THE UNIVERSITY *of* EDINBURGH

Learning Robotic Motor Skills for Dynamic Grasping, Catching, and Dexterous In-hand Manipulation

Wenbin Hu

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2023

Abstract

Endowing robots with human-level grasping and manipulation skills is an appealing yet challenging research topic over decades. Towards more extensive functionalities, the robots should interact with the objects and environment in a more robust, efficient and intelligent way. With the rapid development of deep learning, merging the learning-based algorithms into the perception and control loop of the robot exhibits promising performances. In this thesis, we explore the implementation of model-free deep reinforcement learning (DRL) and learning from demonstrations (LfD) in acquisition of dynamic grasping and manipulation motor skills of a robotic hand-arm system.

We propose a systematic framework for end-to-end learning robotic control policies with model-free DRL, including simulation set-up, reward design and sim-to-real transfer. The DRL-based training framework is evaluated by three case studies with different robotic tasks and research emphases. We first introduce the framework in Chapter 3, with a focus on the design of reward function and special initial training states. The trained policy coordinately controls the robotic hand and arm for dynamic reaching, grasping and re-grasping of objects sliding on the ground. We further propose a multi-modular structure consisting of three control policies trained with proposed framework (Chapter 5). With seamless cross-module integration achieved by the gating policy network, the robot can catch in-flight objects with mitigated impact forces. In Chapter 6, dexterous in-hand manipulation skills with tactile feedback is trained from scratch in simulation and directly transferred to real robot. We focus on the exploitation of tactile perception and sim-to-real transfer methods. Moreover, in Chapter 4 we propose a low-cost method to collect human demonstration data for supervised learning. Using only proprioceptive sensing, the trained neural network based controllers can grasp property-unknown objects with adaptive grasping forces.

Lay Summary

Reactive and efficient skills of grasping and manipulating objects are essential for robots working in domestic environments. While adept in structural and standardized tasks, conventional methods which control the robot by pre-defined rules usually struggle in face of uncertainties, e.g. when the object suddenly moves as the hand approaches, or when the object stiffness is unknown. Quick adaptation to dynamic changes and generalization ability to novel objects are indispensable but still challenging in robotic grasping and manipulation. In this thesis, we aim to propose a systematic framework for acquiring autonomous robotic grasping and manipulation skills. We evaluate the proposed methods by four challenging tasks, covering the three stages of grasping and manipulation. *Pre-grasp stage* when the robot hand moves to a proper pose which is effective to grasp. We realize the reaching, grasping and re-grasping of objects moving on the table (Chapter 3), and the catching of in-flight objects (Chapter 5). *Grasping stage* when the fingers close towards the object and perform the grasp. We propose a neural network based controller to adaptively adjust the grasping force when grasping objects with unknown stiffness (Chapter 4). *Post-grasp stage* when the hand manipulate the grasped object. We study the in-hand manipulation of slender cylindrical objects based on fingertip tactile sensing (Chapter 6).

Benefiting from the rapid breakthroughs of deep learning, the robot control steps into a new era. Taking perception as input and generating control commands in milliseconds, neural network based policies are inherently effective for dynamic tasks. In this thesis, we focus on the application of model-free deep reinforcement learning (Chapter 3, 5 and 6) and learning from human demonstrations (Chapter 4) in robotic grasping and manipulation.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified. The work in this thesis is either published or under review in the following articles with attribution as follows. As the co-author of the second article, I participated in the ideas development, experiments implementation and paper writing.

Wenbin Hu*, Chuanyu Yang, Kai Yuan, Zhibin Li. "Learning Motor Skills of Reactive Reaching and Grasping of Objects" *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021.

Shuaijun Wang*, **Wenbin Hu**, Lining Sun, Xin Wang, Zhibin Li. "Learning Adaptive Grasping from Human Demonstrations" *IEEE/ASME Transactions on Mechatronics*, 2022

Wenbin Hu*, Fernando Acero, Eleftherios Triantafyllidis, Zhaocheng Liu, Zhibin Li. "Modular Neural Network Policies for Learning In-flight Object Catching with a Robot Hand-Arm System" *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023 (Under Review)

Wenbin Hu*, Bidan Huang, Wang Wei Lee, Sicheng Yang, Yu Zheng, Zhibin Li. "Dexterous In-Hand Manipulation of Slender Cylindrical Objects through Deep Reinforcement Learning with Tactile Sensing" *IEEE/ASME Transactions on Mechatronics*, 2023 (Under Review)

Acknowledgements

Firstly, I would like to express my sincere appreciation to my principal supervisor Zhibin (Alex) Li, for his patient and unreservedly guidance during this tough but meaningful journey. I met him in 2017 and since then his enthusiasm towards research, career and life deeply influences me. Through countless times of fruitful discussions, he shows me how to do good research, and more importantly, how to become a strong-willed person. He always fills our research group with tremendous energy and optimism. I will never forget the kindly and inspiring words he said when I was upset about my paper rejection. The lessons I learned from him are the most valuable treasure and weapon to overcome the challenges in my future career.

I would also like to thank my fiancee, Jiaxin. Without her genuine love and heartful support I cannot accomplish this thesis. She effaces my emptiness with companion, cures my depression with cuddles and pacifies my struggles with smiles. I will repay her kindness with all my love.

I would also like to thank my parents and grandparents who raised me. They made me who I am today. The old memory with them are like pure light, warm, sacred and dazzling, which is the inexhaustible source of my intrinsic power. I hope this thesis can make them proud of me.

I would also like to thank all the staff from University of Edinburgh for their remarkable work of making the campus a pleasant place for study and research.

Last but not least, I would like to thank all my friends at Edinburgh, for their advice, laughter and free meals.

Table of Contents

Chapter 1

Introduction

With numerous potential working scenarios, future robots with universal purpose need to possess the athletic intelligence of efficient and robust interaction with the surrounding environment to accomplish tasks. The complexity of human environments suggests that an effective manipulation robot ought to adaptively react to uncertainties and dynamic changes during the motion. In this thesis, we aim to acquire the motor skills of a robotic manipulator in autonomous reaching, grasping and manipulation of objects from a learning perspective.

Robotic grasping is a hybrid task consisting of multiple sub-problems, and amongst which two major topics are *where-* and *how-to-grasp*. Given the observation of target object, the robot end-effector configuration and sometimes the purpose of the grasp, grasp synthesis algorithms solve the *where-to-grasp* problem by generating feasible hand pose and contact locations for grasping. Over the recent decade many analytical and data-driven approaches have been proposed for grasp synthesis for either known or novel (Patten et al., 2020) objects, single object or objects in clutter (Mahler et al., 2017a), using either parallel grippers (Chen and Burdick, 1993) or dexterous hands (Duan et al., 2021), etc. In comparison, less attention has been drawn to the *how-to-grasp* problem, in precise terms, moving

robot hand to the desired grasp pose and performing the grasping motion. Conventional motion planning methods are viable for reaching stationary objects – moving the end-effector to the target pose via a collision-free trajectory and then closing the fingers. However, reaching and grasping moving objects pose various new challenges, especially for multi-fingered robot hands. High dynamic nature of the environment requires the fast response capability to changes and uncertainties of the hand-arm controller. Fine coordination between robot arm and hand (fingers) is also important for efficient and collision-free grasping of moving objects. Moreover, the contact forces applied to the grasped objects should be appropriate, large enough to resist the slippage but not causing excessive deformation or damage. In the first part of this thesis we focus on the *how-to-grasp* problem, namely synergetic hand-arm motion to reach and grasp moving objects (Chapter 3, 5), and performing adaptive grasping forces for objects with unknown physical properties (Chapter 4).

In many cases the robot need to further manipulate the object after it is firmly grasped, e.g. plugging a cable into the socket or using a stirring rod to stir the drink. Multi-fingered hands significantly increase the robotic dexterity, enabling the robot to perform many tasks that is difficult with parallel grippers. Dexterous in-hand manipulation using robotic fingers is a necessary skill, especially when the manipulated objects are small or when the robot arm movements are restricted in the confined workspace. As defined by Bicchi (2000), dexterous in-hand manipulation is the capability of changing the pose of manipulated object within the hand workspace according to the task. Extensive research has been done in the robotic manipulation domain (Yu and Wang, 2022; Billard and Kragic, 2019), but human-level manipulation skills in some tasks still remain challenging for robots, e.g. fingertip manipulation of elongated objects, as one has to continuously change the object pose with fingers while satisfying the grasp criterion. Visual input is often occluded by the robot hand during the in-hand manipu-

lation. Benefit from the development in tactile sensors, robots can get access to direct perception of contacts which significantly improves the performances of dexterous manipulation (van Hoof et al., 2015). In Chapter 6 we study the dexterous in-hand manipulation of slender cylindrical objects, which are more difficult than normal objects because of the narrow contact regions, using the tactile feedback.

Before the remarkable breakthroughs in deep learning, analytic approaches based on physical models of the robot hand and objects dominated the community of robotic grasping and manipulation. However, the performances are usually restricted by the deviations of simplified models and the heavy computation consumption. Deep learning spectrum is undergoing a rapid development, and the artificial intelligence even outperforms human in certain tasks, e.g. the game of go (Silver et al., 2017). Merging the neural networks into the robotic perception (Wei et al., 2022) and decision making (Amiri et al., 2020) modules enhances the efficiency and robustness of integral robotic systems. By leveraging such powerful tools, many learning-based algorithms have been proposed in robotic grasping (J.Bohg et al., 2014; Newbury et al., 2022) and manipulation (Yu and Wang, 2022; Rajeswaran et al., 2018).

Compared to the frameworks where some specific sub-modules are replaced by learned models, deep reinforcement learning provides an end-to-end solution for various robotic tasks, e.g. quadrupedal walking (Yang et al., 2020; Lee et al., 2020), grasping (Quillen et al., 2018) and dexterous in-hand manipulation (Andrychowicz et al., 2020), taking the environmental perception as input and directly generating control commands. In DRL-based algorithms, policy convergence is guided by the reward functions. Sparse reward is straightforward but difficult to learn, especially for complex tasks. We design task-specific dense reward functions for training reaching and grasping policy (Chapter 3), catching

policy (Chapter 5) and dexterous in-hand manipulation policy (Chapter 6), showing that the proposed reward design method can be extended to different robotic grasping and manipulation tasks.

Gathering training data in a trial-and-error manner with real robots is time-consuming and dangerous, especially at the early stage of training. Hence, physics simulators such as PyBullet (Coumans and Bai, 2016–2021) and MuJoCo (Todorov et al., 2012) are widely used for the more efficient and safe training. The policies trained in simulation are ultimately transferred to real robots. To deal with the sim-to-real discrepancies, in Chapter 6 we propose a general method to calibrate the finger joint models. We also simplify the model of tactile sensors and randomize the physical properties of manipulated objects.

Different from pure DRL methods where the policies learn from scratch in a trial-and-error manner, leveraging human demonstrations in training is more sample efficient, and hence enables the learning from real robot data. For grasping objects with adaptive contact forces, it is difficult to precisely simulate the correlation between finger joint commands and the applied grasping forces. In Chapter 4 we encode the finger control policy as neural network and train it with human demonstration data. A low-cost data collection method is proposed, where the human hand motion is recorded by a camera to teleoperate the robotic hand for object grasping.

Compared with other application scenarios of learning-based methods like translating or computer vision, safety issue is more important in robotics domain, since the robots physically interact with environment. A robot arm swinging with high speed is potentially more dangerous than a chatting bot. We abide by this concept throughout the thesis. Effective re-grasp behaviour is specifically learned to avoid the potential collision between the outer side of fingers and object during the dynamic grasping, as described in Section 3.3.2. By learning from human

demonstrations, excessive grasping forces for deformable objects are prevented in Chapter 4. The impact force when a robot hand catches the flying object is also mitigated by special arm movements in Section 5.3.4.

To summarize, in this thesis we study the application of model-free DRL and learning from demonstrations in acquisition of autonomous motor skills of robot manipulator. Four projects cover the tasks from three stages of grasping: pre-grasp stage, grasping stage and post-grasp stage. In the pre-grasp stage, we propose a holistic controller for both robotic hand and arm to seamlessly approach and grasp objects, even when they are sliding on the ground or flying in the mid-air. In the grasping stage, after the hand arrives the desired pre-grasp pose, we propose a neural network based controller to adaptively regulate grasping forces for novel objects with unknown physical properties, without any exteroceptive sensing like vision or tactus. In the post-grasp stage, when the object is already grasped, sometimes we need to adjust the object pose with fingers. We propose a learning-based fingertip in-hand manipulation controller. Leveraging the distributed tactile sensor arrays, the three-fingered robotic hand can manipulate the slender cylindrical objects (e.g. stirring rods) using only fingertips, to follow real-time pose trajectories. We summarize a systematic pipeline of deep reinforcement learning implementation in robotic grasping and manipulation (Chapter 3, 5, 6), including the simulation setup, reward design, initial states setup, model calibration and domain randomization. In Chapter 4 we propose an efficient method for collecting and learning from human demonstrations.

1.1 Coordinated Robotic Hand-Arm Motion

Smooth hand-arm coordination enables human to reach and grasp objects fast and accurately. Endowing the robots with comparable motor skills can significantly improve the grasping efficiency. Conventional methods perform the reaching and

grasping of objects separately in a sequential manner by different controllers: The robot hand first approaches the desired pre-grasp pose guided by the arm controller, and then the grasping motion is applied by the finger controller. Errors of hand pose in the approaching stage directly affects the effectiveness of finger motion in the grasping stage. Controlling the robot hand and arm with separate controllers block the synergy actions between them, and hence limits the reaction ability to environmental changes and resistance to errors.

Lampe and Riedmiller (2013) proposed a hierarchical structure consisting of a long-range controller moving the hand towards the object, and a short-range controller performing the fine adjustments of the hand pose and the final grasping motion. Both controllers cover the motion of robot arm and hand, but the switch between them is determined by a fixed threshold of hand-object distance. Morrison et al. (2018b) proposed a light weight convolutional neural network (CNN) for grasp synthesis that enables the top-down grasp using a two-fingered gripper in real-time. Visual-servoing systems that control the robot with vision feedback are implemented in the combined reaching and grasping tasks (Lampe and Riedmiller, 2013; Ardon et al., 2018), but the start timing of finger closure is given by predefined criterion or a trained module.

