroit hand, around 40 cylinders are drawing compressed air, hence pressure fluctuations from the source cannot be avoided. The pressure variations in the source has to be accounted for to increase the accuracy of the model.

### *4.4 Hardware overview and Control*

AIRPEL anti-stiction cylinders are used in the actuation system of Adroit. Based on the joint range and torque requirements, different cylinders varying in stroke length and bore area are used. The Bio-mimetic hand is actuated using AIRPEL M9D37.5 (AIR37) which has a bore diameter of $9mm$ and stroke length of $37.5mm$ because of the small range of motion required for the finger joints. Whereas Adroit robot arm is actuated using bigger cylinders that can provide larger joint range and more torque at each joint. All the cylinders are linear pneumatic actuators. We test our model on two AIRPEL cylinders, M9D37.5 and AIRPEL M9D200 (AR200). We chose these cylinders to compare the effect of the size of the cylinder on the model performance.

2-dof robot is built using different pneumatic actuators, one is a linear double actuation pneumatic cylinder SMC CQ2A32-25DC (SMC), and the other is a rotary double actuation

pneumatic cylinder, PRNA20S-180-45(PRN). The volume of each of the cylinders used is mentioned in Table 4.3. The cylinder characteristics are mentioned in Table 4.2. These cylinders were chosen to test our model's performance on cylinders that have different leakage rates. FESTO MPYE-5-1/8-LF-010-B valves are used to control all the pneumatic actuators. Valves with different flow characteristics and deadzone reagions, shown in Figure 4.4, are chosen to study the effect of these on pressure dynamics. We used connecting tubes that are $2mm$ in diameter. The maximum compressor pressure used was 60 $kPa$, and to address the issue of fluctuating source pressure we used a pressure sensor to monitor the actual pressure from the source.

The pressure inside the cylinder unit is observed using a (SMC PSE540-IM5H3) pressure sensor. The cylinder piston stroke length using a magnetic length sensor (SICK MPS-032TSTU04). In case of the 2-dof robot, the piston stroke position is inferred from the joint angle sensor readings because the cylinders are non-magnetic in nature. The pressure sensors are sampled at 32KHz and the length sensors are sampled at 9Khz. High frequency components of the sensor readings are filtered out, using low pass filters, before they are made available for use. High sampling rate allows us to perform data filtering without introducing significant delays.

| Cylinder | Characteristics |
|---|---|
| AIRPEL M9D37.5 | High leak, small volume |
| AIRPEL M9D200 | High leak, large Volume |
| SMC CQ2A32-25DC | Low leak, large Volume |
| PRNA20S-180-45 | Low leak, small volume |

Table 4.2: Cylinder Characteristics

| Cylinder | Volume($m^3$) |
|---|---|
| AIRPEL M9D37.5 | 2.3856e-06 |
| AIRPEL M9D200 | 1.2723e-05 |
| SMC CQ2A32-25DC | 2.0106e-05 |
| PRNA20S-180-45 | 3.50e-06 |

Table 4.3: Volume values for each cylinder

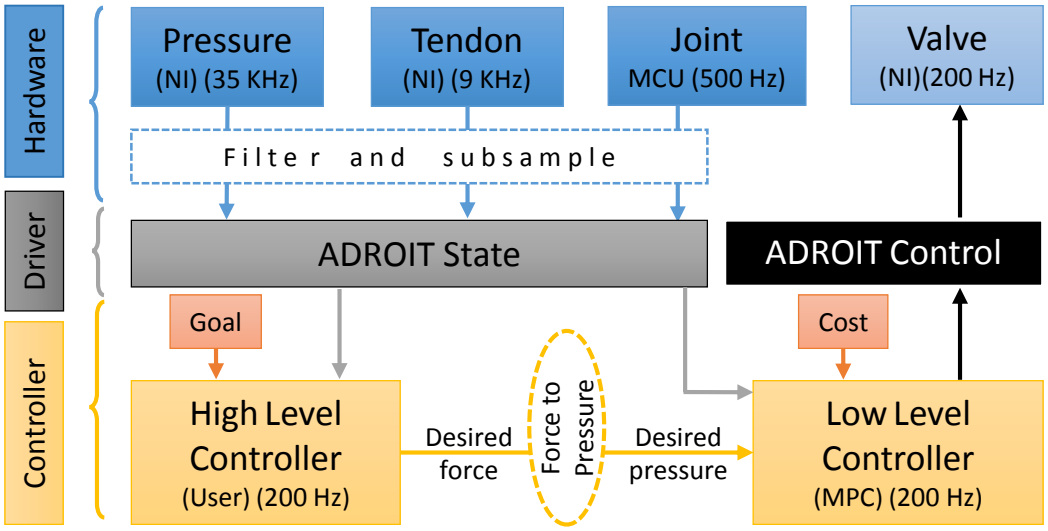



Figure 4.3: The model based controller architecture being developed for Adroit Manipulation Platform

### *4.4.1 Control Architecture*

The overall control architecture is illustrated in the Figure 4.3. The three main components of the architecture are ($a$) Hardware: which consists of all physical components like sensors, control valves and electronics, ($b$) Driver : where the data from the sensors are calibrated and assembled as states and controls, ($c$) Controller : which uses the states as an input and sends the desired controls to the valves. The controller itself operates at two levels. At the beginning of each control cycle (running at 200Hz) using the current state the high level

controller submits a desired torque profile to the low level controller pretending that robot is driven using a ideal torque actuator. The "force to pressure" module accounts for the transmission to map torques to desired pressures. The low level controller subsumes the complexities of the pneumatics and abstracts out a simple torque actuator to the high level controller for planning purposes. The low level controller (detailed in Chapter 5) leverages online trajectory optimization techniques and pneumatic models to handle the complications of the pneumatic actuation in order to execute the request submitted by the high level controller.

## *4.5 Experimentation and Results*

In this section, we describe the experimental method used to identify the parameters of thin port model. First, to understand the flow characteristics we measure the air flow from the valves using a flow meter. Second, we measure the pressure response from pneumatic cylinder for a particular set of command signals. We use this data to optimize for the parameters. The experimental method is explained in detail in the next few paragraphs.

### *4.5.1 Valve Characterization*

As described in the previous section, pressure changes inside a cylinder chamber is a function of the air mass flow. Characterizing this flow from the valve is crucial in understanding the pressure changes inside the chamber. Equation 4.3 describes the mass flow through an area $a$, the valves control this area by using solenoid actuation that moves a spool when a control input is applied. The control range of the valve is $0 - 10V$, we map this to a $-5$ *to* 5 range for easy representation. A command signal $-5$ *to* 0 opens the exhaust port and a command signal 0 *to* 5 opens the inlet port of the valve. To understand this behaviour of the valve as a function of the control input, we used a standard air flow meter (SFAB-200U-HQ8-25V-M12 from FESTO). The flow measured here is in $l/min$. The Figure 4.4 illustrates the data obtained from the flow sensor. It is important to note that the valves are far from ideal and there is some flow in both the directions as the command input is changed. The flat region

around the zero command in Figure 4.4 is the deadzone in the valve.

The bandwidth of the valve is dependent on the amplitude of the spool displacement, and for a full length displacement of the spool, the bandwidth is 125Hz. Since the bandwidth of the pneumatic system is much lower than 100Hz, the dynamics of the valve itself can be neglected while modelling the pressure dynamics.

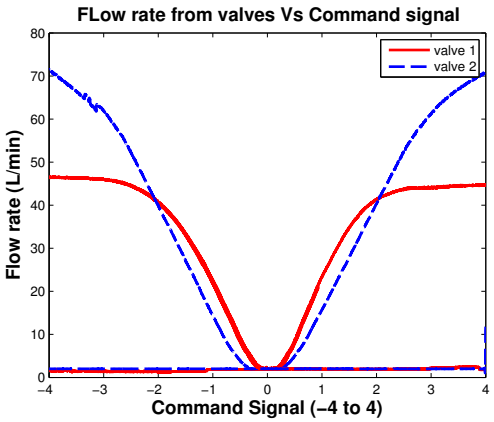



Figure 4.4: This Figure shows flow measurement from two different valves. Because of the deadzone, the Valves have no response for a small range around the zero input command. The response of the two valves differ significantly.

### *4.5.2 Optimization*

Three parameters need to be identified in order to implement the thin port model for pressure predictions, first the mapping from command signal to the inlet and exhaust port area $a_c$ and $a_a$ respectively, the nature of this function is illustrated in Figure 4.4 where we can see that the there is always constant small flow in both the directions due to leakage. The second parameter that needs to be identified is the volume of the cylinder $v$. In addition to this, we also need to estimate the effective leak area $a_l$ described earlier. These parameters can be identified using data driven optimization techniques. We collect the pressure response data by first fixing the cylinder at a particular volume. Then we apply random step commands

to the valve. By minimizing the error between the measured rate of change in pressure and the rate of change in pressure predicted by model we estimate values for the variables $a_c, a_a, a_l, v_o$. This is done for a whole range of control inputs. By doing this, we obtain a mapping of the area with respect to the control signal. If this experiment is repeated at different fixed volumes, then we can also build a function that maps sensor readings to volume of the chamber inside.

This optimization problem can be set up to identify the desired area and volume parameters. In equation 4.12, $\dot{P}_m$ is the measured rate of change of pressure inside the cylinder. The other terms are described in section 4.2.

$$\underset{\mathbf{a_c},\mathbf{a_r},\mathbf{v},\mathbf{a_l}}{argmin}\{\dot{P}_m - \frac{nR_sT}{v}(a_c f(P_c, p) - (a_a + a_l)f(p, P_r))\} \tag{4.12}$$

The built-in optimization tool box in MATLAB was used to solve the optimization problem described above.

*Inlet and Exhaust areas*

The identified input and exhaust valve areas are shown in Figure 4.5(a). It is important that the function that represents the area as a function of the command signal be a continuous function because any discontinuity will introduce undesired non-linear behaviour. We chose the gompertz function, given in equation 4.13, to represent the area function and used standard matlab curve fitting tool to obtain the parameters $offset, a, b, c \; and \; d$. The values obtained for different valves are presented in table 4.4. *cmd* is the control input to the valve.

$$area = offset + (a)exp^{(-b)exp^{(-c)(cmd+d)}} \tag{4.13}$$

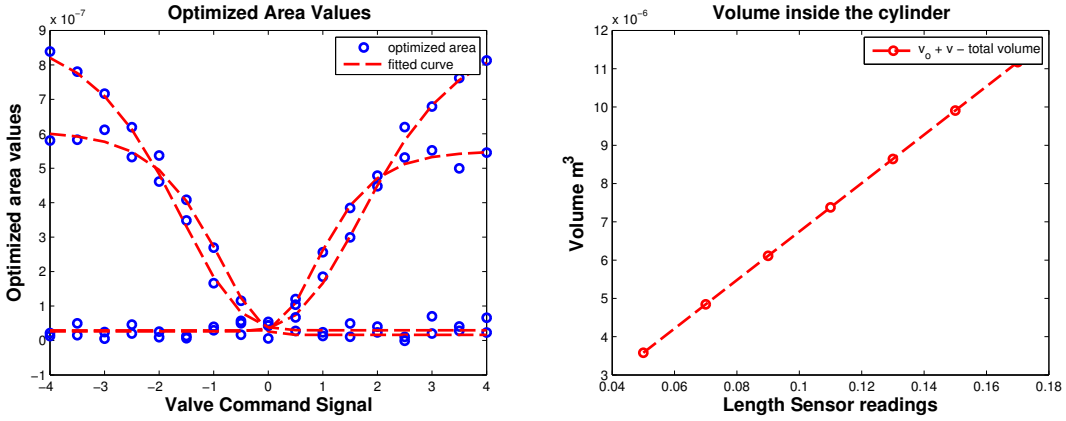
*Volume*

The volume of the cylinder during actuation is extremely important in building an accurate model for the pressure dynamics. The volume inside the pneumatic cylinder can be estimated

| valve | offset | a | b | c | d |
|---|---|---|---|---|---|
| Valve 1 inlet area | 2.9274e-08 | 5.501e-07 | 3.539 | 1.564 | 0.12 |
| Valve 1 exhaust area | 2.6076e-08 | 6.079e-07 | 3.149 | -1.365 | -0.1 |
| Valve 2 inlet area | 2.627e-08 | 8.731e-07 | 4.029 | 0.929 | -0.1505 |
| Valve 2 exhaust area | 2.964e-08 | 8.659e-07 | 3.942 | -0.9816 | 0.1584 |

Table 4.4: Optimized values for the mathematical model of inlet and exhaust ports of the valves. Illustrates how each valve has different characteristics

by sensing the position of the piston and multiplying it with the bore area, but this does not include the dead volume in the chamber. By optimizing for the volume, we are essentially estimating the dead volume of the cylinder also. Based on the optimized values, we represent the volume as a function of the sensor readings directly. As expected, the volume function obtained through optimization was a linear function. This is illustrated in Fig. 4.5(b).




(a) Valve orifice area

(b) Volume

Figure 4.5: The Figure (a) shows the optimized values for two different types of valves. The difference in the flow characteristics is captured by optimization, hence taking into account the valve leakage. Figure (b) shows the optimized volume function

*Leakage area*

For Airpel cylinders, the optimized value of the orifice diameter through which leakage is assumed to take place was found to be $1.9665e-04m$.

## *4.6 Results*

### *4.6.1 Pressure Predictions*

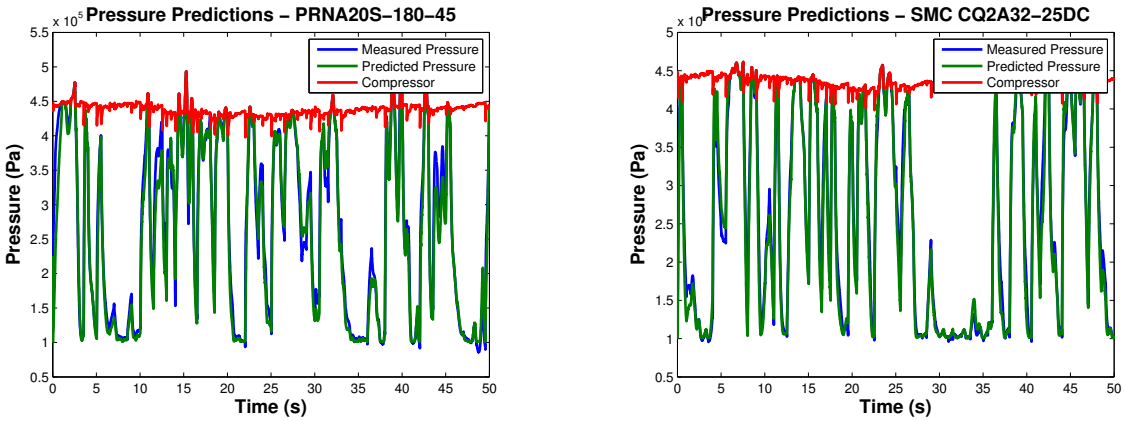
In this section, the pressure modelling results are investigated on actual hardware. To illustrate the effectiveness of the identified model, we apply fast changing random commands to the valves and manually provide aggressive volume changes to the cylinders. Then we compare the pressure changes predicted by the identified model to the measured pressure changes during the hardware experiment. Pressure modelling results on all four cylinders are presented. An example of the control signal and volume change during the data collection is illustrated in Figure 4.8. It is important to note that the pressure is measured in Pascals (Pa).

Figures 4.6b and 4.6a illustrates the pressure modelling achieved from PRN and SMC cylinders.

| Cylinder | RMSE | Percent of pressure range |
|---|---|---|
| SMC | 1.476e+04 | 3.2 |
| PRN | 1.369e+04 | 3.042 |
| AR200 | 5.161e+04 | 11.4 |
| AR37 | 1.370e+04 | 3.04 |

Table 4.5: The RMSE values for the pressure predictions are given in this table.

Figures 4.7a and 4.7b show the pressure modelling achieved in AR37 and AR200 cylinders. In this Figure, the light blue line represents the volume change in the cylinder. As the figures




(a) Pressure predictions for PRN cylinder (b) Pressure predictions for SMC cylinder

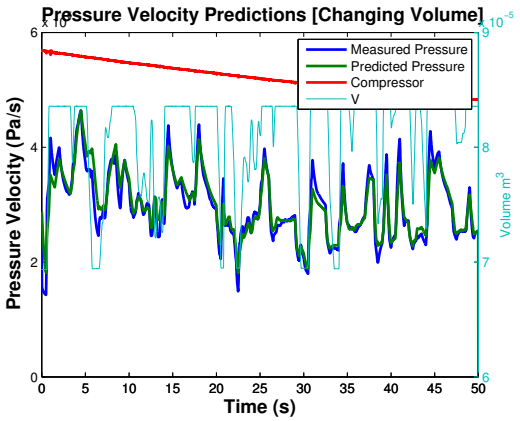
Figure 4.6: Pressure predictions

illustrate the identified model is robust and is able to match the pressure changes measured inside the cylinder. The RMSE values obtained are shown in Table 4.5

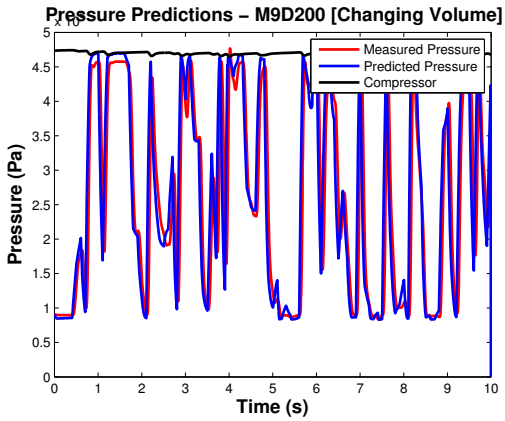
## *4.7 Discussion and Conclusion*

In this section, we analyse the performance of the model based on the error in pressure predictions on the different cylinders. We discuss how our model is affected by parameters such as the leakage and deadzone in the valves, leakage from cylinders and large volume. We also highlight some of the shortcomings of the model and the reasons for them.

The model based low level controller leverages the pneumatic model to look through the planning horizon and selects the optimal policy that results in effective pressure predictions. So, we are most interested in the prediction of the model for a short duration of about $2s$ into the future. We initialize the model to initial measured pressure and let the model predict the pressure changes for the next $2s$. To check the model performance, we take the mean of prediction error for multiple two second trajectories and assess them based on comparing the error as a percentage of the working pressure range.

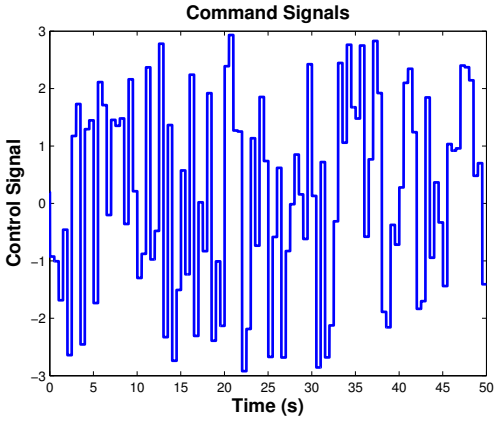



(a) Pressure predictions for AR37 cylinder

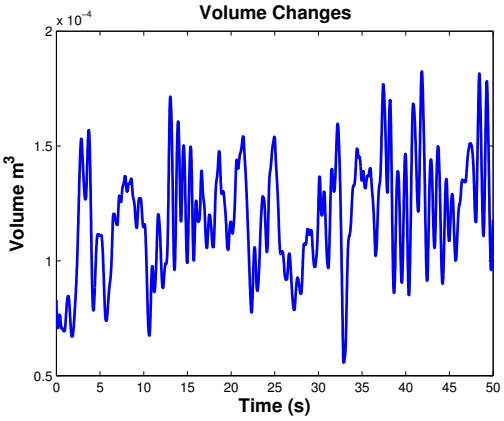



(b) Pressure predictions for AR200 cylinder

Figure 4.7: Results of Pressure modelling for AR37 and AR200. The model is able to predict the pressure changes inside the cylinder based on the control signal and the volume changes, which was the primary goal of the system identification.




(a) n example of the type of random control signal used for prediction




(b) the volume change the cylinder is subjected to during data collection

Figure 4.8: Data collection for pneumatic modelling and testing

From the results we obtained, the pressure predictions were well under 4% of the working pressure range for all the cylinders except AR200. The reason for the relatively poor performance of the model with AR200 is a combination of the leakage and larger volume.

### *4.7.1 Effect of Volume*

Larger volume has an adverse effect on the performance of the model because the $v$ term in the thin port model has more impact on the overall pressure dynamics of the system. This effect is illustrated in the Figure 4.9(a), Both Airpel cylinders suffer from leakage, but it is much harder to predict the pressure changes in the AR200 with a larger volume. Figure 4.9(a) also shows the comparison between SMC and PRN actuators. The prediction error in SMC is more because to its larger volume.

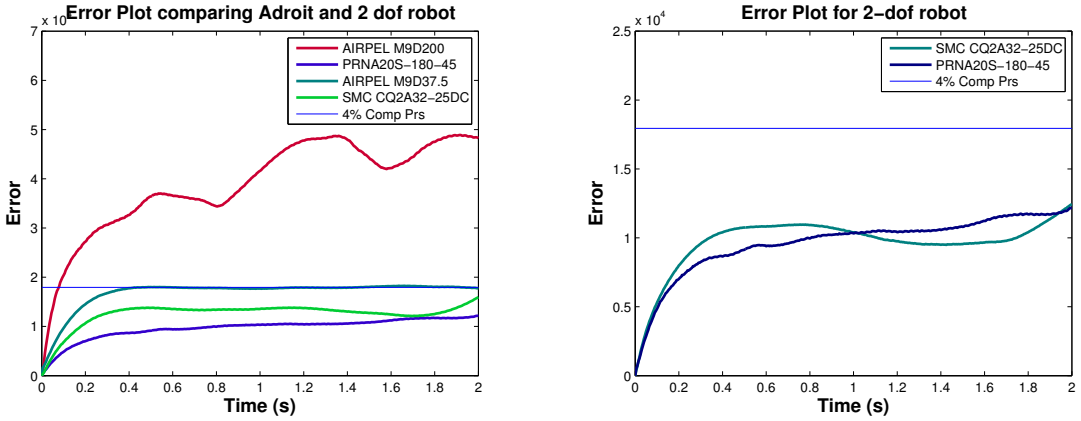
### *4.7.2 Effect of valve deadzone*

The valve characteristics is hard to model around the zero command signal because of the non linear behaviour introduced by the valve deadzone around that control region. This leads to some error in prediction for control signals in this region. This is shown in the Figure 4.10(b), the error obtained by subjecting the pneumatic system to a random fast changing control signal between the range -0.1 to 0.1 is larger than the error obtained for a random control input between the range -3 to 3. This is an important factor when the model is implemented with a controller, because when the pressure in the cylinder has to be maintained at a particular value, the controller operates in this small range around the deadzone. However, the errors are still less than 4% of the working pressure range, so the area function comes close to modelling this region.

### *4.7.3 Effect of leakage in the cylinders*

Figure 4.9(a) compares the performance of the identified model for SMC, PRN, AR37 and AR200 cylinders, as mentioned earlier, the leakage from SMC and PRN cylinders is negligible

compared to the leakage from AR37 and AR200 cylinders. Hence, it is not surprising that the errors in SMC and PRN are lesser. However, Modelling the leak based on our assumption that the leak takes place through an equivalent orifice area from the cylinder does reduce the error. This is illustrated in 4.10(a).



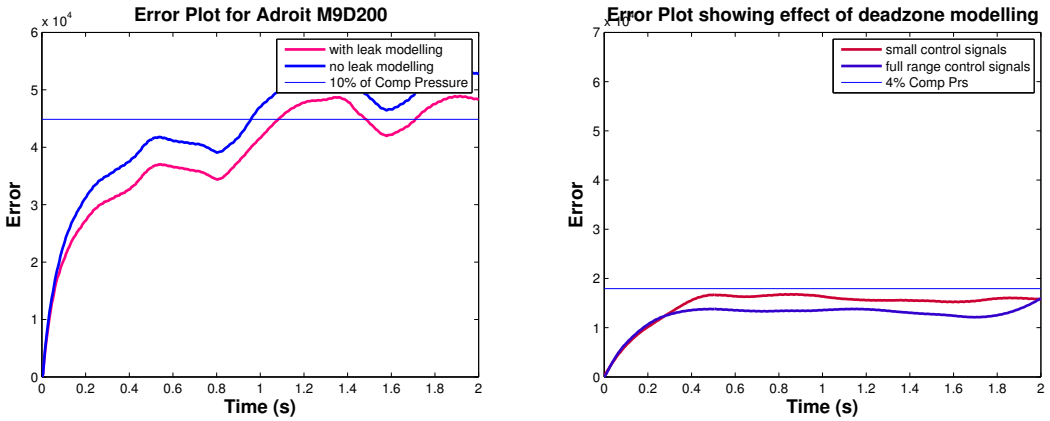

(a) Prediction errors across cylinders (b) Prediction errors in SMC and PRN cylinders

Figure 4.9: Pressure prediction errors across different cylinders. Airpel cylinders that suffer from more leakage have larger error

This chapter presents our method for identifying a pneumatic model for Adroit manipulation platform. The effects of valve leakage, valve dead zone, leakage in cylinders and input pressure variations are taken into account to increase the accuracy of the model. Pressure predictions achieved for the different cylinders satisfy the accuracy and robustness requirement for implementing it in a controller. We also highlighted the factors that determine the difficulty in modeling a pneumatic system by comparing the errors in the prediction.

## *4.8 Future Work*

Including the effects of the tube lengths to improve the overall prediction accuracy is left as a future work.




(a) Reduction in error by modelling the leak (b) Dead zone modelling

Figure 4.10: Figure (b) illustrates the difficulty in modelling the valve around the zero control region because of non linear behaviour due to valve dead zone. The red line is the error in prediction when the control range is close to zero but the blue line is the prediction error over a larger control range. It is easier to estimate the inlet and exhaust port areas for larger control signals.

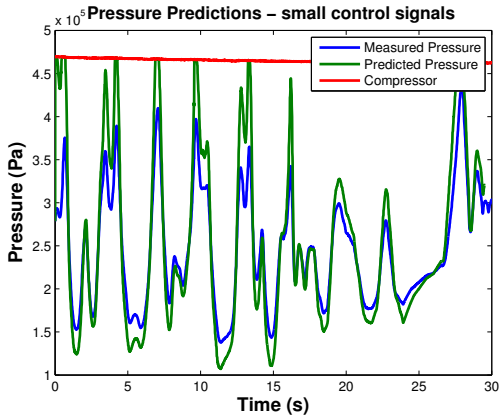



Figure 4.11: This Figure shows the increased error in pressure predictions by the identified model for control signals close to zero.

### *4.9 Next Steps*

In the following chapter, we will leverage the pneumatic models developed here to build high-performance low-level pneumatic controllers. These controllers are built on a key insight – biological muscles have time scales similar to that of pneumatics, yet they are quite effective and extremely efficient. The effectiveness of the biological muscles can be attributed to the fact that they act in anticipation. It's not clear if this insight can be leveraged to extract performance from pneumatics actuators. In next chapter we outline techniques that leverage accurate pressure dynamics models to build anticipation in pneumatic actuators, thereby reducing the overall latency of the system.

# Chapter 5

# *ADROIT* MANIPULATION PLATFORM: HIGH PERFORMANCE PNEUMATICS

High force density, compact form factor and muscle like compliance properties makes pneumatics actuators quite desirable for robotic applications. However, the compressibility of the air not only throttles their bandwidth but also makes them harder to control. In this chapter, we leverage online trajectory optimization techniques to design high performance controllers for pneumatic actuators of ADROIT manipulation platform. These controllers use future predictions to act in anticipation. We show that the proposed controllers prepares the system ahead of time in order to achieve better performance and use less controls while doing so. Hardware results are presented on the pneumatic actuators of the ADROIT platform. Controller's robustness is evaluated using hardware variants.

## *5.1 Introduction*

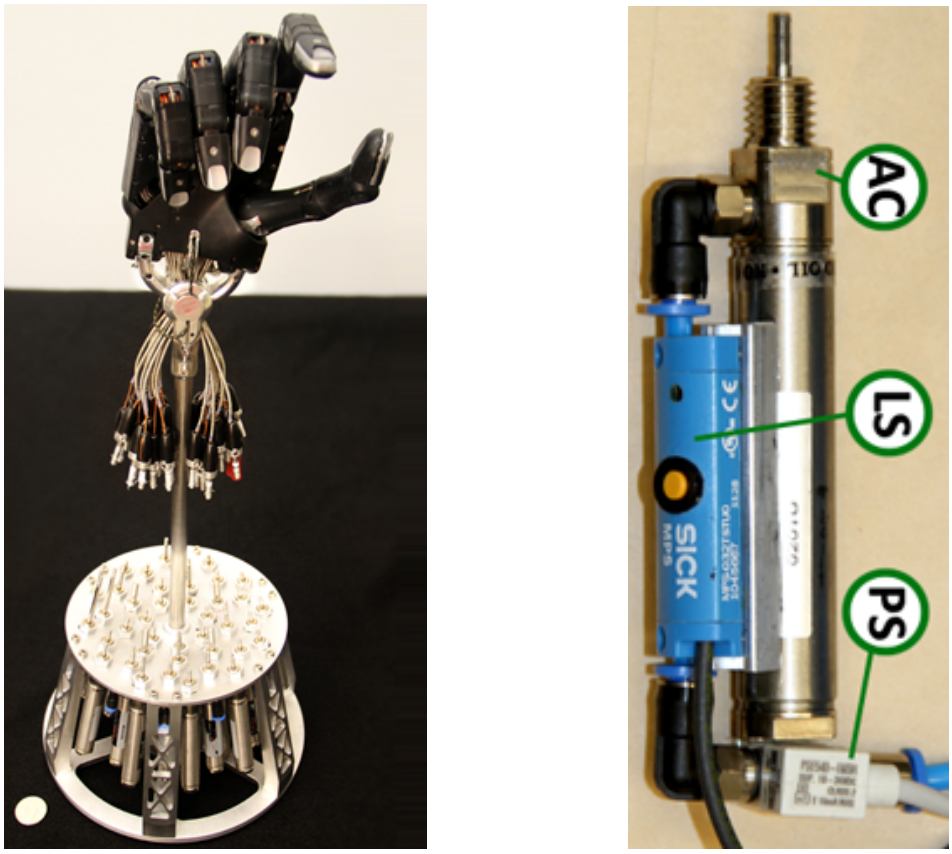
Search for better actuators continues as robotic devices move from being hard and stiff, to being soft, nimble, agile and compliant. Substantial improvements are still needed (spanning safety, form factor, price point etc.) to kick-start the era of personal robotics. Fluid based actuators have several desirable properties relevant to the present and upcoming needs of such devices. They are inexpensive, mechanically simple with few moving parts and light weight with high force to weight ratio. Due to their compact form factor and high force density, they can be mounted directly on the moving degree of freedom (DoF), eliminating the need of gears and transmissions. Robustness, low friction and direct mounted on a DoF makes them ideal for force control applications.

However, the benefits of fluid based actuators have been overshadowed by the complexity

of the control techniques they require. For applications where the dynamics of the fluid doesn't need excitation, deployment of simple linear controllers have made these devices widely successful. They are the defacto actuators for industrial automation needs, commercial heavy weight equipments, load bearing and transmission mechanisms, power tools etc. For applications such as robotics, where the dynamics of the fluid needs to be accounted, controller design still remains a challenge. As the bandwidth of fluid based actuators (specially hydraulics) is improving, with the advancements in the value technology (MOOG), they are increasingly being used in the fast dynamics applications – Spot, Atlas [15], HyQ quadruped [91], Cheetah [96].

Bandwidth of the pneumatics actuators are lower than that of its hydraulic counterpart due to compressibility of the air, resulting in timescales of the order of $100ms$. On the other hand pneumatics is cleaner, lighter, quieter and easier to operate. They have properties similar to biological muscles which make them quite desirable for biological applications seeking compliance – (1) they are back drivable and compliant at the mechanism level; (2) they have internal activation state (air pressure in the case of pneumatics, calcium concentration in the case of muscles) whose dynamics makes the entire system 3rd-order; (3) Pressure dynamics effectively introduces a low pass filter between the command and the force with timescales similar to that of biological muscles; (4) Much like biological muscles, co-contraction can be achieved by tuning the stiffness of the antagonistic counterparts. Compressibility is often considered a liability, but it is quite desirable at the same time. Spring dampers have long been used in mechanisms design for desirable passive behaviors. Pneumatics is the most generic form of an actuated spring damper with tunable gains. There is no doubt that – if efficient controllers are realised, pneumatics actuators (fluid actuators in general) will gain strong traction, specially in robotic applications.

Our interest in synthesising complex behaviors [46] [98] [29] for biological systems and the desirable properties of pneumatic actuators led us into the development of ADROIT manipulation platform [48] – which is a modified Shadow Hand with custom pneumatic actuation (Figure 5.1). Our specific motivation for designing pneumatic controllers roots in the need of




(a) ADROIT hand mounted (tendons disconnected) on the pneumatic muscle assembly

(b) Pneumatic muscle [AC: Actuator, LS: Length Sensor, PS: Pressure Sensor]

Figure 5.1: Hardware setup

high performing low level controllers for ADROIT. In order to extract performance out of a system with large timescale and low bandwidth, effective planning through the dynamics of overall system is required. Pneumatic dynamics modelling has received significant attention in the past – parametric [122] as well as physical models [85] have been developed. We build and improve on these pressure dynamics models as we exploit ideas from the field of model based trajectory optimization to design a high performance pneumatic controller for our system.

## *5.2 system*

We first describe our system (Figure 4.3) with relevant technical details. Later, we present experimental observations that highlight properties relevant for controller design.

### *5.2.1 Hardware*

ADROIT platform houses a 24 DoF bio-mimetic hand (Figure 5.1a). 20 Dofs are independently actuated using 40 antagonistic *pneumatic muscle actuation units* (Figure 5.1b) while the rest (finger's distal joints) are coupled. ADROIT's control-loop runs are 200 Hz, sampling 135 sensors and commanding 48 actuators using a 9 slot NATIONAL INSTRUMENTS 3U PXIe-1078 chassis[1](NI). The chassis is configured from the *computational unit* [2] using a PXIe-PCIe 798MB/s bandwidth data channel.

A low friction, low stiction double acting pneumatic cylinder (*AIRPEL M9 37.5NM, AIRPORT Corporation*, [4]) forms the muscle actuation unit. As muscles can only pull, the rear chamber of the cylinder is left passive and open to the atmosphere. Each unit has stroke length of $37.5mm$, can produce $42N$ of force at 100 PSI and weighs about 37.5 grams. *Airpel* replaces traditional pneumatic seals with "air seals" in order to achieves low friction and low stiction conditions. While air seals are desirable for smooth movements, the constant leakage from the seals makes the non linear pneumatic dynamics further chaotic and hard to model.

The muscle actuation unit is observed using two sensors. The pressure inside a muscle unit is observed using a solid state (*SMC PSE540-IM5H*[3]) pressure sensor. The muscle excursion is measured as piston stork length using a magnetic length sensor (*SICK MPS-032TSTU0*[4]). The pressure sensors are sampled at 32KHz and the length sensors are sampled

[1]The chassis has a data rate of 1Gb/s, 250 Mb/s bandwidth per slot.

[2]12 cores 3.47GHz Intel(R) Xeon(R) processor with 12GB memory running Windows x64

[3]The pressure sensor can measure up to $10^6 pascal$ with $< 2\%$ resolution, $< \pm 0.7\%$ linearity, $< \pm 0.2\%$ repeatability and weighs 4.6 grams

[4]The length sensors can measure up to 32 mm excursions with $0.05mm$ (about $2mm$ in practice) resolution,

at 9Khz. High frequency components of the sensor readings are filtered out, using low pass filters, before they are made available for use. High sampling rate allows us to perform data filtering without introducing significant delays. Joint angle sensors are sampled using an embedded micro-controller(MCU) at 500Hz.

A high flow 5/3 festo proportional valve is used to drive the front chamber of the muscle actuation unit. The proportional valve (MPYE-5-M5-010-B from *FESTO*) has a flow rate of 100 liters/min at 87 PSI, bandwidth of 125 Hz and weighs 290gms.

### *5.2.2 Hardware exploration*

We explored the asymptotic pressure response (Figure 5.2) of the ADROIT's pneumatic muscle unit by subjecting the volume locked assembly to voltage step changes from either extremes. The input voltage affects the flow through the valve. Ideally, the asymptotic response of the value should be a step function. Near the zero point (center of the input range) of the valve, it is partially connected to both the source (compressor) and the sink (atmosphere), resulting in intermediate asymptotic values. The pressure dynamics is significantly slower around the zero point as illustrated by the time required to reach the asymptotic pressure curve. Impulse response obtained from the step changes, for two different tube lengths, are illustrated in Figure 5.3. Higher latencies for the longer tubes can be attribute to the time required for pressure wave to travel though them.