In this thesis we aim to obtain a unified controller of the hand-arm system for seamlessly reaching, grasping and re-grasping of dynamic targets. To evaluate the quick reaction capability of the proposed controllers, we test the robot with extreme tasks. We first train a policy for grasping objects sliding on the table (Chapter 3). The robot can adjust the hand-arm motion as the object moves, and apply efficient re-grasp in case of failures. Then based on the hand-arm controller, we further develop a multi-modular framework to catch in-flight objects (Chapter 5). Since the task is more challenging, we split the hand-arm motion controller into two parts: A reaching controller for moving robotic hand to the

selected pre-grasp pose, and a grasping controller for mitigating the impact force and performing the grasping motion. We train a gating controller for seamlessly coordination of the two aforementioned controllers.

1.2 Adaptive Grasping

Applying adaptive grasping forces for objects with unknown properties is essential but challenging, especially for fragile or deformable objects. The related solutions can be roughly categorized into three groups: methods using sensors for active perception of object surface; methods using specially designed end-effectors; feedback control methods using general robot hands without aforementioned sensors nor special hardware design. Tactile sensing provides dense contact information during the grasping and manipulation. Narita et al. (2020) theoretically modelled the incipient slip between hand and grasped object, and proposed a grasp force controller based on the slip detection. Takahashi et al. (2008) switched between position control and force control based on the contact detection, and the grasping forces were determined by slip measurements. Unlike computer vision, tactile data processing is heavily coupled with the hardware design, and hence it is difficult to obtain a general method to exploit the tactile information. Difficulties in sensor installation and maintenance also restrict the adoption of tactile sensors in the broad robotics community.

With inherent compliance, soft robot end-effectors have more grasping adaptability than rigid ones, showing good performances in industry object picking and fruit harvesting (Gunderman et al., 2022; Hao et al., 2016; Zhou et al., 2018). However, the control approaches are highly correlated with the kinematic structures and hence hard to generalize to different robot end-effectors.

Without tactile sensors and specially designed structures, robot grippers or hands can only use proprioceptive sensing, e.g. joint positions and torques, to infer the

properties of grasped objects. Pfanne et al. (2020) proposed an impedance-based controller for in-hand object re-orientation, where the desired normal contact forces are pre-defined by users and the controller calculates the specific contact force vectors based on the balance and friction. Xiong et al. (2005) explicitly modeled the nonlinear coupling between the contact force and elastic deformation of the object, and proposed an adaptive grasping controller based on the model. Analytical grasping methods are prone to suffer from model errors, while learning-based methods can compensate the such inconsistency with massive training data. Learning from human demonstrations is a powerful mechanism to generate expert policies for training. Lin et al. (2012a) trained a Gaussian Mixture Model with the recorded motion and force data from human grasping demos, and generated the robotic force and motion trajectory with Gaussian Mixture Regression. Misimi et al. (2018b) integrated visual and tactile data of human demonstrations, enabling the autonomous grasping of deformable and fragile objects. In Chapter 4, we propose an efficient method of learning from human demonstrations, where only a RGB-D camera is used to record human grasping motion. Three different neural network based controllers are designed and evaluated for grasping property-unknown objects with adaptive forces. We show that the history data of finger joints is effective in adaptive grasping.

1.3 Dexterous In-hand Manipulation

High dexterity of human hands enable us to interact with various objects freely. Empowering robots with comparable in-hand manipulation skills is an active research topic for decades. Bicchi (2000) provides a clear definition of robot dexterity and summarises the challenges from the mechanical design and control perspectives. Pfanne et al. (2018) detect the contact positions and normal directions between hand and object based on vision and joint measurements, and

further develop an impedance controller for in-hand object reorientation (Pfanne et al., 2020). Analytical methods rely on the explicit models of object and contact, and the variations introduced by the hand-object movements amplify the model errors, influencing the manipulation performances.

Learning-based algorithms become dominant in robotic manipulation in recent decades, where the controllers are parameterized as trainable neural networks. Several surveys review the progress in robot learning for manipulation (Kroemer et al., 2021; Yu and Wang, 2022; Nguyen and La, 2019), and here we selectively discuss the work that is most related to our topic. Andrychowicz et al. (2020) trained the continuous re-orientation of cubes with a dexterous robot hand fixed at the palm-up pose, and Chen et al. (2022) extended the manipulation to any given palm pose. Fingertip manipulation without support of the palm can also be learned, e.g. object re-orientation via finger-gaiting (Sievers et al., 2022; Khandate et al., 2022). Human demonstrations can enhance the sample efficiency and promote the learning of natural hand movements. Solak and Jamone (2019) learned dynamical movement primitives from demonstrations and generalized them to in-hand translation and rotation of unknown objects. Rajeswaran et al. (2018) added few human demonstrations into deep reinforcement learning to reduce the complexity of random policy searching. However, for robot hands different with human hands in structures (e.g. the three-fingered hand we use in Chapter 6), it is difficult to provide effective demonstrations, especially for contact-rich tasks.

In Chapter 6, we focus on a more challenging task: the continuous fingertip manipulation of elongated objects. Different from previous object re-orientation tasks with sparse target poses, manipulation such as stirring needs the object to follow continuous trajectory in real-time. Compared to objects with ordinary shapes and sizes that are widely used in previous research, when manipulating the

elongated object, a large part of the object is not inside the convex hull formed by fingers and palm, and therefore it is more difficult to maintain the object balance. Also, elongated objects have narrow potential contact regions, and hence the perception of contacts needs to be more accurate. Tactile sensing can provide such essential contact information during in-hand manipulation (Yousef et al., 2011). We utilize the distributed sensors on fingertips to infer the contact locations, and evaluate the effectiveness of different types of tactile information in dexterous in-hand manipulation.

1.4 Thesis Structure

This thesis contains seven chapters. In this chapter we introduce the related work in learning-based robotic grasping and manipulation. In Chapter 2 we describe the background of the techniques that are used in this thesis. The subsequent four chapters (Chapter 3, 4, 5, 6) present four related projects with different research emphasises, covering different aspects of grasping and manipulation, from pre-grasp hand-arm motion to post-grasp manipulation.

- **Chapter 3** presents a deep reinforcement learning based controller to autonomous reach, grasp and re-grasp static or moving objects on the table with coordinated hand-arm motion. The proposed dense reward function considers the spatial relation between object and hand before, during and after the grasp. Special initial training states are proposed to enable the robot to perform collision-free re-grasps in case of failures.

  Wenbin Hu, Chuanyu Yang, Kai Yuan, Zhibin Li. "Learning Motor Skills of Reactive Reaching and Grasping of Objects." *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021.

- **Chapter 4** presents a neural network based finger controller to generate

adaptive grasping forces for property-unknown objects. The training data is gathered from human demonstrations. Only the robot proprioceptive information is used as state feedback. We compare three different network structures and demonstrate the effectiveness of history data in adaptive grasping.

> Shuaijun Wang, **Wenbin Hu**, Lining Sun, Xin Wang, Zhibin Li. "Learning Adaptive Grasping From Human Demonstrations." *IEEE/ASME Transactions on Mechatronics*, 2022.

- **Chapter 5** presents a multi-modular framework for catching objects in-flight. We modify the grasping policy learned in Chapter 3, splitting it into a reaching policy to move the hand to the desired pre-catch pose, and a grasping policy to catch the object softly. We also train a gating network to autonomously coordinate these two policies. The proposed framework can react to disturbances on objects quickly, and mitigate the impact force of catching by special arm movements.

  > **Wenbin Hu**, Fernando Acero, Eleftherios Triantafyllidis, Zhaocheng Liu, Zhibin Li. "Modular Neural Network Policies for Learning In-flight Object Catching with a Robot Hand-Arm System" IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023 (Under Review)

- **Chapter 6** presents a deep reinforcement learning based controller for tactile based in-hand dexterous manipulation. With task specific reward function, the robot hand can manipulate the slender cylindrical objects with fingertips to track different continuous pose trajectories. Policies trained with different representations of tactile feedback are analyzed and compared. Sim-to-real transfer methods are leveraged to deploy the policies trained in simulation to the real robot.

Wenbin Hu, Bidan Huang, Wang Wei Lee, Sicheng Yang, Yu Zheng, Zhibin Li. "Dexterous In-Hand Manipulation of Slender Cylindrical Objects through Deep Reinforcement Learning with Tactile Sensing" IEEE/ASME Transactions on Mechatronics, 2023 (Under Review)

In the last chapter we conclude the content and contributions of this thesis, and provide insights of future work based on the thesis.

Chapter 2

Background

In this chapter we describe the technique background of the methods that we use in thesis, namely training the grasping and manipulation policies with deep reinforcement learning in simulation (Chapter 3, 5, 6) and then transfer them to real-world environment (Chapter 6). We also introduce the background of learning from demonstrations, which we use in Chapter 4.

2.1 Deep Reinforcement Learning

Sequential decision-making in an uncertain environment is a core capability for robots with self-autonomy, and reinforcement learning (RL) offers a theoretical solution to this problem. Similar to a living creature, in RL an artificial agent learns a task-conditioned policy by interacting with the environment. Based on the scenario, the agent can be a game player, an autonomous car or a robot. With accumulated experiences during the interactions, the learned policy of the agent is converged and optimized with respect to formulated rewards. In combination with deep learning techniques, deep reinforcement learning (DRL) can address the problems with high-dimensional observation space from a massive amount of training data. DRL has achieved notable success in several tasks, such as playing

video games (Mnih et al., 2015), Go (Silver et al., 2016), autonomous car driving (Pan et al., 2017) and robotics (Yang et al., 2020; Levine et al., 2016).

2.1.1 Problem Formulation

In DRL an agent learns a behaviour/policy in a trial-and-error manner, where it autonomous interacts with the environment and updates the intrinsic policy based on collected information. In this thesis, the process of agent-environment interaction is modeled as a finite-horizon discounted Markov decision process (MDP). The future of the MDP is only determined by the current state, which means we do not need to consider the history information. An MDP is a five-tuple consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a distribution of initial states $p(s_0)$, the state transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Each learning episode starts with a sampled initial state s_0. Thereafter, at each time step, the agent chooses one action based on current state and the policy $\pi(s_t)$ to be executed. After execution, the agent will receive a reward $r(s_t, a_t)$ and the state observation s_{t+1} from the environment. The goal of the agent is to obtain a policy $\pi(s, a)$ that maximize the expected discounted sum of rewards $V^\pi(s) : S \to \mathbb{R}$, which is also called the Value function:

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \right], \tag{2.1}$$

where $\gamma \in (0, 1]$ denotes a discount factor. Equation 2.1 represents the expectation of the discounted reward sum that a policy can get from the starting state s. The discount factor γ makes the sum converge, and serves the idea that the policy cares more about immediate future than the distant ones with more uncertainties. Another important concept is the Q function $Q^\pi(s, a)$ which is defined as follows:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right], \tag{2.2}$$

which measures the value of taking action a at the state s. The optimal policy $\pi^*(s)$ can be derived from the optimal Q function:

$$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} Q^*(s,a), \tag{2.3}$$

where $Q^*(s,a) = \max_{\pi \in \Pi} Q^\pi(s,a)$. The optimal Q function is the expected discounted return when taking a given action a at a certain state s while following the policy π^* thereafter. The advantage function evaluates the effectiveness of taking a specific action a compared to the expected return when following the policy π directly:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{2.4}$$

Based on different criterion, the DRL algorithms can be classified into several categories. *Model-based* methods using a model to describe the state transition dynamics of the environment and the estimated reward functions, which is either explicitly defined, e.g. the games with fixed rules like Go, or learned from data. With the available model, planning algorithms can be used for action recommendation, e.g. look-ahead search for discrete actions and trajectory optimization for continuous action space. Methods without modelling the environment are referred as *model-free* methods, which has two major components: a policy and a value function representing the quality of the state or state-action pair. Solving the problem from different aspects, the model-free methods can be classified into two different branches: *value-based* methods and *policy-based* methods. *Value-based* methods aims to build an optimal value function and subsequently gets the desired policy with Equation 2.3, such as the Q-learning algorithm (Watkins and Dayan, 1992) and the deep Q-network (DQN) (Mnih et al., 2015). *Policy-based* methods use variants of stochastic gradient ascent with respect to the policy parameters to optimize the expected cumulative reward, searching the hyperspace of policies instead of assigning values to state-action pairs. Since the policy-based methods directly optimize what we need, they have stability and reliability

improvements over the value-based methods (Nguyen and La, 2019).

2.1.2 Proximal Policy Optimization

Proximal policy optimization (PPO) is one of the most widely used policy gradient methods due to its robustness, simple parallel implementations and sample efficiency (Schulman et al., 2017). Policy gradient methods estimate the policy gradient and plugs it into a stochastic gradient ascent algorithm in order to optimize a performance objective function $J(\theta)$. The gradient update is of the form $\triangle\theta \propto \nabla_\theta L(\theta)$ where $L(\theta)$ is the parameterized objective function. As an example, one of the commonly used objective function in policy gradient algorithms is

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right] \tag{2.5}$$

where $\hat{A}_t$ is the estimation of the advantage function at timestep t, suggesting whether the action a_t is better or worse than the average action the policy takes at s_t. By differentiating the objective function, the estimation of the gradient is of the form $\hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right]$. Though we can directly optimize the objective loss L^{PG} based on the gradient estimator, empirically it usually causes large policy updates, making the learning unstable and hard to converge. To prevent the excessive vibration in policy updates, Schulman et al. (2015) proposed the trust region policy optimization (TRPO) which maximizes the surrogate objective function subject to a hard constraint:

$$\max_\theta \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \tag{2.6}$$

$$s.t. \quad \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{\text{old}}}, \pi_{\theta_{\text{old}}} \right] \right] \leq \delta. \tag{2.7}$$

where the θ_{old} denotes the policy parameters before the update, and δ denotes the upper bound of the KL-divergence. By making a first-order linear approximation to the surrogate objective and a second-order quadratic approximation to the KL divergence constraint, the optimization problem can be solved with conjugate gradient algorithm.

A major disadvantage of TRPO is that it is computationally expensive. Developed from TRPO, PPO is simpler in formulation and faster in computation. Without the constraint, maximization of Equation 2.6 would result in excessively large policy updates. Let $r_t(\theta)$ denote the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. PPO modifies the objective function, penalizing the changes of the policy that shift $r_t(\theta)$ away from 1. The objective function of PPO is:

$$L^{PPO}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{2.8}$$

where ϵ is a hyper-parameter designed to clip the probability ratio and constrain the policy update. This objective function allows the policy to update towards action distributions with positive advantage while avoiding aggressive policy changes. The second term of the objective function clips the probability ratio by the interval $[1-\epsilon, 1+\epsilon]$. The minimum of the original and clipped objective is taken as the final objective, which is the lower bound of the original objective. PPO has an actor-critic fashion, with the actor consisting of a policy $\pi_\theta(s_t)$ parameterized by θ and a critic consisting of estimated value function $V_\phi(s_t)$ parameterized by ϕ. The goal of PPO is to maximize the objective function $L(\theta)$, therefore π_θ is updated by gradient ascent with respect to Equation 2.8. The estimated value function V_ϕ is trained by minimizing the loss function:

$$L(\phi) = \hat{\mathbb{E}}_t \left[(V(s_t) - R_t)^2 \right], R_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}, \tag{2.9}$$

where R_t is the discounted reward during time-step t, γ is the discount factor and T is the total number of time-step during the sampled path. V_ϕ is updated by gradient descent with respect to Equation 2.9.

2.2 Learning Grasping and Manipulation Policies

Learning-based robotic control is an interdisciplinary topic between robotics and machine learning. From the perspective of learning community, robotic tasks are

appealing scenarios to evaluate and apply the algorithms. From the perspective of robotics community, learning-based algorithms are powerful tools with enormous potential to enable more advanced robotic athletic intelligence. Despite drawing attention from both research communities, there are many obstacles for embedding machine learning algorithms into robot control loop, e.g. how to set up the training environment and how to define the goal or reward. In this section we focus on the related work of utilizing learning-based methods in robotic grasping and manipulation. We first center the discussion around the application of deep reinforcement learning algorithms, where the control policies are trained in simulation and then transferred to reality. Then we review the exploitation of human demonstrations for safe and robust motor skills. We present the open challenges and opportunities for improvements and then lead to our proposed solutions in the next chapters, evaluated by certain tasks.

2.2.1 DRL-based Policy Learning

As aforementioned, the deep reinforcement learning algorithms can be categorized into model-based RL and model-free RL, and both categories achieve notable success in robotic grasping and manipulation. Accurate transition dynamics models are rarely available and difficult to learn from data. However, approximate models can be acquired and applied to guide exploration and policy search (Levine et al., 2015; Finn and Levine, 2017; Schenck and Fox, 2018). The learned models greatly improve sample efficiency, and they are reusable for different tasks in similar environments. The major drawbacks of model-based RL are (1) the difficulty of learning the models, and (2) incorrect models may introduce bias into policy learning, resulting in poor cross-domain performances. For contact-rich grasping or manipulation tasks, it is challenging to precisely model the state transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, especially when the object physical properties are unknown (Chapter 4) or when the finger surfaces are deformable (Chapter 6). With-

out having access to a model, model-free methods learn policies directly through robot-environment interactions. The disadvantages of model-free RL are that (1) learned policies cannot easily adapt to new tasks, and (2) the learning requires large amount of data. The former problem can be addressed by augmenting new training data, empowering the policy with new skills Yang et al. (2020). The requirement of data can be addressed by using simulation, as discussed in the next section. For comprehensive review of DRL-based robot learning we refer the readers to papers by Kroemer et al. (2021); Nguyen and La (2019). In this thesis we use the PPO algorithm as it is one of the most advanced and widely used model-free RL algorithm in robot learning field.

2.2.1.1 Physics Simulators

Training of the policies requires a certain amount of data, even for the sample efficient algorithms. Collecting sufficient training data with real robots is slow and costly, and hence it becomes an appealing option to use the physics simulators which can generate realistic data trajectories faster than real time. Taking advantage of the high-performance computing devices and parallelism techniques, thousands of robots can be learning simultaneously in simulation (Rudin et al., 2021). Another appealing fact of using simulation for training is that in simulation we can easily get the precise information that is difficult to acquire in real-world, e.g. the object translational and rotational velocities used for reward computation. Robotic policies trained in simulation ought to be deployed to the real robots in the real environments ultimately. Therefore the training environment and data generated by physics simulator should be as realistic as possible. The robot should *be* realistic (the robot body model), and *feel* realistically (the perception model), and *act* realistically (the actuation model), and get realistic *responses* from the environment (the contact model).

Several papers reviewed and compared the existing simulators with respect to the

capabilities, efficiency and accuracy (Körber et al., 2021; Choi et al., 2020; Liu and Negrut, 2021). Amongst the vast open-sourced physics simulators emerged in recent decades, Gazebo (Koenig and Howard, 2004), PyBullet (Coumans and Bai, 2016–2021) and MuJoCo (Todorov et al., 2012) are most popular for learning-based robotic research. Gazebo supports four different physics engines: Bullet, DART, ODE and Simbody, and has a close integration with Robot Operating System (ROS). Thanks to the homogeneous ROS connection, one can control the real physical system with the same interfaces for simulation. Hence it is easy to perform the sanity check in simulation before the robot moves in real environment. PyBullet provides a wide range of functions for building customized simulated environments. It has a collision detection engine which is widely used by other engines. The default usage of cone friction model is more accurate. Based on the extensive documentation and wide community, it is easy to learn and use even for inexperienced users. MuJoCo has more tunable solver parameters and settings than other simulators, and its own model description format MJCF (MuJoCo Modeling XML File) is more flexible and comprehensible than URDF (Universal Robotics Description Format). The framework and extensive benchmark environments of OpenAI Gym (Brockman et al., 2016) makes MuJoCo a perfect simulator for learning related research. Nvidia Ominiverse Issac Sim is a newly released robotics simulation toolkit, featured in high resolution rendering and realistic synthetic sensors. Issac Gym provides a framework specially designed for RL policies to interact with the simulation. With GPU acceleration and tensor-based APIs, thousands of environments can run in parallel on a single workstation. Though still under development, it can be a promising option for DRL-based robotic research. In this thesis we use MuJoCo for Chapter 3 and 6, and PyBullet for Chapter 5 due to the license issue of MuJoCo. Note that since November 2021, Deepmind makes MuJoCo completely free and open-sourced.

2.2.1.2 Training Environment Setup

Building up a proper simulated environment is essential for robotic policy learning. Though the training environment is highly correlated to the task, there are some general principles that need to be followed when designing it. The most fundamental aspect is that the simulation should resemble the real scenario, including the robot itself, the interacted objects and rest environment. To mitigate the sim-to-real gap, many parameters should be calibrated before the training, e.g. the collision body, weight and inertia matrix of each robot link, the friction efficient of any related surface. For robotic locomotion (Rudin et al., 2021; Lee et al., 2020) or objects sliding on the ground (Chapter 3), the floor friction and terrain shapes should be carefully designed. For tasks evolving fast motion, e.g. catching in-flight objects, the air resistance should also be explicitly modeled (see Section 5.3.1). Modeling the actuator dynamics accurately also significantly narrow down the reality gap (Sievers et al., 2022; Tan et al., 2018). We calibrate the finger joint dynamics via trajectory following experiments in Section 6.3.4.1. Due to the inherent uncertainty and complexity of real environment, policies trained with single set of simulation parameters lack the generalization and adaptation abilities. Therefore domain randomization techniques are introduced during the training Andrychowicz et al. (2020), which we discuss in detail in Section 2.2.1.4.

The choice of action space is important for policy learning. The control signals sent to the low-level actuators are determined by the output actions of the policy, but in practice the actions cannot be directly used as control signals. An additional controller is often used to bridge the policy actions and actuator commands Sievers et al. (2022), e.g. linear PID controllers or model-based impedance controllers. Adding the controller leverages both the stability and robustness from robot control algorithms Spong et al. (2006) and the emergent intelligence of learned policies. Policy actions can either be defined in the joint space or the

Cartesian task space of the end-effector. For controlling robotic arms, it is significantly easier to train the policies with task space actions, and the joint commands are generated by pre-designed controller Lampe and Riedmiller (2013); Lowrey et al. (2018). For contact-rich dexterous grasping and manipulation tasks, joint space actions are better for the emergent of motor skills (Andrychowicz et al., 2020; Veiga et al., 2020). From the control point of view, the low-level control frequency is usually up to 1kHz, but the high-level learned policies are difficult to reach comparable frequency due to limitation of observation input. A common practice is to repeat the control commands until next policy update. Actions from learned policies can be jerky sometimes, leading to vibrating robot motion which may damage the robot and surrounding environment. There are several options to smooth the jerky actions: reward shaping to penalize motion jerkiness, imitating smooth reference trajectories (Peng et al., 2018), adding smooth term into training loss Yang et al. (2020). In Chapter 5 and 6, we use low-pass filters to process the generated actions.

Grasping and manipulation tasks exhibit a sequential and modular structure. Even the simple benchmark pick-and-place task can be divided into sub-phases like reaching and grasping the object, holding and transporting the object, placing and releasing the object. Learning the long sequential tasks as a whole is challenging, as the subsequent sub-tasks highly depend on how well the previous sub-tasks performed. Decomposing the primary task into smaller and more tractable sub-tasks, and training a modular policy for each sub-task significantly reduces the problem complexity. Modular structure also provides flexibility. Learned modules are reusable and can be assembled in different ways for different tasks. When the task requirements change, sometimes we only need to update few modules instead of retraining the entire policy. In Chapter 5 we train five modules to catch flying objects using a robotic hand-arm system.

2.2.1.3 Reward Function Design

Reward function is an essential component for any RL application. Sparse reward is straightforward to design, for example, 1 for success grasp and 0 for other situations. However it is notoriously difficult to train complex policies with sparse reward functions. The critical states with meaningful rewards are too meagre in the broad state space, which are difficult for the agent to reach with random and undirected exploration, resulting in the training data with low information density. The most intuitive solution to sparse reward problem is reward shaping (Ng et al., 1999). Unlike typical sparse reward triggered by specific events, dense reward evaluates the state of each time-step during the robot-environment interaction. An effective dense reward function should create a quasi attraction region, where the states closer to the final success have larger reward assignments. Similar to the effect of heuristics in the A* algorithm, the shaped reward functions bias the agent to explore promising directions (Eschmann, 2021). Prior knowledge about solving the task can be implicitly imparted to the agent via shaped reward signals. In Chapter 3, 5 and 6, we construct dense reward functions based on task-specific physical properties, achieving reactive reaching, grasping, catching and in-hand manipulation.

2.2.1.4 Sim-to-Real Transfer

Sim-to-real gap is the major cause for the performance decline when deploying policies trained in simulation to the real-world. There are many sources for the gap, including biased dynamics model, incorrect physical parameters and stochastic real environment. Performance decline when transferring the policies trained in simulation to real-world can be addressed from two aspects: (a) mitigating the discrepancies between simulation and reality via precise models, and (b) increasing the robustness of the policies against variations via domain randomization. For the robot body, accurate models of the actuator dynamics (Sievers et al.,

2022; Tan et al., 2018) and perception feedback like tactile (Wang et al., 2022) and vision (James et al., 2019) are important. Domain randomization samples the simulation parameters from certain distributions, exposing the policy to different environment settings. Parameters feasible for randomizing include robot dynamic parameters such as joint dampings and link weights (Peng et al., 2018), environment physical properties such as surface textures and friction coefficient (Tobin et al., 2017). In Chapter 6, we use both model calibration and domain randomization to transfer the dexterous in-hand manipulation policy to real robot hand.

2.2.2 Learning from Demonstrations

In comparison to learning from scratch, learning from demonstrations (LfD) follows the idea that the robots acquire new skills by imitating the demonstrations from an expert. Not simply repeating the prescribed actions or trajectories, LfD algorithms aim to distill implicit task constraints and learn universal control policies that can generalize to varied scenarios. For tasks where the ideal behaviour are difficult to be either explicitly scripted or described as a reward function, LfD is a promising option for robotic policy learning. Adding real robot demonstrations into training data-set alleviates the dependence on simulation and sim-to-real gap. Compared with learning solely from trial-and-error, leverage of expert demonstrations significantly improves the sample-efficiency (Sun et al., 2017). Khansari-Zadeh and Billard (2011b) modeled the robotic motion as a non-linear autonomous dynamical system and learn the model parameters from few demonstrations. Mohseni-Kabir et al. (2015) integrated LfD with hierarchical task networks, and learned the high-level tasks from even a single demonstration.

The choice of method to gather expert demonstrations depends on the robot and task. Based on the acquisition approach of demonstrations, Ravichandar et al.

(2020) divided LfD algorithms into three categories: Kinesthetic teaching where users physically moves the robot (Kopicki et al., 2016); Teleoperation where users control the robot remotely (Song et al., 2020); and passive observation where the robot learns by observing the demonstrators (Dillmann, 2004). Kinesthetic teaching and teleoperation record the robotic joint space actions or trajectories as demonstration data, while the observation of user usually collects task space data.

In the domain of robotic manipulation, various LfD methods have been applied to different robotic tasks, e.g. pouring (Finn et al., 2016), pick-and-place (Deniša et al., 2015; Mohseni-Kabir et al., 2015), polishing (Kronander et al., 2015) and grasping (Bohg et al., 2013). Compared with aforementioned algorithms and applications, in Chapter 4 we use a low-cost teleoperation method to generate demonstration for adaptive grasping, where only a commercial camera is used to record human hand motion. The finger control policies are encoded as neural networks and trained with supervised learning.

Chapter 3

Learning Reaching and Grasping of Objects

3.1 Introduction

Adaptive and reactive grasping of objects is an essential capability of autonomous robot skills. Yet it is challenging to learn such sensorimotor control that can coordinate hand-finger motions and is robust to disturbances and failure grasps. In this chapter, we proposed a learning scheme to train feedback control policies which coordinate reactive reaching and grasping actions. We formulated geometric metrics and task-orientated quantities for the reward design, and enabled efficient exploration of the grasping policy. Further, to improve the success rate, we deployed key initial states of difficult poses in the training which can induce potential failures due to uncertainties. The extensive simulation validations and benchmarking demonstrated that the learned policy was robust to grasp both static and moving objects stably. Moreover, the policy generated successful failure recoveries within a short time in difficult configurations and was still robust with synthetic noises in the state feedback which were unseen during training.