The steady state responses (Figure 5.4) of the muscle unit was obtained as slow varying voltage signals sweep the input range of a volume locked assembly. Normalization wrt the compressor pressure reveals its dependence on the compressure pressure, which changes as we use the system. The fat belly around the center illustrates the nondeterministic nature of the spool (hence the pressure) around the valve's zero position. A pneumatic valve spends most of its time around the nominal positions. The nondeterminism around the nominal position might pose significant challenge for controller design.

---

$< 0.3mm$ linearity, $< 0.1mm$ repeatability, The sensor has sampling time of $1ms$ and can sense movements up to a max speed of $3m/s$.

The muscle assembly was subjected to chirp signals to study the frequency response of the over-all pneumatic circuit. The critical frequency was found to be around $25Hz$.

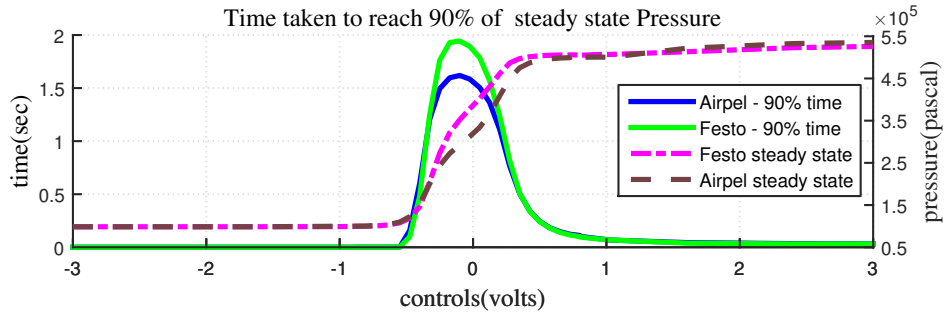



Figure 5.2: Asymptotic pressure response of Adroit's actuators

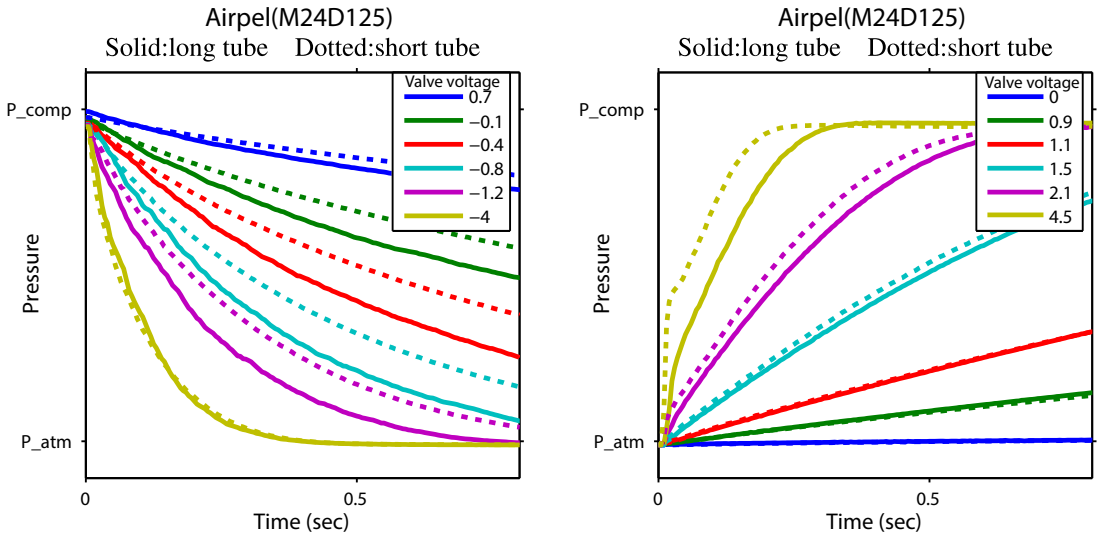
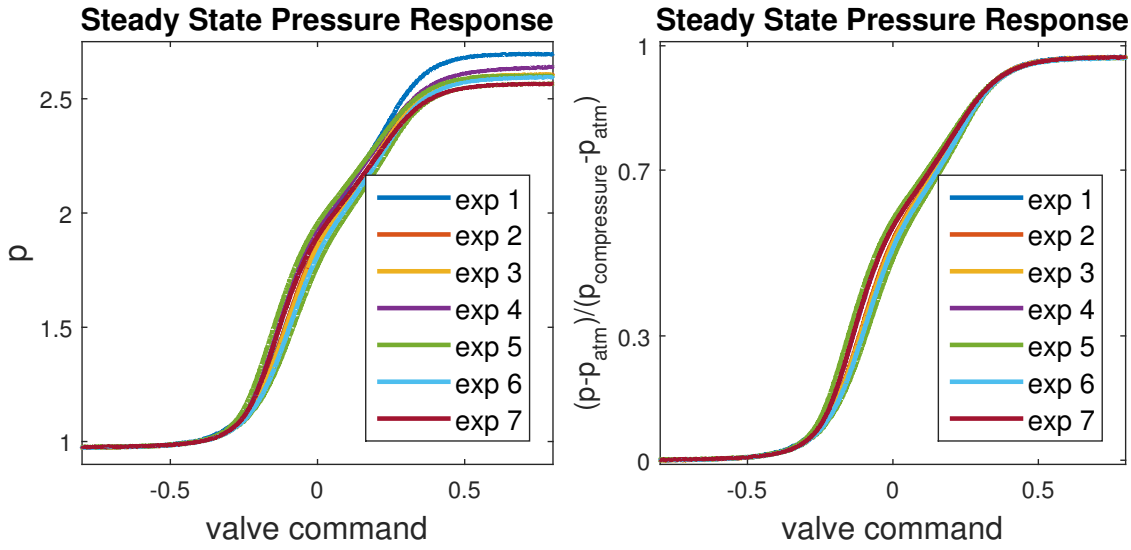



Figure 5.3: Impulse responses for different tube lengths

## *5.3 Pneumatics*

### *5.3.1 Cylinder*

A cylinder (Figure 5.7) is a device with two chambers, separated by a moving bore. Each chamber has an orifice, called port, that connects it to a pressure reservoir. The reservoir




(a) Raw pressure measurements

(b) Normalized relative-pressure

Figure 5.4: Steady state pressure response

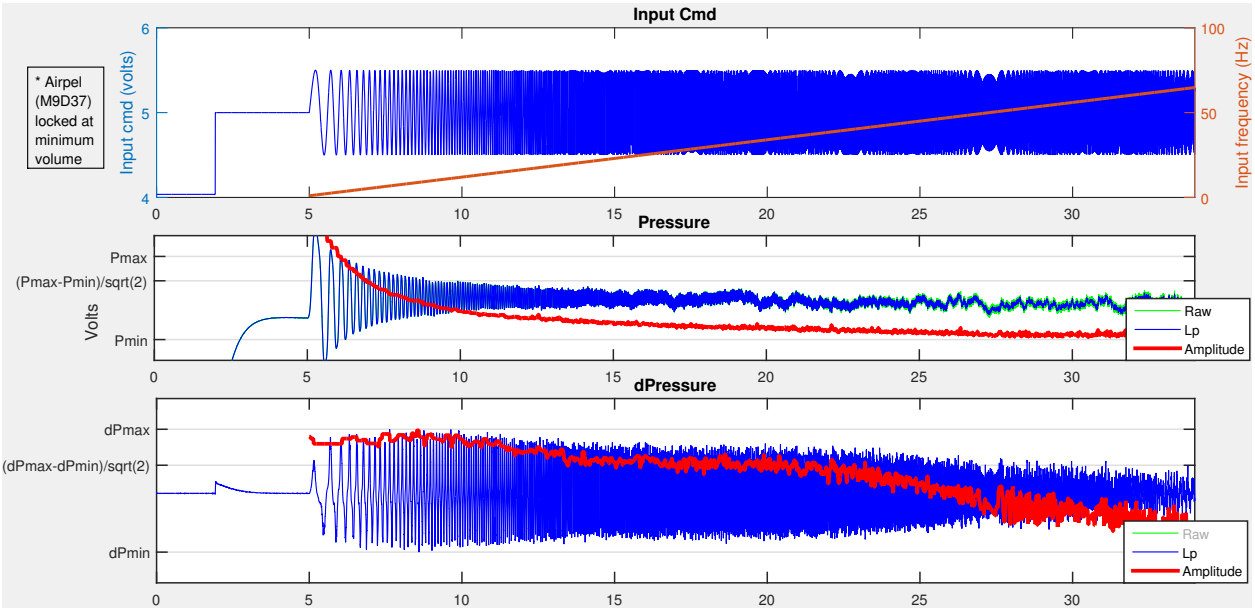



Figure 5.5: Pneumatic system bandwidth

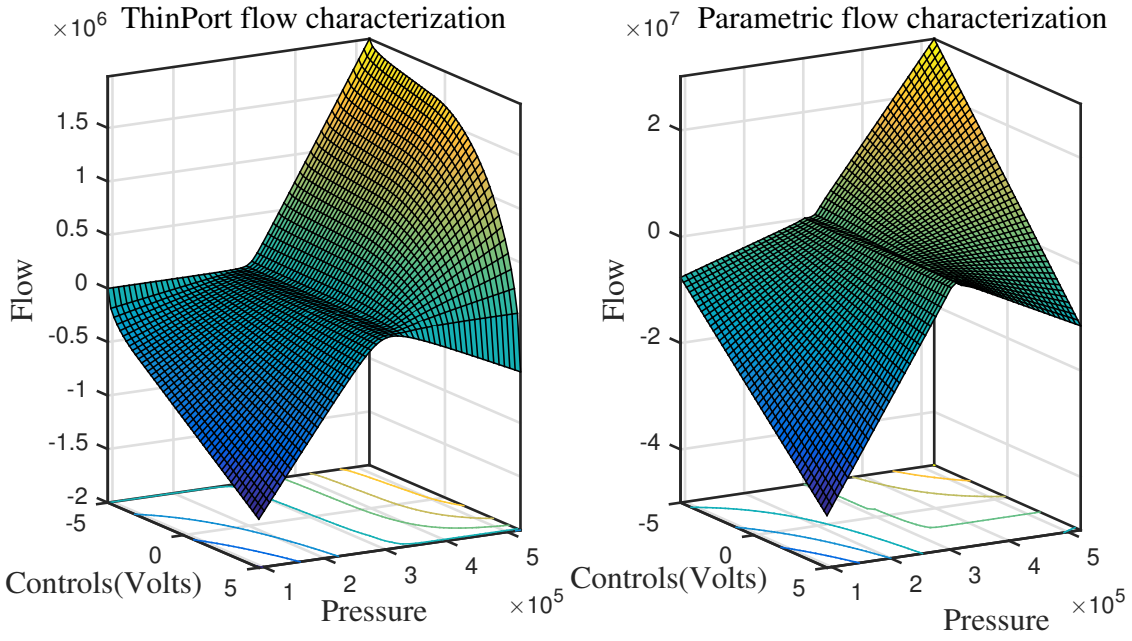



Figure 5.6: Flow characterization of ADROIT's muscle unit

with pressure higher (usually air compressor) than that of the chamber is called the source and the corresponding port is called the supply port. The reservoir with pressure lower (usually atmosphere) than that of the chamber is called the sink and the corresponding port is called the exhaust port.

### *5.3.2 Valve*

The pressure inside a chamber can be regulated by connecting a valve to it's port. A valve is a mechanical device that connects a chamber to multiple reservoirs (usually two - a source and a sink) and regulates the cross-sectional area opening between them. Two commonly used types of valves are (a) binary on/off valves which use Pulse Width Modulation scheme for area modulation and (b) proportional solenoid valves. Unlike binary valves, proportional valves are expensive but provide fine grained control over the port area resulting in smooth operation. A solenoid actuated spool moves inside the valve in response to the

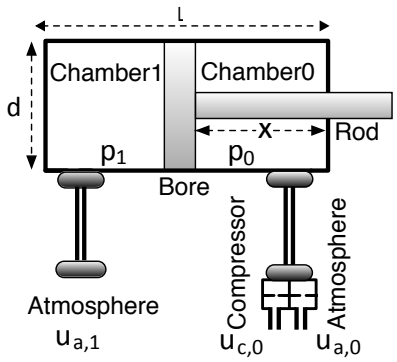



Figure 5.7: Schematic representation of a pneumatic cylinder with a port controlled by a value and the other port open to the atmosphere

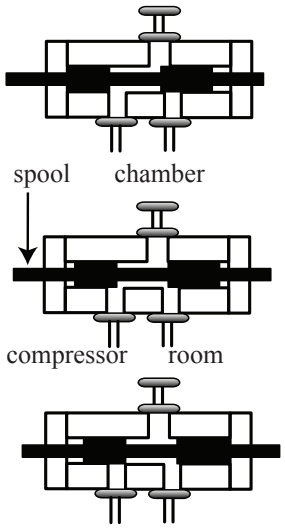



Figure 5.8: Schematic representation of a proportional pneumatic value.(Top: supply port open. Center: Both exhaust & supply port partially open. Bottom: Exhaust port open)

input command thereby smoothly changing the port's cross-sectional area (Figure 5.8).

### *5.3.3 Thin Plate Port pressure dynamics model*

The thin plate port model (Figure 5.9) explains the flow of fluid (air) through a port as a function of the area of the orifice, the upstream pressure $p_u$ and the downstream pressure $p_d$. Assumption being that the plate connecting the chambers is thin, the port area is small, fluid in use is a perfect gas, both the chambers are at the same temperature and the flow is isentropic. Thin plate port model is common in the pneumatics control literature. [85] is an excellent article that reviews and builds on previous work in the light of real time control applications. Modifications, with justified assumptions, have been proposed to handle computational complexity while accounting for prediction accuracy necessary for

such applications. The model is briefly summarized below for completeness.

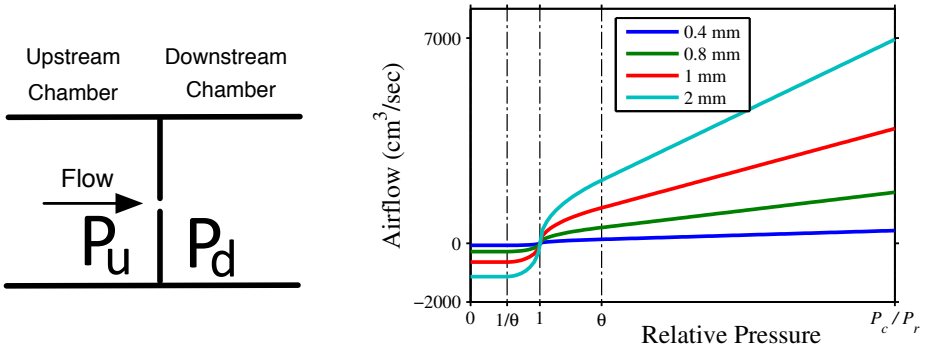



Figure 5.9: (a)**Thin Plate.** (b)**Thin Plate Flow Function.** The 4 curves show the flow rate for 4 orifice diameters. The downstream pressure $p_d$ is kept at room pressure $P_r \approx 100_{kPa}$, and $p_u$ is varied from 0 to the compressor pressure $P_c \approx 620_{kPa} \approx 90_{psi}$. The linear regime is outside the dotted vertical lines.

*Thin plate flow function*

Thin plate flow function $\phi(p_u, p_d)$ describes the mass flow $\dot{m}$ through a port per unit port area $a$.

$$\dot{m} = a \cdot \phi(p_u, p_d) \tag{5.1a}$$

$$\phi(p_u, p_d) = \begin{cases} z(p_u, p_d) & \text{if } p_u \geq p_d \\ -z(p_d, p_u) & \text{if } p_u < p_d \end{cases} \tag{5.1b}$$

$$z(p_u, p_d) = \begin{cases} \alpha\, p_u \sqrt{\left(\frac{p_d}{p_u}\right)^{\frac{2}{\kappa}} - \left(\frac{p_d}{p_u}\right)^{\frac{\kappa+1}{\kappa}}} & \text{for } p_u/p_d \leq \theta \\ \\ \beta p_u & \text{for } p_u/p_d > \theta \end{cases} \tag{5.1c}$$

The physical constants $\kappa$, $\alpha$, $\beta$ and $\theta$ are described in 4.1 and 4.2

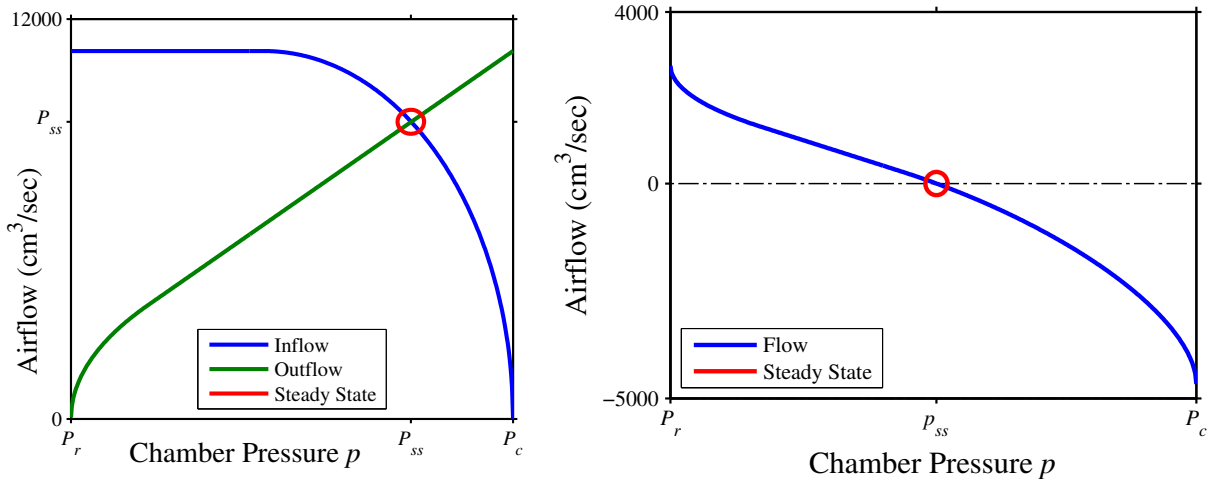
Note that the flow function (Figure 5.9) is continuously differentiable and is linear in the upstream pressure $p_u$ for $p_u > \theta p_d$.

*Two port chamber*

The net mass flow into a chamber with two ports can be described as

$$\dot{m}(p, a_c, a_r, p_c, p_r) = a_c\phi(p_c, p) - a_r\phi(p, p_r) \tag{5.2}$$

where $a_c, a_r$ are the orifice areas connecting the chamber to the compressor and room respectively, and $p_c, p_r$ are the respective constant pressures. Figure 5.10 shows this function to be monotonically decreasing, which corresponds to stable dynamics that converge to a steady-state pressure $p_{ss}$ given by $a_c\phi(p_c, p_{ss}) = a_r\phi(p_{ss}, p_r)$.




(a) Individual flows

(b) Net flow

Figure 5.10: Two port chamber

*Chamber pressure dynamics model*

Using ideal gas laws, two port net flux ((5.2)) and polytropic process with isothermal constraints, pressure dynamics of a single chamber can be written as

$$\dot{p}(p, u, \mathrm{v}, \dot{\mathrm{v}}, p_c, p_r) = \kappa \frac{RT}{\mathrm{v}} \dot{m} - \kappa \frac{\dot{\mathrm{v}}}{\mathrm{v}} p \tag{5.3a}$$

$$\dot{m}(p, u, p_c, p_r) = a_c(u)\phi(p_c, p) - a_r(u)\phi(p, p_r) \tag{5.3b}$$

where v is the volume of the chamber, $\dot{\mathrm{v}}$ is the rate of change of that volume and $\kappa, R$ and $T$ are physical constants. The first term in the pressure dynamics equation ((5.3a)) captures the dynamics dictated by the net flow into the chamber, while the second explains the dynamics due to the changing chamber volume.

### *5.3.4 Parametric pressure dynamics model*

The over all time scale of a pneumatic system depends on the valve dynamics, the delays in the pneumatic circuit, and the chamber volume ((5.3a)). While the pneumatic is slow in general, a fully retracted cylinder with a low chamber volume can have a time constant of the order of microseconds. As a result, a generic parametric form of pressure dynamics will need extremely small time step or a variable time integrator in order to integrate the dynamics forward.

$$\dot{p}(p, u, v, \dot{v})|\mathbf{c} = \big(s(u, \mathrm{v}, \dot{\mathrm{v}})|\mathbf{c} - p\big) \cdot r(u, \mathrm{v}, \dot{\mathrm{v}})|\mathbf{c} \tag{5.4}$$

(5.4) summarises the parametric model that we zeroed in our prior work [122]. Note that it is linear with respect to $p$, allowing allows us to perform analytical integration. As a result it is stable for arbitrary time step, and computationally inexpensive to evaluate. The function $s()$ has a unit of pressure and represents the steady state pressure. The $r() > 0$ has a unit of inverse of time and represents the rate of change. The system is differentiable if $s()$ and $r()$ are differentiable.

Here we present a slightly modified parametric model (note that the parametric form is still the same) that is more general and delivers better predictions.

$$\dot{p}(p,u,v,\dot{v},p_c,p_r)|\mathbf{c} = \big(c_b + c_s sigmoid(c_3\hat{u} + c_4\hat{u}^3) - p\big)\frac{c_7\big(sabs(\hat{u}) + c_8\hat{u}\big) + c_9}{\tau} + \frac{c_5\dot{v}p}{\tau} \tag{5.5}$$

Where $c_b = (p_c + p_a)/2$ represents the mid point pressure, $c_s = (p_c - p_a)/2$ represents the pressure range, $\hat{u} = u - c_2$ is centered control voltage, $\tau = 1 + c_6$, $sigmoid(z) = z/\sqrt{1+z^2}$ and $sabs(z) = \sqrt{z^2 + c_\gamma^2} - c_\gamma$. Note that $c_s$ and $c_b$ are no longer fixed constants based on maximum and minimum pressure (as treated in [122]). They now are inputs to the model that need to be updated using real time pressure readings.

## *5.4 Model identification*

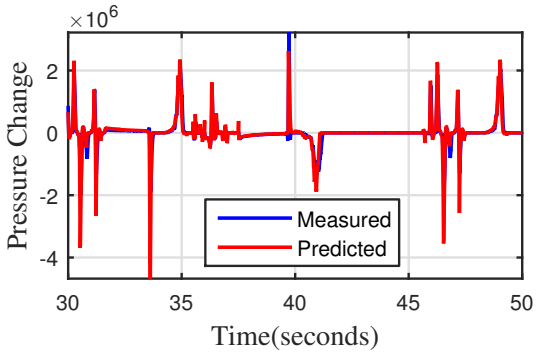
### *5.4.1 Thin port model*

The dependence of port area on input voltage $a_c(\mathbf{u}), a_r(\mathbf{u})$ is not known and is not possible to measure directly. We obtain this dependency using numerical optimization. A volume locked chamber ($\dot{\mathbf{v}} = 0$) was subjected to step voltage changes starting from each voltage extreme. We solve a small optimization problem to recover $\{a_c^{u_i}, a_r^{u_i}\}$ pair for each voltage step. Standard curve fitting techniques was used to fit a parametric form (gompertz function 5.6) to the resulting area pairs. Refer [44] for details.

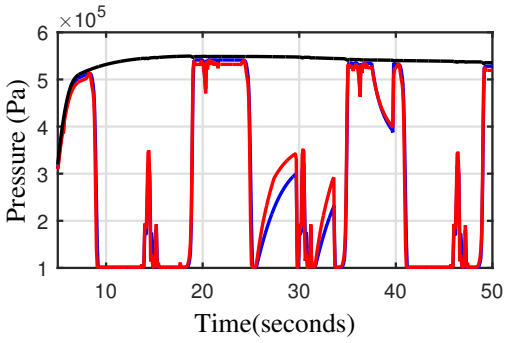
$$\{a_c^{u_i}, a_r^{u_i}, v_o\} = \underset{a_c,a_r,v_o}{argmin}\big\{\dot{P}_{measured}^{u_i} - \kappa\frac{RT}{v+v_o}\big(a_c^{u_i}f(P_c,p) - a_r^{u_i}f(p,P_r)\big)\big\}$$

$$a(\mathbf{u})|c = c_0 + c_1 e^{-c_2 e^{-c_3(\hat{\mathbf{u}}+c_4)}} \tag{5.6}$$

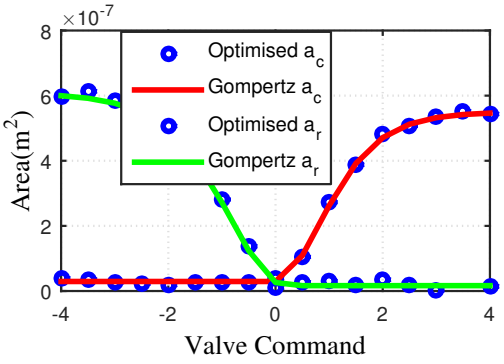
Where $c_0$ is the minimal leak area of the port, $\hat{\mathbf{u}} = (\mathbf{u} - c_1)$ is the centered control voltage. Figure 5.11c outlines the recovered area pairs and the parametric area fits. Figure 5.11a and 5.11b presents the $\mathbf{p}$ and $\dot{\mathbf{p}}$ predictions using the identified model.




(a) pressure change predictions

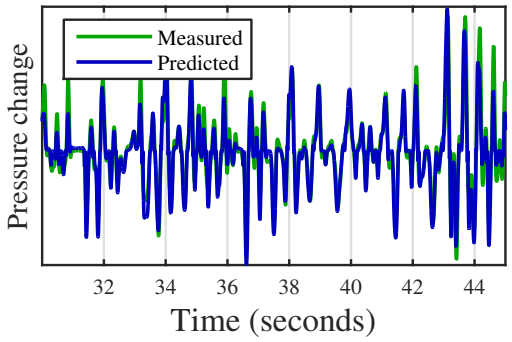



(b) pressure predictions

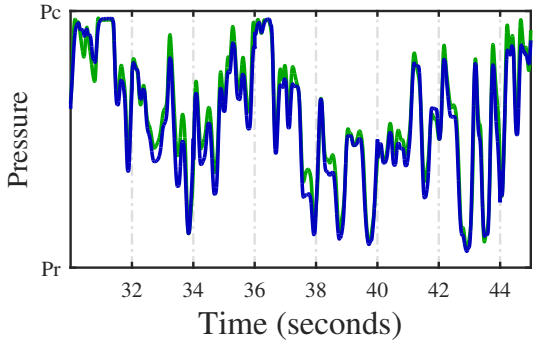



(c) Port area

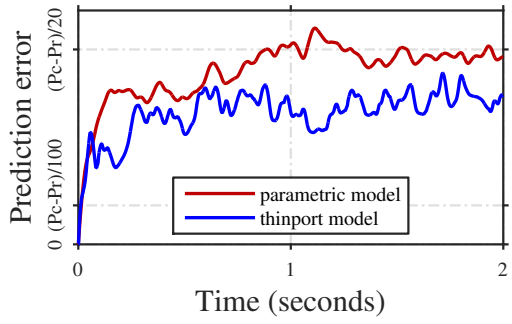
Figure 5.11: **Thin Port model identification**




(a) pressure change predictions




(b) pressure predictions




(c) pressure prediction errors

Figure 5.12: **Parametric model identification**

### *5.4.2 Parametric model*

*Steady state response*

In order to capture the steady state response $s(u, \mathrm{v}, \dot{\mathrm{v}}; \mathbf{c})$ of our pneumatic system, a volume locked chamber was subjected to slow varying control input. The slow varying control input drags the system equilibrium along as it sweeps the entire input range. We fit a steady state model of the form $p_{steady\ state} = c_b + c_s sigmoid(c_3\hat{u} + c_4\hat{u}^3)$ to the resulting data. Figure 5.13 summarizes the system response and the model predictions wrt time(left) and input voltage (right). Unlike the predictions (black curves) from our old model [122] with fixed $c_s$ and $c_b$, the predictions from the model using real time $c_b$ and $c_s$ readings nicely captures the dependence of the steady state pressure on the compressor pressure.

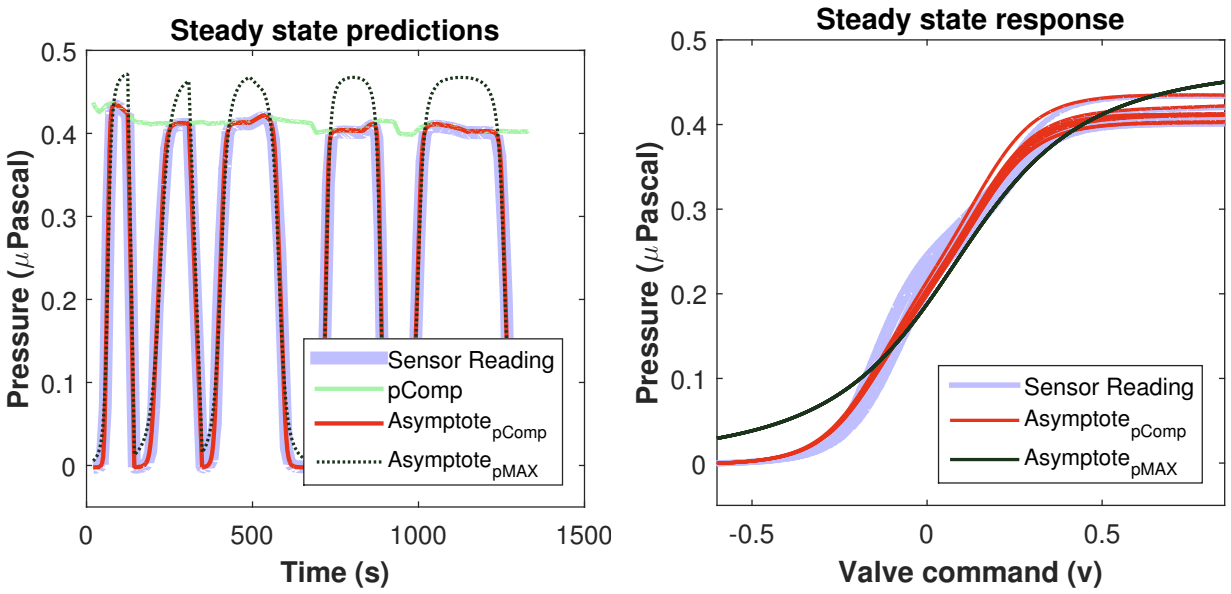



Figure 5.13: **Steady state predictions**. Unlike the predictions $Asymptote_{pMAX}$ from model using constant $c_s$ and $c_b$, the predictions $Asymptote_{pComp}$ from model using real time $c_s$ and $c_b$ nicely captures the dependency of the steady state pressure on compressure pressure.

*Rate*

For data diversity, the cylinder was overdriven by applying external forces to the cylinder's piston (in order to diversify the chamber's volume $v$ and the rate of change of volume $\dot{v}$) while it was subjected to random signals for rate $r(u, \mathrm{v}, \dot{\mathrm{v}}; \mathbf{c})$ identification. As we aren't explicitly modelling the valve dynamics, the random signals were low passed filtered (30 Hz butterworth) below the valve's critical frequency (125 Hz) before they were executed on the hardware.

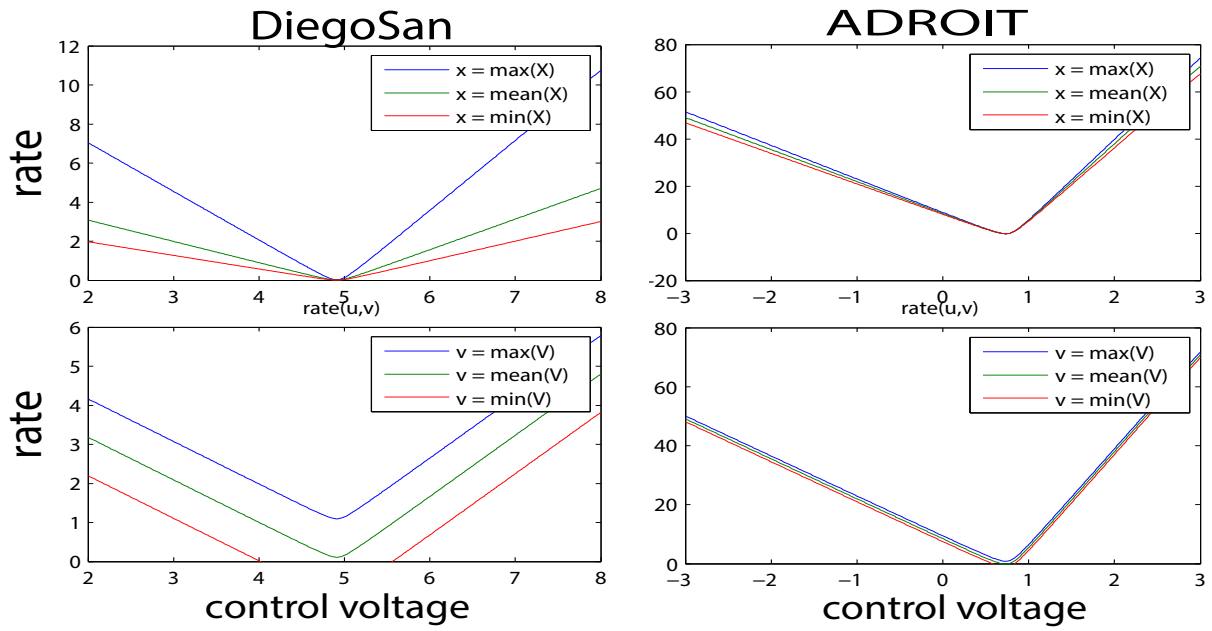



Figure 5.14: Rate comparison between Adroit and DieogSan's [122] actuators. Adroit's actuators have much higher rates and are fairly insensitive to bore position(x) & velocity(v)

The rate (Figure 5.14) reveals valuable information about the dynamics, and its sensitivity, of our pneumatic system. A system with high flow rate valve and small chamber volume should have fast dynamics. Which is indeed the case – the rate of the identified model grows rapidly with control input. While fast dynamics means better throughput, it also means that the system is highly sensitive to the control input and timing delays. Small control input

will be enough to drive the system (Section 5.6) and the wide proportional regime of our expensive valves will be hard to exploit. We also observe that rate is relatively insensitive to the volume $v$ and volume rate $\dot{v}$. This can be attributed to the fact that the chamber has small volume and stroke length. As a result, the changes induced by the bore movement is insignificant. This insensitivity can be exploited to simplify the pressure model[5].

$$\begin{aligned}\dot{p}(u, p_c, p_r)|\mathbf{c} = &\big(c_b + c_s sigmoid(c_3\hat{u} + c_4\hat{u}^3)\\ &- p\big)\big(c_7\big(sabs(\hat{u}) + c_8\hat{u}\big) + c_9\big)\end{aligned} \tag{5.7}$$

### 5.5 *High performance pneumatic controller*

High level motion planners have seen significant advancements in recent times. We now have planners that can plan in real time for high DoFs robots [46] [98] [29]. The planners are no longer short sighted and impulsive. They reason about the movements in longer time horizon and opt for instant replanning if the execution derails form the plan. Most of these improvement primarily stem from the fact that we now have more computational resources at our disposal, that are being utilized for better understanding of the system via simulation, data analysis, optimization, learning etc.

Pneumatic systems are knows to have significant latencies. Compressibility of the air results in long time constants(Figure 5.3) that throttles the bandwidth of the system. Local feedback controllers are unable to deal with these challenges to deliver performance required by a dynamical system. The proposed controller does that by leveraging the dynamics model of the system to unfold the system forward in time and reasoning about the actions over a longer time window (called horizon). The predictive capability from the pneumatics models (from Section 5.4), lets our controller make anticipatory preparation for actions well in advance, resulting in improved performance. We outline the necessary details of our controller below

[5]We haven't simplified the models for our use case at this point as the simplification doesn't hold for the arm cylinders which are significantly bigger than the hand cylinders. From the software architecture standpoint, we would like to treat all the cylinder identical, if possible.

### *5.5.1 Controller design*

Instead of having a myopic view presented by the immediate desired pressure value $\mathbf{p_{dest}}$, our controller design focuses on designing a policy using a macroscopic view as presented by a desired pressure trajectory over a time horizon $\mathsf{T} = \mathsf{N} * \mathbf{dt}$. Given the desired pressure trajectory $\mathbf{P_{des}} = [\mathbf{p_{dest}}, \mathbf{p_{dest+dt}}, .., \mathbf{p_{dest+(N-1)dt}}]$, the goal of our controller is to find the appropriate policy, in the space of valve commands, that guides the system though $\hat{\mathbf{p}}_{des}$ over time. We pose the policy design problem as a finite horizon optimal control problem and deploy standard trajectory optimization techniques, iterative-LQG[109] in this case, for efficient solutions in real time.

### *5.5.2 Finite horizon optimal control*

Given the state $\mathbf{x}_i$, controls $\mathbf{u}_i$ at time $i$, let the discrete time dynamics of a system be described as $\hat{\mathbf{x}} = \mathbf{x}_{i+1} = f(\mathbf{x}_i, u_i)$. The finite horizon optimal control problems can be posed as – starting from an initial state $\mathbf{x}_0$ solve for a time varying control law $\hat{\mathbf{U}}(\mathbf{x})$ that minimizes the cumulative sum of the *running cost* $\mathbf{l}_i(\mathbf{x}_i, \mathbf{u}_i)$ and the *final cost* $\mathbf{l}_f(\mathbf{x_N})$ along a trajectory.

$$\hat{\mathbf{x}} = \mathbf{x}_{i+1} = f(\mathbf{x}_i, u_i) \tag{5.8}$$

$$\hat{\mathbf{U}}(\mathbf{x}) = \text{argmin}_{\mathbf{U}} \sum_{i=0}^{N-1} \mathbf{l}_i(\mathbf{x}_i, \mathbf{u}_i) + \mathbf{l}_f(\mathbf{x_N}) \tag{5.9}$$

### *5.5.3 Trajectory optimization*

Our choice of trajectory optimizer (iterative-LQG[109]) solves the problem above using the principles of dynamic programming. The value $\mathbf{V}(\mathbf{x}, i)$ corresponding to a state $\mathbf{x}$ at time $i$ indicates the minimal cost incurred to optimally solve the problem for remaining $\mathsf{N} - i$ steps. The final values function $\mathbf{V}(\mathbf{x}, \mathsf{N}) = \mathbf{l}_f(\mathbf{x_N})$ is just the final cost. Dynamics programming principle reduces the minimization over a sequence of controls $\mathbf{U}_i$ to a sequence of

**Algorithm 1** Trajectory Optimization

**Input:** Dynamics $f(\mathbf{x}, \mathbf{u})$, running cost $\mathbf{l}_i(\mathbf{x}_i, \mathbf{u}_i)$, final costs $\mathbf{l}_f(\mathbf{x_N})$, current state $\mathbf{x}_0$, warm-start sequence $\mathbf{U}$.

**Output:** Locally optimal control sequence $\hat{\mathbf{U}}$

1: *Rollout*: Integrate $\mathbf{U}$ to get the initial trajectory$(\mathbf{x}_i, \mathbf{u}_i)$

2: *Derivatives*: Get derivatives for $\mathbf{l}_i$ and $\mathbf{l}_f$

3: *Backward pass*: Calculate the second order approximations of $\mathbf{V}(\mathbf{x}, i)$. Obtain a search direction $\Delta\mathbf{U}$ as 2nd-order solution to the (5.10)

4: *Forward Pass*: Rollout $\mathbf{x}_0$ and $\mathbf{U} + \alpha\mathbf{U}$ forward with different line search parameters $0 < \alpha < 1$ to pick the winner

minimization over a single control, proceeding backwards in time

$$\mathbf{V}(\mathbf{x}, i) = \min_{\mathbf{u}}[\mathbf{l}(\mathbf{x}, \mathbf{u}) + \mathbf{V}(f(\mathbf{x}, \mathbf{u}), i + 1)] \tag{5.10}$$

An iteration starts with a *Rollout*, which integrates the dynamics (5.8) forward in time. *Backpass* computes the second-order approximations of the derivatives of $\mathbf{V}$ wrt $\mathbf{x}_i$, to obtain a search direction $\Delta\mathbf{U}$ as 2nd-order solution to the (5.10). A line search is performed along the search direction, as part of the *Forward Pass*, by rolling out multiple trajectories with control sequences $\mathbf{U}_0 + \alpha\Delta\mathbf{U}$ for different values of $\alpha$ $(0 < \alpha < 1)$, to select the winner.

The algorithm is outlined in Algorithm 1. We recommend [109] for in-depth analysis and [46] [98] [29] for more applications.

### *5.5.4 Model Predictive Control (MPC)*

The goal of this chapter is to abstract out the pneumatic actuators as ideal torque actuators. The controller in consideration will form the low level controller of the ADROIT actuation system. The goal is to design low level controllers that emulate ideal force controllers using non-liner pneumatic actuator. As the system is always in motion for the low level controllers, the $\mathbf{P_{des}}$ (specified by the high level controllers) is constantly flowing with time, even when

the high level controller is demanding a constant torque. If the system dynamics was linear and the cost we deploy was quadratic we will find the optimum in single iteration of the trajectory optimization Algorithm 1. The pressure dynamics is highly nonlinear and the cost we use is not quadratic either. Therefore, the algorithm needs multiple iterations to work through the linear approximation of the dynamic and quadratic approximation of the cost to converge on an optima.

As the system is always in motion, we deploy the trajectory optimization in a Model Predictive Control (MPC) fashion. Which means instead of solving for the optimum, starting from our current state estimates, we will only take few iteration of the algorithm, improve the policy for the current estimates, and then opt for an estimates update. There are multiple rational behind this choice – (1) The model used for planning will never be perfect and we will never reach the true optimum even if we have it; (2) Solving for optimum is computationally expensive. Fast policy update provides better performance by dragging the system closer to the optimum with each update; (3) The low level controller has no controls over the demands of the high level planner. The high level planner can decide to abruptly change the plan. The best choice for the low level controller is to respect its demands as soon as possible.

### *5.5.5 State and System Dynamics*

The state $\mathbf{x} = [\mathbf{p}, \mathbf{w}]^T$ of our system is of dimensionality $2n_a$, where $n_a$ is the number of actuators in the system (40 for ADROIT hand). It consists of *cylinder pressure* $\mathbf{p}$ and *valve memory* $\mathbf{w}$, which is the controls (valve voltages) from the previous time step. System dynamics can be written as

$$\mathbf{x} = [\mathbf{p}, \mathbf{w}]^T ; \mathbf{x_{t+dt}} = f(\mathbf{x_t}, \mathbf{u_t})|_{\mathbf{v},\dot{\mathbf{v}}} = [\mathbf{p_{t+dt}}, \mathbf{u_t}]^T \tag{5.11}$$

Where $\mathbf{p_{t+dt}}$ is obtained using the pressure dynamics models of likes outlined in Section 5.3.3 (by euler integration of Eq5.3) and 5.3.4 (Eq5.4 can be exactly integrated forward). It is important to note that the volume $\mathbf{v}$ and volume rate $\dot{\mathbf{v}}$ is not a part of the system's state $\mathbf{x}$. They are required as inputs to the pressure dynamics and enter the system as external

sensor readings. This is possible because, given the volume $\mathbf{v}$ and volume rate $\dot{\mathbf{v}}$, the pressure dynamics is independent of the dynamics of the piston (and the external load its driving). Similarly, given the net force on the piston, the piston dynamics is independent of the pressure dynamics. Since the sensor for measuring chamber pressures and piston stroke lengths are cheap and reliable (and available in ADROIT), we exploit this independence.

### *5.5.6 Cost Function*

The running cost of the system is of the form

$$\begin{aligned} \mathbf{l}_i(\mathbf{x}_i, \mathbf{u}_i) = \alpha_{\mathbf{p}}.||\mathbf{p}_{\mathbf{des}i} - \mathbf{p}_i|| + \alpha_{\mathbf{u}}.||\mathbf{u}_{\mathbf{des}i} - \mathbf{u}_i|| \\ + \alpha_{\mathbf{w}}.||\mathbf{u}_i - \mathbf{w}_i|| \end{aligned} \tag{5.12}$$

The final cost $\mathbf{l}_f(\mathbf{x}_\mathsf{N})$ is same as the running cost. The justification for including the *valve memory* $\mathbf{w}$ in the state can be found in the last term of the cost i.e. $\alpha_{\mathbf{w}}.||\mathbf{u}_i - \mathbf{w}_i||$. This term forces the optimizers to pick similar controls for the time steps adjacent to each other. Thus ensuring smoothness in the final control sequence that executes on the hardware. Without this term, the optimizers are free to pick arbitrary sequence of chattering controls, as long it minimizes the cost objective. Rapidly chattering control sequence keeps the pneumatic valves always active and on the edge of their critical frequency, resulting in decreased performance and hardware wear. $\alpha_{\mathbf{w}}$ can be adjusted to tune the amount of smoothing required.

### *5.5.7 Simulation results*

We use a 1 DoF platform with pneumatic actuator to compare the performance of different controllers under identical conditions. We prefer simulated platform for this comparison as it allows us to study a controller's performance independent of the unpredictability of the real world. Figure 5.15 outlines the performance of our trajectory optimization based controller with the PIDF controller [104]. Unlike the myopic nature of the PIDF which acts only after the change is demanded, anticipatory nature of the iLQG controllers results in lower latency

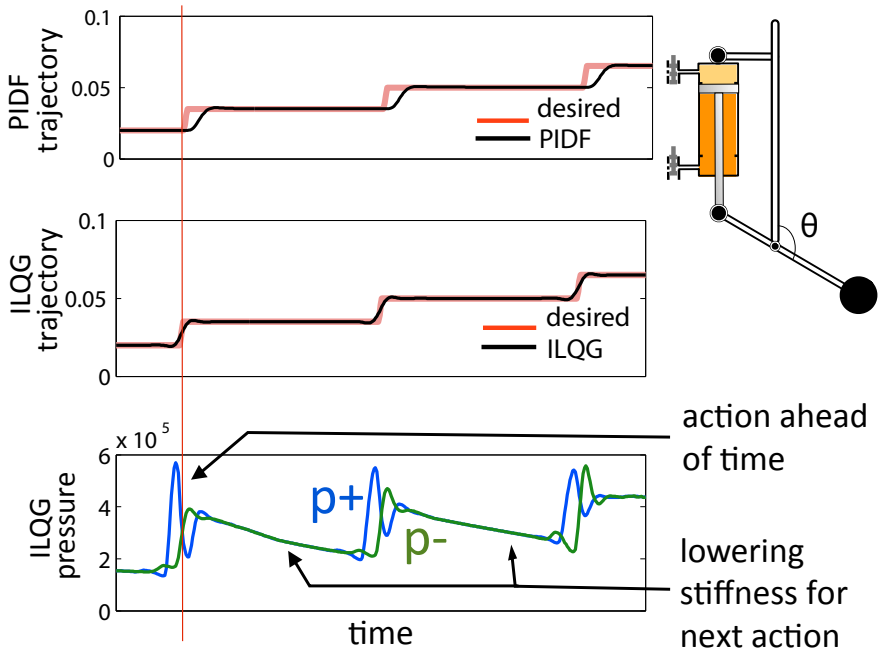



Figure 5.15: Controller comparison

and better performance. Close attention to the pressure highlights pneumatic activity ahead of time and stiffness lowering when possible.

## *5.6 Controller performance*

While identifying the parametric model (Figure 5.14) we learned that, because of the small volume of our cylinders, the pressure dynamics is quite insensitive of piston position and velocity. Leakage of the ADROIT pneumatic muscle unit further complicates the nonlinear dynamics. To circumvent these complications, we first present our controller's performance (Figure 5.16 & 5.17) on a leak free volume locked cylinder (Festo DSN104P).

### *5.6.1 Trajectory Optimization and MPC details*

To have all the optimization entities roughly in the same scale, we use pressure in Mega Pascal. The pneumatic controller and the hardware driver are two separate processes com-

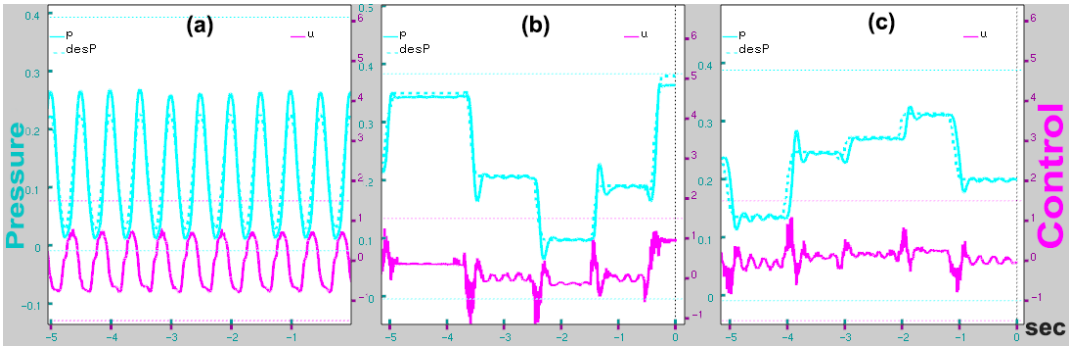



Figure 5.16: pressure tracking for a volume locked FESTO cylinder+ Thin Port + short tube. (a) 2Hz (b) Random (c) Instantaneous change

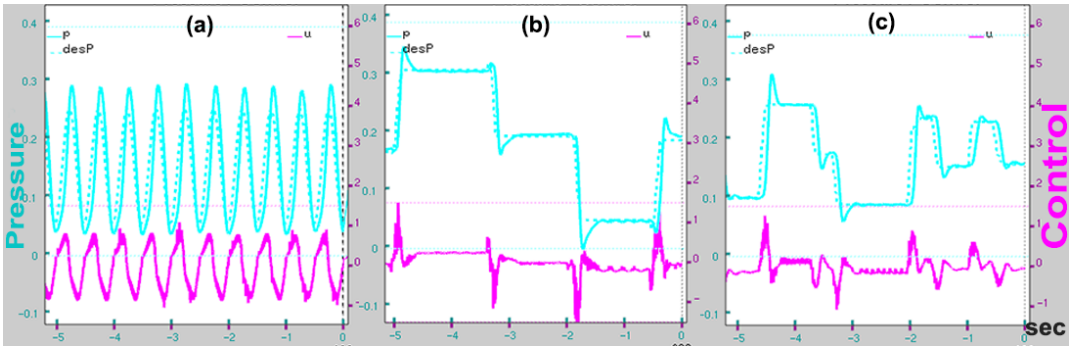
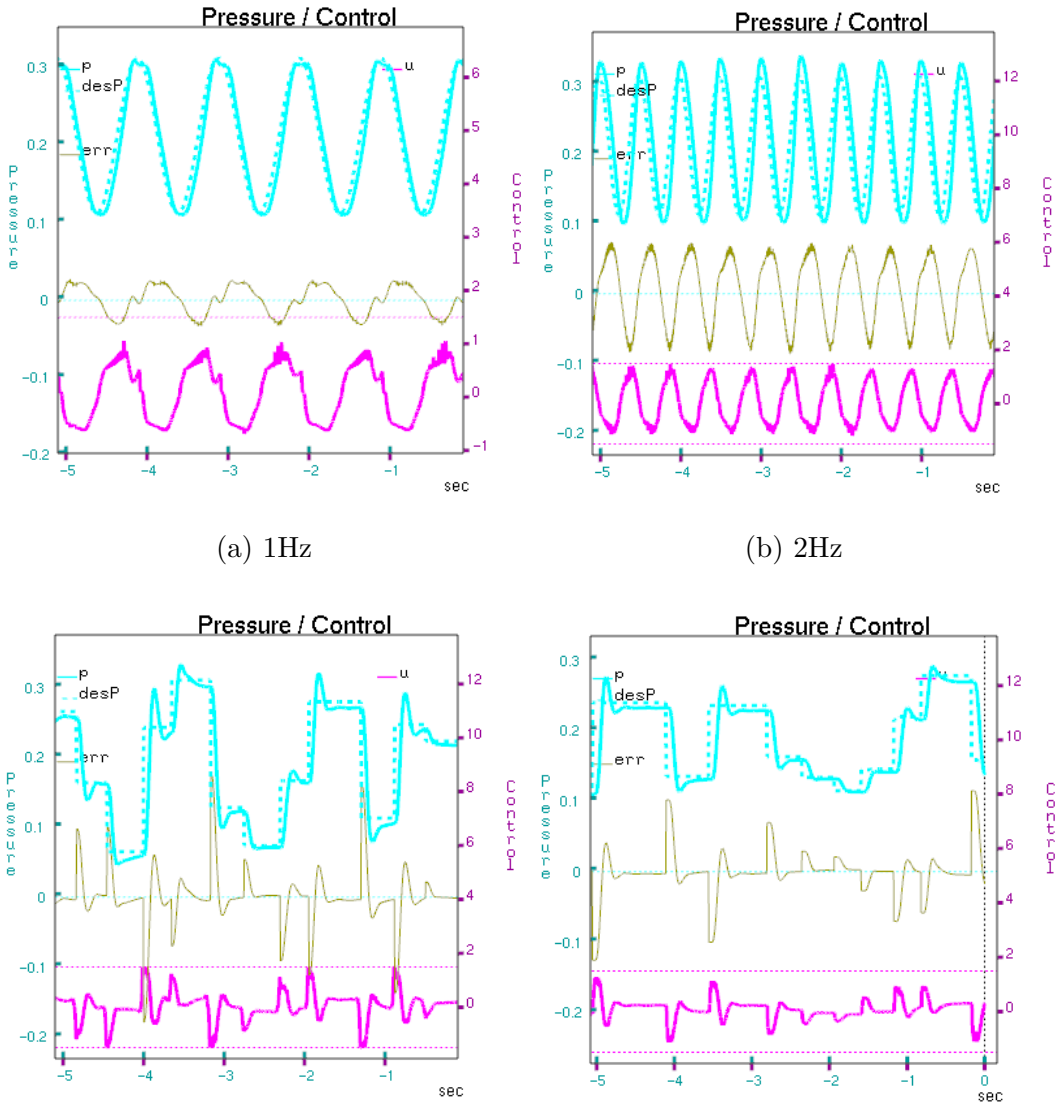



Figure 5.17: pressure tracking for a volume locked FESTO cylinder+ parametric model+ short tube. (a) 2Hz (b) Random (c) Instantaneous change

municating using sockets. At the onset of a control loop, the controller receives the current estimates from the hardware driver and responds with the controls queried using the current policy. The iLQG is parallelized across 8 physical cores. The planning horizon is 200 ms (N = 100, dt = 0.005s) long. The policy lag is between 7.5 to 10 ms. The cost coefficients are $\alpha_{\mathbf{p}} = 1, \alpha_{\mathbf{u}} = 10^{-4}, \alpha_{\mathbf{w}} = 10^{-4}$.




(a) 1Hz

(b) 2Hz

(c) Random

(d) Instantaneous changes

Figure 5.18: Controller's performance on a free to move (Airpel M09D37) cylinder with leak, while using pressure dynamics model learned on a leak-free volume-locked ( FESTO DSN104) cylinder.

### *5.6.2 Controller's robustness*

In order to demonstrate (Figure 5.18) the effectiveness of our approach and the robustness of the resulting controller, we artificially induce modelling errors in the system, by replacing the volume locked leak free cylinder with moving cylinder with leak (ADROIT pneumatic muscle unit). In order to evaluate our controller's ability to deal with abrupt change in the demands from the high level planner (Figure 5.16c & 5.17c), we subject the controller to instantaneous random changes in entire desired future sequence starting from current time. These instantaneous changes renders the current policy useless due to the local nature of the controller's linear feedback policies. The optimizer acts on these instantaneous demands only after the next policy is pushed to the hardware.

### *5.6.3 Real world considerations*

Results on a hardware platform is a combination of sound theory, good implementations and few practical considerations that get highlighted only in the real world. For reproducibility of the results, we outline our experiences learned while deploying the system.

1. *Timing is the key to performance*: The inherent pneumatic latency doesn't leave a lot of leeway for the controllers to drive the system through dynamic movements. Every effort was taken to minimize the latency of the system to the extent possible. Special attention was required for hardware-controller clock synchronization, data communication latencies, sensor data filtering latencies, estimates and policy lag, optimizer's computational needs and execution latencies (as the policy is queried over the network).

2. *Pneumatic nonlinearities*: The discontinuity in port areas as valve's spool moves across zero position induces severe non-linearities. The noise in the spool movement makes these nonlinearities unpredictable and hard to model. Special attention was required while modeling the pneumatic dynamics near the valve's zero point.

3. *Smooth operations*: As the valve chatters around its zero position making minor improvements, the pneumatic nonlinearities around zero position severely degrade the controller's performance. Introduction of valve memory $\mathbf{w}$ and cost $\alpha_{\mathbf{w}}.||\mathbf{u}_i - \mathbf{w}_i||$ for smoothing the controls significantly improves controller's performance.

4. *Operational regime*: Smoothness requirements promote the controller to leverage the inherent smooth dynamics of the system over abrupt changes in the value controls resulting in plans with small and smooth controls. In practice, the controller uses only 40% of the valves input range. In light of this observation, control limits were enforced on the controller using [99] and iterative model learning methods were used to refit the pneumatic models to the restricted range of operation. Restricting the operation range, allows us to run the control loop faster ($200Hz$) as the critical frequency of a valve is much wider for small movements.

5. *feedback*: Our system tends to be more stable delivering better performance with partial feedback (around 20%).

### *5.7 Scaling up to Adroit Hand*

Thus far we explained how a high-performance force actuator can be abstracted out from a pneumatic actuator. However, the actual goal of this exercise is to build a high-performance actuation for *ADROIT* , which has a muscle actuation unit consisting of 48 cylinders. Scaling trajectory optimization from a single actuator to the whole *ADROIT* system roughly requires 50 times scale up. Trajectory optimization for each actuator is computationally expensive, and requires a massively parallel architecture that needs to guarantee real-time performance. 50 folds scaling of this infrastructure is quite a monumental task.

While we played with various software architectures, the one that we converged on is outlined in Figure 5.19. It consists of multiple isolated "Abstraction-Pools", each dealing with a fixed number of actuators. An Abstraction-Pool is a virtual sandbox that runs an instance of trajectory optimization in isolation. The idea is to distribute the 48 actuators across these isolated pools to evenly distribute the load. Various configuration of actuator

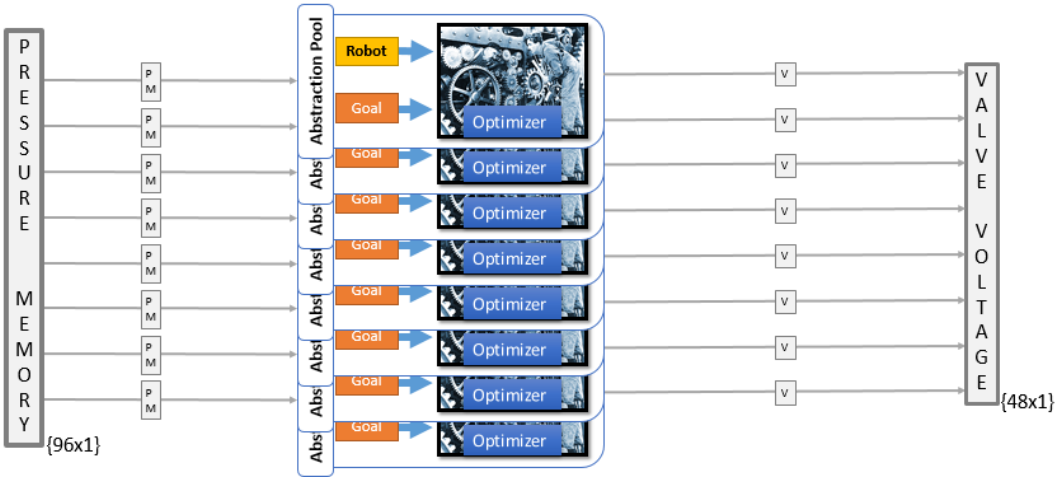
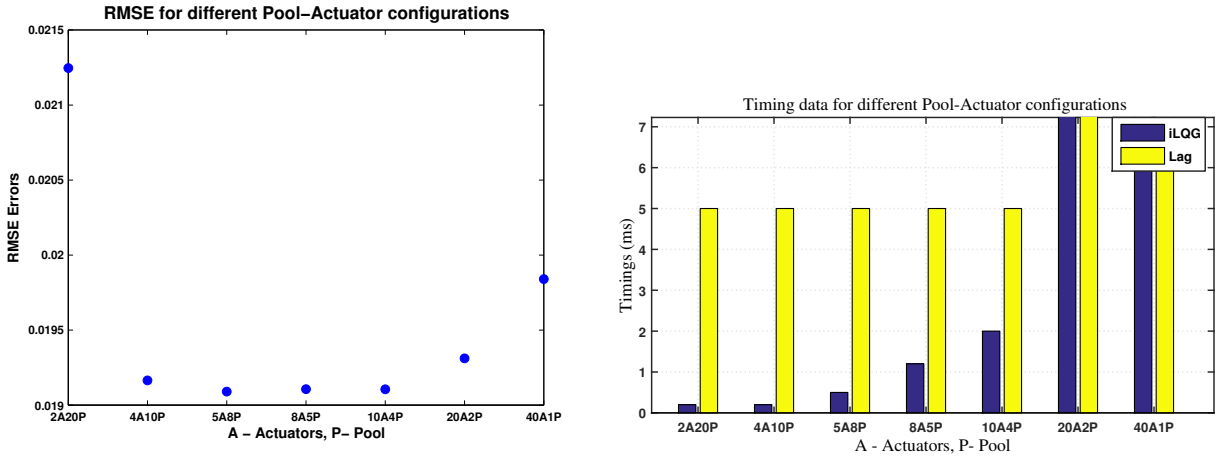



Figure 5.19: Actuator Pool configuration

distribution per pool was investigated. The results are presented in Figure 5.20. The final configuration that we picked was 5 pools with 8 actuators each.

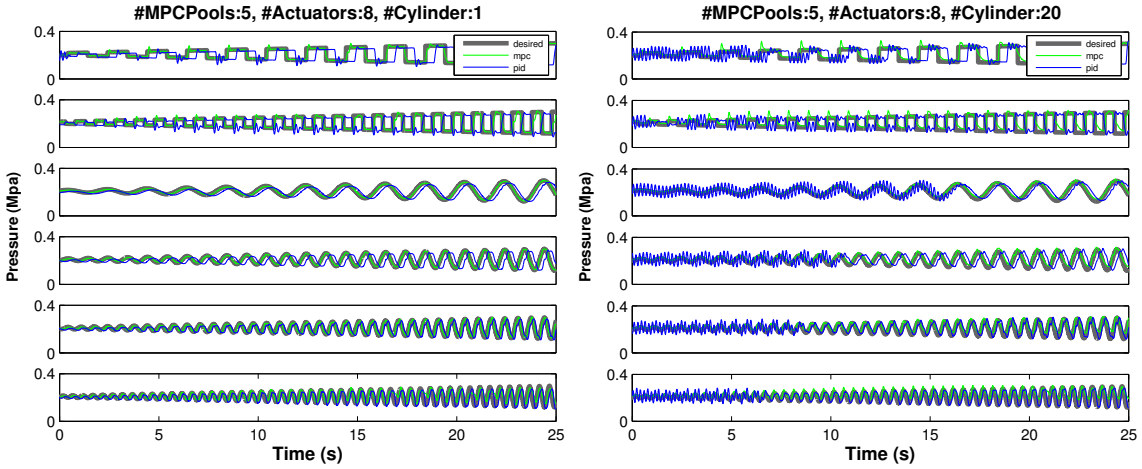



(a) Tracking performance for various actuator-pool configurations

(b) Timings for various actuator-pool configurations

Figure 5.20: Performances analysis of various actuator-pool configurations

Pressure tracking performance of two representative actuators are presented in Figure 5.21. Figure 5.21a shows the best case performance of the PID based system. Figure 5.21b displays an average performance we receive from PID based controller. We observe that PID based controller significantly lags and often enters resonance. Our trajectory optimization based controller is quite robust and performs quite well, even in high dynamics ranges.




(a) Pressure tracking performance for various inputs(best PID performance)

(b) Pressure tracking performance for various inputs(usual PID performance)

Figure 5.21: Pressure tracking performance

## *5.8 Next Steps*

Chapter 4 and Chapter 5 conclude our efforts towards taming the complexities of pneumatics actuation. To summarize, leveraging the pneumatic models developed in chapter 4 we have developed techniques to abstract out high-performance force actuators using pneumatics actuators. The force abstraction has been scaled up to full *ADROIT* system. Moving further, we will refer to *ADROIT* as a force actuated system.

Now we will switch our attention to the complexities posed by the dimensionality of the system. *ADROIT* is a high dimensional system capable of hosting dexterous manipulation.

However, due to the curse of dimensionally, synthesising dexterous manipulation policies for such a complex system is quite challenging. Moving forward, we will focus on techniques capable of synthesising contact rich dexterous manipulation behavior in high dimensional spaces. In next chapter we investigate behavior synthesis for *ADROIT* from a pure optimal control perspective. Later, we will leverage tools from reinforcement learning in association with tools from optimal control to further our efforts in contact rich behaviour synthesis for dexterous manipulation.

# Chapter 6

# SYNTHESISING DEXTEROUS HAND MANIPULATION BEHAVIOR VIA MPC USING GLOBAL MODEL

Dexterous hand manipulation is one of the most complex types of biological movement, and has proven very difficult to replicate in robots. The usual approaches to robotic control – following pre-defined trajectories or planning online with reduced models – are both inapplicable. Dexterous manipulation is so sensitive to small variations in contact force and object location that it seems to require online planning without any simplifications. Here we demonstrate for the first time online planning (or model-predictive control) with a full physics model of a humanoid hand, with 28 degrees of freedom and 48 pneumatic actuators. We augment the actuation space with motor synergies which speed up optimization without removing dexterity. Most of our results are in simulation, showing non-prehensile object manipulation as well as typing. In both cases the input to the system is a high level task description, while all details of the hand movement emerge online from fully automated numerical optimization.

## *6.1 Introduction*

Dexterous manipulation is one of the most complex and expressive classes of biological movement. Unlike other dynamic movements such as locomotion – which have been discovered by evolution in many different forms – manipulation appears to be challenging even for biology. Indeed a large part of the primate brain is devoted to controlling the hand and its complex musculo-tendon network. Manual dexterity is perhaps the most distinguishing sensorimotor capability of humans. Therefore, if we aspire to integrate robots into our human-centered world, we must equip them with manipulators and control strategies of similar dexterity.

Dexterous manipulation not only involves effective co-ordination of a high degree-of-freedom (dof) manipulator, but also requires stabilizing an object that only becomes controllable through contacts. High dof, including mutually coupled dof, operate in a very compact space co-inhabited by the object being manipulated. This results in a large number of contacts and dynamic phenomena such as rolling, sliding and deformation. Movement of objects within the workspace causes contacts to appear and disappear often, which greatly complicates motion planning.

In addition to high dof, a dexterous manipulator must also exhibit a well balanced combination of capabilities such as speed, strength, low loop latency, and compliance. High number of actuated dof in a small space makes it hard to design such manipulators. Dexterous robotic hands exist [92] [36] [27], but the difficulty in controlling them has motivated many researchers to focus on more limited mechanisms. Our goal instead is to develop general-purpose control solutions that can be applied to mechanisms as complex as a human hand.

Most prior work on hand control has focused on achieving stable grasp. While a lot of progress has been made on the theoretical side, practical applications are largely limited to using task-specific manipulators in carefully controlled environments. Part of the difficulty in deploying manipulation in less constrained environments comes from challenges in sensing and state estimation – due to visual occlusions and limited tactile sensing. But control is also challenging. Even if we assume a perfect model, most existing methods for grasp synthesis are too slow to operate in real-time, making it impossible to adapt to an uncertain environment.

For applications of dexterous manipulation in unconstrained human environments, we need the capability to plan in real time. This is because no pre-computed solution or simplifying set of assumptions are likely to work for such a complex behavior, that is so exquisitely sensitive to small variations in contact force or object position. Planning needs to be error tolerant and capable of quickly generating corrective maneuvers that stabilize the object in the face of unexpected disturbances. This is particularly challenging in non-prenehsile

manipulation, i.e. in the absence of a statically-stable grasp. Numerical optimization is the only general approach we are aware of that is at least in principle capable of "inventing" complex movements fully automatically, without intervention from a human. This is why our approach is centered on optimization.

The challenges of legged locomotion bare resemblance to those of dexterous manipulation: in both cases, an effective solution must deal with the forces acting between the robot's body, over which it exerts full control, and the unactuated dynamics – the root joint for locomotion, and the object in manipulation. After decades of research in legged locomotion, we are now capable of producing careful walking with humanoid robots [39] [2]. However these developments have not yet had an impact on hand movement control, in part because they rely on domain-specific abstractions such as ZMP, capture points, inverted pendulum approximations, etc. In fact this may be why manipulation is so much harder: in manipulation, constructing a reduced model that captures the essence of the behavior seems impossible. Any such model will clearly need to include the object and all the hand surfaces that can interact with it, along with the kinematic limitations of the manipulator – so it is not really a reduction.

From a computational perspective, hand movement synthesis is challenging because high dof (both actuated and unactuated) results in high dimensionally of the search space. Inter-finger contacts and object-finger interactions produce large number of contacts that impose discontinuities in the search space. The optimization landscape is further complicated by active involvement of dynamic phenomenon such as rolling, sliding and large number of contacts that make and break often.

In this chapter we describe real-time planning (or model-precitive control, MPC) of dexterous manipulation. The most interesting results are obtained in simulation, we present these results in light of a hardware platform *ADROIT* – a ShadowHand skeleton that we have equipped with faster and more compliant actuation [48]. This hardware platform called "*ADROIT* " is described in section 6.2. MPC control of object manipulation in simulation is presented in section 6.4. We also introduce an element which is often considered in hand ma-

nipulation but is novel to the MPC setting: namely hand synergies. Synergy-space planning techniques are described in section 6.4.5.

Our movement synthesis for dynamic hand manipulation builds upon our previous work [98] [29] which was able to synthesize full body movements. However, it was unable to scale up for motion synthesis in case of dexterous hand manipulating owing to difference in the nature of the optimization manifold. After carefully investigating the challenges, we introduce synergy-space planning and demonstrate for the first time online synthesis of dexterous object manipulation. The input to the planner is a high-level task description: we only specify the desired position of the object being manipulated, and the entire hand movement is then synthesised automatically. Similarity, in case of typing, we specify the sequence of desired key presses and the behavior emerges automatically. Instead of imposing these specification as hard constraints, we pose them as costs and mix them with other intuitive costs such as being gentle and moving at a nominal speed.

## *6.2 Manipulation platform*

ADROIT (figure 3.9) is a reconfigurable platform with 24 dof hand mounted on a 4 dof arm. All the joints can be actuated independently by its two exclusive opposing cylinders; with the exception of DIP and PIP joints of the four fingers that are mutually coupled.

The skeleton is mostly dominated by the Shadowhand skeleton with modifications to accommodate our pneumatic actuation system. Other major modifications include, the reinforcement of base joint to support upside down mounting. A low level driver written in 'C' talks to the system. The system is capable of sampling all the tendons length and pressure sensors at 9000Hz. A PIC microprocessor embedded in the palm samples all the joint angle and reports data at 500Hz via a CAN channel. The pneumatic valves can be commanded at 125Hz.

The entire platform is in a stage of constant evolution with the iterative improvements between (a) the hardware requirements, as demanded by the controller while synthesising dynamic manipulation behaviors in simulation and (b) manipulation capabilities, given the

development in our control strategies and hardware limitations.

### *6.3 Modelling*

In order to evaluate the hardware and test our control strategies, the ADROIT simulator (figure 6.1) is developed using the Mujoco Physics engine. Mujoco [105] is a new physics engine that works with generalised co-ordinates and supports a number of unique features including tendons actuation. Let $\mathbf{q}$ denote the vector of joint angles. Mujoco represents tendons $L(q)$ as path via routing points (i.e. sites), such that it does not penetrate any geometric wrapping objects (sphere and cylinders). At each time step Mujoco automatically computes the tendon's moment arms $\frac{\partial L(q)}{\partial q}$ which links tendon velocities $\dot{L}$ to the joint velocities $\dot{q}$ and joint torques $\tau$ to the scalar tension $f$ applied on the tendon by the corresponding linear actuator.

The upper bound on tension $f$ is determined by the type of actuator connected to the tendons. The lower bound $f_{slack}$ needs to be carefully maintained to avoid tendon slack. While the upper bound is strict, lower bound depends on the physical properties of the actuator (like friction, damping, stiction of the actuator and corresponding joints) and the nature of task being performed.

Mujoco's tendon length constraints are exploited to model PIP and DIP joint coupling. These joints are coupled using a tendon network that constraints the DIP flexion to be greater that PIP flexion. Figure 6.1 illustrates how this coupling is modelled in Mujoco using a soft tendon length constrain.

The simulator supports two different actuator mechanisms. First, a linear actuator that directly acts on a tendon to produce tension. Second, third order pneumatic actuator that uses pressure dynamics to produce tendon tension. [122] presents the modelling results of pressure dynamics of our muscle assembly (pneumatic cylinder, pipeline and valves). We present a generic parametric model of pressure dynamics that allow us to predict pressure upto 5 seconds in the future (given the current state and future voltage trajectories). The linear actuator provides a nice abstraction secluding pneumatics from the rest of the robot.