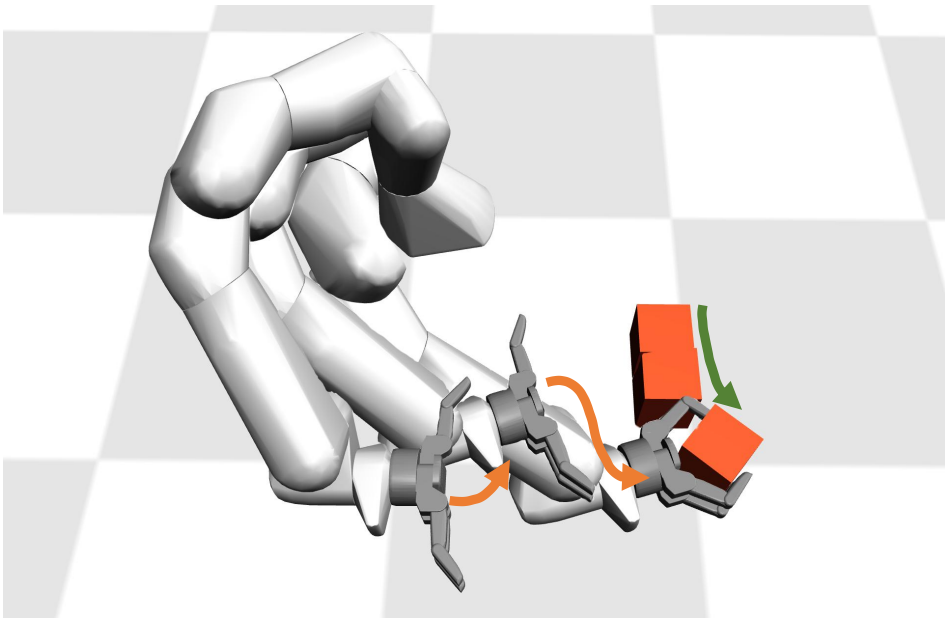


Figure 3.1: Snapshots of grasping a moving object. The emergent hand-finger coordination enables the robot to actively change the approaching direction and grasp the object.

Reactive grasping of static and moving objects and robust recovery from failure grasps are important features, which underpin autonomous grasping and increase the capabilities. They are essential in many robotic manipulation scenarios. However, a coherent control scheme combining both reaching and grasping in a dynamically changing setting remains an area to be researched.

In the conventional approaches, reaching and grasping are separated as planning and control and executed in a cascade manner. Marturi et al. (2019) developed an approach of tracking a moving object and planning grasp posture online, where the grasp motion was determined by a human operator. It is often implemented in an open-loop or partially reactive controlled manner that plan of the combined reaching and grasping motion, which is also time-consuming (Kroemer et al., 2010). Generally, conventional planning based methods have good results in solving either reaching (Suzuki et al., 2015) or grasping problem (Hang et al.,

2017) individually, but the switch between controllers is designed manually. As a next-level performance with increased robustness, reaching, grasping, and even failure recovery need to be addressed *simultaneously within one unified policy.*

Data-driven methods provide a promising option for autonomous control, as they alleviate the requirements of tedious manual design, autonomously explore the operation space, and react in potential corner cases. Recently, vision-based learning methods show prominent performances in the optimization of grasping static objects (J.Bohg et al., 2014; Lu et al., 2018). Compared with classical grasp synthesis, learning-based approaches improve the performance of grasping unknown objects (Kappler et al., 2015; Pinto and Gupta, 2015; Levine et al., 2016; Mahler et al., 2017a). However, it is data hungry to train such models. To collect data either from simulation (Kappler et al., 2015) or self-supervised real robot experiments (Pinto and Gupta, 2015) is time-consuming. Furthermore, sampling and ranking of grasp candidates often takes long computation time, which limits the capability of reactive control (Pinto and Gupta, 2015; Mahler et al., 2017a). In case of failure, the whole pipeline has to reset and repeat another attempt, instead of an online reactive recovery strategy (Kappler et al., 2015).

Recent research in Deep Reinforcement Learning (DRL) has shown promising capabilities of solving continuous control tasks of high-dimensional state space, such as pouring liquid (Hangl et al., 2015), multi-finger grasping (Merzic et al., 2019a), in-hand manipulation (OpenAI et al., 2018), and bipedal locomotion (Yang et al., 2018). In the DRL framework, an agent learns the policy from scratch by maximizing the expected cumulative return from autonomous interactions with environment. In contrast to other Machine Learning techniques, such as supervised and unsupervised learning, no pre-collected training data is required as the agent autonomously generates it by interacting with the environment, and infers the quality of the state-action pairs through reward signals.

This chapter focuses on learning a unified control policy for reactive reaching and grasping of objects in presence of certain uncertainties, which requires synergetic behaviors and fine coordination between the hand and fingers. This unified control policy learns from the proposed framework without the need of handcrafted control rules and has emerged behaviors of adapting to moving objects and objects of different shapes.

While both reaching ability of planning methods and the grasp quality of cutting-edge data-driven methods are important capabilities, this work contributes to *learning a unified grasping policy with coordinated motor skills using high dimensional state feedback in a closed-loop manner*. Most importantly, the policy is capable of *recovering quickly in case of failures*: a problem partially addressed by aforementioned methods. Real-time and reactive failure recoveries can significantly increase robustness and efficiency, and further enable real-world applications, compared to the methods which simply restart the cascaded planning-and-control pipeline.

The main contributions of this chapter are threefold:

(1) A state-action feedback policy of reactive and coordinated reaching and grasping learned from the proposed DRL framework.

(2) A task-orientated reward function with geometric metrics for better policy convergence and the proposed state initialization for learning robust failure recoveries.

(3) The learned feedback policy is robust in both trained and unseen situations: Grasping static or moving objects (see Figure 3.1); Adjusting motions online under sudden changes to object position (see Figure 3.10c); Recovering quickly after failures even in some challenging configurations (see Figure 3.10b and 3.10a).

3.2 Related Work

Visual servoing methods offer a solution to the integration of reaching and grasping, and the real-time action is determined by the current vision input. Such methods have the potential to achieve reactive control (K. Hashimoto, 1993). However, most implementations required large amounts of prior knowledge of the tasks (Hong and Slotine, 1997), or complex control architectures (Namiki et al., 2003) which limits the capability of handling environmental changes, such as the sudden movement of the object.

According to the grasping task, force (Pastor et al., 2011) or tactile sensing (Calandra et al., 2018b; Church et al., 2019) has been used as feedback to achieve close-loop control and generate re-grasping motions. These approaches precisely controlled the hand posture and the applied finger forces for a static object, but neither addressed the problem of grasping moving object, nor the coordination between hand and fingers.

Other research solved the reaching and grasping problem through a combination of trajectory planning with policy learning (Kroemer et al., 2010) or imitation learning (Ratliff et al., 2007). Lampe and Riedmiller (2013) combined the classic visual servoing method with DRL. The proposed system included two parts – long-range controller for reaching and short-range controller for more precise motion of reaching and grasping – which were triggered and switched by handcrafted condition. Compared to the manually partitioned controllers (Lampe and Riedmiller, 2013), our approach learns the policy of reaching and grasping in a holistic manner as one policy.

One major deficiency of learning-based grasp detection via visual input is the long computation time caused by large CNN and individually sampled and ranked grasp candidates (Pinto and Gupta, 2015; Mahler et al., 2017a). Morrison et al. (2018a) overcame this problem by a lightweight network structure that enabled re-

active close-loop control. The learned controller dynamically tracked and grasped novel objects in clutter and achieved a high success rate. However, this approach has not yet considered recovery strategies in case of failures. In our approach, we propose the use of challenging configurations as state initialization that induce failure grasps more often which can train recovery policies more effectively.

3.3 Method

In this section, we present the details of learning a unified policy for reactive grasping as shown in Algorithm 1. First, we introduce the system's framework and the simulation environment. Then, we introduce the state and action space, as well as the reward design. Finally, we introduce the collection of a representative data trajectory.

We formulate the robot control policy as a neural network taking state observation as input and generating control commands. The policy is trained using deep reinforcement learning, more specifically, the proximal policy optimization algorithm (Schulman et al., 2017). Both of the policy π_θ and the value function V_ϕ are parameterized with a fully-connected neural network with two hidden layers.

3.3.1 Control Framework

The system's schematics is shown in Figure 3.3. One training iteration consists of two procedures: data acquisition and update of network weights. Guided by the policy, the agent actively interacts with the environment and gathers the state-action-reward data tuples, and then the weights of actor and critic networks are updated.

We designed a hierarchical control framework, consisting of a high-level and a low-level loop, as shown in Figure 3.2. The sensory feedback processor filters

Algorithm 1: Policy Learning for Dynamic Grasping

Input: Training Epoch N, Probability of Using Special Initial States β, Episode Length T

for $k \in \{1,...,N\}$ **do**

  if $rand(0,1) < \beta$ **then**

    Normal initial state with random object position;

  else

    One special initial state in Figure 3.4;

  for $t \in \{1,...,T\}$ **do**

    Get the current state $\mathcal{S} = \{\boldsymbol{P}, \boldsymbol{q}, \theta, \boldsymbol{\tau}, \boldsymbol{d}, \boldsymbol{V}\}$;

    Run policy π_θ and get the action a_t;

    Execute a_t based on low-level controller in Figure 3.2;

    Compute reward r_t with Equation 3.1;

    Collecting tuple $\{s_t, a_t, r_t\}$;

  $\pi_{\text{old}} \leftarrow \pi_\theta$;

  Update θ by gradient ascent w.r.t. $\hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$;

  Update ϕ by gradient descent w.r.t. $\hat{\mathbb{E}}_t \left[(V(s_t) - \sum_{l=0}^{T-t} \gamma^l r_{t+l})^2\right]$;

the raw states and extracts the sparse object point cloud. The high-level controller generates desired actions at a frequency of 50Hz, such as target hand pose and velocity as well as desired joint angles of all fingers, based on the processed state observation. Given the actions from the high-level, the joint-level proportional-derivative (PD) controller computes all joint torques applied in the physics simulator at 500Hz.

3.3.2 Simulation and Training Setup

For stable, realistic contacts and dynamics, we use the physics simulator MuJoCo (Todorov et al., 2012), where the Barrett Hand is the end-effector that attached on the Franka Emika robot arm. For policy learning, the training objects consist of a standard cube and three household objects, as shown in Figure 3.3.

For the object observation, we used processed and down-sampled surface point

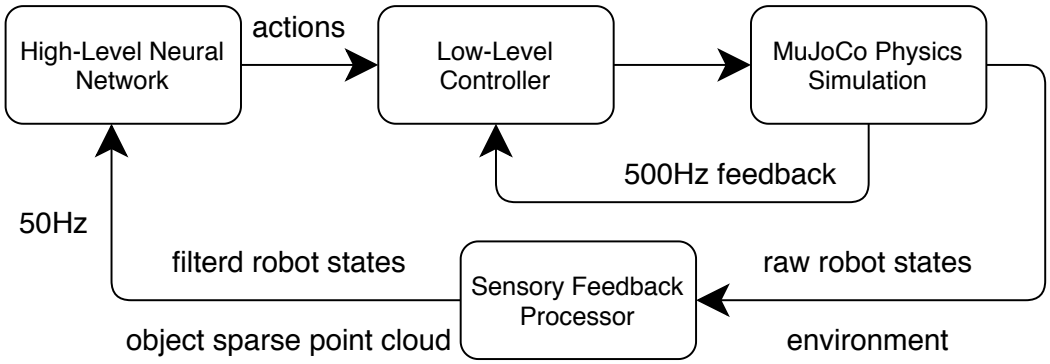


Figure 3.2: Control framework embedding the neural network policy and robot joint-level control.

cloud with simulated synthetic noises (see discussion in Section 3.3.4). Specifically, the object meshes and point clouds were imported from YCB database (Çalli et al., 2015). This ensures the sparse and noisy point cloud is as realistic as possible.

The state space includes three parts: object observation, robot proprioception data, and the hand-object spatial relation. The measured joint torques are used to approximate contact information due to the fact that tactile sensors are difficult to simulate and are not commonly available as hardware neither.