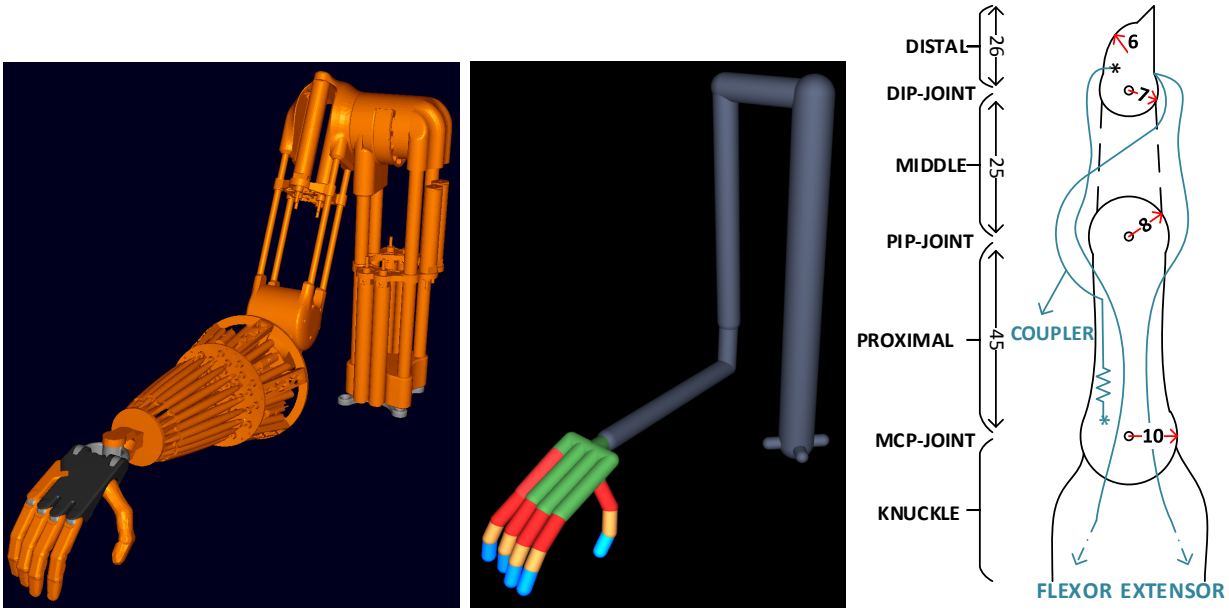



Figure 6.1: ADROIT Simulator visualizations (table mount) ***Left:*** A full CAD model of our 28-DOF platform. ***Center:*** An equivalent simplified model using capsules to simplify contact detection. The kinematic tree is the same in both models. ***Right:*** Tendon schematics representing DIP-PIP joint coupling.

Since our pressure dynamics are 3rd order, secluding pneumatics removes pressure from the list of state variables, thus reducing the dimensionality of the state space. This reduction is extremely helpful while exploring hardware capabilities independent of pneumatic actuation.

The dynamic manipulation results and the typing results mentioned in this chapter were generated using linear actuators.

### *6.4 Behavior synthesis*

Our approach for synthesising dynamic hand manipulation behaviors is to use MPC (detailed in sections 5.5.2, 5.5.3, 5.5.4), also known as online trajectory-optimization or receding-horizon control. Each MPC iteration begins with updating the state of the system using the latest estimates from the estimator. An estimator processes readings from all the sensors

and maintains a coherent estimate of the state at all times. A single iteration of a trajectory-optimization algorithm is applied starting from this state, and the initial part of the resulting policy is applied to the system until the process can be repeated and the policy updated again. For MPC to succeed, large updates to the policy must be made in each step so that the optimizer can "keep-up" with the changing state. This means that the regularization parameter $\mu$, which slows the optimizer should be small and carefully chosen.

### *6.4.1 State*

ADROIT platform has 28 dof and is actuated using 48 pneumatics cylinders. Depending on wether linear or 3rd order pneumatic actuators are in use, our state space is either 56 ($q$ + $\dot{q}$ for 28 joints) or 104 ($q$ + $\dot{q}$ for 28 joints + $p$ for 48 tendons) dimensional. Manipulation example adds another 12 dimensions for the object state and typing on keyboard adds extra 24 dimensions for the keyboard state. While Adroit is equipped with effective sensing capabilities (joint, touch, pressure and tendon length sensors) which makes its state observable, additional sensing options are required for sensing the bottle and the keyboard. We use the PhaseSpace 3D tracking system.

### *6.4.2 Trajectory optimization*

We have demonstrated earlier [98] [29] that iLQG can be used under MPC setting to generate full body movements in real time, while the very same approach failed in producing dynamic hand manipulation behaviors.

After careful investigation we attributed its inability to the nature of optimization space to be navigated and to the difference in the morphology of hardware and actuation mechanism other than direct torque control. Unlike full body movements, large number of dof exists in a very compact workspace in hand manipulation, resulting in a compact high dimensional optimization landscape. High dof to workspace volume ratio results in large number of contacts that make and break too often embedding discontinuities in the already compact high

dimensional space. High dexterity allows multiple solution to co-exists inviting optimizers to be stuck in local minima.

In following section we will address these challenges one by one and discuss ways to mitigate them.

### *6.4.3 Contact pruning*

In MuJoCo, the most intensive part of computing a single step is the handling of contacts. Unlike full body movements where the number of active contacts are small with low variance, manipulation with high dof manipulator involves large number of active contacts. Self contacts in case of full body movements are rarely active and few when active. For dexterous hands like ours, dof to workspace volume ratio is very high which results in most of the contacts being active all the time. Number of contacts with the environment in the former case is mostly small (feet-ground contacts). However in the later case, object under manipulation interacts with almost all the links of the hand resulting in large number of active contacts. Moreover these contacts make and break too often producing huge dynamic non-linearities in the search space. Computation time is severely affected as number of contacts (inter-finger and finger-object) increases. High variability in the number of contacts introduces variability in computation timings and hence policy lags.

For real-time behaviours synthesis, mesh collisions were never an option. Even simple geometric collision models using capsule resulted in too many active contacts (40 on average) for real time behaviours. To speed up the computations, contact space was very carefully populated considering kinematic constraints and joint couplings. Furthermore, nature of these contacts were carefully picked. Mujoco supports 3 types of contacts: 1-D contacts, 3-D contacts, 6-D contacts. 1-D contacts are only capable of producing normal forces. All inter-finger contacts are 1-D contacts. 3-D contacts are capable of producing normal and surface friction. In addition to normal and surface friction, 6D contacts produces rolling and torsional friction. All object finger contacts are 3D contacts, except finger tip-object contacts which are 6D.

It should be noted that we never made compromises while pruning the contact space. If violations were ever found, we immediately add the respective contacts back.

### *6.4.4 Passive springs and armature inertia*

Low-weight finger segments, strong and low friction actuators provide ADROIT its unique combination of speed, strength and compliance [48]. Such a combination is desirable for any robot hardware but not-so-desirable for numerical optimizers and naive controllers. Numerical optimizers enjoy accelerating the light weight segments producing behaviours that are either unsafe or look unnatural. Weak joint springs were added around the resting configuration of the hand (as shown in Figure 6.1) in the model that was available to the optimizers for planning. These springs acted as a passive attractors towards the resting configuration of the hand and prevented optimizers from choosing unnatural behaviours. Behavioural artifacts resulting from high acceleration were mitigated by adding armature inertia to the joints proportional to the link masses. It not only removed the artifacts but also improved convergence thereby improving the quality of the solution.

### *6.4.5 Dimensionality Augmentation using synergy spaces*

Animal life form exhibits complex and diverse set of behaviours. These behaviours are produced by numerous bio-mechanical muscles that transmit force using skeletal tendons. Unlike the rest of the body, hands exhibit complicated tendon network that span across multiple joints that introduce substantial coupling between joints. For full body movements, its possible to have individual control over each joints. Kinematics constraints and a regulariser around default body posture have long been exploited to synthesize movements close to animal life forms. However, these tricks were not enough for synthesis of human like hand movements. Optimization rarely produces desired behaviors and if it does, it produces a solution that is too rich to be exhibited by human hands.

Dimensionally reduction techniques have often been used as a response to the curse of dimensionality. Dimensionality reduction is known to produce good results but these results

Table 6.1: Passive joint spring constants and armature inertia

| Joints | Spring Constants | Armature inertia |
|---|---|---|
| Torso | 0.00 | 0.60 |
| Shoulder | 0.00 | 2.40 |
| Elbow flex | 0.00 | 0.45 |
| Elbow roll | 0.00 | 0.03 |
| Wrist ad-ab | 0.05 | 0.06 |
| Wrist flex | 0.05 | 0.06 |
| Finger MCP1 | 0.20 | 0.06 |
| Finger MCP2 | 0.05 | 0.06 |
| Finger PIP | 0.10 | 0.06 |
| Finger DIP | 0.15 | 0.06 |
| Little metacarpal | 0.15 | 0.06 |
| Thumb MCP1 | 0.05 | 0.06 |
| Thumb MCP2 | 0.05 | 0.06 |
| Thumb PIP1 | 0.15 | 0.06 |
| Thumb PIP2 | 0.15 | 0.06 |
| Thumb DIP | 0.15 | 0.06 |

come at a cost of reduced dexterity. Dimensionality reduction limits the solution along the synergy subspace thus imposing restrictions on the expressiveness of the solution. Given, the goal of this thesis is to push the expressiveness and dexterity of hand manipulation behaviors, dimensionality reduction is not quite acceptable. We instead develop a counterintuitive framework called "dimensionality augmentation".

Key insight being dimensionality augmentation is to make the optimization space more

tractable for numerical optimizers by augmentation of the space. These augmented dimensions are subspace along which we expect rapid progress. Synergy spaces can often be used as the supplementary dimensions. The optimizers used these subspaces to make rapid progress and then resorts to the individual dimensions to the finer details.

We introduced five hand synergy dimension [107] on top of individual control dimensions and made them slightly cheaper for the optimizer to choose them. First four synergies individually influence the curl of the four fingers and the fifth synergy influences the spread of the four fingers. Addition of synergy dimensions increased the space of control dimensions by five but made the problem more tractable for the optimizer, without compromising the control over individual joints (hence dexterity) of the hand. Synergies help the optimizer to quickly find a pre-grasping pose, after which individual joints dominate to produce expressive behaviours.

This is for the first time that iLQG has been demonstrated to be amenable in synergy spaces. Traditionally synergy spaces have always been exploited as a mechanism for dimensionality reduction to make a high dimensional problem trackable. However, here we are adding synergy dimensions without removing existing dimensions, thus expanding the dimensionality of an already high dimensional problem. This is a little counterintuitive. We found that the optimizer exploits synergy spaces to quickly navigate through the space full of nonlinearities and discontinuities to localize itself in the correct neighbourhood, where dimensions other than synergies (i.e. individual joint actuation) dominate to produce expressive behaviours.

### *6.5 Cost terms*

Synthesis of behaviours involves specification of high level cost function that encodes the task objectives. All behaviours presented in this chapter were generated using very simple high level cost functions. These cost function are composed of few simple terms with intuitive meaning associated with each term that talks about the task. Note that no cost terms outlining the movement of the hand (or grasp metric) are provided. We focused our efforts

on generic behaviors like dynamic manipulation and object relocation which constitutes significant portion of our daily activities. Once the framework was in place it was easily extendible for specific tasks like typing. In addition to the regular control penalization, which was common for all behaviors, the following other cost terms were used:

*Dynamic manipulation:*

Dynamic manipulation behaviours include dynamic catching of a falling objects, grasping and stabilizing of unstable objects. Cost terms penalizes

- Control synergies. These bear slightly lower penalty than the rest of the independent actuators.
- Velocity of the object being manipulated.
- Distance of the object to its desired goal configuration (position and orientation)
- The last cost term, with a small coefficient penalizes the distance of the object from the center of hand's grasp envelop.

Ideally, if the MPC horizon is long enough, we will not need the last cost term. In practice, real-time constraints restricts us from using very long horizon. In absence of this term, if the object is far away, the optimizer will not be able to find an improvement within the given time horizon. In such cases, this term acts as hint term for the hand to make an initial approach towards the object. We used a static point equidistant from - the palm base, finger's (Index, middle, ring and little) middle segments and thumb's distal segment as the center of the grasp envelop. Once the initial approach has been achieved, the first three terms dominate. Note that we used no grasp-specific costs promoting force closure or other supposedly desirable grasp properties. All the behaviors seen in the movie [goo.gl/WPTjcS]emerged from these simple cost terms.

*Object relocation:*

All costs for relocation were same as for dynamic manipulation except object desired configuration cost. Desired configuration was exposed as a parameter to the user. Users were allowed to change this cost by dynamically changing desired configuration in middle of the simulation, resulting in interactive grasping and relocation.

Object relocation starts as a dynamic stabilization sequence, but with a pre-specified intermediate goal configuration, called the 'hold configuration'. Any relaxed position of the hand where it can stably hold the object away from all the obstacles can be used as the hold configuration. The hand waits at the hold configuration for the user to specify the final desired relocation configuration, after which the final relocation maneuver is attempted. If the desired configuration has been preemptively specified by the user, the hand immediately exits the hold configuration and attempts to position the object at the desired configuration. Once the user is satisfied with the final configuration of the object he/she triggers the release, which switches off the grasp cost resulting in hand gracefully leaving and moving away from the object once its stable.

*Typing:*

This behavior include interactively typing specific numbers on a number pad. Keys on number pad are spring loaded to make the problem challenging. Key press is not detected until the key is fully pressed. Typing results exploit keyboard heat maps to pre-allocate key-finger pair. One rational behind this assumption is that although every individual has his/her own typing preferences, key-finger pairs are almost static. Downside of this approach is that if fingers get stuck in local minima, alternative options will never be explored by the optimizer. Behaviors include cost on

- Desired key pressed
- Desired key approach by the assigned finger's tip.

- Next key hover by the next assigned finger: This cost encouraged the next assigned finger, to make the initial approach towards its key if not being used thus speeding up the typing.

- Auto-correct. This terms kicks-in when accidental typing mistakes are made. A back space is pressed before moving ahead.

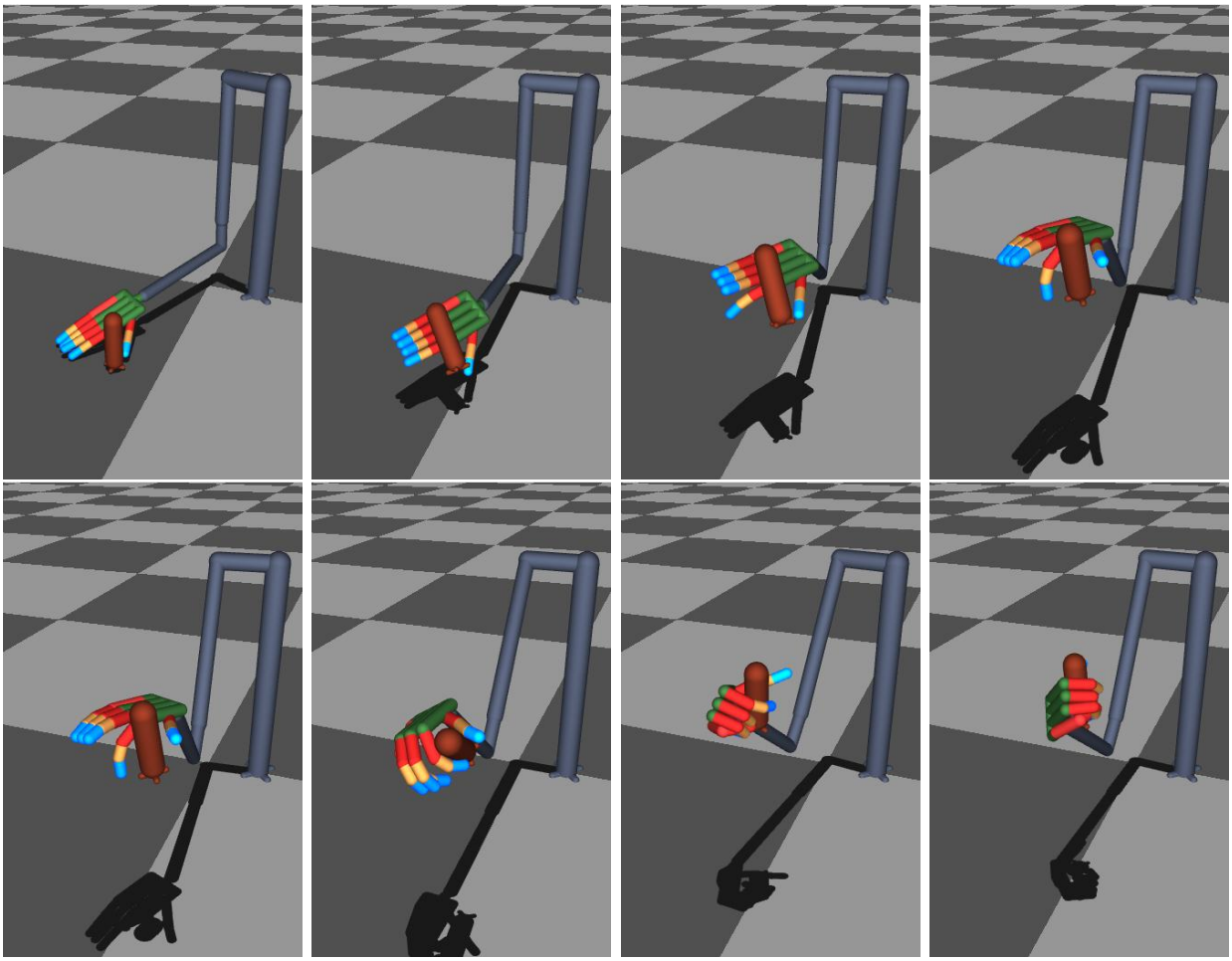
Our approach towards typing is event driven. Once a key press is detected, the typing sequences increments and the entire plan is recomputed for the next assignment. In other words, optimizer is not able to plan through the transitions. To allow the optimizer to plan through the transition, an additional cost term 'next key press' is appended. Smooth step-up $S_{up}$ and step-down $S_{dn} = (1 - S_{up})$ functions are used as cost coefficients for 'desired key press' and 'next key press' respectively. When key event is detected the steps are switched which allows the cost to smoothly transition from key hover to key press.

$$S_{up}(t) = \frac{1}{2}\left(\frac{(t - a_1)}{\sqrt{(t - a_1)^2 + b_1^2}} - \frac{(t - a_2)}{\sqrt{(t - a_2)^2 + b_2^2}}\right)$$

where $t$ is time, $a_1$ and $a_2$ are the step locations, and $b_1$ and $b_2$ are the step smoothness.

### *6.6 Results and discussions*

We will now present the real time interactive behaviour synthesis results for ADROIT in general tasks like dynamic manipulation, object relocation and specific tasks like typing. All the results mentioned in this chapter are generated using a Intel Xeon X5690 @3.47 Ghz processor with 12GB of memory running Windows7. Typical timings for different parts of the computation can be found in table 6.2. Optimization parameters can be found in table 6.3. To better appreciate our results we highly recommend watching the video attachment. Our latest results can be found at [goo.gl/WPTjcS].



### *6.6.1 Hand manipulation behaviors*

Characteristic behaviours in the dynamic hand manipulation sequence involve catching falling objects, grasping objects from different configurations and stabilization of unstable objects. Figure 6.2 shows frames (150ms apart) of two agile recovery maneuvers.

Use of carefully picked contacts pairs of different nature helped reduce the average number of active contacts from 40 to 20 which provided a major boost in simulation timings. For dynamic grasping, we observed that our optimizers enjoy slightly smoother contacts over the hard ones. Large number of frequently changing contacts introduce discontinuities in the search space. Smoother contacts makes these discontinuities better behaved for optimisers

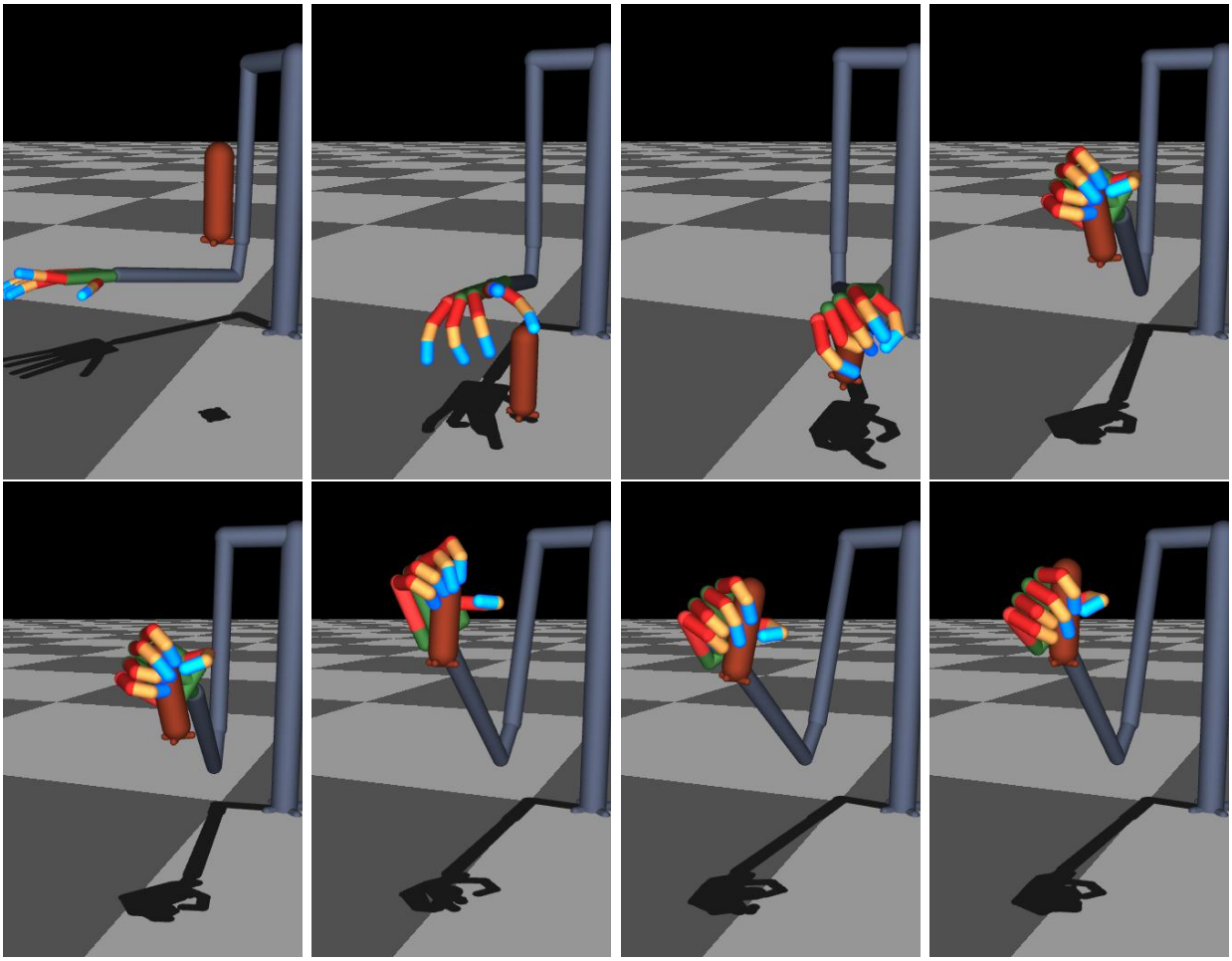


Figure 6.2: Two sequences from the accompanying movie. The time between consecutive frames is 150ms. Both sequences show an agile recovery manoeuver. The top sequence begins with a back-handed fumble of the object, which is then caught and brought into the desired position. In the second sequence the initial grip on the object is not robust and the object begins to slip. The controller releases the slipping object and re-grips it in mid-flight.

to plan through them. Use of armature inertia to tame large accelerations worked much better than trying to force it using velocity cost. The success rate for dynamic manipulation was around 80%. For unsuccessful cases, either the hand gets stuck in some local minima and never emerges from it or is unable to make a good grasp due to faulty initial approach.

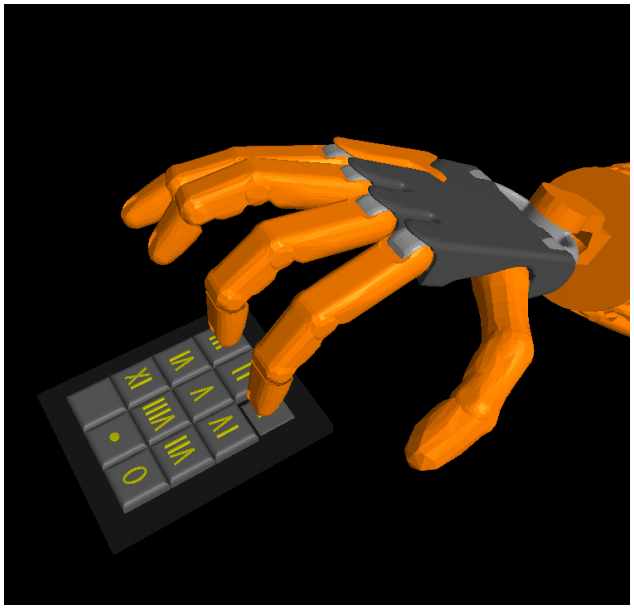


Figure 6.3: ADROIT while typing the key-1 on the numeric keypad

### *6.6.2 Typing on a numeric keypad*

Figure 6.3 shows a snapshot from the typing sequence while ADROIT is pressing the first key. Average number of active contacts for typing stayed relatively low (5 on average, mostly inter finger). Our typing worked fairly well. Though rare, we did observe the fingers getting stuck in local minima for some sequences. Due to preassigned key-finger pairs, optimizers don't even explore alternative approaches to make progress, when stuck. Naive approach to get out of the local minima will be to momentarily pause typing and relax all fingers before moving ahead. We leave key-finger assignment problem as future work. Mistyping due to accidental contact of finger with the edge of the adjacent key is also observed.

## *6.7 Discussions*

The motivations and the goals of this thesis are to realize dynamic hand manipulation on our Adroit hardware platform. To the best of our capabilities, utmost measures were taken to not make any assumptions in our simulation results to maintain its applicability on the

Table 6.2: MPC Timings (sec) for Intel Xeon X5690 @3.47 Ghz, 12GB memory, Windows7.

| Timings | Dynamic Manipulation | Typing |
|---|---|---|
| Total | 0.059 | 0.047 |
| Policy lag | 0.050 | 0.038 |
| Rollout | 0.006 | 0.004 |
| Derivatives | 0.036 | 0.025 |
| Cost computations | 0.001 | 0.001 |
| Backpass | 0.010 | 0.010 |
| Line search | 0.007 | 0.006 |

Table 6.3: Optimization parameters

| Specifications | Dynamic Manipulation | Typing |
|---|---|---|
| Simulation dt (sec) | 0.005 | 0.001 |
| Optimizer dt (sec) | 0.005 | 0.020 |
| Horizon (sec) | 0.300 | 0.640 |
| Mindist(m) | 0.010 | 0.001 |
| Contact softness(m) | 0.005 | 0.001 |

real hardware. Simulation results were used to inform the hardware design of *ADROIT* . Later, simulation results were adapted to the design modifications made to the *ADROIT* hardware. This iterative process tremendously helped to improve the design as well as the behavior synthesis framework.

### *6.8 Future directions*

Pneumatic actuation of the ADROIT has the capability to emulate biological muscles. One unexplored, yet promising, direction is to leverage biological muscle models for movement generation. Leveraging biological muscles models for movement generation leads to more naturalistic movements.

With respect to individual behaviors, our approach towards typing is shortsighted as the optimizer only sees and plans for present and the next key in the typing sequence, even if the entire sequence is specified in advance. One option is to incorporate a phase variable (encoding the development made by the trajectory) in the optimization and let the optimizer choose the number of key presses and plan through it. Piano playing is another problem with a temporal objective that will require specific attention to key-finger assignment problem.

### *6.9 Next steps*

Hand manipulation evolves in a very compact space. The object maneuvers through the workspace by establishing a sequence of contacts. Contacts make and break quite often as the object maneuvers within the workspace. The non-linearity, resulting from the onset and offset of the contacts and various limits, makes the state space quite complicated and sensitive to minor perturbation. It's extremely unlikely that a plan can be fully executed without significant deviation from the nominal course. The reactive nature of model predictive control seems like the right paradigm to address the flexibility and reactive replanning requirements posed by dexterous manipulation policies.

The compactness of the space in which hand manipulation evolves makes it extremely hard for any estimation module to provide a reasonable estimate of the state. While MPC is quite effective, its effectiveness is dependent on the estimates of the real world. Initial testing of our behavior synthesis framework on the actual hardware indicates its sensitivity to the modeling and estimation errors. In the following chapter, we further explore the modeling and estimation challenges in realizing true dexterous manipulation on real hardwares.

# Chapter 7

# CHALLENGES WITH HARDWARE & VIABLE OPTIONS

Although remarkable progress has been reported in chapter 3 and chapter 6 in terms of a dexterous manipulator capable of hosting contact rich complex manipulation, and reactive controllers capable of synthesising complex behaviors on the simulated systems, significant challenges still need to be addressed before dexterous manipulation is realised on real systems under real-world conditions. In this chapter we will focus on identifying these real world constraints and challenges. We will also investigating methods and techniques that are more likely to scale and produce results on the real systems.

## *7.1 Challenges*

Planning for manipulation gets increasingly challenging as the dexterity of the manipulator increases. Existing techniques struggle to solve for dynamic dexterous manipulation behavior due to the complexity of the optimization landscape. The compact optimization landscape, cohabited by the object being manipulated, is full of nonlinearities and discontinuities that arise from the finger-finger interaction, finger-object interaction, joint limits, and the dynamic constraints of the manipulator. The landscape morphs and evolves as the object maneuvers in the workspace, which further complicates the planning. Unlike previous approaches, which resorting to quasi-static movements for simple manipulators to avoid planning complexities, planning manipulation behaviours is no longer the limiting factor for us in realizing dynamic dexterous manipulation. In chapter 6, we presented a framework capable of dealing with the mentioned complexities and synthesizing dynamic dexterous manipulation behaviours in real time.

The discrepancies between the planner and the real world pose significant challenges as we

try to port our simulation results to the real hardware. These discrepancies can be attributes to two factors – modelling errors and estimation errors. The sensitivity towards modelling errors is not due to the high dimensionality/ dexterity of our manipulator, but is fundamental to the nature of the movements we are interested in. Unlike quasi-static movements, where the state of the system and the time are loosely related, they are tightly coupled for systems undergoing dynamic movements. As a result, for dynamic movements, the controller's policy is quite sensitivity to the variations in the state and the state transitions (i.e. the model of the dynamical system). As the space where manipulation happens is high dimensional, compact and densely populated (cohabited by the manipulators DoFs and the DoFs of the object being manipulated) with constraints and nonlinearities, small variations in the state can produce significantly variation in the optimization landscape as the number and location of contacts changes. In addition to model discrepancies, state estimation is of prime concern as well for achieving dynamics dexterous manipulation with real hardware. The compact and crowded space pose significant sensing challenges. State estimators bear an unreasonable expectation of delivering perfect estimates of an extremely sensitive state space with partial observability. These are fundamentally grave concerns that can't be slid under the rug of "technological and computational advancements will solve these problems". Moving forward we will explicitly recognize these challenges while picking strategies and designing solution techniques.

### *7.2 Real world constraints*

Dexterous manipulators are quite complex at the mechanism level. The space limitation restricts direct actuation, which leads to further mechanical complexity coming from the transmission mechanisms (tendons, four bar linkages etc). Owing to the space and weight considerations, there are physical limitations in terms of sensing, thus the over all observability of the manipulators. ADROIT platform is a highly observable system with 135 high performing sensors (measuring individual joint angles, tendon excursions, tendon forces, forces at finger tips). With this level of sensing one would assume that it will be possible to

get a perfect state reconstruction. This is however impossible to achieve in practice for two reasons

1. *Imperfect calibration*: The small workspace resulting from the small link lengths presents significant challenge while calibrating sensors of the dexterous manipulators. Calibrations jigs are quite ineffective due to the compactness of the workspace. Optimization based automatic methods relying on external sensing like motion capture. Motion capture systems are ineffective due to marker obstruction and marker confusion. The small spacial movements being read by these sensors are comparable to the sensor positioning offset and sensor noise, which provides misleading information to the optimization based techniques. [115] and [20] presents two different efforts I perused towards improving sensor calibration (Not detailed in this report).

2. *Observability*: Compactness of the workspace and the occlusions from the manipulator itself pose significant challenges while sensing the object being manipulated. The complexity of the dexterous manipulators are such that its nearly impossible to achieve full observability of the manipulator with existing technologies. For ADROIT platform in particular, as the tendon tension is sensed directly using the pressure sensor mounted on the pneumatic cylinder, the loss in tendon forces due to stiction and the friction resulting from the tendon routing are unobservable. The touch sensors are available only on the finger tips. As the hand interacts with the object using entire inner surface of the palm and the fingers, interactions between the hand and the passive object are unobservable.

### *7.3 Viable approaches for handling model discrepancies*

Even with complete observability and careful system identification, a perfect computational model of the hardware is impossible to realize. Planners and controllers will have to work with this restrictions. Multiple approaches have been proposed in the literature to design policies that either entirely model free or are based on model approximation/ adaptation

via exploration. Owing to the high dimensionality of the dexterous manipulators and the sensitivity of the policy for dynamic movements, model free methods are hard to scale and generalize across different task. This body of works will explore and build on the model approximation/ adaptation approaches. Few viable approaches that have the potential to scale and generalize, in the light of dynamic dexterous manipulation on real hardware, are outlined below

### *7.3.1 Fast replanning under Model Predictive Control framework*

Model Predictive Control (MPC) works on the principle of constantly updating the policy at fast rate, assumption being that the next policy is updated before the previous policy derails the system towards unrecoverable catastrophic failure. As the system is always in motion, a standard trajectory optimization is deployed in a Model Predictive Control (MPC) fashion. Which means instead of solving for the optimum, starting from the current state estimates, the algorithm takes only take few, improves the policy for the current estimates, and then opt for an estimates update. There are multiple rational behind this choice

1. The model used for planning will never be perfect and true optimum will never be achieved even if we have it

2. Solving for optimum is computationally expensive. Fast policy update provides better performance by dragging the system closer to the optimum with each update.

Our approach towards behavior generation as outlined in chapter 6 is based on the ideas mentioned here. It uses iterative-LQG as the trajectory planner. In depth analysis of iterative-LQG can be found in [109]. Refer to [46] [98] [29] for more applications.

### *7.3.2 Data driven model learning*

This body of work, outlined in chapter 8, combines iteratively refitted time varying linear models with trajectory optimization, and can be seen as an instance of model-based rein-

forcement learning or as adaptive optimal control. We rely on a model, but that model does not have any informative predefined structure. Instead, it is a time-varying linear model learned from data, using a generic prior for regularization. Deploying trajectory optimizers over the learned linear model, we can achieve sample-efficient learning of tasks that involve intermittent contact dynamics and under-actuation. The appeal of the method lies in its ability to handle challenging problems with surprisingly little data.

### *7.3.3 Hardware adaptation*

The debate between model free vs model based approaches has never been settled. Both methods have their own benefits and limitations. Model free approaches on one hand work on the real systems, but are hard to scale and generalize across tasks and manipulators. Model based approaches on the other hand, produces extremely detailed behaviours and generalizes well, but are difficult to scale to real systems. There is not dispute that the combination of model free and model based approach is more likely to generalize and scale well in the real world problems, but the right paradigm in order to do so seems to be a topic of constant debate.

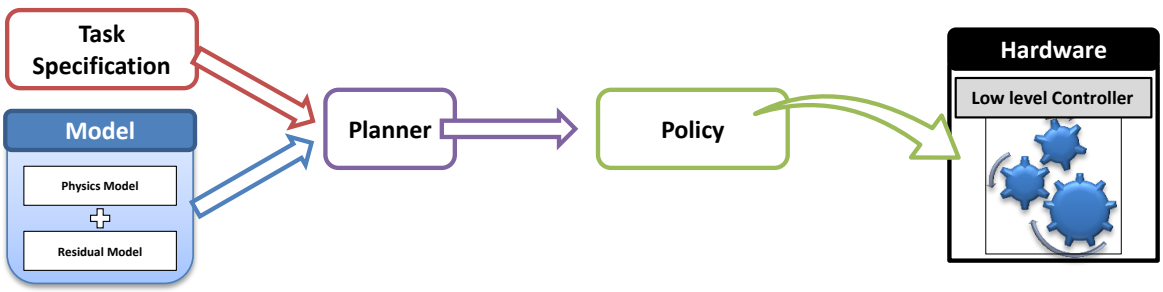



Figure 7.1: Model adaptation

Unlike previous approaches (figure 7.1), where model discrepancy are patched by adapting the model used for planning to better explain the real hardware, our approach try to patch the discrepancy from the reverse direction. We propose to adapt the hardware instead,

in order for it to behave more like the model available for planing. The rational behind this choice lies in the fact that planning through ideal physics model is much easier than planning through the patched model, where the adapted component (residual model) can be quite nonlinear and ill-behaved.