The state vector is $\mathcal{S} = \{\boldsymbol{P}, \boldsymbol{q}, \theta, \boldsymbol{\tau}, \boldsymbol{d}, \boldsymbol{V}\}$, where $\boldsymbol{P}$ refers to the object point cloud positions relative to the hand, representing the object's shape and pose; $\boldsymbol{q}, \theta$ refer to the finger joint positions and hand rotation angle around Z axis; $\boldsymbol{\tau}$ refers to the finger joint torque measurements, implying the contact status of each finger; $\boldsymbol{d}, \boldsymbol{V}$ refer to the distances and unit vectors between the in-hand key points and their nearest object surface points. The in-hand key points locate at the inner surfaces of palm and three fingers. $\boldsymbol{d}, \boldsymbol{V}$ are also related to the reward computation, accelerating the policy convergence. In the computation of $\boldsymbol{d}, \boldsymbol{V}$, we use the processed dense object point cloud for better precision; but in $\boldsymbol{P}$, the number of points is downsampled to 32 for reducing the dimension of the input vector for the neural networks. The object point cloud and in-hand key points are in the

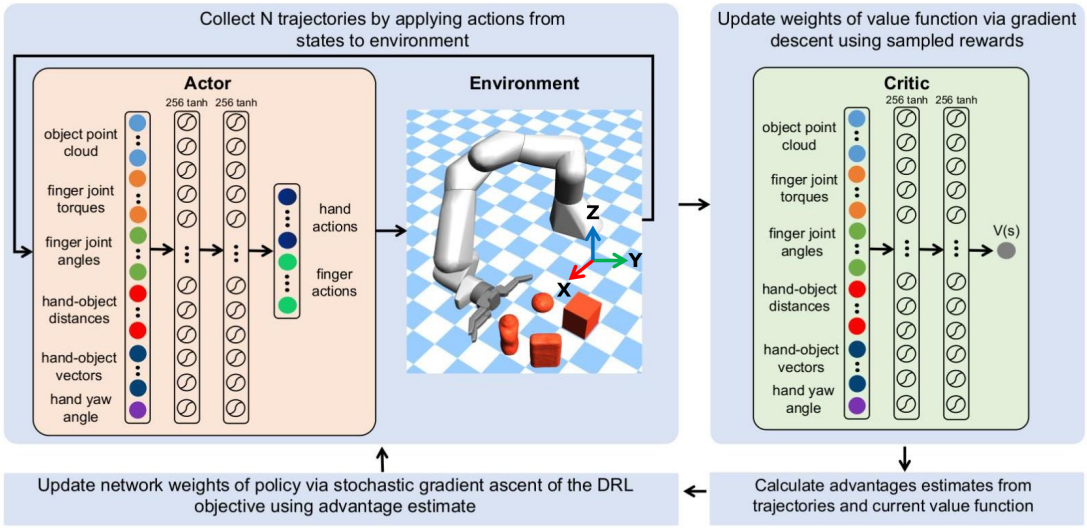
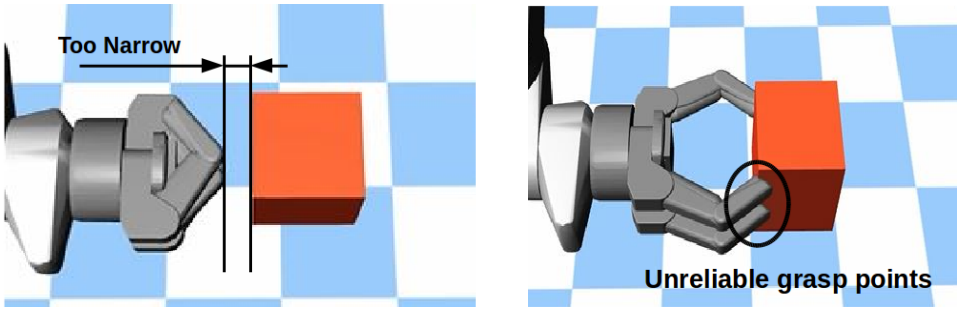


Figure 3.3: Framework of learning motor skills for reactive grasping, detailing state observation and action space.

world frame.

A general policy for 6D grasp pose and grasp points based on the object's shape is beyond the scope of this chapter. As a proof of concept for learning reactive reaching and grasping, we therefore focus on a planar case with restricted degrees of freedom (DoF) of the end-effector moving at the horizontal plane. We constrain the translational motion of end-effector to the XY plane at a height of 0.06 m, and only the yaw rotation around Z axis is allowed. Hence, the action space $\mathbf{A}$ consists of the hand translational velocities, hand rotational velocity around Z axis, and the target finger joint positions. We designed the action space at the end-effector level, which is hence robot independent and off-the-shelf inverse kinematics solvers can be used to map the task-space motions to the joint-space. We also set early termination signal if self-collision and joint limit are violated, and therefore the learned motion is safe and collision-free.

One data acquisition episode starts with randomizing the object position and orientation within the workspace of the robot arm. The robot interacts with the environment and gathers data for 8 seconds. Any self-collision will trigger the



(a) Potential collision. (b) Unreliable grasp pose.

Figure 3.4: Critical and difficult poses included in the state initialization for training policies against failure grasps.

early termination of current episode.

The failure recovery requires synergized motions of fingers and hand. To obtain the collision-free failure recovery skill, apart from the normal training episodes, we introduced two special initial states, as shown in Figure 3.4. These initial states simulate the challenging configurations that the agent can encounter during the grasp: collisions between hand and object, and unreliable grasps.

3.3.3 Reward Function Design

We formulated task-related quantities and proposed the following reward function, which is the weighted linear combination of positive rewards and negative penalties:

$$\mathbf{R} = \sum_{i=1}^{5} \omega_i r_i(\mathcal{P}^{\boldsymbol{o}}, \mathcal{P}^{\boldsymbol{h}}), \tag{3.1}$$

where $[\omega_1, \omega_2, \omega_3, \omega_4, \omega_5] = [2, 1, 1, 2, 0.3]$. The coefficients are determined empirically, and tuned via several rounds of policy training. $\mathcal{P}^{\boldsymbol{o}}, \mathcal{P}^{\boldsymbol{h}}$ represent the object point cloud and the robot in-hand key-points.

The term r_1 minimizes the distances between in-hand surface and the object, encouraging the hand to contact the object as much as possible, which is formulated

as

$$r_1 = \exp(-\sum_{i=1}^{N_\mathrm{h}} \left\| p_i^{\boldsymbol{o}} - p_i^{\boldsymbol{h}} \right\|), \tag{3.2}$$

where $p_i^{\boldsymbol{h}} \in \mathbb{R}^3$ denote the positions of in-hand key points, which distribute on the inner surfaces of fingers and palm. For each in-hand key point $p_i^{\boldsymbol{h}}$, we find the nearest point in object point cloud $p_i^{\boldsymbol{o}}$. $p_i^{\boldsymbol{o}}$ are illustrated in orange in Figure 3.5a. N_h is the number of hand key points. Exponential function is used to regulate the value between $(0, 1]$.

For a firm and secured grasp, the contact surfaces between hand and object should comply with each other; and at each contact point, the normal vectors of both surfaces should well align with each other. Considering computing the surface normal vectors from the point cloud is expensive, as an approximation, we use the vector pointing from in-hand point to its nearest object point as a substitute.

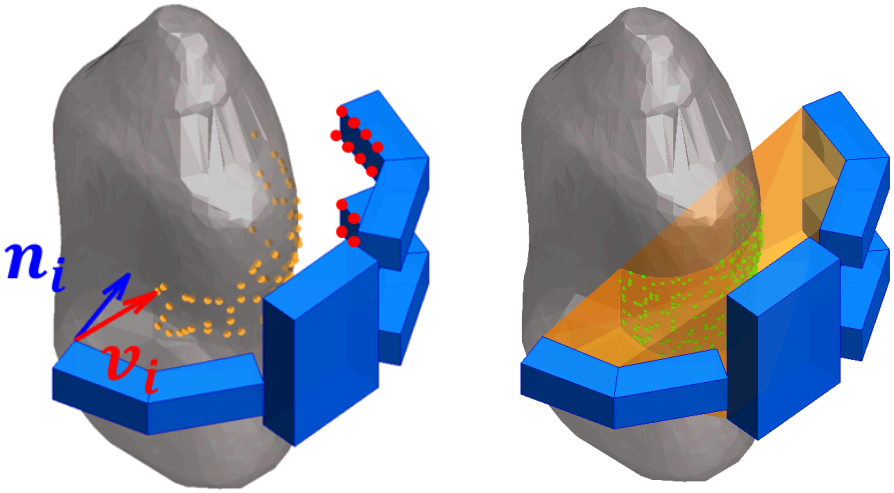
Therefore, the divergence between hand surface normal and such a shortest distance vector can mathematically reflect how well this alignment is. This divergence term orients finger normal to the object surface and is formulated as:

$$r_2 = \frac{1}{N_\mathrm{h}} \sum_{i=1}^{N_\mathrm{h}} \vec{n}_i \cdot \vec{v}_i, \tag{3.3}$$

where $\vec{n}_i, \vec{v}_i$ denote the unit hand surface normal vectors and unit shortest distance vectors, illustrated as blue and red vectors in Figure 3.5a.

In addition to the quantitative measurements of surface and geometry proximity given by Equation 3.2, as well as the contact surface fitting given by Equation 3.3, we still require one more metric to evaluate how well the hand enclose the object, as an index to reflect the *form closure*. A convex polyhedron is formed by all in-hand key points, and the term r_3 is the percentage of object surface points which are enclosed inside the polyhedron:

$$r_3 = \frac{N_\mathrm{in}}{N_\mathrm{obj}}, \tag{3.4}$$



(a) Divergence (b) Polyhedron

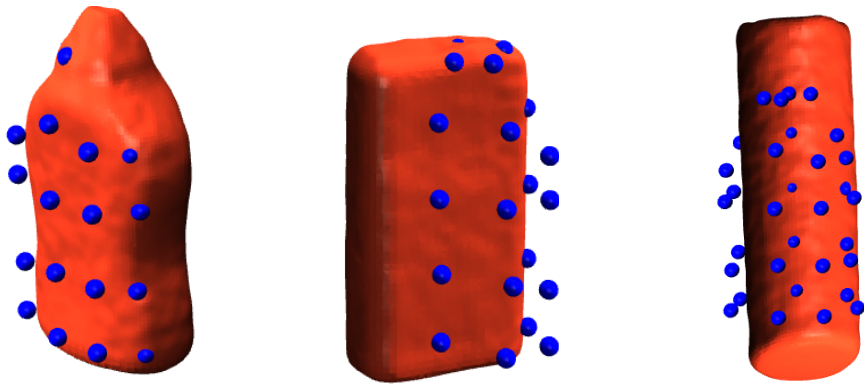
Figure 3.5: (a) Divergence reward term: red dots on hand denote the in-hand key points, and the orange dots denote the object surface points that are closest to them. Blue vector denotes the unit normal vector of in-hand key point, and the red vector denotes unit vector pointing from in-hand key point to its closest object point. (b) Polyhedron reward term: the green dots denote the object surface points that are inside the orange convex polyhedron formed by the hand.

where N_{obj} denotes the total number of object surface points. As shown in Figure 3.5b, the hand polyhedron is displayed in orange, and the enclosed object points are displayed in green. Although this reward term encourages the hand to wrap the object as much as possible, it does not guarantee the form closure criteria since the object shape and contact locations are not considered.

A contact term is added to encourage power (enveloping) grasp, which is more stable than precision (fingertip) grasp:

$$r_4 = \text{mean}\left(\frac{\tau}{\tau_{max}} \cdot e^{-(\frac{\dot{q}}{\dot{q}_{max}})^2}\right), \tag{3.5}$$

where τ_{max} and $\dot{q}_{max}$ denote the upper limit of joint torque and velocity. This term gives more reward when the policy generates larger joint torque and $\dot{q}$ is smaller during the physical contact and grasp of the object.



(a) Bottle (b) Box (c) Chips can

Figure 3.6: Observation of downsampled sparse point clouds with synthetic noises, approximating the features of offset and noises of the real visual data.

A penalty on the translational velocity of the object $\vec{V}_{\text{obj}}$ is introduced to reward smoother grasping behavior and prevent unwanted movements when an object is grasped:

$$r_5 = e^{-\left\|\vec{V}_{\text{obj}}\right\|} - 1. \tag{3.6}$$

3.3.4 Synthetic Visual Noises

The prior-perceptive data are usually of good quality due to high resolution encoders and force transducers, e.g., joint positions and torques. The largest uncertainty comes from the visual feedback: Though the majority of object point cloud can be obtained from multi-view cameras, the visual data have inevitable occlusion (bottom of the objects) and noises in depth images.

Therefore, to evaluate the robustness of the learned policy, we introduced synthetic sensory noises in visual feedback during testing with unseen objects. The noisy observation of the object point cloud is produced as

$$\mathcal{P}^{o}_{\text{noisy}} = \mathcal{P}^{o} + \|\epsilon\|\,\vec{u}, \tag{3.7}$$

where $\vec{u}$ is a unit vector in random directions and ϵ is sampled from a zero-mean Gaussian distribution with standard deviation $\sigma = 1.0\ cm$, which covers

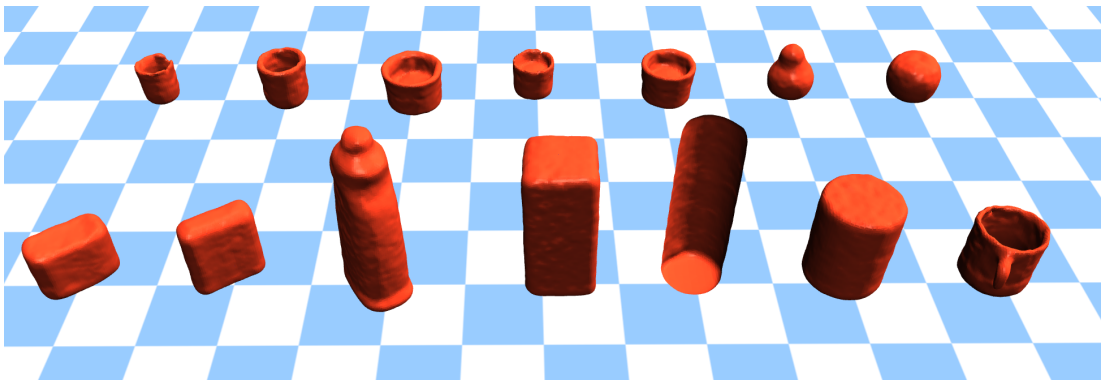


Figure 3.7: Testing objects used in validations that were very different and unseen during training.

the uncertainties of depth images from most common cameras, e.g., the accuracy is 0.5-1.0 cm from RealSense D435 camera at 1 m distance which can be more accurate at a closer distance. Figure 3.6 shows the examples of downsampled point clouds of three different objects with the synthetic noises as in Equation 3.7. Although the policy is trained with noise-free state feedback, the results in Table 3.1 suggest that the policy is robust to the noisy visual input which was not seen before.

3.4 Validation

In this section, we present the results of validations in simulation, and evaluate the capabilities of reaching, grasping and failure recovery with different metrics and tasks. Moreover, we discuss the influence of each reward term and the effectiveness of the initial training states.

As a baseline for benchmarking, we designed a hand-craft grasping policy: The hand keeps approaching the object with open fingers until the distance to the object is below the threshold; Once within proximity, finger torques are applied to generate the grasping motion.

To evaluate the policy's robustness and generalization ability, during the test we

Table 3.1: Success rate of the policy evaluated in different tasks using object point cloud with and without sensor noises.

	Noise-free			Noisy		
	Lift[%]	Shake[%]	Recover[s]	Lift[%]	Shake[%]	Recover[s]
Baseline (Static)	82	64	∅	84	61	∅
Static Object	94	85	∅	85	78	∅
Moving Object	88	79	∅	77	69	∅
Reactive Grasp	83	76	1.59	80	74	1.44
Close Fingers Recovery	92	85	0.87	84	73	0.98
Shallow Grasp Recovery	97	88	0.72	78	66	0.62

used 14 unseen objects with very different sizes, shapes, and masses, as shown in Figure 3.7. We also introduced synthetic noises into object point cloud and evaluated the robustness under imperfect and noisy visual feedback. The statistics of the success rate is shown in Table 3.1.