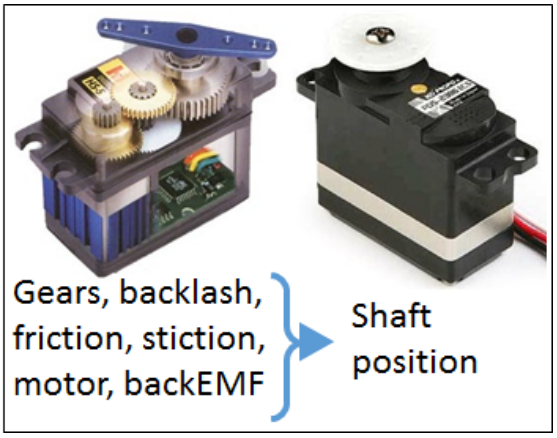



Figure 7.2: Position servo hardware abstraction

To better understand our choice, let's take the examples of the position servo actuators (Figure 7.2). They subsume the complexity of the internal mechanism consisting of gears, motors (that is a dynamical system of its own) and presents a simple hardware abstraction. This abstraction allows the users to ignore the complexities of the internal mechanisms (friction, stiction, gears, backlash, motor dynamics, back EMF etc) and focus on pure kinematic planning.

The rational behind our choice is some what similar. We want to abstract out the hard-to-model complexities of hardware in a way that it behaves as an "ideal hardware". This ideal hardware abstractions can take any desired form –

1. *Ideal Actuators*: It can be a ideal position, velocity or torque control

2. *Model available to the planner*: Not only actuator complexity, hardware abstraction can be made to subsume the un-modelled dynamics of the overall system. This is quite

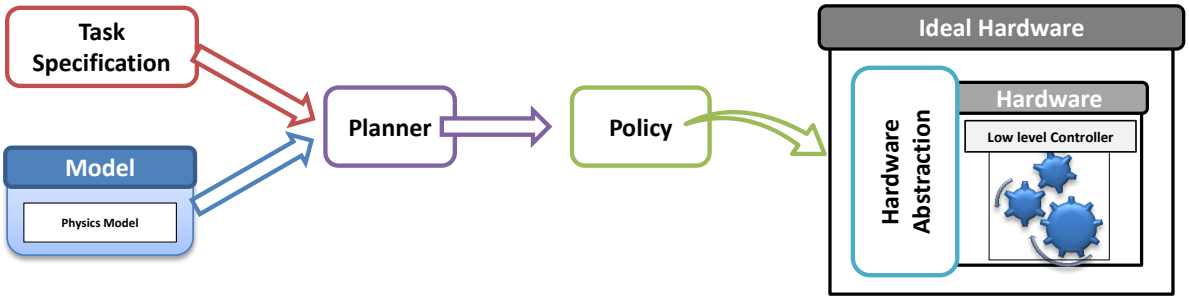



Figure 7.3: Hardware adaptation

desirable. As the real hardware starts to behave more like the physics model available to the planner, all results presented in chapter 6 can now be directly executed on the real hardware.

3. *Model Reduction*: Planner can be made to plan in reduced dimensions and the job of mapping the low dimensional plans to the full state space can be subsumed in the hardware abstractions.

If effectively achieved, it will be a universal solution for a hardware, that needs to be prepared once and will generalize across all use cases. Next, we outlines hardware abstractions ideas for ADROIT manipulation platform.

## *7.4 Torque actuator abstractions for the pneumatic actuators*

High force density, compact form factor and muscle like compliance properties makes pneumatics actuators quite desirable for robotics applications. However, the compressibility of the air not only throttles their bandwidth but also makes them harder to control. The key idea behind improving the controllability of the system is to hide the complexities of pneumatics inside the hardware abstraction such that the resultant hardware acts like a normal torque actuator (Figure 7.4). In chapter 5 we leveraged trajectory optimization techniques to design high-performance controllers for pneumatic actuators that live inside the hardware

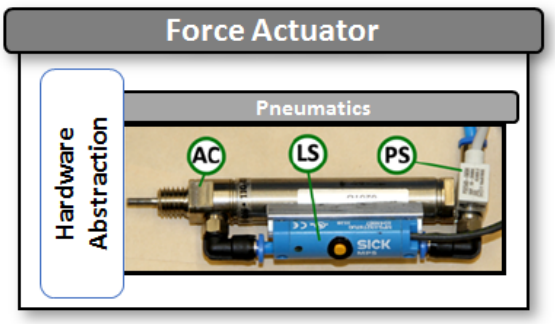



Figure 7.4: Abstracting force actuators out of pneumatics actuators

abstraction layer. These controllers use future predictions in order to act in anticipation. The proposed controller prepares the system ahead of time in order to achieve better performance and uses less controls while doing so.

## *7.5 Next steps*

This chapter is a crucial pivot point for this thesis. It outlines the high-level philosophy and key ideas on which the fundamentals of this thesis is based. Although we didn't explicitly stress on, design philosophy and key insights of all the previous chapters discussed thus finds their roots embedded in the ideas presented here. More specifically, chapter 6 used the key ideas from section 7.3.1 to formulate a controller capable of fast replanning. Chapter 5 leverages ideas from the section 7.4 in order to formulate hardware abstraction that simplified complex pneumatic force actuators into a simple torque actuator. Moving forward we view *ADROIT* as a force controller system.

In following chapters we build upon the techniques outlined in section 7.3.2. More specifically, we will leverage tool from model-based reinforcement learning to learn models directly on the hardware using raw sensors in order to sidestep the challenges posed by modeling and estimation errors.

ToDo: clean it up This system is used in three sets of experiments: the first set of experiments

examines learning dexterous manipulation skills from scratch using trajectory-centric reinforcement learning, the second set of experiments is focused on learning more complex skills with a combination of trajectory-centric reinforcement learning and learning from demonstration, and the third set of experiments examines how various approximation methods, including nearest neighbor and deep neural networks, can be used to learn from multiple learned behaviors to acquire a single generalizable skill that succeeds under various circumstances.

The trajectory-centric reinforcement learning algorithm that we use combines the linear-quadratic regulator (LQR) algorithm with learned time-varying local linear models. This algorithm, which follows previous work [51], is described in Section 8.2. We then present results on both a real-world and simulated version of the ADROIT platform using the algorithm, in Section 8.4. This first set of experiments focuses primarily on the capability of the trajectory-centric reinforcement learning method to efficiently learn viable and robust manipulation skills.

The second set of experiments, presented in Section 10.1, is aimed at evaluating how human demonstrations can be used to aid learning for more complex skills. In this section, we examine a complex grasping scenario, where trajectory-centric reinforcement learning on its own does not produce sufficiently successful behaviors, while human demonstrations alone also do not achieve a sufficient degree of robustness in the face of perturbations. We demonstrate that combining demonstrations with trajectory-centric reinforcement learning produces effective skills with a high degree of robustness to variation in the initial placement of objects in the world.

Our final set of experiments, presented in Section 11.2, address the question of generalization: can we use multiple skills, learned with a combination of imitation and trajectory-centric reinforcement learning, to acquire a single robust and generalizable dexterous manipulation policy? To that end, we explore the use of deep neural networks to achieve generalization by combining the behaviors of multiple skills from the previous section. We also compare to nearest neighbor as a baseline method. We demonstrate that deep neural

networks can learn time-invariant manipulation policies that acquire the strategies represented by the time-varying controllers learned with trajectory-centric reinforcement learning, and furthermore can perform those skills using onboard sensing in a simulated experiment, without knowledge of the true position of the manipulated object.

# Chapter 8

# LEARNING DEXTEROUS HAND MANIPULATION BEHAVIOR VIA EXPERIENCE USING LEARNED LOCAL MODELS

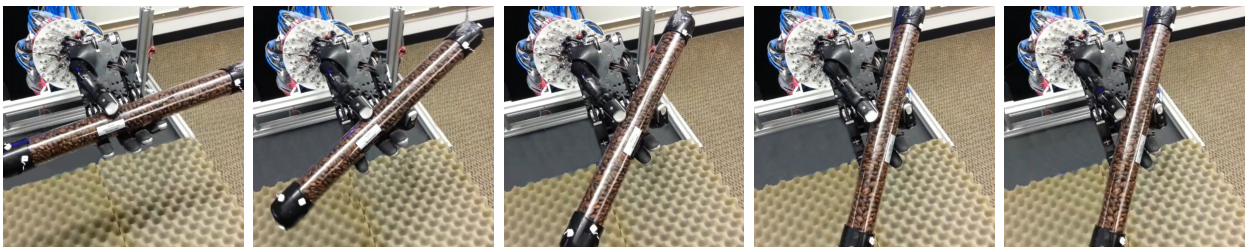


Figure 8.1: Learned hand manipulation behavior involving clockwise rotation of the object

This chapter describes a method for learning dexterous manipulation skills with a pneumatically-actuated tendon-driven 24-DoF hand. The method combines iteratively refitted time-varying linear models with trajectory optimization, and can be seen as an instance of model-based reinforcement learning or as adaptive optimal control. Its appeal lies in the ability to handle challenging problems with surprisingly little data. We show that we can achieve sample-efficient learning of tasks that involve intermittent contact dynamics and under-actuation. Furthermore, we can control the hand directly at the level of the pneumatic valves, without the use of a prior model that describes the relationship between valve commands and joint torques. We compare results from learning in simulation and on the physical system. Even though the learned policies are local, they are able to control the system in the face of substantial variability in initial state.

## *8.1 Introduction*

Dexterous manipulation is among the most challenging control problems in robotics, and remains largely unsolved. This is due to a combination of factors including high dimensionality, intermittent contact dynamics, and under-actuation in the case of dynamic object manipulation. Here we describe our efforts to tackle this problem in a principled way. We do not rely on manually designed controllers. Instead we synthesize controllers automatically, by optimizing high-level cost functions. The resulting controllers are able to manipulate freely-moving objects; see Fig 8.1. While the present results are limited to learning local models and control policies, the performance we obtain, together with the small amount of data we need (around 60 trials) indicate that the approach can be extended with more global learning methods such as [51].

We use our Adroit platform [48], which is a ShadowHand skeleton augmented with high-performance pneumatic actuators. This system has a 100-dimensional continuous state space, including the positions and velocities of 24 joints, the pressures in 40 pneumatic cylinders, and the position and velocity of the object being manipulated.

Pneumatics have non-negligible time constants (around 20 ms in our system), which is why the cylinder pressures represent additional state variables, making it difficult to apply torque-control techniques. The system also has a 40-dimensional continuous control space – namely the commands to the proportional valves regulating the flow of compressed air to the cylinders. The cylinders act on the joints through tendons. The tendons do not introduce additional state variables (since we avoid slack via pre-tensioning) but nevertheless complicate the dynamics. Overall this is a daunting system to model, let alone control.

The aim of our approach is to avoid the need for explicit analytic modeling of the hand and its interactions with other objects, and instead directly learn generalizable controllers that can perform a useful task in a variety of situations. Such controllers must be both flexible and responsive, so as to react to the particular variables in the environment (including the configuration of the hand and external objects) that are most important for completing the

**Algorithm 2** RL with linear-Gaussian controllers

1: initialize $p(\mathbf{u}_t|\mathbf{x}_t)$
2: **for** iteration $k = 1$ to $K$ **do**
3: run $p(\mathbf{u}_t|\mathbf{x}_t)$ to collect trajectory samples $\{\tau_i\}$
4: fit dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ to $\{\tau_j\}$ using linear regression with GMM prior
5: fit $p = \arg\min_p E_{p(\tau)}[\ell(\tau)]$ s.t. $D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \leq \epsilon$
6: **end for**

task. However, choosing the right controller representation by hand for such complex systems is exceedingly difficult

The manipulation skills we learn are represented as time-varying linear-Gaussian controllers. These controllers are fundamentally trajectory-centric, but otherwise are extremely flexible, since they can represent any trajectory with any linear stabilization strategy. Since the controllers are time-varying, the overall learned control law is nonlinear, but is locally linear at each time step. These types of controllers have been employed previously for controlling lower-dimensional robotic arms [53, 57].

### *8.2 Reinforcement Learning with Local Linear Models*

In this section, we describe the reinforcement learning algorithm (summarised in algorithm 2) that we use to control our pneumatically-driven five finger hand. The derivation in this section follows previous work [51], but we describe the algorithm in this section for completeness. The aim of the method is to learn a time-varying linear-Gaussian controller of the form $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{C}_t)$, where $\mathbf{x}_t$ and $\mathbf{u}_t$ are the state and action at time step $t$. The actions in our system correspond to the pneumatic valve's input voltage, while the state space is described in the preceding section. The aim of the algorithm is to minimize the expectation $E_{p(\tau)}[\ell(\tau)]$ over trajectories $\tau = \{\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$, where $\ell(\tau) = \sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)$ is the total cost, and the expectation is under $p(\tau) = p(\mathbf{x}_1) \prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)p(\mathbf{u}_t|\mathbf{x}_t)$, where $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ is the dynamics distribution.

### *8.2.1 Optimizing Linear-Gaussian Controllers*

The simple structure of time-varying linear-Gaussian controllers admits a very efficient optimization procedure that works well even under unknown dynamics. The method is summarized in Algorithm 2. At each iteration, we run the current controller $p(\mathbf{u}_t|\mathbf{x}_t)$ on the robot to gather $N$ samples ($N = 5$ in all of our experiments), then use these samples to fit time-varying linear-Gaussian dynamics of the form $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) = \mathcal{N}(f_{\mathbf{x}t}\mathbf{x}_t + f_{\mathbf{u}t}\mathbf{u}_t + f_{ct}, \mathbf{F}_t)$. This is done by using linear regression with a Gaussian mixture model prior, which makes it feasible to fit the dynamics even when the number of samples is much lower than the dimensionality of the system [51]. We also compute a second order expansion of the cost function around each of the samples, and average the expansions together to obtain a local approximate cost function of the form

$$\ell(\mathbf{x}_t, \mathbf{u}_t) \approx \frac{1}{2}[\mathbf{x}_t; \mathbf{u}_t]^{\mathrm{T}} \ell_{\mathbf{xu},\mathbf{xu}t}[\mathbf{x}_t; \mathbf{u}_t] + [\mathbf{x}_t; \mathbf{u}_t]^{\mathrm{T}} \ell_{\mathbf{xu}t} + \text{const.}$$

where subscripts denote derivatives, e.g. $\ell_{\mathbf{xu}t}$ is the gradient of $\ell$ with respect to $[\mathbf{x}_t; \mathbf{u}_t]$, while $\ell_{\mathbf{xu},\mathbf{xu}t}$ is the Hessian. The particular cost functions used in our experiments are described in the next section. When the cost function is quadratic and the dynamics are linear-Gaussian, the optimal time-varying linear-Gaussian controller of the form $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{C}_t)$ can be obtained by using the LQR method. This type of iterative approach can be thought of as a variant of iterative LQR [55], where the dynamics are fitted to data. Under this model of the dynamics and cost function, the optimal policy can be computed by recursively computing the quadratic $Q$-function and value function, starting with the last time step. These functions are given by

$$\begin{aligned} V(\mathbf{x}_t) &= \frac{1}{2}\mathbf{x}_t^{\mathrm{T}} V_{\mathbf{x},\mathbf{x}t}\mathbf{x}_t + \mathbf{x}_t^{\mathrm{T}} V_{\mathbf{x}t} + \text{const} \\ Q(\mathbf{x}_t, \mathbf{u}_t) &= \frac{1}{2}[\mathbf{x}_t;\mathbf{u}_t]^{\mathrm{T}} Q_{\mathbf{xu},\mathbf{xu}t}[\mathbf{x}_t;\mathbf{u}_t] + [\mathbf{x}_t;\mathbf{u}_t]^{\mathrm{T}} Q_{\mathbf{xu}t} + \text{const} \end{aligned}$$

We can express them with the following recurrence:

$$
\begin{aligned}
Q_{\mathbf{xu},\mathbf{xu}t} &= \ell_{\mathbf{xu},\mathbf{xu}t} + f_{\mathbf{xu}t}^{\mathrm{T}} V_{\mathbf{x},\mathbf{x}t+1} f_{\mathbf{xu}t} \\
Q_{\mathbf{xu}t} &= \ell_{\mathbf{xu}t} + f_{\mathbf{xu}t}^{\mathrm{T}} V_{\mathbf{x}t+1} \\
V_{\mathbf{x},\mathbf{x}t} &= Q_{\mathbf{x},\mathbf{x}t} - Q_{\mathbf{u},\mathbf{x}t}^{\mathrm{T}} Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u},\mathbf{x}t} \\
V_{\mathbf{x}t} &= Q_{\mathbf{x}t} - Q_{\mathbf{u},\mathbf{x}t}^{\mathrm{T}} Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u}t},
\end{aligned}
$$

which allows us to compute the optimal control law as $g(\mathbf{x}_t) = \hat{\mathbf{u}}_t + \mathbf{k}_t + \mathbf{K}_t(\mathbf{x}_t - \hat{\mathbf{x}}_t)$, where $\mathbf{K}_t = -Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u},\mathbf{x}t}$ and $\mathbf{k}_t = -Q_{\mathbf{u},\mathbf{u}t}^{-1} Q_{\mathbf{u}t}$. If we consider $p(\tau)$ to be the trajectory distribution formed by the deterministic control law $g(\mathbf{x}_t)$ and the stochastic dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, LQR can be shown to optimize the standard objective

$$
\min_{g(\mathbf{x}_t)} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)]. \tag{8.1}
$$

However, we can also form a time-varying linear-Gaussian controller $p(\mathbf{u}_t|\mathbf{x}_t)$, and optimize the following objective:

$$
\min_{p(\mathbf{u}_t|\mathbf{x}_t)} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] - \mathcal{H}(p(\mathbf{u}_t|\mathbf{x}_t)).
$$

As shown in previous work [52], this objective is in fact optimized by setting $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t, \mathbf{C}_t)$, where $\mathbf{C}_t = Q_{\mathbf{u},\mathbf{u}t}^{-1}$. While we ultimately aim to minimize the standard controller objective in Equation (8.1), this maximum entropy formulation will be a useful intermediate step for a practical learning algorithm trained with fitted time-varying linear dynamics.

### *8.2.2 KL-Constrained Optimization*

In order for this learning method to produce good results, it is important to bound the change in the controller $p(\mathbf{u}_t|\mathbf{x}_t)$ at each iteration. The standard iterative LQR method can change the controller drastically at each iteration, which can cause it to visit parts of the state space where the fitted dynamics are arbitrarily incorrect, leading to divergence. Furthermore, due

to the non-deterministic nature of the real world domains, line search based methods can get misguided leading to unreliable progress.

To address these issues, we solve the following optimization problem at each iteration:

$$\min_{p(\mathbf{u}_t|\mathbf{x}_t)} E_{p(\tau)}[\ell(\tau)] \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \leq \epsilon,$$

where $\hat{p}(\tau)$ is the trajectory distribution induced by the previous controller. Using KL-divergence constraints for controller optimization has been proposed in a number of prior works [9, 79, 80]. In the case of linear-Gaussian controllers, a simple modification to the LQR algorithm described above can be used to solve this constrained problem. Recall that the trajectory distributions are given by $p(\tau) = p(\mathbf{x}_1)\prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)p(\mathbf{u}_t|\mathbf{x}_t)$. Since the dynamics of the new and old trajectory distributions are assumed to be the same, the KL-divergence is given by

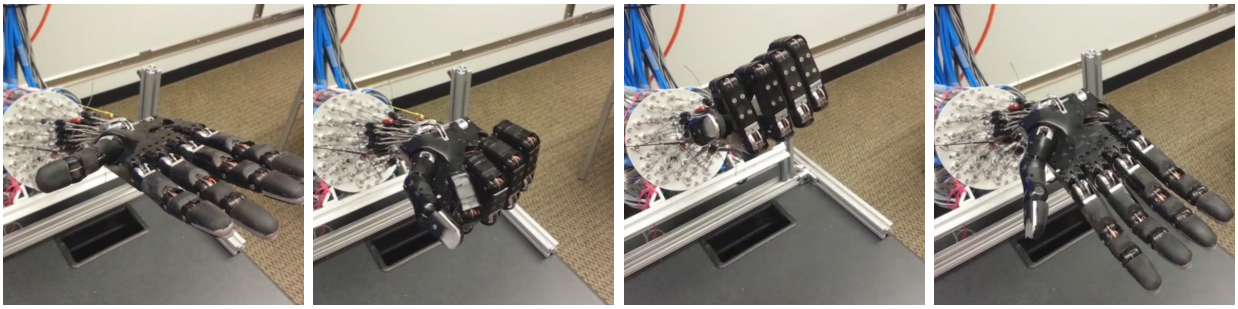
$$D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) = \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\log \hat{p}(\mathbf{u}_t|\mathbf{x}_t)] - \mathcal{H}(p),$$

and the Lagrangian of the constrained optimization problem is given by

$$\mathcal{L}_{\mathrm{traj}}(p, \eta) = E_p[\ell(\tau)] + \eta[D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) - \epsilon] =$$
$$\left[\sum_t E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \eta \log \hat{p}(\mathbf{u}_t|\mathbf{x}_t)]\right] - \eta\mathcal{H}(p(\tau)) - \eta\epsilon.$$

The constrained optimization can be solved with dual gradient descent [17], where we alternate between minimizing the Lagrangian with respect to the primal variables, which are the parameters of $p$, and taking a subgradient step on the Lagrange multiplier $\eta$. The optimization with respect to $p$ can be performed efficiently using the LQG algorithm, by observing that the Lagrangian is simply the expectation of a quantity that does not depend on $p$ and an entropy term. As described above, LQR can be used to solve maximum entropy control problems where the objective consists of a term that does not depend on $p$, and another term that encourages high entropy. We can convert the Lagrangian primal minimization into a problem of the form

$$\min_{p(\mathbf{u}_t|\mathbf{x}_t)} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\tilde{\ell}(\mathbf{x}_t, \mathbf{u}_t)] - \mathcal{H}(p(\mathbf{u}_t|\mathbf{x}_t))$$



(a) Start pose, against gravity (b) End pose, against gravity (c) Start pose, with gravity (d) End pose, with gravity

Figure 8.2: Positioning task

by using the cost $\tilde{\ell}(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{\eta}\ell(\mathbf{x}_t, \mathbf{u}_t) - \log \hat{p}(\mathbf{u}_t|\mathbf{x}_t)$. This objective is simply obtained by dividing the Lagrangian by $\eta$. Since there is only one dual variable, dual gradient descent typically converges very quickly, usually in under 10 iterations, and because LQR is a very efficient trajectory optimization method, the entire procedure can be implemented to run very quickly.

We initialize $p(\mathbf{u}_t|\mathbf{x}_t)$ with a fixed covariance $\mathbf{C}_t$ and zero mean, to produce random actuation on the first iteration. The Gaussian noise used to sample from $p(\mathbf{u}_t|\mathbf{x}_t)$ is generated in advance and smoothed with a Gaussian kernel with a standard deviation of two time steps, in order to produce more temporally coherent noise.

### *8.3 System and task description*

Modularity and the ease of switching robotic platforms formed the overarching philosophy of our system design. The training can either happen locally (on the machine controlling the robot) or remotely (if more computational power is needed). The learning algorithm (algorithm 2) has no dependency on the selected robotic platform except for step 3, where the policies are shipped to the robotic platform for evaluation and the resulting trajectories are collected. For the present work, the system was deployed locally on a 12 cores

3.47GHz Intel(R) Xeon(R) processor with 12GB memory running Windows x64. Learning and evaluated was studied for two different platforms detailed below.

### *8.3.1 Hardware platform*

The Adroit platform is described in detail in [48]. Here we summarize the features relevant to the present context. As already noted, the hand has 24 joints and 40 Airpel cylinders acting on the joints via tendons. Each cylinder is supplied with compressed air via a high-performance Festo valve. The cylinders are fitted with solid-state pressure sensors. The pressures together with the joint positions and velocities (sensed by potentiometers in each joint) are provided as state variables to our controller.

The manipulation task also involves an object – which is a long tube filled with coffee beans, inspired by earlier work on grasping [6]. The object is fitted with PhaseSpace active infrared markers on each end. The markers are used to estimate the object position and velocity (both linear and angular) which are also provided as state variables. Since all our sensors have relatively low noise, we apply a minimal amount of filtering before sending the sensor data to the controller.

### *8.3.2 Simulation platform*

We model the entire system, including the air dynamics, tendon actuation and hand-object interactions, in the MuJoCo simulator we have developed [106]. Here MuJoCo is used as a MEX file called from MATLAB. Simulating a 5 s trajectory at 2 ms timestep takes around 0.7 s of CPU time, or around 0.3 ms of CPU time per simulation step. This includes evaluating the feedback control law (which needs to be interpolated because the trajectory optimizer uses 50 ms time steps) and advancing the physics simulation.

Having a fast simulator enables us to prototype and quickly evaluate candidate learning algorithms and cost function designs, before testing them on the hardware. Apart from being able to run much faster than real-time, the simulator can automatically reset itself to a specified initial state (which needs to be done manually on the hardware platform).

(a) End pose, learned (b) End pose, human (c) End pose, learned (d) End pose, human

Figure 8.3: Object rotation task: end poses for the two rotation directions (a-b: clockwise, c-d: anticlockwise), comparing the learned controller to the movement of a human who has not seen the robot perform the task.

Ideally, the learning on the hardware platform should be seeded from the results of the software platform. This however is hard in practice due to (a) the difficulty in aligning the state space of the two platforms, (b) the non-deterministic nature of the real world. State space alignment requires precise system identification and sensor calibration which otherwise are not necessary, as our algorithm can learn the local state space information directly from the raw sensor values.

## *8.4 Learning Policies from Experience*

In this section, we will describe our first set of experiments, which uses the trajectory-centric reinforcement learning algorithm in the previous section to learn dexterous manipulation skills from scratch on both the physical and simulated ADROIT platform. The experiments in this section are aimed to ascertain whether we can learn complex manipulation behaviors entirely from scratch, using only high-level task definitions provided in terms of a cost function, with the controller learned at the level of valve opening and closing commands. The particular tasks are detailed in Table 8.1 and 8.2 and shown in the accompanying video, and include both hand posing behaviors and object manipulation skills.

Table 8.1: Different hand positioning task variations learned

| **Platform** | **# different task variations learned** |
|---|---|
| Hardware | 5 (move assisted by gravity)<br>2 (move against gravity) |
| Simulated | 5 (move assisted by gravity)<br>3 (move against gravity) |

Table 8.2: Different object manipulation task variations learned

| **Platform** | **# different task variations learned** |
|---|---|
| Hardware +<br>an elongated<br>object (Fig:8.3) | 2 ({clockwise & anti-clockwise}<br>object rotations along vertical) |
| Simulated +<br>4 object<br>variations | 13 ({clockwise, anti-clockwise,<br>clockwise then anti-clockwise}<br>object rotation along vertical<br>8 ({clockwise, anti-clockwise}<br>object rotation along horizontal) |

### *8.4.1 Hand Behaviors*

In the first set of tasks, we examine how well trajectory-centric reinforcement learning can control the hand to reach target poses. The state space is given by $\mathbf{x} = (q, \dot{q}, a)$. Here, $q$ denotes the vector of hand joint angles, $\dot{q}$ is the vector of joint angular velocities, $a$ the vector of cylinder pressures, and the actions $\mathbf{u}_t$ correspond to the valve command signals, which are real-valued and correspond to the degree to which each valve is opened at each time step. The tasks in this section require moving the hand to a specified pose from a given initial pose. We arranged the pair of poses such that in one task-set the finger motions were helped by gravity, and in another task-set they had to overcome gravity, as shown in Figure 8.2. Note that for a system of this complexity, even achieving a desired pose can be challenging, especially since the tendon actuators are in agonist-antagonist pairs and the forces have to balance to maintain posture. The cost function at each time step is given by

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = ||q_t - q^*||^2 + 0.001||\mathbf{u}_t||^2,$$

and the cost at the final time step $T$ emphasizes the target pose to ensure that it is reached successfully:

$$\ell(\mathbf{x}_T, \mathbf{u}_T) = 10||q_t - q^*||^2.$$

### *8.4.2 Object Manipulation Behaviors*

The manipulation tasks we focused on require in-hand rotation of elongated objects. We chose this task because it involves intermittent contacts with multiple fingers and is quite dynamic, while at the same time having a certain amount of intrinsic stability. We studied different variations (table 8.2) of this task with different objects: rotation clockwise (figure 8.1), rotation counter-clockwise, rotation clockwise followed by rotation counter-clockwise, and rotation clockwise without using the wrist joint (to encourage finger oriented maneuvers) – which was physically locked in that condition. Figure 8.3 illustrates the start and end poses and object configurations in the task learned on the ADROIT hardware platform. The

running cost was

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = 0.01||q_t - q^*||^2 + 0.001||\mathbf{u}_t||^2 + ||q_t^{pos} - q^{pos*}||^2 + 10||q_t^{rot} - q^{rot*x}||^2$$

where $\mathbf{x} = (q, q^{pos}, q^{rot}, \dot{q}, \dot{q}^{pos}, \dot{q}^{rot}, a)$. Here $q$ denotes the vector of hand joint angles, $q^{pos}$ the object positions, $q^{rot}$ the object rotations, $a$ the vector of cylinder pressures, and $\mathbf{u}_t$ the vector of valve command signals. At the final time we used

$$\ell(\mathbf{x}_t, \mathbf{u}_t)_{t=T} = 2[0.01||q_t - q^*||^2 + ||q_t^{pos} - q^{pos*}||^2 + 10||q_t^{rot} - q^{rot*x}||^2].$$

Here, the cost function included an extra term for desired object position and orientation. The final cost was scaled by a factor of 2 relative to the running cost.

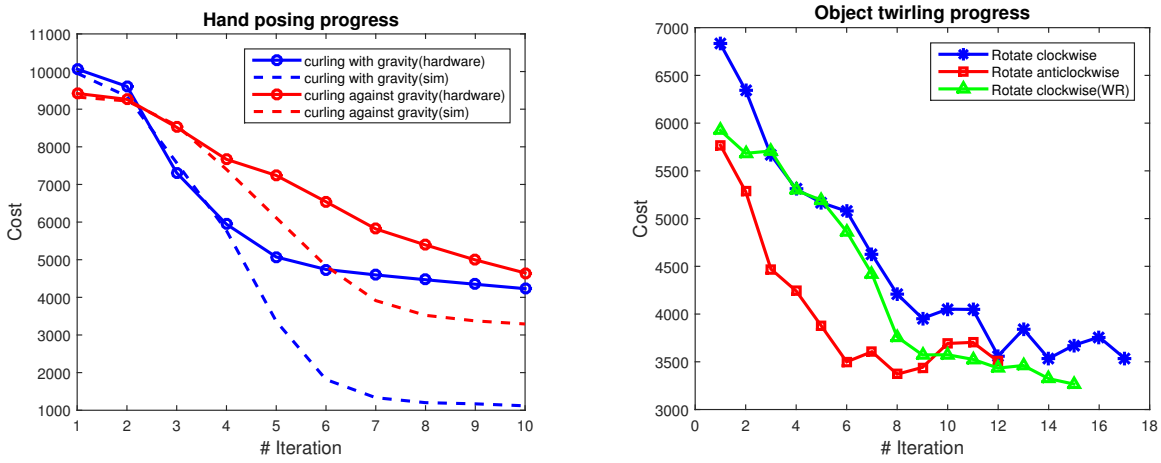



Figure 8.4: Learning curves for the positioning (top) and manipulation (bottom) tasks. [1]

## *8.5 Results*

The controller was trained as described above. Once the order of cost parameters were properly established, minimal parameter tuning was required. Training consisted of around 15 iterations. In each iteration we performed 5 movements with different instantiations of the

exploration noise in the controls. The progress of training as well as the final performance is illustrated in the video, and in the figure 8.1.

Here we quantify the performance and the robustness to noise. Figure 8.4 shows how the total cost for the movement (as measured by the cost functions defined above) decreased over iterations of the algorithm. The solid curves are data from the physical system. Note that in all tasks and task variations we observe very rapid convergence. Surprisingly, the manipulation task which is much harder from a control viewpoint takes about the same number of iterations to learn.

In the positioning task we also performed a systematic comparison between learning in the physical system and learning in simulation. Performance early in training was comparable, but eventually the algorithm was able to find better policies in simulation. Although it is not shown in the figure, training on simulation platform happens a lot faster, because the robot can only run in real-time while the simulated platform runs faster than real-time, and because resetting between repetitions needs to be done manually on the robot.

We further investigated the effects of exploration noise magnitude injected during training. Figure 8.5 shows that for a relatively small amount of noise performance decreases monotonically. As we increase the noise magnitude, sometimes we see faster improvement early on but the behavior of the algorithm is no longer monotonic. These are data on the Adroit hardware platform.

### *8.5.1 Delayed Robustification*

Finally, we used the simulation platform to investigate robustness to perturbations more quantitatively, in the manipulation task. We wanted to quantify how robust our controllers are to changes in initial state (recall that the controllers are local). Furthermore, we wanted to see if training with noisy initial states, in addition to exploration noise injected in the controls, will result in more robust controllers. Naïvely adding initial state noise at each

[1] At each iteration, the current controller $p(\mathbf{u}_t|\mathbf{x}_t)$ is deployed on the robot to gather $N$ samples ($N = 5$ in all of our experiments).

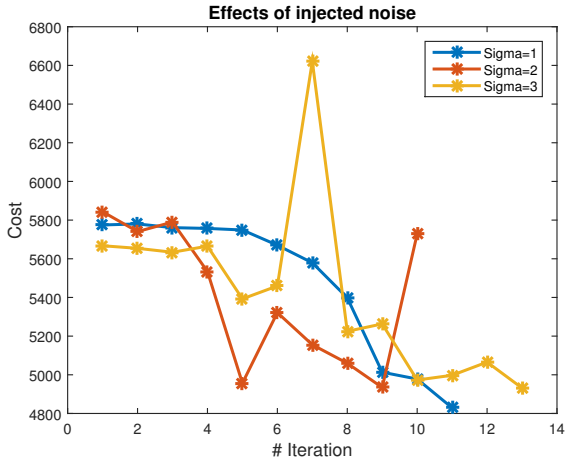



Figure 8.5: Effect of noise smoothing on learning. Sigma=1 (width of the Gaussian kernel used for smoothing noise), takes a slow start but maintains a constant progress. Higher sigma favors steep decent but it fails to maintain the progress as it is unable to successfully maintain the stability of the object being manipulated and ends up dropping it. The algorithm incurs a huge cost penalty and restarts its decent from there. [1]

iteration of the algorithm (Algorithm 2) severely hindered the overall progress. However, adding initial state noise after the policy was partially learned (iteration $\geq 10$ in our case) resulted in much more robust controllers.

The results of these simulations are shown in Figure 8.6. We plot the orientation of the object around the vertical axis as a function of time. The black curve is the unperturbed trajectory. As expected, noise injected in the initial state makes the movements more variable, especially for the controller that was trained without such noise. Adding initial state noise during training substantially improved the ability of the controller to suppress perturbations in initial state. Overall, we were surprised at how much noise we could add (up to 20 % of the range of each state variable) without the hand dropping the object, in the case of the controller trained with noise. The controller trained without noise dropped the object in 4 out of 20 test trials. Thus injecting some noise in the initial state (around 2.5 %) helps im-

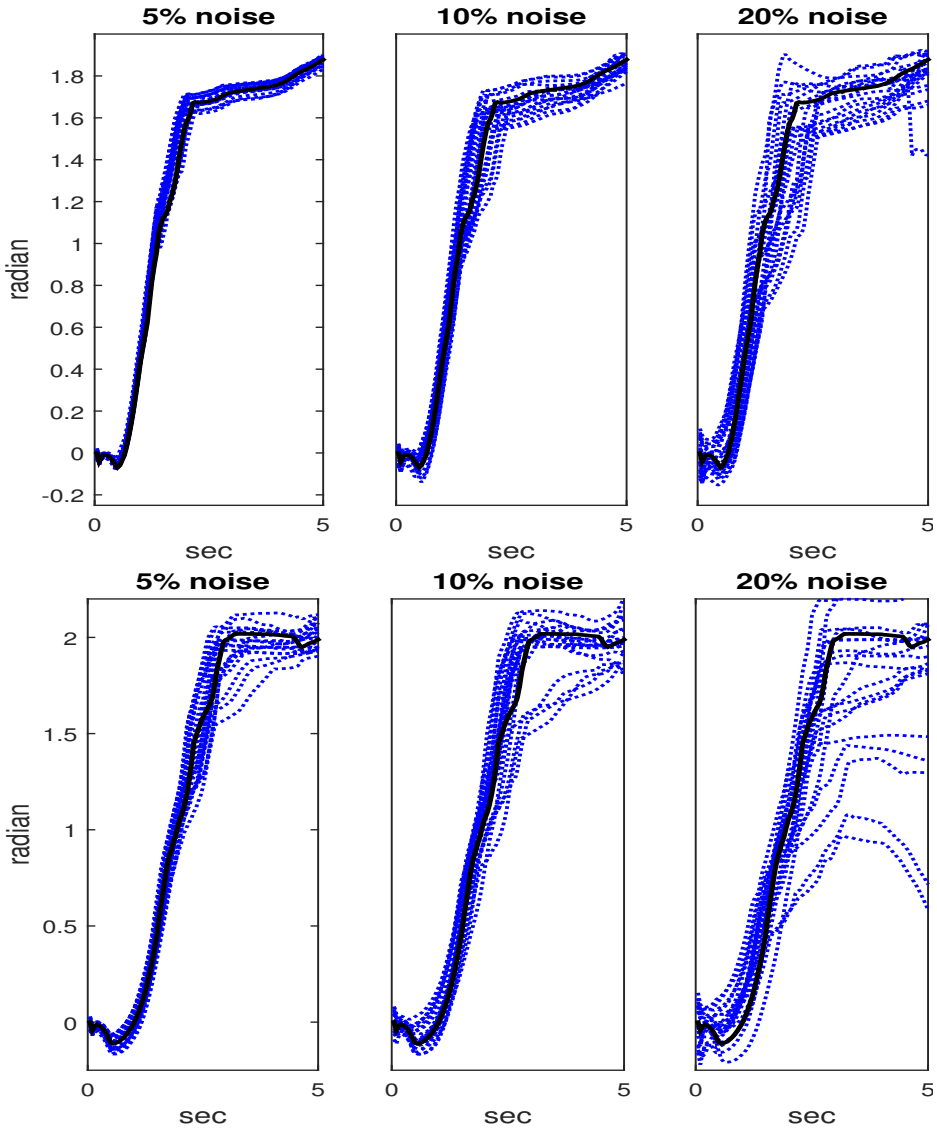



Figure 8.6: Robustness to noise in initial state. Each column corresponds to a different noise level: 5, 10, 20 % of the range of each state variable. The top row is a controller trained with noise in the initial state. The bottom row is a controller trained with the same initial state (no noise) for all trials.

prove robustness. Of course on the real robot we cannot avoid injecting such noise, because exact repositioning is very difficult.

## *8.6 Summary*

We demonstrated learning-based control of a complex, high-dimensional, pneumatically-driven hand. Our results show simple tasks, such as reaching a target pose, as well as dynamic manipulation behaviors that involve repositioning a freely-moving cylindrical object. Aside from the high-level objective encoded in the cost function, the learning algorithm does not use domain knowledge about the task or the hardware, learning a low-level valve control strategy for the pneumatic actuators entirely from scratch. The experiments show that effective manipulation strategies can be automatically discovered in this way.

## *8.7 Next Steps*

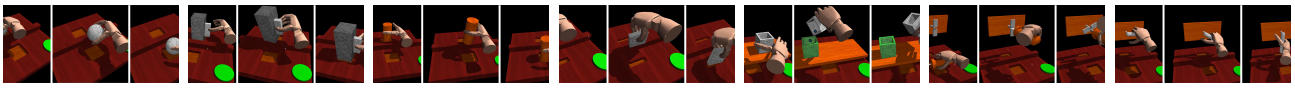
While the linear-Gaussian controllers we employ offer considerable flexibility and closed-loop control, they are inherently limited in their ability to generalize to new situations, since a time-varying linear strategy may not be effective when the initial state distribution is more diverse. Furthermore, due to the local nature of the learning from experience techniques presented in this chapter, it can get stuck in local minima. In next few chapters we will focus on dealing with the challenges faced while scaling techniques developed here to more complicated tasks. We will finally address generalization. More specifically, in chapter 10 we will leverage guided exploration (i.e. exploration guided using expert demonstrations) in order to scale our current techniques to complex tasks and leverage machine learning tools to achieve generalization.

While leveraging an expert is quite desirable in a wide variety of situations, deploying and exploiting an expert is exceptionally difficult for dexterous manipulation. This is due to two principal factors. First, dexterous manipulation strategies are extremely sensitive to minor variations in the contact forces, contact locations, and object positions. Thus, minor deviations from the expert demonstrations render the demonstration useless. Second, technology to capture the details of hand manipulation is often unreliable. Unlike full body movements, hand manipulation behaviors unfold in a compact region of space co-inhabited by

the objects being manipulated. This makes motion capture difficult, due to occlusions and, in the case of passive systems, marker confusion. Manipulation also involves large numbers of contacts, including dynamic phenomena such as rolling, sliding, stick-slip, deformations, and soft contacts. The human hand takes advantage of these rich dynamics, but recording the data and interpreting it with regard to well-defined physics models is challenging and hasn't been demonstrated yet. Before we detail guided exploration using an expert, its important to capture an expert with necessary details. In the next chapter, we will develop a technology capable of hosting and capturing physically realistic dexterous manipulation.

Chapter 9

# VIRTUAL REALITY SYSTEM FOR RECORDING DEXTEROUS HAND MANIPULATION BEHAVIORS



Data-driven methods have lead to advances in multiple fields including robotics. These methods however have had limited impact on dexterous hand manipulation, partly due to lack of rich and physically-consistent dataset as well as technology able to collect them. To fill this gap, we developed a virtual reality system combining real-time motion capture, physics simulation and stereoscopic visualization. The system enables a user wearing a CyberGlove to "reach-in" the simulation, and manipulate virtual objects through contacts with a tele-operated virtual hand. The system is evaluated on a subset of tasks in the Southampton Hand Assessment Procedure – which is a clinically validated test of hand function. The system is also being used by performer teams in the DARPA Hand Proprioception & Touch Interfaces program to develop neural control interfaces in simulation. The software is freely available at www.mujoco.org

## *9.1 Introduction*

Dexterous hand manipulation lags behind other areas of robotics such as kinematic motion planning or legged locomotion. While there are multiple reasons for this, here we focus on the challenges specific to data-driven approaches. Unlike full body movements, hand manipulation behaviors unfold in a compact region of space co-inhabited by the objects being manipulated. This makes motion capture difficult, due to occlusions as well as marker con-

fusion in the case of passive systems. Manipulation also involves large numbers of contacts, including dynamic phenomena such as rolling, sliding, stick-slip, deformations and soft contacts. The human hand can take advantage of these rich dynamics, but recording the data and interpreting it with regard to well-defined physics models is challenging and has not been done in a systematic way.

The solution we propose is to leverage the adaptation abilities of the human brain, and move the data collection from the physical world to a physically-realistic simulation. The simulation is based on the MuJoCo physics engine we have developed [105]. We have recently shown [30] that MuJoCo outperforms a number of alternative simulators in terms of both speed and accuracy on modelling systems relevant to robotics, especially simulated hands grasping objects. We augment the simulator with real-time motion capture of arm and hand movements, and stereoscopic

visualization using OpenGL projection from the viewpoint of the user's head (which is also tracked via motion capture.) The resulting system has empirically-validated end-to-end latency of 42 msec. It creates a sense of realism which is sufficient for untrained human users to interact with virtual objects in a natural way, and perform tasks selected from the Southampton Hand Assessment Procedure (SHAP).

The system is called MuJoCo HAPTIX. We have developed it for DARPA, with the goal of facilitating research in the ongoing Hand Proprioception & Touch Interfaces (HAPTIX) program. A number of performer teams are already using it to explore novel neural interfaces for prosthetic hands. Here present our own tests of the system's latency and usability, and show that humans can indeed perform manipulation tasks with virtual objects. This clears the way to collecting rich and physically-consistent dataset of hand-object interactions. Since the interaction happens in simulation, we can record every aspect of it – including joint kinematics and dynamics, contact interactions, simulated sensor readings etc. There are no sensor technologies available today that could record such rich dataset from hand-object interactions in the physical world. Furthermore, since the interaction is based on our simulation model of multi-joint and contact dynamics, the dataset is by definition physically-

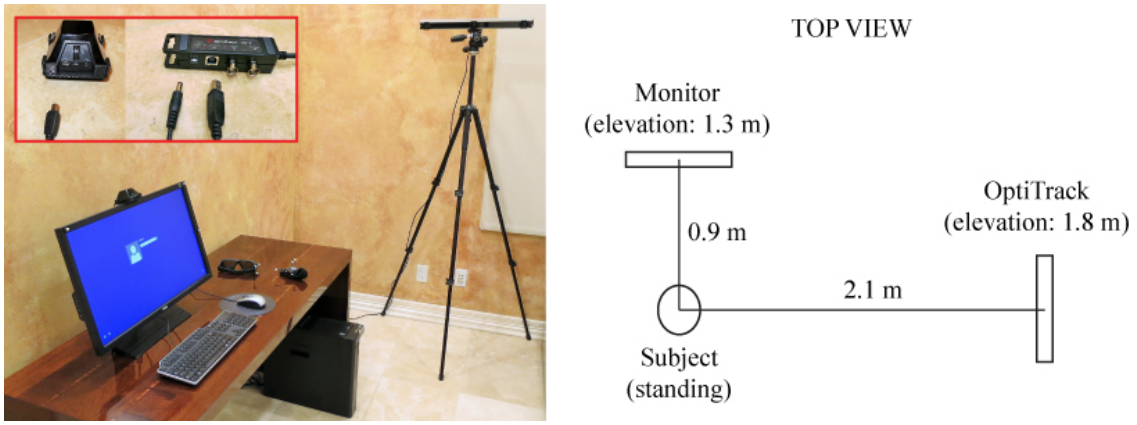



Figure 9.1: System overview. Left: Overall system. Right: Schematic representation of the VR system with relative distances.

consistent. Systematic collection of larger dataset is left for future work. Here we focus on describing the system and demonstrating its capabilities.

The key ingredients of Mujoco Haptix system are (a) Realistic real-time physics simulation with one-to-one mapping to the physical world, and (b) Ability to record physically-consistent dataset with dynamic interaction phenomena including contact forces, rolling, slip, deformation etc. The human user is allowed to make arbitrary hand movements, and the environment responds correctly, enabling virtual object manipulation with minimal adaptation on behalf of the user.

## *9.2 MuJoCo HAPTIX*

### *9.2.1 Hardware*

MuJoCo HAPTIX uses standard hardware components, except for the 3D-printed attachments for the motion capture markers and the glasses emitter (Fig 9.3). The default computer system is a Dell Precision T5810 workstation with Intel Xeon E5-1650 v3 processor, NVidia Quadro K4200 video card, 8GB of 2133MHz DDR4 RAM, 256GB SSD. The software relies on quad-buffered OpenGL for stereoscopic 3D rendering. It is not memory or I/O intensive and the CPU is mostly idle – which is because the MuJoCo physics engine can run realistic

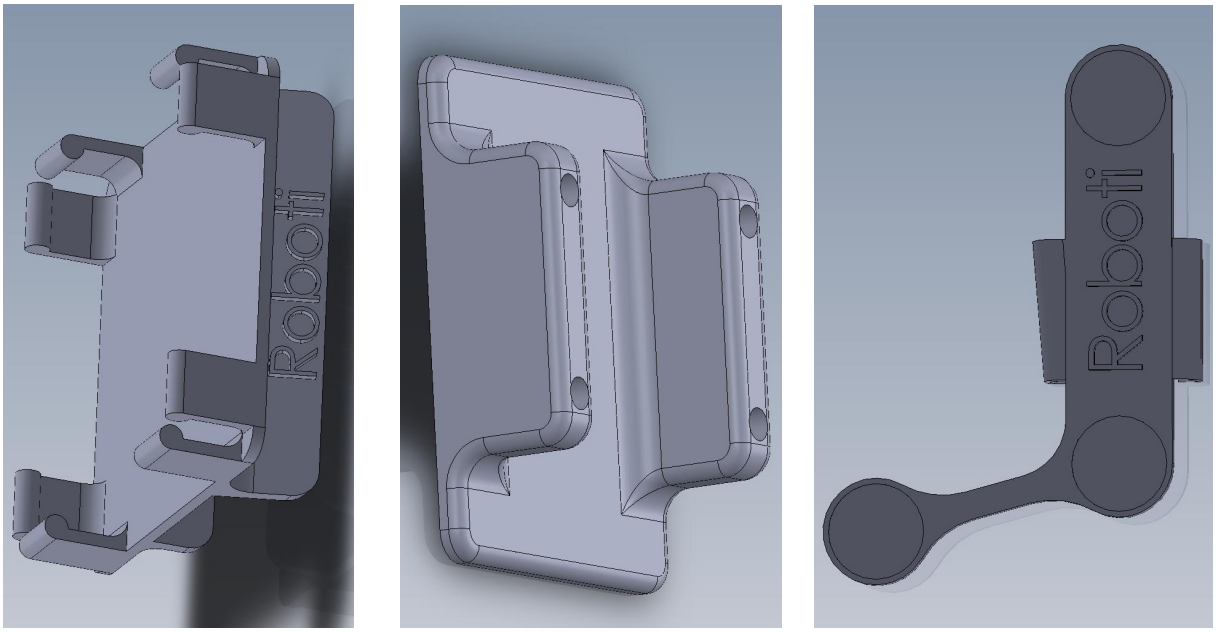
Figure 9.2: Motion capture markers

simulations much faster than real-time. NVidia 3D Vision 2 glasses are used to stereoscopic visualization, together with a BenQ GTG XL2720Z stereo monitor. MoCap markers are attached to the glasses for head tracking using 3D printed attachment (Fig 9.3c). The emitter is fitted in a 3D printed holder (Fig9.3a) and placed on top of the monitor. This avoids blanking of the stereo glasses due to occlusion of the emitter via hand. MoCap markers are also attached to the monitor and used to define the frame for the motion capture data; in this way the infrared cameras can be moved during a session without any effect on the simulation and rendering.

Motion capture is based on the OptiTrack Trio:V120 system [76]. This device has tracking speed and accuracy comparable to devices that cost substantially more. Its main limitation is that all three cameras are mounted in one elongated bar. This is sufficient to achieve stereo vision, but since all three views are quite similar, the system cannot track the hand in situations where markers are occluded or overlap. For head and monitor tracking this is not an issue, and the hand tracking workspace is sufficient for object manipulation tasks. Finally, we use a CyberGlove for wrist and finger tracking[22].

### *9.2.2 Markers and attachments*

The MoCap markers and their attachments are shown in figure 9.2 and 9.3 respectively. For the head and the hand tracking we use custom 3D-printed parts to which the markers are glued. There are three markers per rigid body. The hand-tracking body uses 7/16”




(a) Emitter holder (b) Hand attachment (c) 3D-Glasses

Figure 9.3: 3D printed marker attachments

markers from NaturalPoint (makers of the OptiTrack), while the rest are 9mm removable-base markers from MoCap Solutions. The head-tracking attachment (Fig 9.3c) slides on the stereo glasses (Fig9.2-left pane) to track the head position of the user. The hand-tracking attachment (Fig 9.3b) has Velcro on the bottom, and attaches to a mating velcro strap that goes over the wrist (Fig 9.2-mid pane) for tracking the base of the hand. Note the orientation of the hand-tracking body. Attaching it in the wrong orientation significantly reduces the usable workspace in terms of forearm pronation-supination. The monitor markers are attached directly to the monitor bezel. The positioning of these markers (as shown in Fig 9.2-right pane) is important, because the software uses them to compute the position, orientation and size of the LCD panel – which in turn is needed for rendering from a head camera perspective.

### *9.2.3 Tracking*

MuJoCo HAPTIX expects a real-time stream with information about the 6D (3D position plus 3D orientation) orientation of the monitor, the users's head and the user's hand. This is provided by the OptiTrack library which is loaded in MuJoCo HAPTIX. For hand tracking a combination of data stream from OptiTrack and CyberGlove is used.

#### *Head tracking*

The user's head and the screen are tracked via the markers attached to the screen and glasses. The markers attached to the monitor are also used to measure the physical dimensions of the screen, which are then taken into account to create appropriate projection. The 6D head orientation is used to render the scene from the physical location of the eyes using oblique projections to create the impression that the virtual world is glued to the monitor.

#### *Hand tracking*

The hand-tracking body attached to the user's wrist controls the base of the simulated hand, but this control is not direct. "Direct control" would involve setting the position and orientation of the virtual hand base equal to the motion capture data. This has undesirable effects in terms of physics simulation and sensor modeling. Instead we use the motion capture data to set the pose of a dummy body, and connect this body to the base of the virtual hand with a soft equality constraint. The constraint is enforced by the MuJoCo solvers that compute the contact forces and the joint friction and the joint limit forces. By adjusting the relative softness of the different constraints, we can set their priority. For example, if the user attempts to move the hand into the table, there is a tradeoff between assigning priority to the non-penetration constraints (in which case the dummy body will move into the table but the virtual hand will remain on the surface), and priority to tracking the dummy body as faithfully as possible.

A CyberGlove, calibrated for the hand model in use, is used to set the position offset

of the PD controller driving the finger joints. The OptiTrack data stream fused with the CyberGlove data stream enables hand tracking.

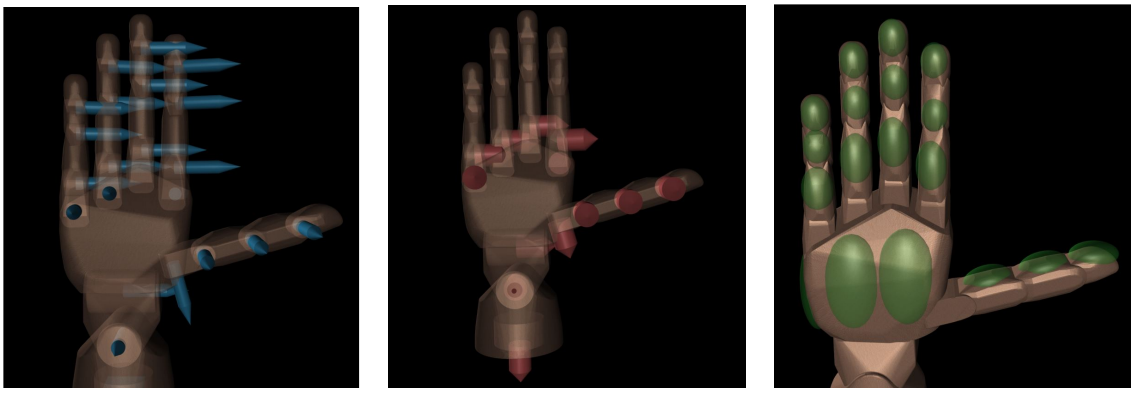
### *9.3 Modelling*

We modelled the Modular Prosthetic Limb (MPL) [66] and ADROIT hand [47] (Fig 9.5) for testing purposes. The two hand models are expressed in MuJoCo's model definition language called MJCF. These models can be populated inside diverse Virtual Environments, also expressed in MJCF.

MPL model consists of 22 dof (Fig 9.4a) (19 in the fingers and 3 in the wrist) and 13 actuated dof (Fig 9.4b). Finger flexion and extension are coupled using differential arrangement and actuated using a single actuator. Ring and little finger's adduction-abduction are coupled together and actuated using a single actuator. MuJoCo's equality constraints are leveraged to model all couplings. Index finger MCP adduction-abduction, all thumb joints and all wrist joints have independent actuation. Convex meshes derived from the original CAD models are used for collision purposes. In accordance with the sensing capabilities of the real hand, the state of the system is exposed to the users via MuJoCo's simulated sensors. Sensory capabilities include - joint position and velocity sensors on all 22 joints, actuator position, velocity and force sensors on all 13 actuators, touch sensors (Fig 9.4c) and inertial measurement units on all 5 finger tips.

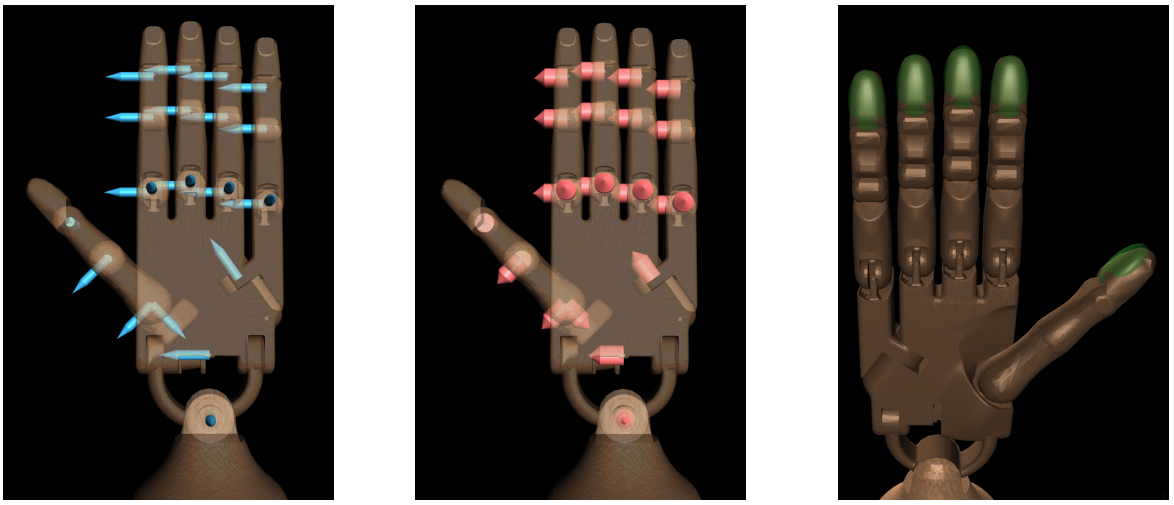
ADROIT model consists of 24 dof (Fig 9.5a) with 20 independently actuated dof (9.5b). Finger's distal joints are coupled with the proximal joints using MuJoCo's tendon constraints. Convex meshes derived from CAD models are used for collision detection. Sensing capabilities include – joint sensors on all 24 joints, position and force sensors on all 20 actuators, and touch sensors on the distal segments of all fingers (Fig 9.5c). Here we focus on MPL hand, but similar results are available for ADROIT hand as well.

Virtual environments (Fig 9.8 & 9.9) were primarily modelled using geometric shape primitives for fast and well behaved collisions. Note that MuJoCo doesn't support fluid modelling. Multiple small and smooth balls were instead used as a replacement.



(a) Joints (b) Actuators (c) Touch sensors

Figure 9.4: MPL sensor configurations



(a) Joints (b) Actuators (c) Touch sensors

Figure 9.5: Adroit sensor configurations

## *9.4 Applications*

MuJoCo HAPTIX is multipurpose platform, capable of supporting various applications across domains. Most applications emerge from its primary design objective i.e. realizing physically consistent data collection from demonstrations. One of the primary application cluster is centered around data driven machine leaning techniques for understanding movements and synthesising behaviours. Leaning from demonstration techniques in robotics can also leverage the capability of our system to capture physically realistic rich demonstrations.

Another use case cluster centers around medical robotics, rehabilitation and prosthesis applications. For these applications, information like joint torques, contact events, actuator forces etc can help study the effectiveness of the mechanisms and its effects on the users. MuJoCo HAPTIX is fast with negligible latency. Access to physical entities can also be leveraged to investigate the sensory motor feedback required in order to close the loop between the prosthesis and its human counterpart. To further this endeavour, Mujoco HAPTIX is being used for Darpa's ongoing Hand Proprioception & Touch Interfaces (HAPTIX) program (see section 9.6).

In section 9.5.3 we elaborate on how the platform can be leveraged to perform hand design iterations prior to manufacturing resulting in cost and time benefits. The diverse dataset collected from demonstrations (say from experiments mentioned in sec 9.5.2 & 9.5.3) conceals rich information about how human brain leverages phenomena like contacts, friction, stiction, deformation, sliding, rolling etc to perform dexterous manipulation. Inverse optimal control techniques can be deployed to study and uncover the underlying objectives being achieved.

## *9.5 Evaluation and results*

### *9.5.1 Latency-tests*

The latency of a virtual environment is an important factor affecting human sensorimotor performance. End-to-end latency, from movement of the physical MoCap marker to change of the pixels on the monitor, of our VR system was established to be between 42 and 45

msec. This was determined empirically in two different ways.

The first – the last two marker positions obtained from the MoCap was used to extrapolate its position some time into the future (45 msec in this case) under constant velocity assumption. Both the current (white) and extrapolated (green) positions were rendered (Fig 9.6), using projection to the surface of the monitor. Hence marker movement closer to the monitor allowed a visual comparison of the physical and the rendered marker positions. The prediction interval was adjusted such that the extrapolated (green) marker neither leads nor lags the physical marker, but instead was aligned with it on average. Extrapolation was done under a constant velocity assumption, thus non-zero acceleration would affect the measurement, but hand acceleration is zero on average (it changes direction) hence the result is unbiased. In figure 9.6 the hand is moving and the green marker cannot be seen because it is exactly under the physical marker. In the right panel the hand is stationary, thus the white and green markers are on top of each other. They appear above the physical marker because it is some distance away from the monitor, and the camera is above the hand.

The second - a tap was applied to the hand tracking body with a pen (figure 9.6 right panel), recorded with a 120Hz camera, and the video frames separating the physical contact event and the change in virtual marker speed (which was printed on the screen) was counted. Similarly, we also moved the physical marker up and down across a horizontal line, and counted the number of video frames between the physical and virtual markers crossing the line. These tests showed around 42 msec overall latency, and the frame counts were very consistent (5 frames in almost all the cases).

The total amount of time that motion capture data spends in the VR pipeline - from the time it is delivered by the motion capture library to the time the video driver reports that rendering is finished, is between 6 msec and 12 msec, because the relative timing of the motion capture and video card fluctuates. It includes processing of the motion capture data, simulation and rendering. The rest is due to latency in the hardware devices in use.

Overall, the virtual environment is very responsive and usable.

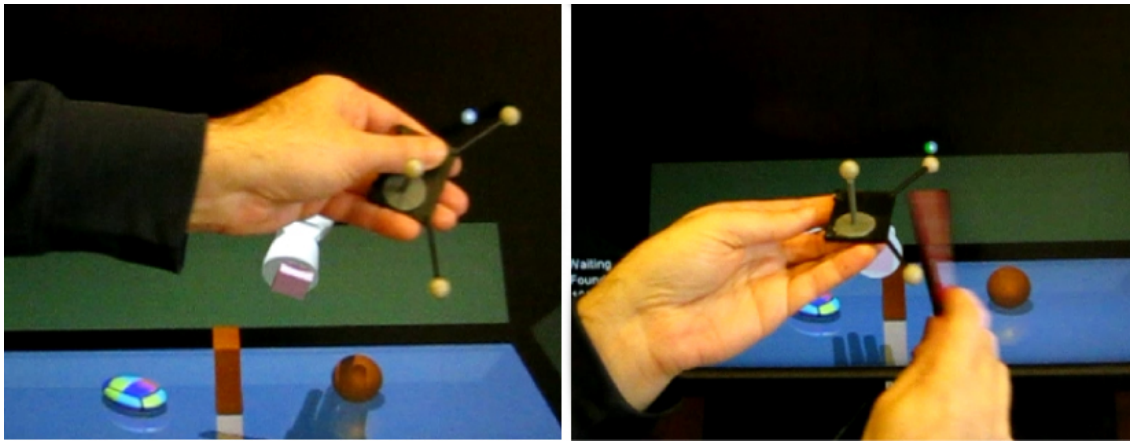


Figure 9.6: Latency experiments. (Left) - The hand is moving. The green marker is exactly below the physical marker. (Right) - The hand is stationary. The green and the white marker are on top of each other.

### *9.5.2 SHAP tests*

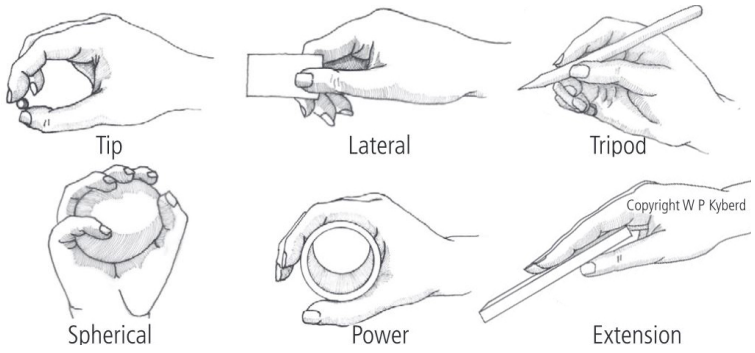
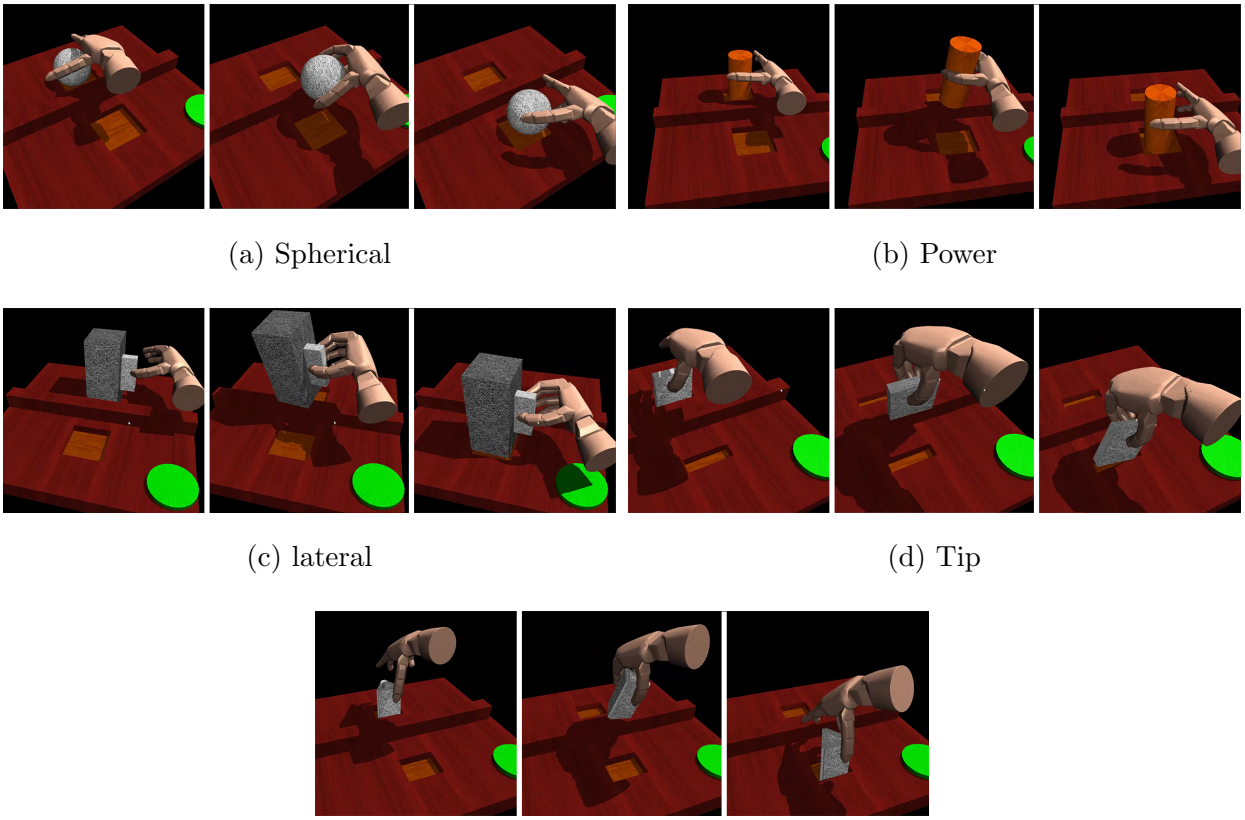



Figure 9.7: Six grip classifications used in Southampton Hand Assessment Procedure assessment (courtesy- W P Kyberd)

To evaluate the manipulation capabilities, we modelled Southampton Hand Assessment Procedure (SHAP) [56] in our virtual environment. SHAP is a clinically validated hand function tests developed by Colin Light, Paul Chappell and Peter Kyberd in 2002 at the University of Southampton. Originally developed to assess the effective functionality of upper limb prostheses, the SHAP has now been applied for assessments of musculoskeletal

and unimpaired participants too. The SHAP is made up of 6 abstract objects (AO) and 14 Activities of Daily Living (ADL). Each task is timed by the participant, so there is no interference or reliability on the reaction times of the observer or clinician. Listed below are the SHAP procedures we tested against. Note that we ignored the procedures that can't be modelled using rigid body dynamics.



(a) Spherical

(b) Power

(c) lateral

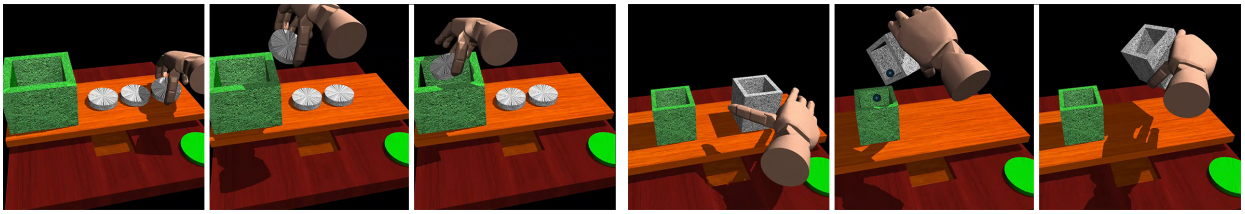
(d) Tip

(e) Tripod

Figure 9.8: Abstract Objects SHAP test sequence

a. AO-Spherical (Fig 9.8a) evaluates spherical gripping capabilities as the user moves a spherical object over a barrier to a slot in front.

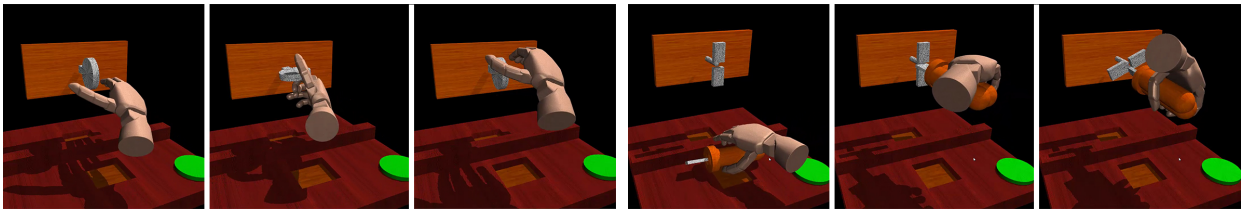
b. AO-Power (Fig 9.8b) evaluates power grasping capability as the user moves a cylindrical object over a barrier to a slot in front.



(a) Pick Coins

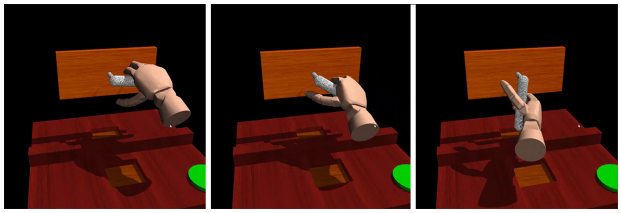


(b) Pouring



(c) Key

(d) Screw



(e) Handle

Figure 9.9: Activity for daily learning SHAP test sequence

c. AO-Lateral (Fig 9.8c) evaluates lateral grasping capability as the user repositions an object with a handle over a barrier.

d. AO-Tip (Fig 9.8d) evaluates grasping capabilities as the user deploys trip grasp to reposition an object over a barrier.

e. AO-Tripod (Fig 9.8e) evaluates tripod grip by asking the user to move an triangular pipe shaped object over a barrier to a slot in front.

f. ADL-Pick Coins (Fig 9.9a) uses tripod or tip grip to drop a sequence of coins from the table into the jar.

g. ADL-Pouring (Fig 9.9b) uses lateral or power grip to pour liquid (roll small rigid balls in this case) from one jar to the other.

h. ADL-Key (Fig 9.9c) evaluates rotation of a key by 90 degrees

i. ADL-Screw (Fig 9.9d) evaluates rotation of a screw by 90 degrees using screwdriver (power grasp)

j. ADL-Handle (Fig 9.9e) evaluates rotating door handle using power grasp.

Our user group comprised of 7 subjects between the age of 22-28. Each subject starts by calibrating the CyberGlove against the hand model in use. She is then allowed some exploration time before the trial starts. She begins by pressing the button to start the timer, performs the task and then presses the button again to stop the timer. Each SHAP procedure is repeated 10 time. Each user performs the evaluation for two different version of the hand under study - fully actuated and with coupled actuation. The resulting 1400 user demonstrations across 10 tasks provide convincing results to back the claims made in chapter regarding usability of the system and its ability to support manipulation.

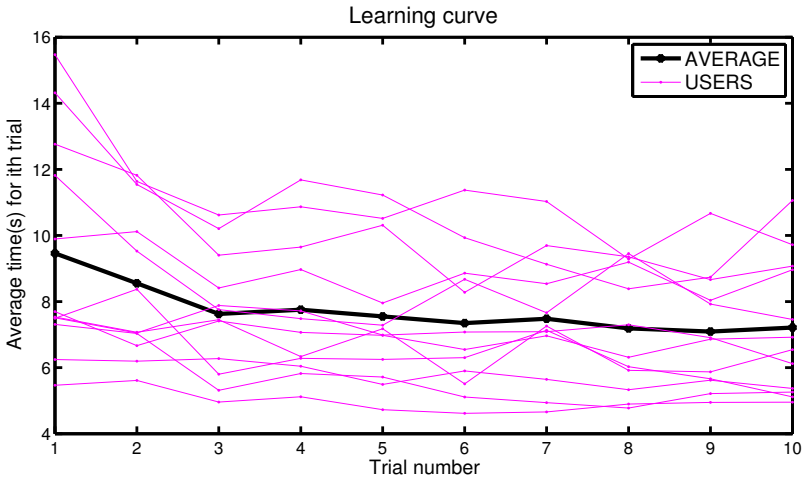



Figure 9.10: Learning curve of the MuJoCo HAPTIX system

The average of the time taken by a user on any particular trial across all SHAP procedures serves as metric to evaluate the usability of the system. Lower the average time, more usable the system. The usability metric plotted over all the trial (Fig 9.10) reveals the learning curve of our system. The curve ramps down slowly and plateaus after 4 trial indicating the low learning curve of our system.