3.4.1 Evaluation Metrics

We evaluated the performance of learned policy by two metrics: the success rate in lift-and-shake tests, and the reaction time specially for failure recoveries. Five seconds after the hand firstly contacts the object, we started to evaluate the grasping performance. In the lifting test, the robot lifts up the hand to a certain height; and in the shaking test, the robot firstly lifts the hand and then an external disturbance force of $12N$ is applied on the object from random directions. If the object does not fall from the grasp for 10 seconds, then this trial is regarded as a success. We also recorded the time for the policy to recover from the failure and achieve a successful re-grasp, i.e. a repeated grasp. For each type of object, we repeated 10 times as one test scenario.

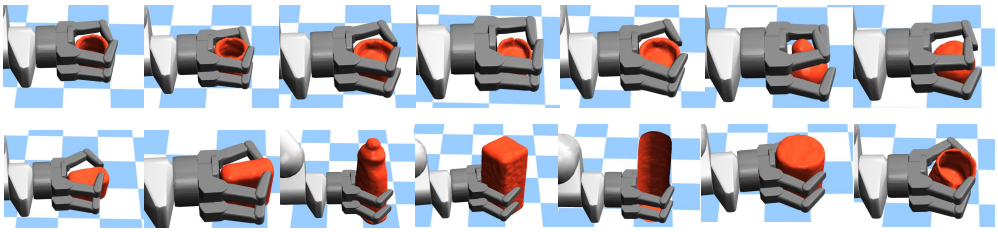


Figure 3.8: Successful grasps of 14 testing objects that were unseen during training.

In Table 3.1 we present the results of different tasks with different metrics. The results indicate that the achieved grasps are robust for lifting and can resist external disturbances. The learned control policy can react to changes rapidly and perform a new grasp in case of failures, even under challenging configurations as shown in Figure 3.4.

3.4.2 Grasp Static and Moving Objects

This test is to reach and grasp the object placed at random positions. The success rate in the second row of Table 3.1 shows that the learned policy can achieve a stable grasp of a static object even under sensory noises.

Figure 3.8 shows 14 representative grasps of unseen objects and Figure 3.9 shows typical control actions of such a task. When the hand is relatively far from the object, the target finger joint positions for next time step are negative, which means the fingers are opening, enlarging the grasping area. The moving velocity of hand reaches the maximum at the beginning and converges to zero in the end, indicating that the agent learns to slow down when the hand approaches the object.

With an effective state-action mapping, the learned policy has good tracking performance and is able to follow and grasp a constantly moving object, as long as the object's velocity and position is within the physical limit of the robot arm. As shown in Figure 3.1, the hand adapts the moving direction based on the

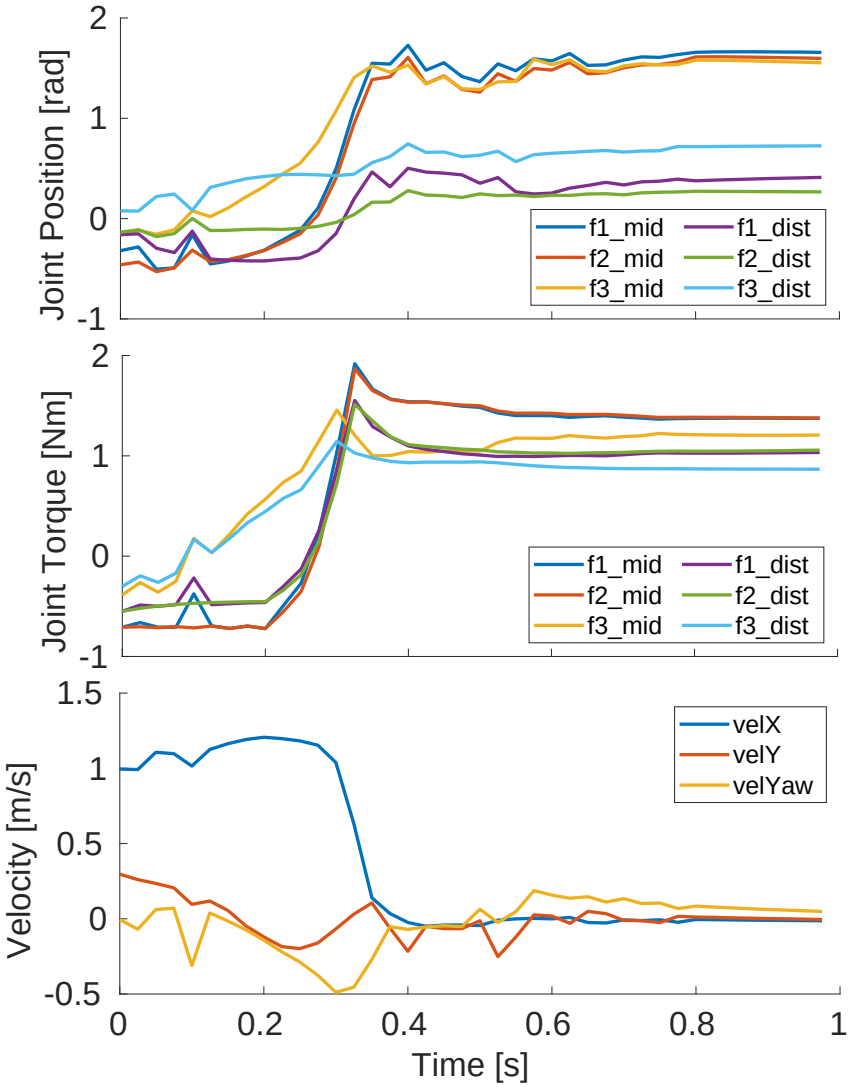
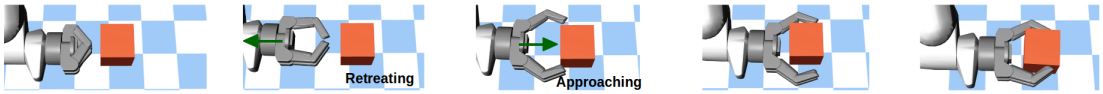


Figure 3.9: Joint positions and torques of fingers and velocities of the hand during a representative grasping of a static object.

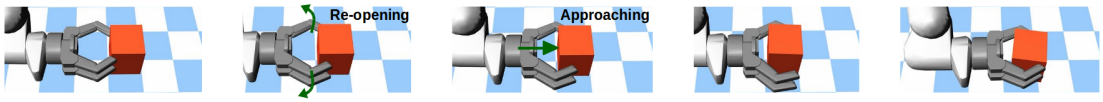
object's motion. Table 3.1 shows that the learned policy can track and achieve secured grasp of moving objects with sensor noises as well.

3.4.3 Reactive Grasp and Failure Recovery

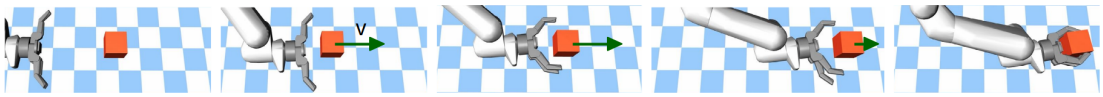
To evaluate the reactiveness under sudden changes of the object's state, we push the object out of the hand when the distance between fingertips and the object is below a threshold at the first time, which leads to two situations: (1) The object escapes from the closure formed by fingers and the palm completely, then the



(a) Coordination and recovery from potential collisions.



(b) Coordinated collision-free re-grasping from an unreliable grasp.



(c) Grasping of a dynamically moving object.

Figure 3.10: Grasping motion generated by the learned policy: (a) Recovering from the initial configuration where the object is too close to the fingers; (b) Re-grasping from the unreliable grasp; (c) Chasing and grasping of a continuously moving object.

fingers would stop closing and re-open; (2) If the object would collide with the fingers and prevent opening, then the hand would need to retrieve backwards to create enough space before re-grasping. In the second situation, even if the object is grasped by the fingertips in motion as an unreliable grasp, the trained agent learns to first release and then re-grasp the object.

Figure 3.10a and Figure 3.10b show the learned recovery motions from failure in two aforementioned challenging configurations. In Figure 3.10c, the object is suddenly pushed to right side at the 2nd snapshot. As a responsive coordination, the fingers re-open and the hand chases the object until executing another grasping attempt within close proximity. According to the results listed in Table 3.1, the learned policy can recover from failures within 2 seconds, which is more efficient than repeating a cascaded planning-and-control pipeline, and it can also achieve secured grasps that pass the lift-and-shake tests with high success rates.

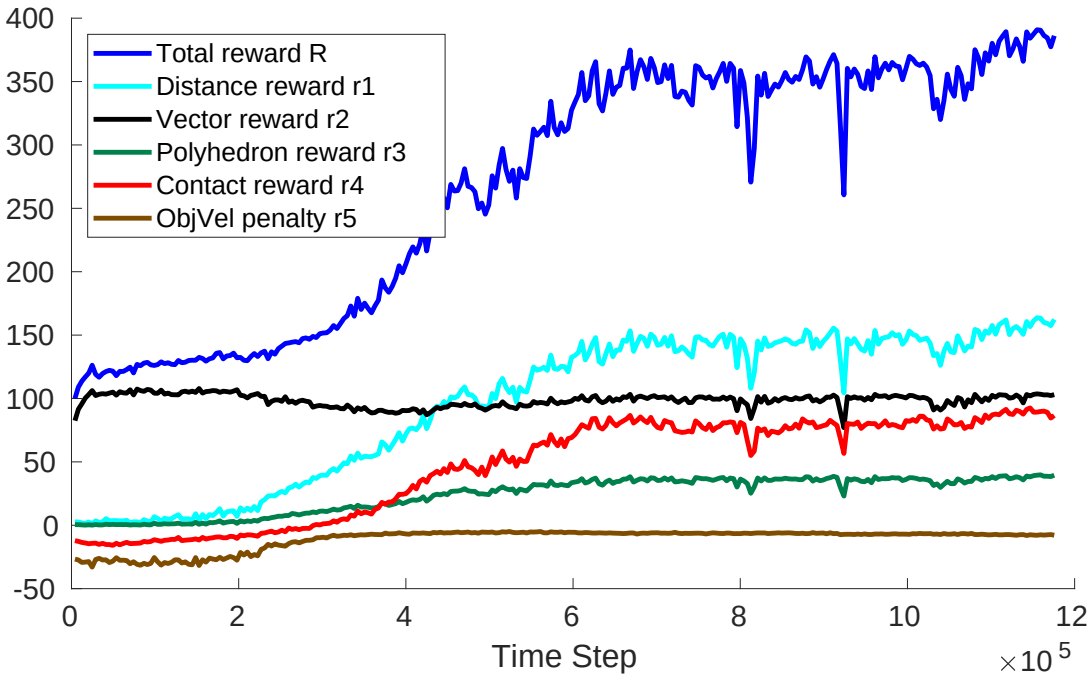


Figure 3.11: Learning curves of the total reward and each term.

3.4.4 Ablation Study

The learning curves shown in Figure 3.11 show the contribution of each reward term during the training. In order to show the necessity and effectiveness of each reward term, we formulated different combinations of the reward function, and compared the resulted learned policies. Table 3.2 shows the capabilities of different policies trained with different reward functions.

The two special initial states in Figure 3.4 are critical for learning the collision-free failure recovery. Without these two initial states, the hand would not be able to learn the retrieving motion and the exteriors of fingers would collide with the object while re-opening.

The r_1 and r_2 terms guide the agent to approach the object. Without r_1, the hand cannot learn to reach the object. r_2 guides the contact surface conformation before and during the grasp, and without it, the agent fails to learn the grasping motion.

Table 3.2: Capability tests with various reward combinations.

Reward combination	Reach	Grasp	Failure Recovery	Lift
No key initial states	✓	✓	✗	✓
No r_1	✗	✗	✗	✗
No r_2	✓	✗	✗	✗
No r_3	✓	✗	✗	✗
No r_4	✓	✗	✗	✗
No r_5	✓	✓	✓	✓
Complete reward set	✓	✓	✓	✓

The r_3 term is important in the early stage of learning after the hand learns to approach the object. It increases reward as the fingers wrap around the object, and also compensates the penalty on object velocity caused by the random actions from policy search. After removing r_3, the hand only learns to stay at a distance from the object with the fingers open. r_4 encourages the joints to generate enough torques and leads to a firm grasp. After removing r_4, instead of contacting and grasping the object, the hand only learns to stay at a closer distance from the object and the fingers fail to form a closure or to have physical contacts.

The penalty on object velocity r_5 encourages gentle grasp motions and prevents the hand from moving the object. The agent can still learn the reaching and grasping without this reward term, but the hand will have undesirable movements after a successful grasp.

3.5 Conclusion

We used model-free reinforcement learning to acquire a unified reactive control policy of reaching and grasping motor skills. The agent explores and optimizes the policy guided by the proposed reward functions, which we mathematically formulated with physical interpretation related to grasping. We applied initial state randomization of the object configuration, and particularly incorporated two challenging initial states to induce failure grasps and train recovery skills.

The training results showed that the learned agent can reach and grasp static and moving objects, and generate a collision-free recovery motion reactively after a failure and re-grasp within 2 seconds, which shows the advantage of the online closed-loop control. Through ablation studies and benchmarking, we analyzed the influence of each reward term and showed our proposed initial states indeed improved the capability and robustness of the learned policy.

In this project we did not consider the object shape in reward design, and the finger contact locations were not explicitly determined. The robotic hand merely approach the object, surround it with fingers and squeeze the object as hard as possible. Therefore the learned grasping policy might perform poorly for small objects or objects with complex shapes.

For future work, we will implement the proposed scheme on a real robot hand and arm, and study the sim2real transfer and investigate new techniques for improving the real-world performance. It is also interesting to explore new sensing technologies, such as tactile sensors and robotic skins, and incorporate multi-modal feedback into the learning framework.

Chapter 4

Learning Adaptive Grasping from Human Demonstrations

4.1 Introduction

The work presented in this chapter is collaborated with Shuaijun Wang, where I participated in the network structure design, network training, comparison study of human and robotic grasping policies, and experiments implementation. This work studied a learning-based approach to learn grasping policies from teleoperated human demonstrations which can achieve adaptive grasping using three different neural network (NN) structures. To transfer human grasping skills effectively, we used multi-sensing state within a sliding time window to learn the state-action mapping. By teleoperating an anthropomorphic robotic hand using human hand tracking, we collected training datasets from representative grasping of various objects, which were used to train grasping policies with three proposed NN structures. The learned policies can grasp objects with varying sizes, shapes, and stiffness. We benchmarked the grasping performance of all policies, and experimental validations showed significant advantages of using the sequential his-

tory states, compared to the instantaneous feedback. Based on the benchmark, we further validated the best NN structure to conduct extensive experiments of grasping hundreds of unseen objects with adaptive motions and grasping forces.

With limited prior-knowledge of the objects, humans can grasp objects with diverse shapes and stiffness and adapt grasping forces during the interaction. In contrast, for robotic grasping, despite the significant progress in object recognition (Gou et al., 2021) and grasp synthesis (Mahler et al., 2017b; Kalashnikov et al., 2018; Morrison et al., 2018b), generating adaptive grasping forces to various objects remains an open problem. Most object grasping uses manually designed rules that apply constant joint torques or fix a threshold of motor current, which is practical for most rigid objects but not suitable or adaptive for various deformable and fragile items, and therefore such simple control policies limit the performance, and can potentially damage the objects.

Robotic grasping is an integrated task of object recognition, motion planning and reactive grasp control. Visual information can be used for selecting contact points and pre-grasp poses, but unknown physical properties such as stiffness cannot be inferred from vision. Therefore, during physical interactions, proprioceptive, force, and/or tactile information is needed to generate appropriate adaptive grasping motions and forces, which are crucial for handling a large variety of daily objects with different stiffness.

Conventionally, separate controllers are designed for objects with different physical properties (Mason et al., 2012; Yoshikawa, 2010), which requires prior-knowledge of target objects, and lacks the adaptability to various object properties. The scope of this work is to study effective learning to extract policies based on limited real demonstration data, and to achieve local feedback control policies of robotic fingers that can adapt grasping forces to various daily objects.

Learning based methods such as deep reinforcement learning (DRL) have promis-

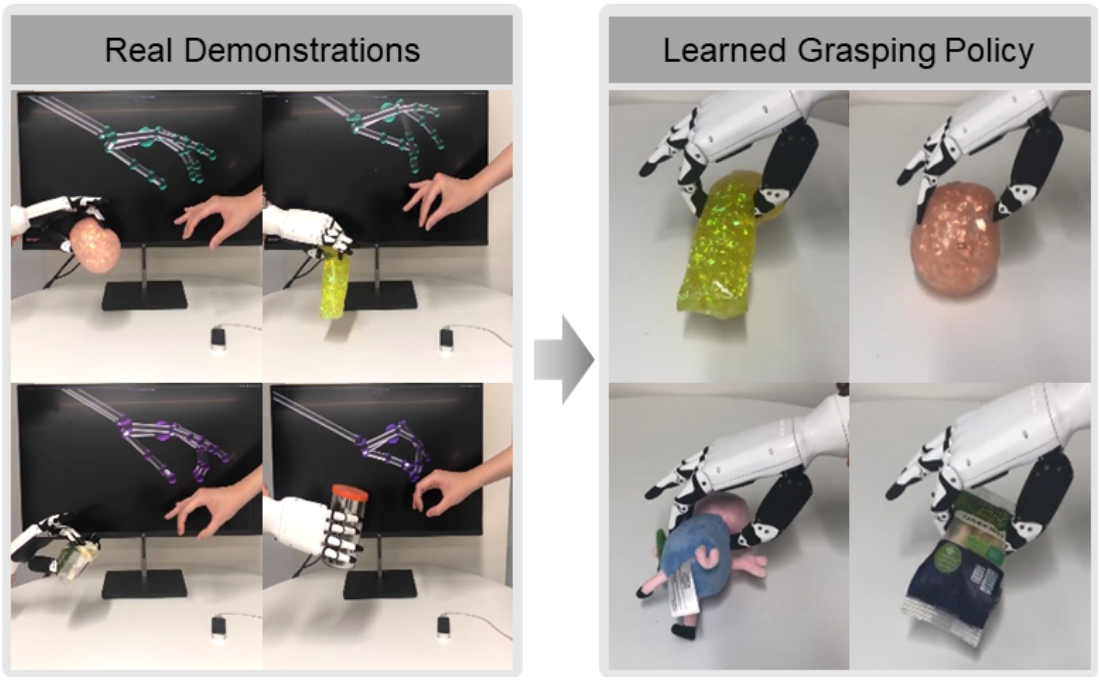


Figure 4.1: Learning from human demonstrations using small real-world data to transfer skills for robotic grasping of various objects.

ing performances in dexterous manipulation. However, learning from scratch without demonstrations requires large training data and long training hours (Andrychowicz et al., 2020), which is not a problem in simulation but problematic with real hardware in the loop. Moreover, the unpredictable emergent behaviours can be unnatural or unsafe on real robots (Rajeswaran et al., 2018). Unlike the traditional problems that can be modeled by rigid body dynamics, for complex physical interactions such as grasping soft deformable objects, there is no high-fidelity and yet computationally efficient simulations to support the trial-error approach of learning in simulation that can be deployed on real systems directly.

Real experimental data is very scarce for grasping various assorted daily objects that are soft, irregular-shaped, deformable, or rigid. To attain sample-efficiency in the cases where only real experimental data are available, an effective approach for the control design is to learn from demonstrations and capture the policies of human grasping skills. Methods for providing human demonstrations include

motion capture (Ficuciello et al., 2019), customized tools Song et al. (2020); Lu et al. (2021) or human signals detection device (Wen et al., 2020; Martin et al., 2004). With appropriate learning frameworks, the real grasping data can be used to train deep neural networks to map multi-modality sensory feedback to the control actions.

Compared to the aforementioned human-robot motion transfer, gesture recognition requires only one depth camera to map human-robot motions, alleviating the complexity in experiment setup and dependency on extra devices. In this work, to collect demonstration data, we used the vision-based human hand tracking to teleoperate a robotic hand to grasp objects with distinct physical properties, as shown in Figure 4.1. The proprioceptive data from the demonstrations are recorded and used to extract the underlying human state-action mapping.

The proposed learned-based grasping can produce adaptive grasping forces solely based on the measured proprioceptive data (joint positions and forces), which can be integrated with many existing motion planning algorithms that generate pre-grasp poses, or used for prosthetic hand control. Given guided hand poses, the learned controller can produce fine motor skills with adaptive grasp motions. In contrast to the analytical approaches, learning-based controllers encode the grasping policies into the weights and biases of neural networks, which are efficient to store and easy to update when new policies are learned by augmenting better demonstrations.

To compare the effectiveness of learning from demonstrations, we studied different combinations of the sensory data feedback and different neural network designs. Specifically, spatiotemporal data with history information and instantaneous data without history information are compared and evaluated. We realized effective learning of reactive motor control of the anthropomorphic robotic fingers from few representative demonstrations, with adaptive grasping motions and forces.

The contributions of this work are as follows:

- A framework of learning adaptive robotic grasping from human demonstrations using an anthropomorphic robotic hand;
- Study and evaluation of the effectiveness of different state feedback for learning the state-action mapping;
- Comparison study of three neural network structures with and without the history input, and their performance validation by grasping objects with various shapes and stiffness.

4.2 Related Work

4.2.1 Analytic methods

Conventional grasping controllers are designed using analytic models based on the feedback of actuator torques and positions (Pfanne et al., 2020; Wimböck et al., 2012; Pfanne et al., 2018; Psomopoulou et al., 2018), but subject to limited adaptive ability, especially in grasping objects with various physical properties.

Pfanne et al. (2020) proposed an object-level impedance controller for dexterous in-hand manipulation capable of handling dynamic changes in the grasp configuration. The proposed algorithm in (Takahashi et al., 2008) switches between force and position control according to the external force. Romano et al. introduced a framework which divided the grasping process into discrete phases based on the tactile information (Romano et al., 2011). Most of this paradigm of solutions are based on human ingenuity and handcraft of control rules (Kaboli et al., 2016).

4.2.2 Learning-based methods

Many recent work on autonomous grasping used learning-based approaches (Bohg et al., 2014; Merzic et al., 2019c; Yuan et al., 2017b; Kopicki et al., 2016). Lee et al. (2019) used multimodal sensory fusion, including visual, kinesthetic and ontological information, to achieve decision making and continuous control based on self-supervised learning. Shahid et al. (2020) leveraged deep reinforcement learning to train a unified policy for reaching, grasping, and lifting the objects in the simulation. The work in (Deng et al., 2020) used spatio-temporal information and an long short-term memory (LSTM) network to learn the classification of object materials and grasping phases. The high-dimensional saptio-temporal human grasping data can be embedded into a lower-dimensional space for modeling and recognition of grasping actions (Romero et al., 2010), which demonstrated the effectiveness of saptio-temporal data in learning grasping policies. Also, the history data is effective in slippage detection (Deng et al., 2020; Zapata-Impata et al., 2019).

4.2.3 Learning from human demonstrations

Learning the robotic grasping from human demonstrations has been widely researched recently (Ravichandar et al., 2020; Lin et al., 2012b). Misimi et al. (2018a) used the support vector regression (SVR) to formulate the grasping controller with both vision and tactile feedback, in order to grasp the compliant food objects. Ekvall and Kragić (2005) used sequential grasping data to achieve grasp recognition, which shows the potential of history proprioceptive data to realize autonomous grasping. Most of the prior works using human demonstrations focused on the grasp planning that generates the pre-grasp pose and contact points for the target object (De Coninck et al., 2020; Misimi et al., 2018a; Song et al., 2020). While little attention has been paid to the finger control and the grasp

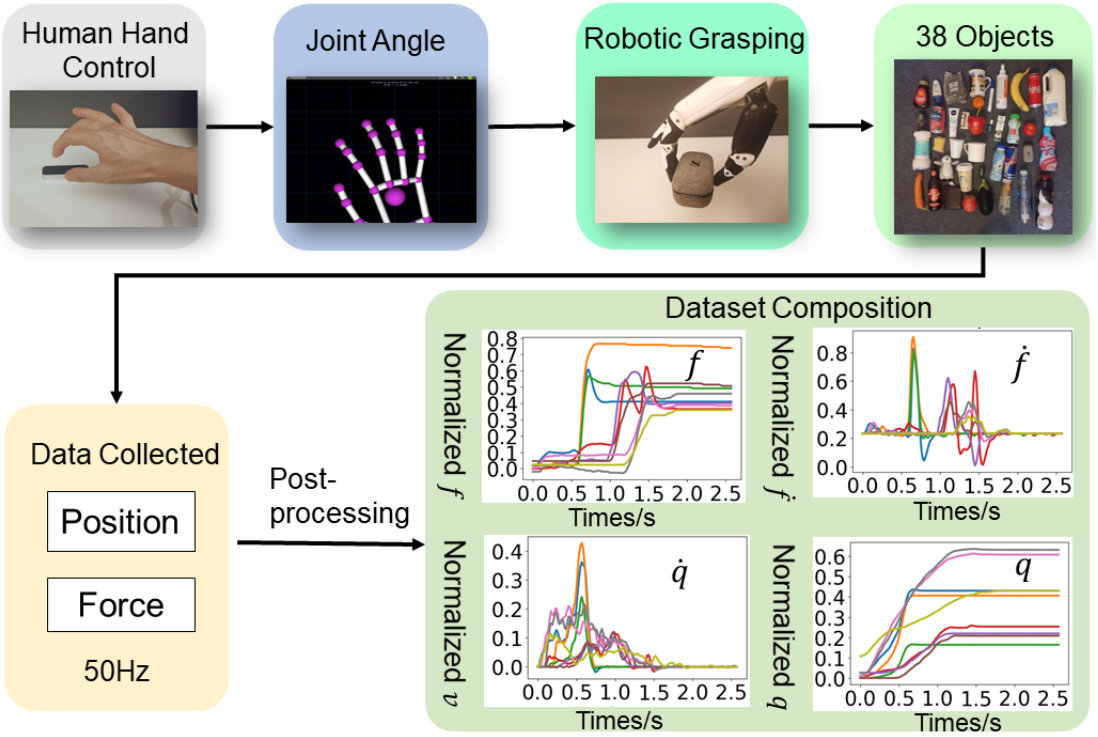


Figure 4.2: Dataset collection for teleoperated learning from demonstrations.

execution. In this work, we utilize the history data of multi-sensing feedback in the framework of learning from human demonstrations, to directly learn the policy of continuous, adaptive grasping of a wide range of objects.

4.3 Method

4.3.1 Human demonstration data

This section presents the tele-operation system and the collection of human demonstration data, and the composition of the dataset that used for training.

4.3.1.1 Teleoperation and data collection

We use the leap motion hand tracking system to detect real-time human fingers motion. As shown in Figure 4.2, the camera is used to detect the real-time human finger joint position, then the corresponding robotic finger actuator positions

are determined by kinematic mapping. The human demonstrator is trained to teleoperate the robotic hand before data gathering, to make sure that the mapped grasping motion is natural. During the grasping, the robot proprioception data including actuators forces, positions and their first order derivatives are recorded as the training data. In this work, the thumb, index and middle fingers are used for grasping a single object.

At the beginning of each demonstration, given a randomly placed object, the robot hand is held and placed at a proper pre-grasp pose by the human operator. During the grasp, the pose of the robotic hand remains fixed. Then, the robotic fingers are teleoperated by the demonstrator to grasp the object, with adaptation to the object's physical properties. Note that the human hand is merely providing the grasping motion, without grasping any real object.

The teleoperation approach of providing demonstrations can mitigate the discrepancies between robot hand and human hand, because the operator can learn to adapt motor skills such that the reflected skills at the robot side are feasible and suitable for the robot itself, so as to ensure successful grasping. Though the robot hand has less degrees of freedom compared to human hand, our trials show that the robotic three-finger grasping motion is very similar to that of humans, since the robot hand has a similar size and morphology to the human hand.