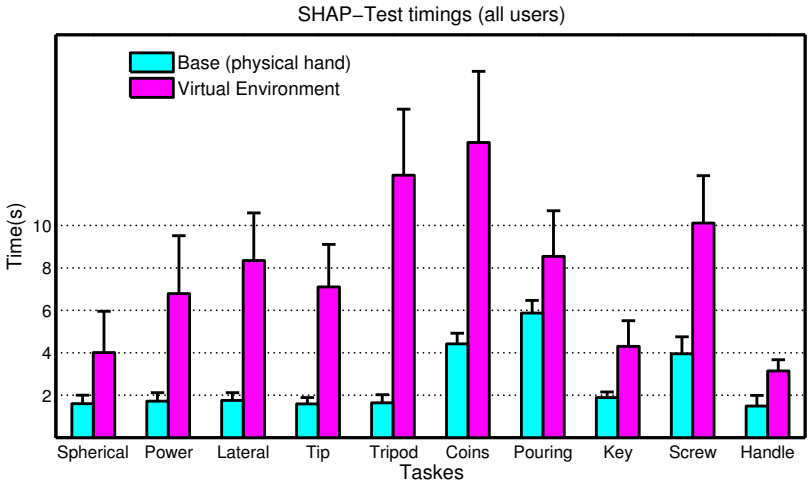



Figure 9.11: SHAP timing comparison between physical tasks and tasks performed in MuJoCo HAPTIX, by healthy control groups

Figure 9.11 summarises the SHAP timing from MuJoCo HAPTIX, along with the results originally reported in [56] (base-line) using physical objects from a healthy control group of twenty-four volunteers selected on the basis of optimum hand function using these criteria: age (range, 18-25y), and no adverse hand trauma, neurologic condition, or disabling effects of the upper limb. All the users were able to finish all tasks well within the time limit. Based on the timing results, we establish that our system supports rich manipulation capabilities to fulfil SHAP within reasonable time frame. On average a user took 3 times more time than the base-line. We attribute this slowdown to a number of factors - primarily to the lack of haptic feedback. All results reported in this chapter are without any user feedback (except visual feedback).

### *9.5.3 Hand design evaluation*

SHAP provides a quantifiable assessment of hand capabilities. These assessment can further be treated as a measure of hand's dexterity under the engaged controller. By changing the hand designs without changing the controller (i.e. the user) we obtain a novel mechanism of relative evaluation of hand designs. Given the plasticity and expertise of the human brain in controlling human hands, we assume that we have engaged a rather smart controller for the hand design under evaluation. Plasticity of the assessment pose a significant advantage. Random perturbations or physical artifacts (like wind, low friction coefficient) can be easily added to increase the severity of the task. Alternatively, evaluation intensity can be reduced by providing help clues (like grasp metrics etc) to the users. SHAP can not only help with design evaluation but also provide assessments like – exceptional power and spherical grip, or impaired ability to perform finer manipulations with tip and tripod grasps. In addition, a 'SHAP Index of Function score' can also be generated, which is one number that provides an overall assessment of hand function under the engaged controller.

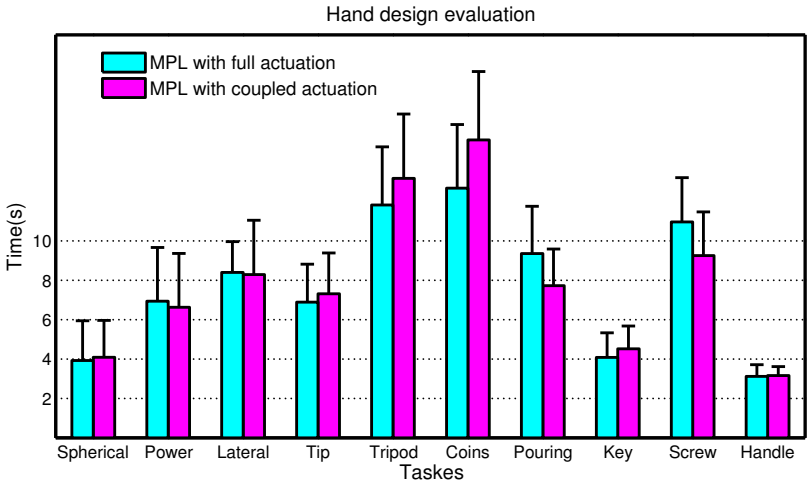



Figure 9.12: Design evaluation of MPL hand - with full and with coupled actuation

A fully actuated system is maximally capable of exploiting the dexterity of a hardware. However, it can be hard to manufacture due to cost and design constraints, specially in case

of anthropomorphic hand designs due to high dof density in the workspace. Couplings and transmissions help solve some of these constraints at the cost of reduced dexterity. Couplings morph the workspace, which can either help the hardware in achieving some tasks better by creating favourable subspaces or can hinder some functionality by constraining dof of the system. One can use above mentioned pipeline to evaluate and iterate over the coupling options that least affects the functionally of the overall system. The pipeline facilitates fast evaluation before manufacturing, resulting in huge cost and time advantage.

Table 9.12 presents the design evaluation result between two MPL hands – one with the coupled actuation as described in [66] and the other with full actuation. Users were asked to perform SHAP tests with both the hands without information about which one they were using. While MPL with coupling performs at par (or slightly better) in most SHAP tests, increase in timings for tip, tripod, coins and key highlight reduction in ability to perform fine manipulation, essential for these SHAP-tests.

## *9.6 User's feedback*

We received strong positive feedback from the participants of the user studies (section 9.5). Users felt that the VR was intuitive and capable. They were often found trying out complicated maneuvers, beyond what was asked for the user study. One common concern was the lack of haptic feedback from the virtual reality. Users also expressed concern about their sudden inability to perform manipulation in few test cases. These cases involved testing design parameters for hand design iteration studies (section 9.5.3), that effectively rendered the hand model incapable of certain tasks. We also received positive feedback from out initial deployment to the teams of Darpa's HAPTIX program. Few feedback snippets from participating teams are appended below

- My overall experience with Mujoco has been very positive. The software is easy to install, runs out of the box, and the API interface is simple to use. We are using the Mujoco HAPTIX VRE for the development and testing of a bi-directional prosthetic

hand interface. In an upcoming study, a volunteer will control the VRE hand via signals decoded from neural and muscular implants. Additionally, the volunteer will receive haptic feedback through neural implants when the virtual hand touches objects in the virtual environment.

- I am currently using Mujoco as part of the HAPTIX project. In my work, I am designing closed-loop control of a virtual hand through a neural interface. By continually pulling in sensor values via the MATLAB API, I am able to calculate neural stimulation parameters for our neural interface device. Overall, my experience with Mujoco has been good. The Mujoco HAPTIX environment is pretty intuitive and easy to use.

## *9.7 Future Works*

Although figure 9.11 provides convincing proof that MuJoCo HAPTIX is capable of supporting manipulation, it also highlights a large divide between demonstrations performed in reality vs demonstrations done in the virtual environment. A number of factors can be attributed to explain the divide - including but not limited to the lack of haptic feedback, off-calibration of CyberGlove etc. Exploration of haptic feedback techniques can be one promising future direction. Other non-standard feedback modalities like visual cues, audio cues etc to convey information about contact events, rolling, sliding, stick-slip etc can be other possible direction in which this work can be extended.

Medical robotics, rehabilitation, and prosthesis can further use MuJoCo HAPTIX for training and testing purposes. Analysis of the resulting dataset can help understand the effectiveness of the mechanism and its effects on the users. MuJoCo HAPTIX is fast with negligible latency. Access to physical entities can be used for real-time feedback. To further this endeavor, MuJoCo HAPTIX is being used for Darpa's ongoing Hand Proprioception & Touch Interfaces (HAPTIX) program for investigation of minimum sensory-motor feedback required in order to close the loop between a prosthesis and its human counterpart.

### *9.8 Next Steps*

Diverse physically consistent dataset (concealing rich information about phenomena like contacts, friction, stiction, deformation, sliding, rolling etc) can be leveraged by data-driven machine learning techniques and inverse optimal control techniques for understanding movements and synthesizing behaviors. Moving forward we will use Mujoco Haptix framework to collect expert demonstrations. Expert demonstrations are extremely valuable for machine learning techniques in order to tackle sample complexity. These expert demonstrations can be leveraged for techniques like guided exploration, mixture of experts, nearest neighbors etc. In the following chapter we will leverage expert demonstrations to scale our learning from experience techniques (chapter 8) to more complicated tasks.

Chapter 10

# LEARNING DEXTEROUS HAND MANIPULATION BEHAVIOR VIA IMITATION USING LEARNED LOCAL MODELS

In the previous chapter we explored learning-based approaches for feedback control of a dexterous five-finger hand performing non-prehensile manipulation using learned local controllers that were able to perform the task starting at a predefined initial state. These controllers were constructed using trajectory optimization with respect to locally-linear time-varying models learned directly from sensor data. In this chapter we will explore learning via imitation and experience in order to scale to more complicated tasks. For learning more complex manipulation skills, we explore the use of human demonstrations to initialize the controllers. Complex tasks with delayed rewards, such as grasping and in-hand repositioning of a heavy object, are difficult to learn from scratch. We show that a teleoperation system can be used to provide example demonstrations from a human operator using a glove-based interface, and that these demonstrations can be used to initialize learning for complex skills. We demonstrate that such controllers can perform the task robustly, both in simulation and on the physical platform. Results are summarized in the supplementary video: https://youtu.be/E0wmO6deqjo

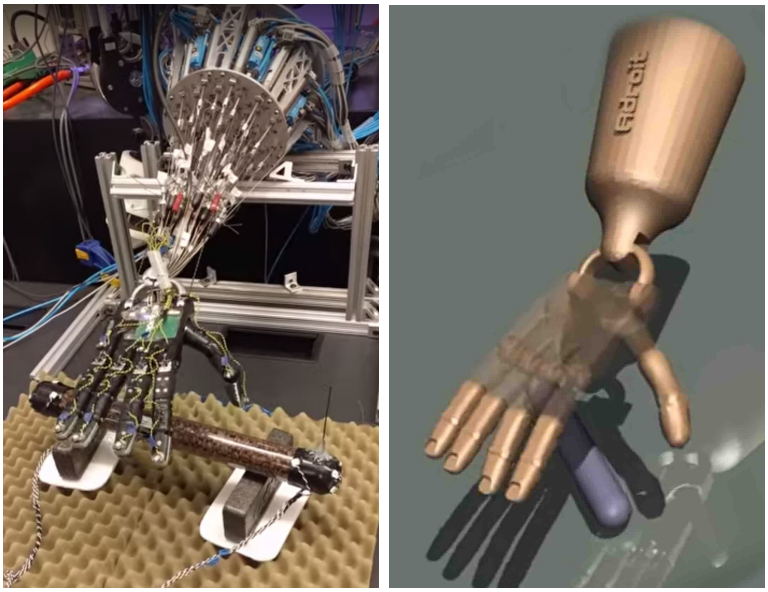
## *10.1 Learning Policies from Experience and Imitation*

Section 8.2 highlighted the strengths of our reinforcement learning algorithm, in synthesizing the details of dexterous manipulation strategies while still preserving sample efficiency. However, as our algorithm makes progress by optimizing a quadratic approximation of the cost over a local approximation of the dynamics, it can become stuck in local minima if

the approximation of the learned dynamics isn't sufficiently accurate, or when the cost is not convex. In principle, arbitrary precision can be achieved by increasing the number of Gaussian kernels used for the dynamics prior, and by increasing the number of trajectory samples $N$ used for the fitting the dynamics. In practice, collecting an arbitrary number of samples might not be feasible due to time and computational limitations, especially when physical robots are involved. Furthermore, many useful task goals are non-convex, and while the method can optimize non-convex cost functions, like all local optimization methods, it does not necessarily converge to a global optimum. Random exploration can help mitigate some of these issues. To encourage exploration, we add random noise while collecting trajectory samples(step 3 of the algorithm 2). For tasks where the reward is delayed and there are multiple local minima, randomly exploring around and hoping to get lucky takes a big toll on sample efficiency. A well know strategy to overcome some of these challenges is to imitate an expert, in order to effectively steer the learned controller towards an effective solution. However, simply following an expert-provided behavior does not necessarily produce behavior that is robust to perturbations. In this section, we describe how we can combine learning from expert teleoperation with trajectory-centric reinforcement learning to acquire more complex manipulation skills that overcome local optima by following expert demonstrations while still retaining the robustness benefits of learning from experience.

### *10.2 Task Details*

The task in this second set of experiments consists of picking up an elongated tube from the table. This task is challenging because the cost depends on the final configuration of the tube, which depends in a discontinuous manner on the positions of the fingers. Furthermore, grasping the tube from various initial poses involves the use of multiple different strategies. Note that the hand is rigidly mounted, so the picking must be done entirely using motion of the wrist and fingers. The shape of the tube further complicates the task. Naïve strategies like aligning the hand to the object's principal axis and force-closure will not perform well, since only the wrist is available for positioning the hand. Instead, the hand must use complex



(a) Hardware platform (b) Simulation platform

Figure 10.1: Initial pose for the pickup task

finger gaits to maneuver the object into position. The significant weight of the tube (in comparison to the lifting capability of the hand) constantly tilts the object out of the grasp. Successful strategies need to discover ways to use the thumb and fingers to act as 'support' and 'pivot' for each other, in order to reposition and reorient the object leading to a successful grasp and pick up. Failure to establish either the support or pivot results in the tube flying out of the workspace due to a net unbalanced force on the object.

The hand is mounted approximately thirty degrees to the horizontal. The task starts with the hand in zero position (Figure 10.1). The goal is to lift the object from a known initial condition (the object is being tracked using the PhaseSpace motion capture system). The tube is considered lifted if it is in complete control of the hand (i.e. not falling out of the hand or resting on the table) and all points on the object are above the ground by a certain height. Note that for the resemblance between the hardware and the simulated platform, the contact between the fingers and the ground plane is disabled for the simulated platform

in order to allow the fingers to curl from below the object.

### *10.3 Expert Demonstrations*

We use Mujoco Hapix (Chapter 9) to capture expert demonstrations. The expert first goes through a regression based calibration process, that maps the Cyberglove sensors to the ADROIT joint space. To enable hand manipulation, we map the joint angle $q^c$ reported by the Cyberglove to actuator commands using the Equation (10.1):

$$\mathbf{u}_t = \mathbf{k} J^{tendon}(\mathbf{q}_t - \mathbf{q}_t^c), \tag{10.1}$$

where $\mathbf{q}_t$ is the current joint configuration of the hand, $J^{tendom}$ is the tendon jacobian that maps the joint space to the tendon space, and $\mathbf{k}$ is the gain vector. $\mathbf{u}_t$ is applied as controls to the pneumatic actuators and we let the simulation evolve by stepping the physics of the world forward in time. Hand-object interactions evolve as the simulation steps forward. The expert is in a tight feedback loop with the simulation via the stereoscopic rendering. As the user interacts with the system by changing its strategy, hand manipulation behaviors emerge. We record the state $\mathbf{x}_t$ and the control trajectory $\mathbf{u}_t$ over time as demonstration trajectory. where $\mathbf{x} = (\mathbf{q}, \mathbf{q}^{pos}, \mathbf{q}^{rot}, \dot{\mathbf{q}}, \dot{\mathbf{q}}^{pos}, \dot{\mathbf{q}}^{rot}, \mathbf{a})$. Here $\mathbf{q}$ denotes the vector of hand joint angles, $\mathbf{q}^{pos}$ the object positions, $\mathbf{q}^{rot}$ the object rotations, $\mathbf{a}$ the vector of cylinder pressures, and $\mathbf{u}$ the vector of valve command signals.

### *10.4 Learning Imitation Policies*

We first attempted to learn the pickup task from scratch using the following cost function:

$$\begin{aligned}\ell(\mathbf{x}_t, \mathbf{u}_t) =& \alpha_1 ||q_t - q^*||^2 + \alpha_2 ||\mathbf{u}_t||^2 + \\ & \alpha_3 ||q_t^{pos} - q^{pos*}||^2 + \alpha_4 ||q_t^{rot} - q^{rot*x}||^2,\end{aligned}$$

where $q^{pos*}$ and $q^{rot*x}$ denote the goal pose of the object, and $q^*$ is a target joint angle configuration that corresponds to a grasp. Simple learning by experience fails to come up with a strategy that can pick up the rod even after significant cost parameter tuning. The

observed behavior is a combination of the following: (a) Hand randomly explores for a while and fails to find the object; (b) The fingers eventually find the object and knock it away from the manipulate-able workspace; (c) The fingers keep tapping on the top of the tube without any significant improvement. The difficulty in finding a successful pickup behavior stems from a combination of factors. First, the cost function only produces a meaningful signal toward the end of the episode, once the object is already grasped. Second, almost every strategy that can succeed has to first clear the object and only then dig the fingers into the space below the tube, with fingers on both sides to restrict the object's motion. Lastly, the high dimensionality of the ADROIT platform means that the space of successful solutions is very narrow, and a huge variety of movements are available that all fail at the task. Note that, for cases where the object isn't aligned well with the palm, additional reorientation maneuvers are required in order to pick up the object, due to the hand's restricted lateral mobility.

Our learning process for these skills combines learning from demonstrations with learning from experience, with the expert demonstration used to bootstrap the learning process. During the learning from experience phase, we use an additional shaping cost, which is a weak cost term that prevents the learning process from deviating too far away from the expert demonstration. The assumption here is that the expert demonstration is already quite good and we don't need to deviate too far to improve the solution. The strength of the shaping cost controls the amount of deviation from the expert demonstration.

For bootstrapping the learning using expert demonstrations, instead of using a random policy in step 3 of the Algorithm 2 at the initial iteration, small random noise is injected into the control trajectory of the expert demonstration to collect trajectory samples $\{\tau_i\}$. The running cost we use is

$$\ell(\mathbf{x}_t, \mathbf{u}_t) = ||\mathbf{q}_t - \hat{\mathbf{q}}_t||^2 + 0.1||\mathbf{u}_t||^2 + 50||q_t^{posZ} - 0.12||^2,$$

where $\hat{\mathbf{q}}_t$ is the hand configuration of the expert at time $t$ and $q_t^{posZ}$ is the vertical height of the object at time $t$. The first term is the shaping cost that restricts the learning from

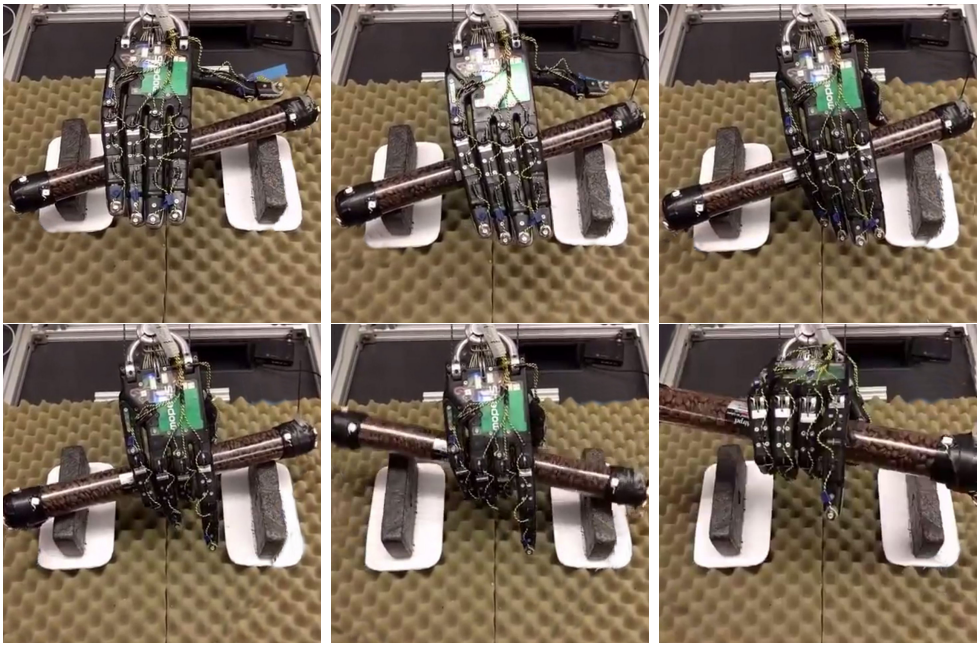


Figure 10.2: Pick up strategy learned using learning via imitation. The movement starts with aligning the palm with the object, then curling the fingers under the object followed by an object-reorientation movement that further aligns the object with the palm. The final movement engages the thumb is squeeze hard again the palm before lifting the wrist upward.

deviating too far from the demonstration, the second term is the control cost and the final term encourages picking up the object to a height of 12 cm above the ground. The final cost is the same as the running cost. Figure 10.2 presents a representative pickup behavior achieved using this method.

### *10.5 Next Steps*

While the learned local model based techniques developed here and in chapter 8 is quite successful in generating contact rich dexterous manipulation, they are local in nature i.e. each local controller is associated with a set initial condition. Any change in the initial state of the system requires learning from scratch again. In the following chapter, we explore

techniques to generalize the learned local controllers into a single global controller.

Chapter 11

# GENERALIZING DEXTEROUS HAND MANIPULATION BEHAVIORS

In chapter 8 and 10 we developed learning-based approaches for feedback control of a dexterous five-finger hand performing non-prehensile manipulation for a limited range of initial conditions around the trained starting state. In this chapter we outlines two interpolation methods for generalizing to a wider range of initial conditions: deep learning, and nearest neighbors. We find that nearest neighbors achieve higher performance. Nevertheless, the neural network has its advantages: it uses only tactile and proprioceptive feedback but no visual feedback about the object (i.e. it performs the task blind) and learns a time-invariant policy. In contrast, the nearest neighbors method switches between time-varying local controllers based on the proximity of initial object states sensed via motion capture. While both generalization methods leave room for improvement, we observe that (i) local trajectory-based controllers for complex non-prehensile manipulation tasks can be constructed from surprisingly small amounts of training data, and (ii) collections of such controllers can be interpolated to form more global controllers. Results are summarized in the supplementary video: https://youtu.be/E0wmO6deqjo

## *11.1 Introduction*

To move beyond local policies that can succeed from only a narrow range of initial states, we explore generalization through two distinct approaches. In the approach method, we train a collection of local policies, each initialized with a different initial demonstration, and then use a nearest neighbor query to select the local policy for which the initial conditions best match the current pose of the manipulated object. In the second method, we use a deep

neural network to learn to mimic all of the local policies, thus removing the need for nearest neighbor queries. Our experimental results show that the nearest neighbor approach achieves the best success rate, but at the cost of requiring the variables for the nearest neighbor queries (the pose of the object) to be provided by hand. We also show that the deep neural network can learn a time-invariant policy for performing the task without requiring knowledge of the object pose at all, using only onboard sensing on the five-finger hand to perform the task.

## *11.2 Policy Generalization*

Section 8.4 and Section 10.1 explores the capabilities of trajectory-centric reinforcement learning to learn robust "local" policies for dexterous manipulation of freely-moving objects. However, the resulting local policies succeed from specific initial states, and are not designed to handle variation in the initial conditions, such as different initial placement of the manipulated object. In this section, we will discuss how we can use the local policies to learn a single global policy that generalizes effectively across different initial conditions.

Generalizable global policies can typically be learned using reinforcement learning with more expressive function approximators. Deep neural networks represent a particularly general and expressive class of function approximators, and have been recently used to learn skills that range from playing Atari games [65] to vision-based robotic manipulation [54]. In this section, we evaluate how well deep neural networks can capture generalizable policies for the pickup skill discussed in the previous section, where variation in the task corresponds to changes in the initial pose of the rod. Our goal is to explore the generalization capabilities of deep neural networks along two axes: the ability to handle variability in the initial conditions, and the ability to handle partial observability and limited sensing. To that end, we will evaluate deep neural network policies that perform the task either with full state observation, or with only the onboard sensing available on the ADROIT platform, without external motion capture markers to provide the pose of the rod.

### *11.3 Task Details*

We evaluate generalization on the same pickup task as outlined in Section 10.2. To analyze generalization, we vary the orientation of the rod at the initial state. The goal is to learn a strategy for this task that succeeds for any initial rod orientation. This is particularly challenging since the robot cannot translate or reorient the wrist (since the hand is stationary), and therefore must utilize substantially different grasping strategies for different rod orientations, including the use of auxiliary finger motions to reposition the rod into the desired pose. Besides this challenge, the task also inherits all of the difficulties detailed in Section 10.1, including the high dimensionality of the system and the challenge of delayed rewards. To mitigate the challenge of local optima, we again use expert demonstrations. We collected a set of 10 demonstrations across 180 degrees of variation in the rod orientation. Figure 11.1 shows the 10 initial configurations from which an expert was engaged for providing demonstrations.

### *11.4 Local Policies*

Before evaluating generalization, we first analyze the performance of the individual local policies trained with imitation and learning from experience. For each task execution, we evaluate the success or failure of trial according to the following criteria: a successful picking trial must result in the object being stationary, both extremely of the rod being above the ground by a certain height, and the entire rod aligned with the $x$-axis, so as to ensure a successful grasp into the desired goal position. Note that partially successful executions, where the tube's center of mass is lifted up but one of the end points remains on the ground, are still marked as failures. This allows us to determine not only whether the tube was picked up, but whether it was also positioned in the desired pose – an important condition for any subsequent manipulations that might be applied to it.

In Figure 11.2a, we analyze the performance obtained by following the demonstrated behaviors directly, for variation in the rod angle within the local neighborhood of the rod

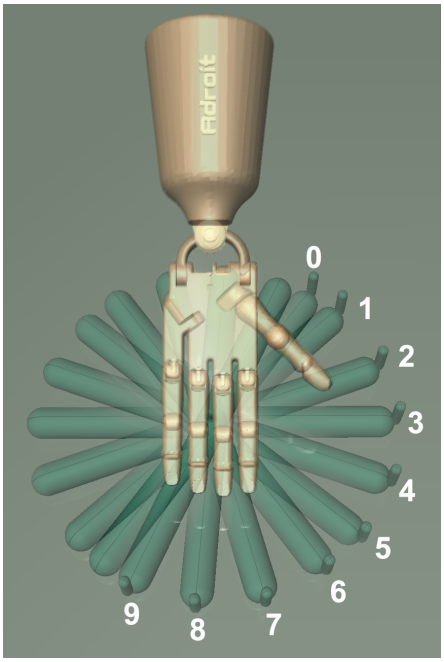
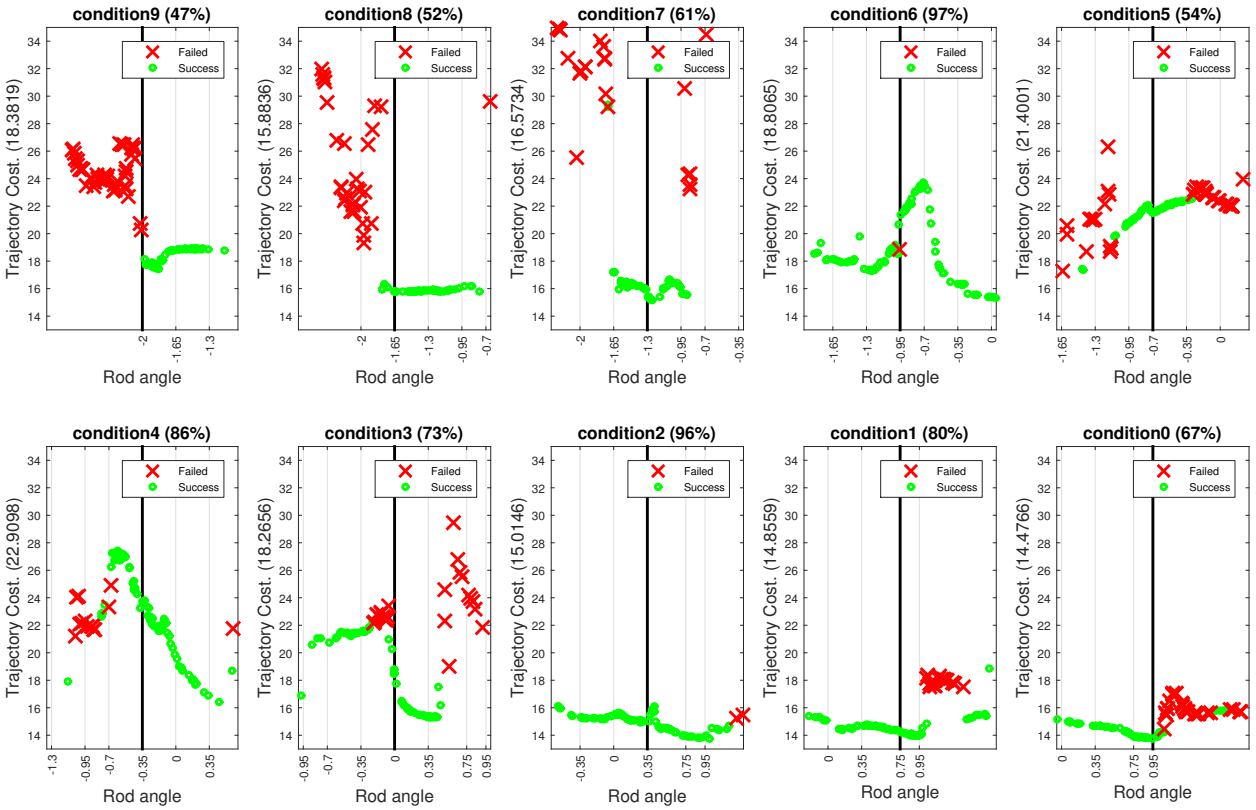



Figure 11.1: The initial tube configuration of the expert demonstration set.

pose for which the demonstration was generated. The numbers correspond to the conditions illustrated in Figure 11.1. The $x$-axis corresponds to variation in the orientation of the rod. Successful trials are marked as green circles, while failures are marked as red crosses. Overall, we observe that most demonstrations are somewhat successful for the particular rod angle for which they were created, but the success rate decreases sharply under variation in the rod pose, with discontinuous boundaries in the success region (particularly for conditions 9, 3, and 0). Some conditions, such as condition 6, are typically successful, but exhibit a brittle strategy that sometimes fails right at the demonstration angle, while others, such as conditions 3 and 1, fail sporadically at various rod poses.

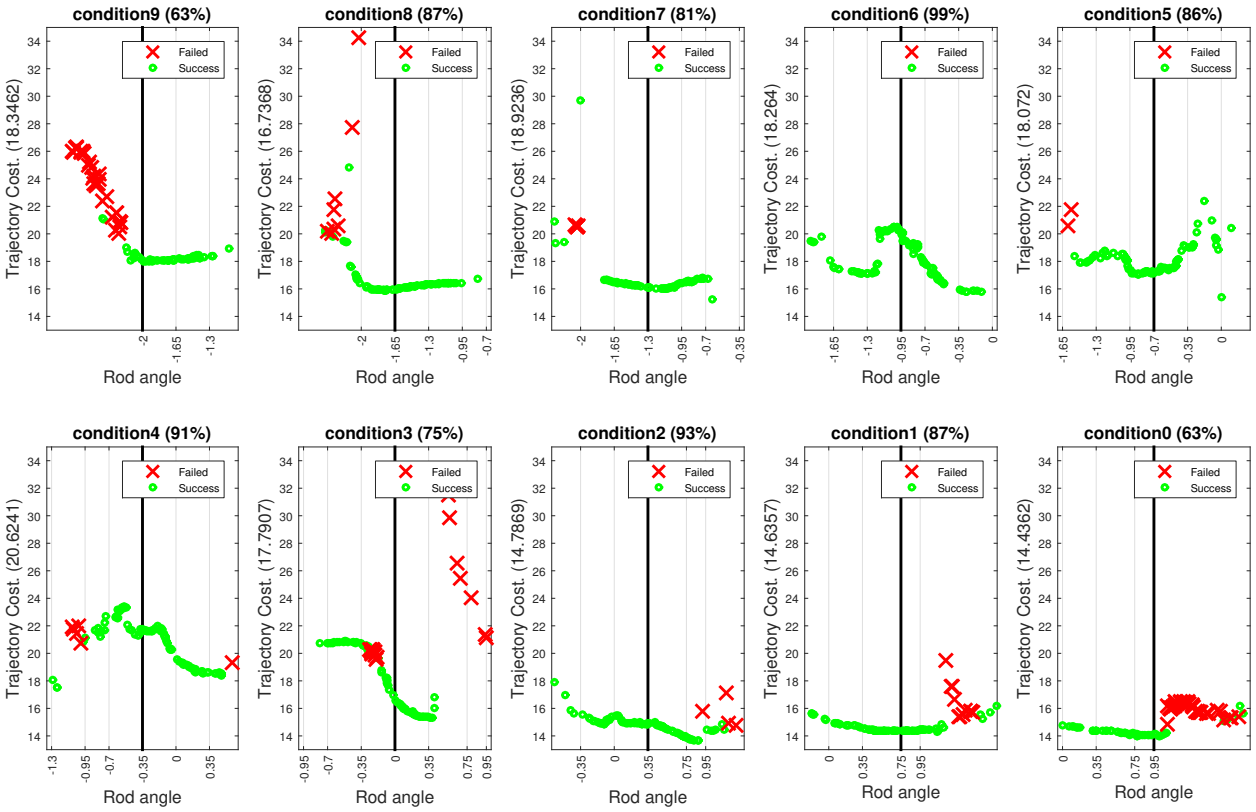
As shown in Figure 11.3a, we can improve the robustness of the local policies corresponding to each condition by further training each of those policies from experience, following the method described in Section 10.1. Each demonstration was subjected to 10 iterations of learning from experience, and the resulting controllers were evaluated using the same proce-




(a) Testing robustness of the expert demonstration in its local neighbourhood.

Figure 11.2: Different plots represent different conditions. Ticks on the X axis marks the different rod angles, corresponding to each condition where expert demonstration was collected. The true angle of the plot is marked with solid black line. Each condition was tested for 100 trials. Successful trials are marked in green and unsuccessful are marked in red. Success percentage are marked in the title and the average cost are provided in the Y label.

dure as in the previous paragraph. The controllers succeed in a wider neighborhood around their default rod pose, with the brittle strategy in condition 6 becoming much more robust, and the region of success for conditions 9, 3, and 0 expanding substantially on both sides. The overall costs and success rates for individual conditions are summarized in Figure 11.4. While we observe that the overall cost remained similar, the success rates for varying rod




(a) Testing robustness of the local policies trained around the demonstrations in its local neighbourhood.

poses increased substantially, particularly for conditions 6, 8, 9, and 10. Figure 11.13 evaluates the effectiveness of the local policies for the entire range of task variation. We observe that the policies are less effective as they move away from the zero point (marked with dark vertical line).

### *11.5 Nearest Neighbor*

We can observe from the results in Figure 11.3a that, after training, each local policy succeeds in a neighborhood that extends to the boundary of the next local policy. This suggests that a relatively simple nearest-neighbor technique could in principle allow for a nonparametric

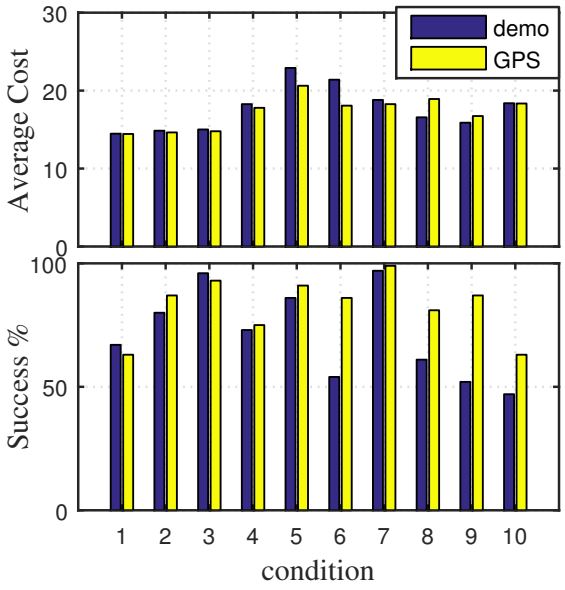



Figure 11.4: Performance comparison between the expert demonstrations and the local policies trained around the expert demonstrations in the local neighbourhood of the task (i.e. the rod angle with the vertical)

strategy to expand the success region to the entire range of rod orientations. In Figure 11.5, we evaluate the performance of this nearest neighbor strategy, which simply deploys the local policy trained for the rod orientation that is closest to the orientation observed in the initial state, in terms of Euclidean distance. Successful trials are marked in green and failures in red, and the overall success rate is 90.8%. Although the nearest neighbor strategy is successful in this case, it requires retaining the full set of local policies and manual specification of the variables on which the nearest neighbor queries should be performed (in this case, the rod pose). Furthermore, the nearest neighbor strategy requires us to know the rod pose, which in a physical experiment would involve the use of external motion capture markers. Can we instead learn a generalizable policy for grasping the tube that does not use any external sensing, only the onboard sensing available on the hand? In the next section, we will discuss how deep neural network function approximators can leverage the local policies

to learn generalizable strategies.

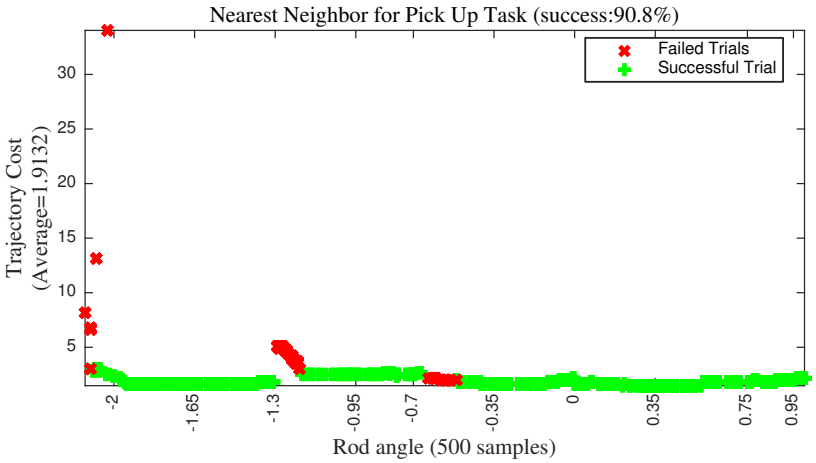



Figure 11.5: Performance of the nearest neighbor policy with full state information. Test trials are collected from random initial conditions of the object, with the local policy corresponding to the nearest rod orientation used in each trial.

## *11.6 Generalization with Deep Neural Networks*

As discussed previously, although the nearest neighbor strategy has an extremely high success rate for the pick-up task being considered, it suffers from several limitations. First, nearest neighbor methods suffer from the curse of dimensionality, which limits their applicability to high-dimensional spaces. Indeed, the nearest neighbor technique described in the preceding section requires us to manually specify that the nearest neighbor queries should be performed with respect to the pose of the rod. This information may not be readily available in general robotic manipulation tasks, where the robot must choose the strategy based on high-dimensional raw sensory information, such as camera images or inputs from tactile sensors. Second, the nearest neighbor method described above still corresponds to a time-varying control strategy, which can be limiting in cases where the robot might begin the task from arbitrary initial states. Finally, it requires storing all of the individual trajectory-centric

controllers.

In this section, we evaluate whether a deep neural network representation of the rod grasping manipulation skill can be learned with comparable generalization capability and limited sensing. Deep neural networks provide a powerful and flexible class of function approximators that can be readily combined with high-dimensional inputs and complex sensing. Furthermore, the use of deep neural networks as the final policy representation does not require us to manually select a small set of state variables for nearest neighbor queries, making them a more general choice that is likely to extend to a wide variety of dexterous manipulation tasks.

To evaluate the use of deep neural networks for learning generalizable dexterous manipulation strategies, we used the collection of local policies trained from initial demonstrations to produce training data for neural network training. Each of the local policies discussed in Section 11.4 was used to generate 50 sample trajectories by executing each policy from its corresponding initial state. These samples were then used to train a deep neural network with 6 fully connected layers and 150 recitified linear (ReLU) hidden units in each layer, as shown in Figure 11.6. The network was trained either with the full state information provided to the local policies (which includes the pose and velocity of the rod), or with partial information reflecting onboard sensing, with knowledge about the rod pose excluded from the inputs to the neural network.

The results of the neural network policy with full state observations are shown in Figure 11.7. This result indicates that the neural network policy trained from the local policies is generally not as successful as the nearest neighbor strategy in the fully observed case. This is not particularly surprising: the nearest neighbor strategy already uses a set of very successful local policies that can succeed up to one angle increment, and thus form overlapping regions of effectiveness. However, the neural network must distill the distinct strategies of different local policies into a single coherent function, and therefore generally performs worse in this case. Figure 11.8 and Figure 11.9 show the performance of a large and small neural network trained without the rod pose provided as input, instead using inputs from

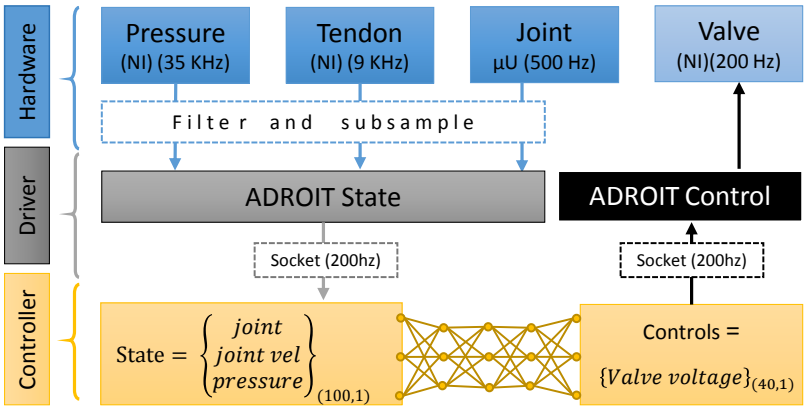



Figure 11.6: Over architecture of the system using the network as controller.

the hand's onboard tactile sensors in the fingertips. When the network is trained without observations of the rod pose, it achieves a success rate of 74%, nearly as high as the fully observed condition, which is substantially higher than the best local policies, as shown in Figure 11.13.[1] These results also show that the larger network is essential for properly handling this condition. Interestingly, when the neural network is not provided with the tactile sensors either, as shown in Figure 11.10, it achieves nearly the same performance, indicating that the neural network is able to make use of proprioceptive sensing to determine which strategy to use. This is also not entirely surprising, since collisions and contacts result in motion of the fingers that can be detected from proprioception alone. These results indicate that the neural network policy that is only allowed to use onboard sensing learns a robust feedback behavior that can adjust the grasping strategy based on proprioceptive feedback, something that is not possible with the nearest neighbor strategy, which must choose the local policy to deploy at the beginning of the episode (before any proprioceptive sensing can take place). This suggests one of the benefits of deep neural networks for dexterous control.

[1]It is worth noting here that the hand is reset to the starting state of the trajectory controller with the closest angle for each rod pose. While these states are very similar, they may nonetheless provide additional cues to the network. Our ongoing experiments, which we will include in the final version, will address randomized initial poses to control for this potentially confounding factor.

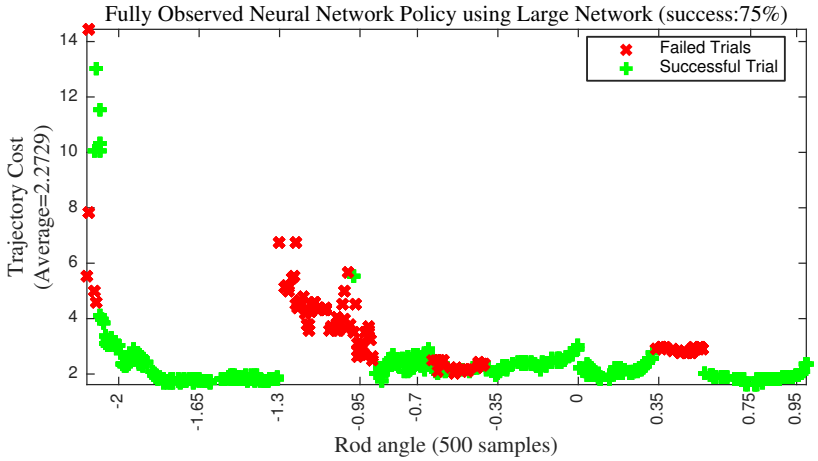



Figure 11.7: Neural network policy performance with fully observed state, at random initial object poses. The network consisted of 6 layers with 150 hidden units each, and was trained on the local policies from conditions 1, 2, 3, 4, 5, 6, and 8.

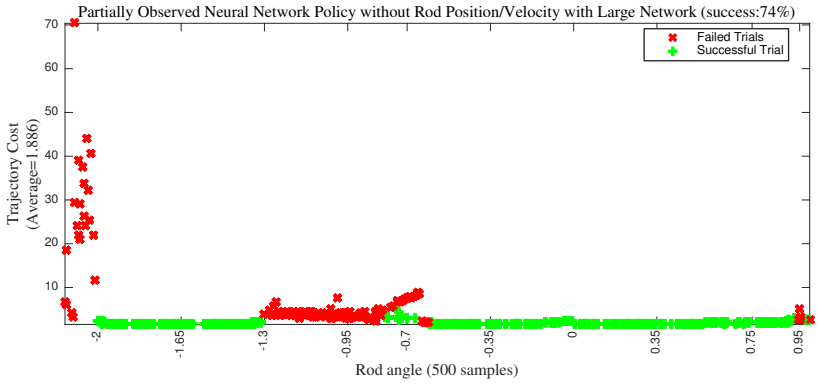



Figure 11.8: Performance of a large neural network (6 layers, 150 units) with partial observations and touch sensors. The rod position is not provided to the network, but instead the network can use inputs from the tactile sensors on the fingertips. Note that overall performance exceeds the best single local policy.

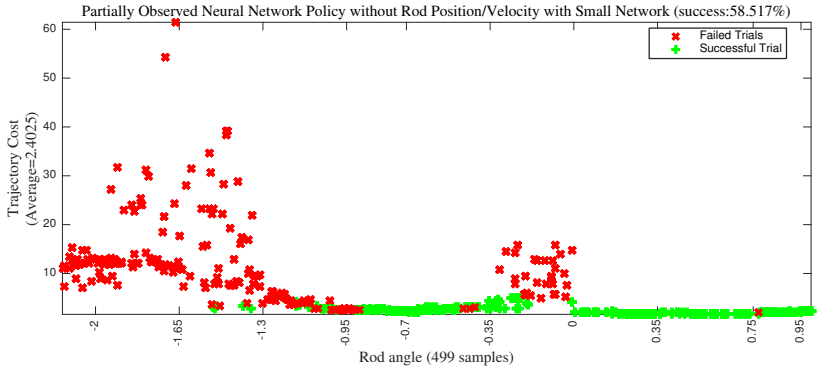



Figure 11.9: Performance of a small neural network (4 layers, 80 units) with partial observations and touch sensors. The rod position is not provided to the network, but instead the network can use inputs from the tactile sensors on the fingertips. Note that overall performance of the smaller network is substantially degraded.

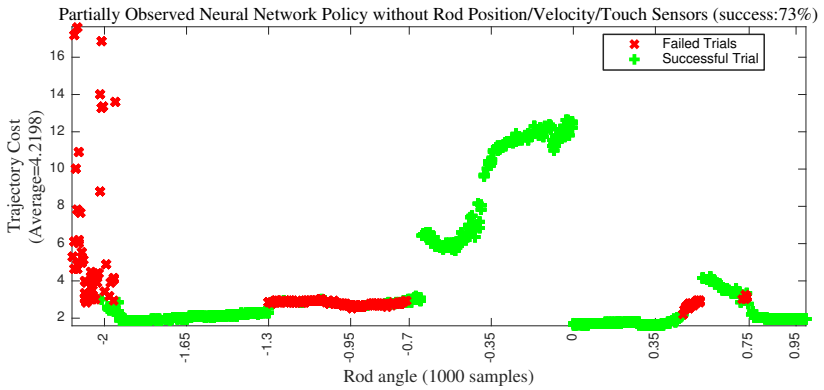



Figure 11.10: Performance of a large neural network (6 layers, 150 units) with partial observations and without touch sensors. The rod position is not provided to the network. Note that the large network performs well even without the touch sensors, indicating that the network is able to use proprioceptive inputs to determine the strategy.

### *11.7 Results on ADROIT Hardware Platform*

In this section, we present our generalization results for the pick up task on the ADROIT hardware platform. The details of the task are the same as mentioned above. However, we

consider generalization both in the position and orientation of the object. The orientation generalization is in a smaller neighborhood than the simulated experiments above, and we train a 6 layer fully connected neural network with 120 rectified linear (ReLU) units at each layer. The training samples are collected by sampling the local controllers learned around the 4 demonstrations provided by the expert user. The expert demonstrations were collected at the initial condition as shown in Figure 11.11. Local controllers were trained using 10 iterations of learning via imitation strategy as mentioned in Section 10.1. The training set for the network consists of 20 samples for each condition (i.e. 80 samples in total). In figure Figure 11.12, we cross-validate each policy for different condition, including random conditions. We found that local controllers are partially successful in a neighborhood wider than just their own. There is no local controller that works well for all the conditions. Overall the neural network policy performs at par with the local controllers on the respective conditions. The neural network policy, however, generalizes better than the individual local policies as conveyed by its higher success in picking object from random initial configurations.

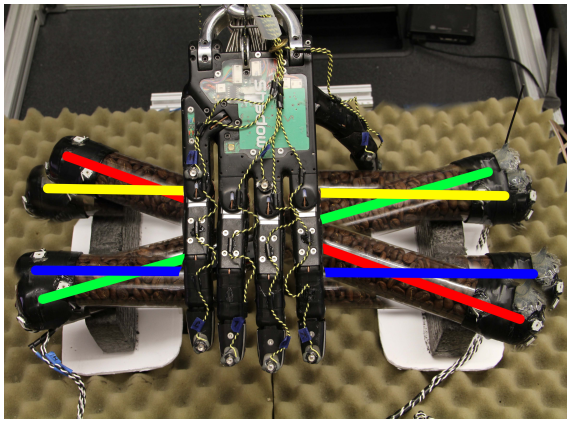


Figure 11.11: Different conditions used for expert demonstrations

| | Initial condition | | | | |
|---|---|---|---|---|---|
| | A | B | C | D | Random |
| Policy-0 | 100% | 80% | 0% | 60% | 40% |
| Policy-1 | 60% | 100% | 40% | 0% | 60% |
| Policy-2 | 60% | 100% | 100% | 0% | 50% |
| Policy-3 | 100% | 40% | 0% | 100% | 60% |
| Policy-NN | 80% | 100% | 100% | 100% | 90% |
| | | | | | |
| | | 5 trials | 10 trials | | |

Figure 11.12: Cross validation of different policies under different conditions on ADROIT Hardware platform

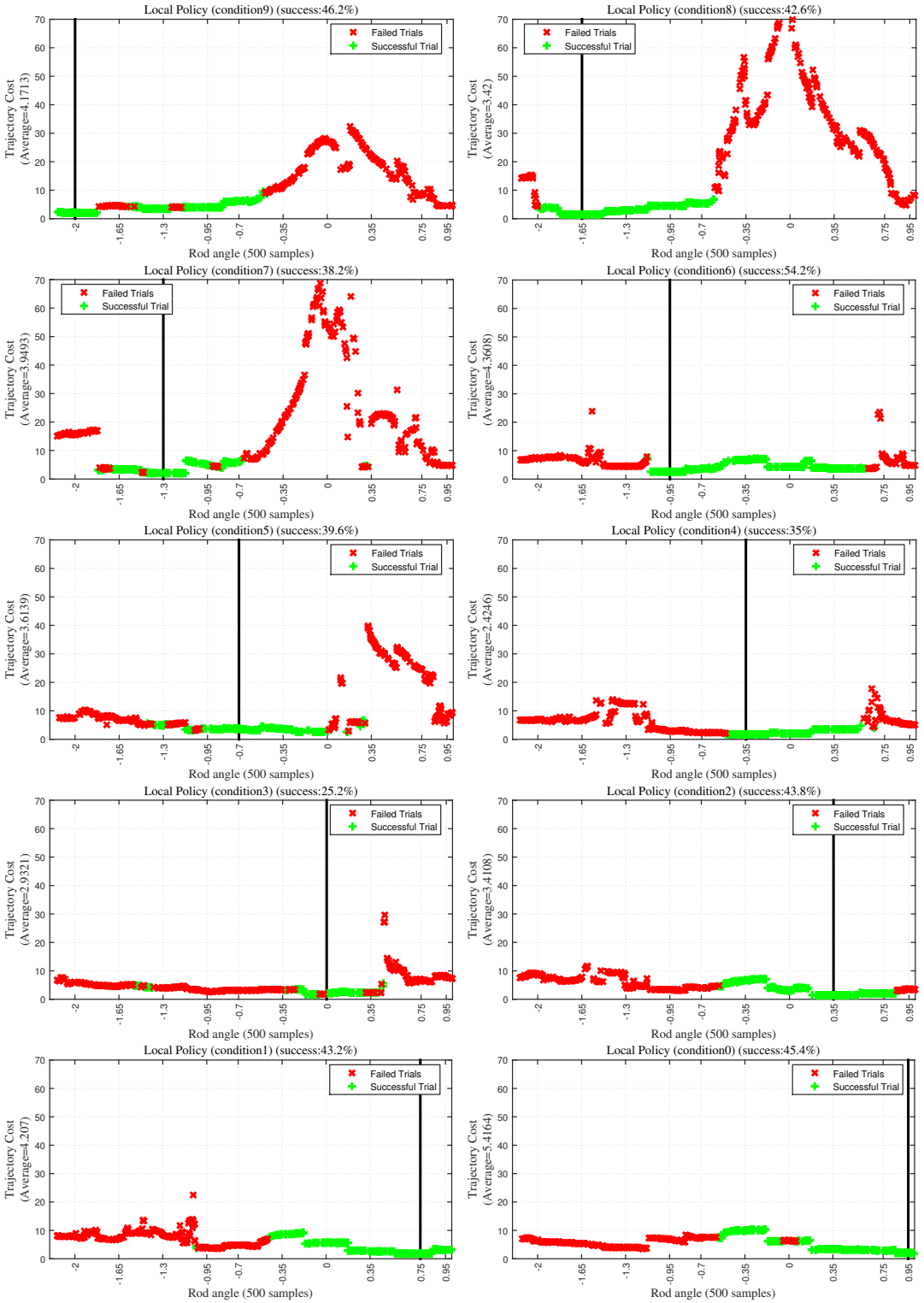



Figure 11.13: Performance of the local policies across the task variations

# Chapter 12

# SUMMARY & FUTURE DIRECTIONS

While realizing true dexterous manipulation approaching human hand capabilities is a monumental task which will take a while to realize, this thesis makes significant progress towards pushing the state of the art. Dexterous manipulation is investigated at the intersection of hardware and computation. Challenges, both from the hardware's (Chapter 3) as well as control's stand point (Chapter 5), have been addressed in unison to develop *ADROIT* manipulation platform.

The *ADROIT* manipulation platform is fast, dexterous and extremely compliant. Synthesizing dexterous manipulation behaviours for such high dimensional system is quite challenging – (a) manipulation evolves in a compact high dimensional space full of constraints and discontinuities which makes manipulation policies extremely difficult to solve for (b) the pneumatic actuation, while quite desirable, makes control quite challenging due to the low bandwidth the 3rd order pneumatic dynamics. The inherent complexity of the problems requires solution strategy that are flexible and reactive to perturbations and failures.

Owing to the reactivity and generalization capability of the model based optimal control techniques, chapter 6 developed a trajectory optimization under model predictive control based "behavior synthesis" framework that is capable of synthesizing contact rich dexterous manipulation in real time. The framework produced extremely agile manipulation behaviors from scratch but failed to scale to physical hardware due to its reliance on the accuracy of the models. In chapter 8 we addressed the accuracy requirements of the models by building "Learning via Experience" framework. The framework constructs local controllers using trajectory optimization with respect to time varying linear models that are learnt directly on the sensors reading, thus making the framework scale to the physical hardware. To scale

to further challenging tasks that require repositioning of an object in the hand and handling delayed rewards, the optimizer were empowered with human demonstrations (Chapter 10) collected via tele-operation in a virtual environment (Chapter 9) leading to the development of "Learning from Experience and Imitation" framework. Aside from the high-level objective encoded in the cost function and the demonstrations, the framework does not use domain knowledge about the task or the hardware.

While generation of isolate movements such as reaching a target pose, as well as dynamic manipulation behaviors that involve repositioning a freely-moving cylindrical object have been achieved in this thesis, composing these movements for solving challenging tasks is a natural continuation and a promising future direction. Generalization capabilities are also evaluated using deep neural network controllers and nearest neighbour switching controllers. While both methods were able to generalize to some extent, the performance of the nearest neighbour method was higher. The two methods have relative strengths and weaknesses and can be improved and perhaps combined in future work, as described below.

The neural network controller was given access to proprioceptive and tactile sensory input, but not to vision input about the object state. Thus it was solving a harder problem, with the caveat that the initial pose of the hand was correlated with the initial pose of the object, and we need to investigate in more detail to what extent the network can indeed perform the task blindly. Blind manipulation is of course not a requirement, given that many vision sensors now exist and real-time state estimators [89] can provide object states. It will be interesting to test the network performance with more complete sensory input.

The nearest neighbour controller is currently time-varying: the switch happens at the beginning of the movement based on the initial state of the object, and then we use the selected local feedback controller for the rest of the movement. Another approach would be to store the dataset of states, controls and feedback gains along the local trajectories, and use nearest neighbour queries at each point in time. This requires more storage and processing than evaluating a neural network, but should not be a challenge for modern computing hardware. Furthermore, it may be possible to store a reduced set of controllers

as opposed to storing the states along all trajectories. For example, Simbicon [121] uses a small collection of time-invariant feedback controllers together with a suitable switching policy, and performs bipedal locomotion remarkably well. The Boston Dynamics approach to control, while not described in detail, also appears to be related.

It may be possible to combine the benefits of the two generalization methods considered here. While deep learning is currently popular, there is no reason to limit ourselves to generic networks. Instead we could consider a mixture-of-experts architecture [40], where the gating network corresponds to the switching mechanism in our current nearest neighbor approach, while the expert networks correspond to our local trajectory controllers. We would still want to leverage the power of trajectory optimization in training the experts, and perhaps use the current switching mechanism to pre-train the gating network.

Another direction for future work is to expand the set of tasks under consideration. While the present tasks are quite complex, they have been selected for their intrinsic stability. For example, consider our object-spinning task. If we were to attempt an identical task but with the palm facing down, our present approach would not work. This is because the object would drop before we have had time to interact with it and learn anything. In general, data-driven approaches to robotics require either an intrinsically stable task (which may be rare in practice), or a pre-existing controller that is able to collect relevant data without causing immediate failure. In this particular case we could perhaps use tele-operation to obtain such a controller, but a more general and automated solution is needed.

As with every other instance of learning from limited and noisy data, the best results are likely to be obtained when data is combined with suitable priors. In the case of robotics, physics provide a strong prior that can rule out the large majority of candidate models and control policies, thereby improving generalization from limited data on the physical system. Existing physics simulators can simulate complex systems such as the one studied here much faster than real-time, and can be run in parallel. This functionality can be used for model-predictive control, aided by a neural network representing a controller and/or a value function [125]. The model itself could be a hybrid between a physics-based model with

a small number of parameters learned from data, and a neural network with a larger number of parameters used to fit the residuals that the physics-based model could not explain.

# BIBLIOGRAPHY

[1] Grasp quality measures. http://personalrobotics.ri.cmu.edu/courses/papers/SuarezEtal06.pdf.

[2] Simulation and control of biped walking robots. http://mediatum.ub.tum.de/doc/997204/997204.pdf.

[3] P. Abbeel, A. Coates, M. Quigley, and A. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[4] Airpel. http://www.airpot.com/html/airpels.html.

[5] Airpel AntiStiction. http://www.airpot.com/html/anti_stiction.html.

[6] John R Amend Jr, Eric Brown, Nicholas Rodenberg, Heinrich M Jaeger, and Hod Lipson. A positive pressure universal gripper based on the jamming of granular material. *Robotics, IEEE Transactions on*, 28(2):341–350, 2012.

[7] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[8] C. Atkeson and J. Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[9] J. A. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[10] E.J. Barth, Jianlong Zhang Jianlong Zhang, and M. Goldfarb. Sliding mode approach to PWM-controlled pneumatic systems. *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, 3:2362–2367 vol.3, 2002.

[11] Peter beater. Pneumatic Drives.

[12] R. Bellman and R. Kalaba. A mathematical theory of adaptive control processes. *Proceedings of the National Academy of Sciences*, 8(8):1288–1290, 1959.

[13] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 348–353. IEEE, 2000.

[14] Jeannette Bohg, Aythami Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesisa survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

[15] Boston Dynamics. http://www.bostondynamics.com.

[16] Doug A Bowman and Larry F Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 35–ff. ACM, 1997.

[17] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[18] G. Carducci, N. I. Giannoccaro, a. Messina, and G. Rollo. Identification of viscous friction coefficients for a pneumatic system model using optimization methods. *Mathematics and Computers in Simulation*, 71(4-6):385–394, 2006.

[19] João Falcão Carneiro and Fernando Gomes De Almeida. Reduced-Order Thermodynamic Models for Servo-Pneumatic Actuator Chambers. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 220(4):301–314, 2006.

[20] Visak Chadalavada and Vikash Kumar. Precise calibration of robots with small link lengths using kinematic extensions.

[21] EY Chao, JD Opgrande, and FE Axmear. Three-dimensional force analysis of finger joints in selected isometric hand functions. *Journal of Biomechanics*, 9(6):387–396, 1976.

[22] Cyber Glove Systems. www.cyberglovesystems.com/.

[23] Andrea d'Avella, Philippe Saltiel, and Emilio Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 2003.

[24] Raphael Deimel and Oliver Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *The International Journal of Robotics Research*, page 0278364915592961, 2015.

[25] M. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[26] M. Deisenroth, C. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2011.

[27] A.D. Deshpande, Z. Xu, M.J.V. Weghe, LY Chang, BH Brown, DD Wilkinson, SM Bidic, and Y. Matsuoka. Mechanisms of the anatomically correct testbed (act) hand. *IEEE/ASME Trasactions on Mechatronics*, 2011.

[28] Ashish D. Deshpande, Ravi Balasubramanian, Ralph Lin, Brian Dellon, and Yoky Matsuoka. Understanding variable moment arms for the index finger mcp joints through the ACT hand. In *IEEE BIOROB*, 2008.

[29] Tom Erez, Kendall Lowrey, Yuval Tassa, Vikash Kumar, Svet Kolev, and Emanuel

Todorov. An integrated system for real-time model predictive control of humanoid robots. In *International Conference on Humanoid Robots*, 2013.

[30] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *IEEE International Conference on Robotics and Automation'15*.

[31] Festo Company. www.festo.com.

[32] Google cardboard. www.google.com/get/cardboard/.

[33] N. Gulati and E.J. Barth. Non-linear pressure observer design for pneumatic actuators. *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, c(August 2005):783–788, 2005.

[34] Navneet Gulati and Eric J. Barth. A globally stable, load-independent pressure observer for the servo control of pneumatic actuators. *IEEE/ASME Transactions on Mechatronics*, 14(3):295–306, 2009.

[35] A. Gupta, C. Eppner, S. Levine, and P. Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstration. In *IROS 2016*, 2016.

[36] S. Haidacher, J. Butterfass, M. Fischer, M. Grebenstein, K. Joehl, K. Kunze, M. Nickl, N. Seitz, and G. Hirzinger. DLR hand ii: hard- and software architecture for information processing. *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, 1:684–689 vol.1, Sept. 2003.

[37] Kawasaki Haruhisa, Mouri Tetsuya, and Ueki Satoshi. *Virtual Robot Teaching for Humanoid Both-Hands Robots Using Multi-Fingered Haptic Interface, The Thousand Faces of Virtual Reality.*

[38] S. Jacobsen, E. Iversen, D. Knutti, R. Johnson, and K. Biggers. Design of the

utah/m.i.t. dextrous hand. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1520 – 1532, apr 1986.

[39] Chestnutt Joel, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *IEEE International Conference on Robotics and Automation*, 2005.

[40] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[41] Sing Bing Kang and Katsushi Ikeuchi. Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps. *Robotics and Automation, IEEE Transactions on*, 13(1):81–95, 1997.

[42] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotic Research*, 32(11):1238–1274, 2013.

[43] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Robotics: Science and Systems (RSS)*, 2010.

[44] C. V. Visak Kumar and Vikash Kumar. Pneumatic modelling for adroit manipulation platform. Manuscript under review, Humanoids'16.

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

[46] Vikash Kumar, Yuval Tassa, Tom Erez, and Emo Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

[47] Vikash Kumar, Yuval Tassa, Tom Erez, and Emo Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6808–6815. IEEE, 2014.

[48] Vikash Kumar, Zhe Xu, and Emanuel Todorov. Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands. In *IEEE International Conference on Robotics and Automation*, 2013.

[49] Jason J Kutch and Francisco J Valero-Cuevas. Challenges and new approaches to proving the existence of muscle synergies of neural origin. *PLoS computational biology*, 8(5):e1002434, 2012.

[50] Leap Motion. www.leapmotion.com.

[51] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[52] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, 2013.

[53] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[54] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015.

[55] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[56] Colin M. Light, Paul H. Chappell, and Peter J. Kyberd. Establishing a standardized clinical assessment tool of pathologic and prosthetic hand function: Normative data,

reliability, and validity. *Archive of Physical Medicine and Rehabilitation*, 2002. http://eprints.soton.ac.uk/256475/.

[57] R. Lioutikov, A. Paraschos, G. Neumann, and J. Peters. Sample-based information-theoretic stochastic optimal control. In *International Conference on Robotics and Automation (ICRA)*, 2014.

[58] Daniel B Lockhart and Lena H Ting. Optimal sensorimotor transformations for balance. *Nature neuroscience*, 2007.

[59] Raymond R Ma, Lael U Odhner, and Aaron M Dollar. A modular, open-source 3d printed underactuated hand. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2737–2743. IEEE, 2013.

[60] Microsoft HoloLens. http://www.microsoft.com/microsoft-hololens.

[61] Microsoft Kinect. www.microsoft.com/en-us/kinectforwindows/.

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

[63] Bud Mishra. Grasp metrics: Optimality and complexity. In *Proceedings of the workshop on Algorithmic foundations of robotics*, pages 137–165. AK Peters, Ltd., 1995.

[64] D. Mitrovic, S. Klanke, and S. Vijayakumar. Adaptive optimal feedback control with learned internal dynamics models. In *From Motor Learning to Interaction Learning in Robots*, volume 264, pages 65–84. 2010.

[65] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[66] Modular Prosthetic Limb, Johns Hopkins Applied Physics Lab. http://www.jhuapl.edu/prosthetics/scientists/mpl.asp.

[67] I. Mordatch, Z. Popovic, and E. Todorov. Contact-invariant optimization for hand manipulation. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 137–144. The Eurographics Association, 2012.

[68] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012.

[69] Tetsuya Mouri, Haruhisa Kawasaki, Keisuke Yoshikawa, Jun Takai, and Satoshi Ito. Anthropomorphic robot hand: Gifu hand iii. In *Proc. Int. Conf. ICCAS*, pages 1288–1293, 2002.

[70] National Instruments. X series user manual: Multichannel scanning considerations.

[71] National Instruments Company. www.NI.com.

[72] National Instruments Company. Knowledgebase 3l8ietlo: How do i eliminate ghosting from my measurements?

[73] Oculus Rift. http://oculusrift.com.

[74] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 255–262. IEEE, 2000.

[75] Omron Company. http://www.omron.com.

[76] OptiTrack. www.optitrack.com.

[77] Shunmugham R. Pandian, Fumiaki Takemura, Yasuhiro Hayakawa, and Sadao Kawamura. Pressure observer-controller design for pneumatic cylinder actuators. *IEEE/ASME Transactions on Mechatronics*, 7(4):490–499, 2002.

[78] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[79] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

[80] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4), 2008.

[81] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*. Atlanta, 2010.

[82] PhaseSpace Company. www.phasespace.com.

[83] Jean-Alban Rathelot and Peter L Strick. Subdivisions of primary motor cortex based on cortico-motoneuronal cells. *Proceedings of the National Academy of Sciences*, 106(3):918–923, 2009.

[84] E. Richer and Y. Hurmuzulu. A High Performance Pneumatic Force Actuator System Part 1 - Nonlinear Mathematical Model. *ASME Journal of Dynamic Systems, Measurement, and Control*, 122(3):416–425, 2001.

[85] Edmond Richer and Yildirim Hurmuzlu. A high performance pneumatic force actuator system: Part1 nonlinear mathematical model. *Journal of dynamic systems, measurement, and control*, 122(3), 2000.

[86] E. Rombokas, M. Malhotra, E. Theodorou, E. Todorov, and Y. Matsuoka. Tendon-driven variable impedance control using reinforcement learning. 2012.

[87] Alla Safonova and Jessica K Hodgins. Construction and optimal search of interpolated motion graphs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 106. ACM, 2007.

[88] Marco Santello, Martha Flanders, and John F Soechting. Postural hand synergies for tool use. *The Journal of Neuroscience*, 1998.

[89] T. Schmidt, R. Newcombe, and D. Fox. Dart: dense articulated real-time tracking with consumer depth cameras. In *Autonomous Robots*, 2015.

[90] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking. *Proceedings of Robotics: Science and Systems, Berkeley, USA*, 2, 2014.

[91] Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Eng.*, 2011.

[92] Shadow Robot Company. www.shadowrobot.com.

[93] SICK Company. www.sick.com.

[94] SMC Company. www.smcworld.com.

[95] Sony Head mounted display. www.sony.co.uk/electronics/head-mounted-display/t/head-mounted-display.

[96] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 2013.

[97] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[98] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Conference on Intelligent Robots and Systems*, pages 4906–4913, 2012.

[99] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 1168–1175. IEEE, 2014.

[100] Yuval Tassa, Tingfan Wu, Javier Movellan, and Emanuel Todorov. Modeling and identification of pneumatic actuators. *2013 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2013*, pages 437–443, 2013.

[101] R. Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[102] Gerald Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2):215–219, 1994.

[103] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[104] E. Todorov, Chunyan Hu, A. Simpkins, and J. Movellan. Identification and control of a pneumatic robot. In *3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, sept. '10.

[105] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012.

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

[107] Emanuel Todorov and Zoubin Ghahramani. Analysis of the synergies underlying complex hand manipulation. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*. IEEE, 2004.

[108] Emanuel Todorov, Chunyan Hu, Alex Simpkins, Javier Movellan, and Computer Science. Identi fi cation and control of a pneumatic robot.

[109] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the American Control Conference*. IEEE, 2005.

[110] Matthew C Tresch and Anthony Jarc. The case for and against muscle synergies. *Current opinion in neurobiology*, 2009.

[111] Matthew C Tresch, Philippe Saltiel, and Emilio Bizzi. The construction of movement by the spinal cord. *Nature neuroscience*, 1999.

[112] Herke van Hoof, Tucker Hermans, Gerhard Neumann, and Jan Peters. Learning robot in-hand manipulation with tactile features. In *Humanoid Robots (Humanoids)*. IEEE, 2015.

[113] Richard M Voyles, J Dan Morrow, and Pradeep Khosla. Gesture-based programming for robotics: Human augmented software adaptation. 1999.

[114] Xue Song Wang, Yu H. Cheng, and Guang Zheng Peng. Modeling and self-tuning pressure regulator design for pneumatic-pressure-load systems. *Control Engineering Practice*, 15(9):1161–1168, 2007.

[115] Tingfan Wu, Yuval Tassa, Vikash Kumar, Javier R Movellan, and Emanuel Todorov. Stac: Simultaneous tracking and calibration. In *Humanoids*, pages 469–476, 2013.

[116] Zhe Xu, Vikash Kumar, and Emanuel Todorov. A low-cost and modular, 20-dof anthropomorphic robotic hand: Design, actuation and modeling. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 368–375. IEEE, 2013.

[117] Zhe Xu and Emanuel Todorov. Design of a highly biomimetic anthropomorphic robotic hand towards artificial limb regeneration.

[118] Yale Open Hand Project. https://www.eng.yale.edu/grablab/openhand.

[119] Katsu Yamane, James J Kuffner, and Jessica K Hodgins. Synthesizing animations of human manipulation tasks. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 532–539. ACM, 2004.

[120] Yuting Ye and C Karen Liu. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (TOG)*, 31(4):41, 2012.

[121] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Trans. Graph.*, 26(3):Article 105, 2007.

[122] Tassa Yuval, Tingfan Wu, J Movellan, and E Todorov. Modeling and identification of pneumatic actuators. In *International Conference on Mechatronics anf Automation (ICMA)*, 2013.

[123] Xu Zhe, Vikash Kumar, Yoky Matsuoka, and Emanuel Todorov. Design of an anthropomorphic robotic finger system with biomimetic artificial joints. In *IEEE BIOROB*, 2012.

[124] Joshua Z Zheng, Sara De La Rosa, and Aaron M Dollar. An investigation of grasp type and frequency in daily household and machine shop tasks. In *IEEE International Conference on Robotics and Automation'11*, pages 4169–4175. IEEE, 2011.

[125] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov. Value function approximation and model-predictive control. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2013.

[126] Zspace. http://zspace.com/.