4.3.1.2 Dataset composition

The training data consists of the forces and positions of the linear actuators that drive the fingers during the grasping motion. 38 objects with variations in shape, size and stiffness are used to provide the grasp demonstrations. For each object, we repeat the grasping attempts for 5 times with different object pose, and the corresponding grasping pose is determined by the human demonstrator. Note that in this project we focus on the generation of grasping forces, and therefore

we assume the grasping poses and finger contact locations accord with human intuition. To fully take advantage of the data, the K-fold cross-validation method is applied in the training, where K is empirically set to 10. We randomly choose 5 objects as the validation dataset and the rest is the training dataset.

We define q and f as the measurements of positions and forces of the linear actuators, and hereby their computed derivatives are $\dot{q}$, $\dot{f}$ respectively. The linear actuator drives the intermediate linkage mechanism, which then enables the revolute finger joint to rotate. Note that the force sensor is placed outside of the drive chain so the measured forces are directly applied to the finger joints via the linkage mechanism.

During the online grasping, the q and f are recorded at 50Hz and post-processed by lowpass filters (first-order with cutoff frequency of 10 Hz).

4.3.2 Framework of learning from grasping demonstrations

The policy of adaptive grasping which maps the robot proprioception to the control signals, can be represented as

$$\dot{q}_{\mathrm{d}} = \pi(s_t), \tag{4.1}$$

where $\dot{q}_{\mathrm{d}}$ is the desired velocities of the finger linear actuators, which are monotonic with the finger joint velocities. s_t denotes the vector of the state feedback at t time (see more in Section 4.3.3); π denotes the policy that maps the state feedback to the desired actions, which is represented by a neural network trained from human demonstration data.

Figure 4.3 demonstrates the algorithm framework which consists of three modules: data generation, offline training, and online grasping. The training dataset is obtained from human demonstrations in the data generation module. Given the training data, the grasping skills from human demonstrations are transferred to

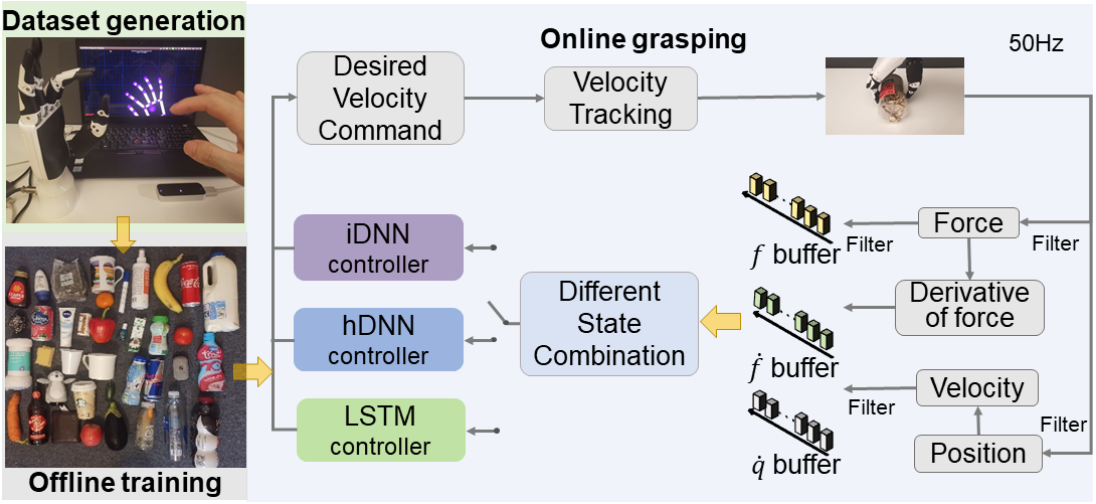


Figure 4.3: The proposed framework include three parts: data generation module, offline training module and online grasping module.

the NN-based controllers via supervised learning in the offline training module.

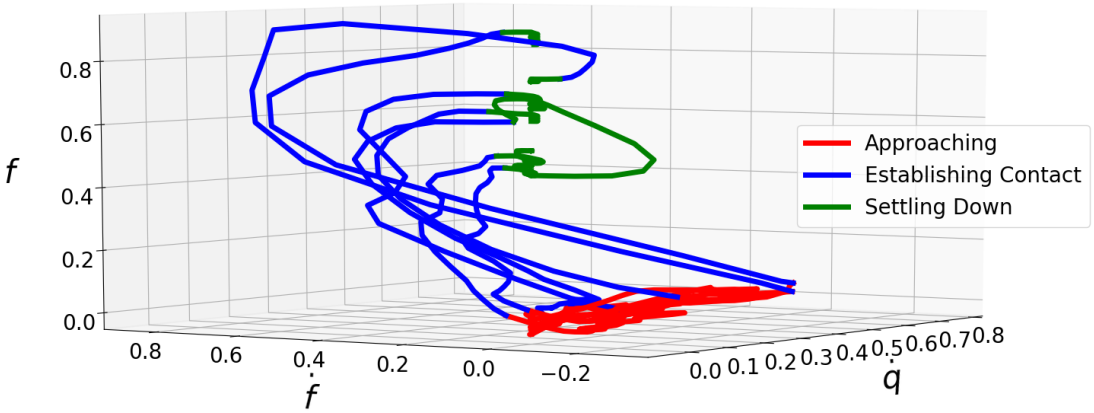
We propose and evaluate three controllers with different structures (see more in Section 4.3.4). Once trained, the learned NN-based policies are used in the online grasping module as a feedback controller, where the measured robot proprioceptive data are post-processed and fed as input. As shown in Figure 4.3, the outputs are the desired velocity commands for the finger actuators.

4.3.3 Data analysis and state combination selection

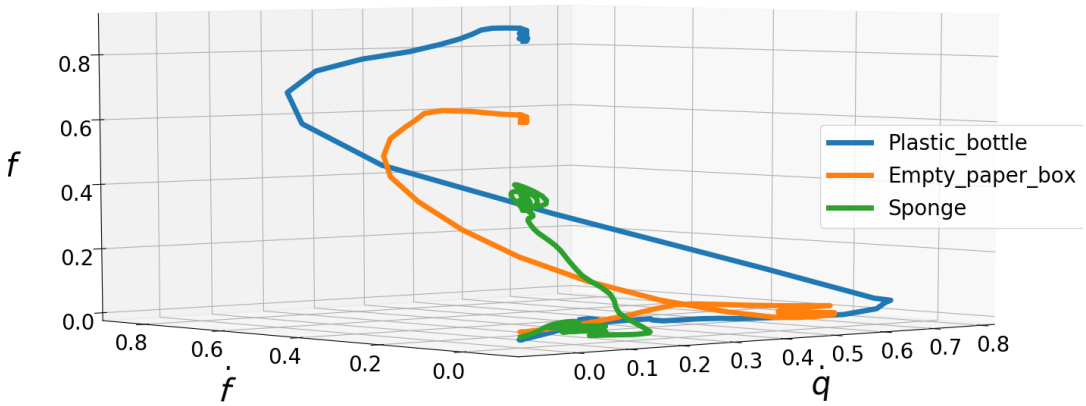
In this section, we analyse the characteristics of robot proprioceptive data during grasping, in order to provide theoretical support for the selection of effective input states combination for learning the policy.

4.3.3.1 Data analysis of the grasping process

During the grasping, the state vector $[\dot{q}, f, \dot{f}]$ can be used to distinguish grasping phases. The actuator position q is less indicative, because it can not demonstrate the phase of establishing contact, and the equilibrium of q depends on



(a) Grasping phases: different colors represent different phases.



(b) Grasping of objects of different stiffness.

Figure 4.4: Normalized feedback states from the index finger during the grasping of three representative objects.

the shape/size of the object which is unnecessary for our controller. When a grasp reaches the equilibrium, $\dot{q}$ and $\dot{f}$ converge to zero, and f converges to a settled value. $\dot{q}, \dot{f}$ can encode the information of object stiffness during the early stage of contact. f indicates the grasping force, and $f, \dot{f}$ can reflect the contact transitions. Therefore, the tuple of state vector $[\dot{q},\ f,\ \dot{f}]$ is used for the policy learning.

Figure 4.4 shows representative trajectories of $[\dot{q},\ f,\ \dot{f}]$ from the index finger during the grasping of different objects. In this state space, grasping different objects shows different trajectories, where the process can be categorized into 3

Table 4.1: Clustering analysis on different feedback states.

NO. of clusters	$f, \dot{f}$	$\dot{q}, f$	$\dot{q}, \dot{f}$	$\dot{q}, f, \dot{f}$	$q, \dot{q}, f, \dot{f}$
10	0.0058	0.0060	0.0049	**0.0061**	0.0051
12	0.0058	0.0086	0.0112	**0.0115**	0.0051

sequential phases as: approaching, establishing contact, and settling down.

- *Approaching:* $\dot{f}$, f are around 0, and $\dot{q}$ becomes non-zero due to the movement.

- *Establishing contact:* $\dot{f}$, $\dot{q}$, f are evolving during this transition. $\dot{f}$ and $\dot{q}$ will rise from 0, reach peaks and then drop to 0. f will rise and reach a constant. Objects with different sizes and stiffness will result in distinct and different trajectories in the state space.

- *Settling down:* f maintains at a constant value, while $\dot{f}$, $\dot{q}$ settles around 0.

4.3.3.2 Clustering analysis of feedback states

To select the most effective combination of input states that can differentiate different physical interaction phases with different objects, the complete training dataset is partitioned into a number of clusters (10 and 12 are used here based on the scale of the dataset) by the unsupervised clustering method *K-means* (Forgy, 1965). Every data-point for clustering is the temporal state feedback within a fixed time-window $s_{t-H:t}$, here H refers to the size of time window. The clustering results are evaluated by *Dunn index* (Dunn†, 1974), i.e. a larger number indicates the more distinguishable clusters. Table 4.1 shows that the state vector $s_t = [\dot{q}, f, \dot{f}]$ has the highest Dunn indexes and captures at least 10 to 12 most distinct phases in temporal sensory measurements.

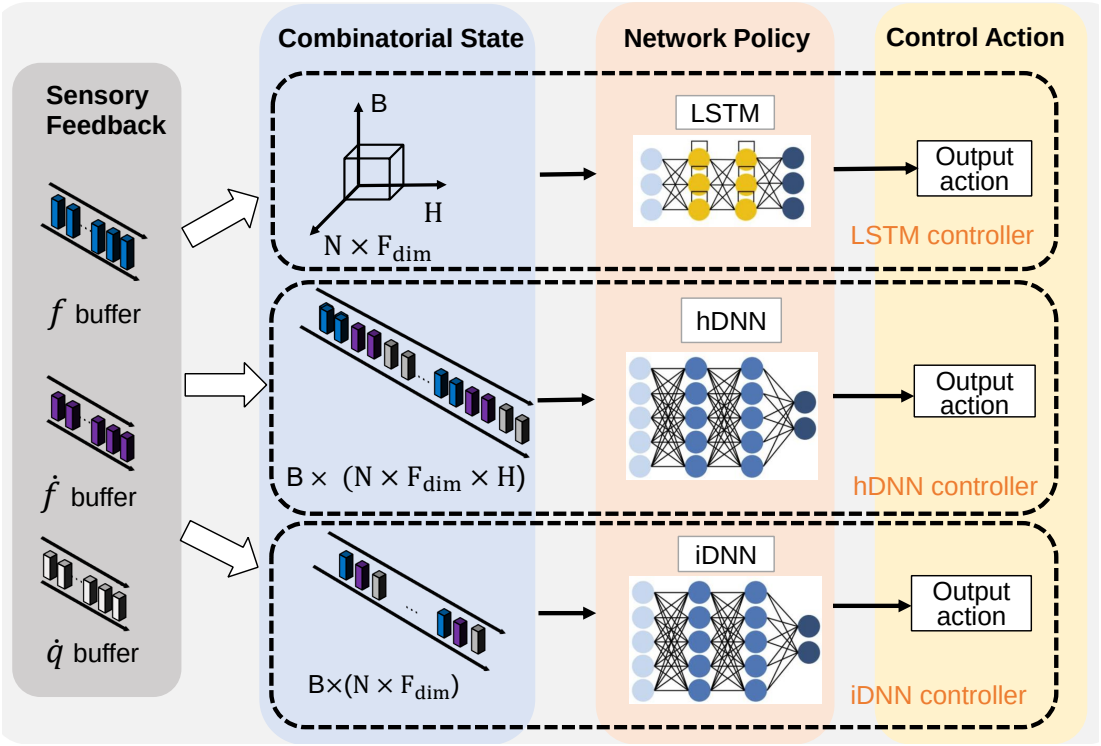


Figure 4.5: Three network structures and their feedback states.

4.3.3.3 Selection of state input

Based on the grasping data analysis and clustering analysis above, the state combination $s_t = [\dot{q}, f, \dot{f}]$ can differentiate object properties and characterize the grasping phases. To further evaluate the effectiveness of history data, we designed two types of state input for policy learning: (1) the instantaneous state vector s_t at the current timestep, and (2) the temporal state tuple $s_{t-H:t}$ using history data within a fixed time window.

4.3.4 Design of grasping controllers

As for the effective skill transfer, we used supervised learning, which is computationally efficient to train the grasping policy π directly with the demonstration data. To focus on the evaluation of history information, and to alleviate any influence to the results introduced by the network structures, we used the simplest network structure – fully connected neural network – as the structure of iDNN

(DNN with instantaneous information) using s_t, and hDNN (DNN with history information) using $s_{t-H:t}$. LSTM network is widely used in processing sequential data. Hence we also designed an LSTM-based controller using history input $s_{t-H:t}$. The time window used in this work is 0.4s, which can cover the transition phase of contact in most robotic grasping. Moreover, an over-long history will include unneeded information and increase computation, while a too-short history is not enough to distinguish different grasping phases. Empirically, we choose this parameter based on the empirical knowledge of the average contact phase during most tasks. The detailed structures and state inputs of three controllers are shown in Figure 4.5.

4.3.4.1 iDNN

The input is the instantaneous state $s_t = [\dot{q}, f, \dot{f}]$, with dimension $I_1 = B \times (N \times F_{dim})$, where B denotes the batch size. N denotes the number of finger joints (also degree of freedom here) and F_{dim} denotes the state dimension of each joint. The output is the finger action vector with dimension $N \times 1$. The network has two fully connected hidden layers.

4.3.4.2 hDNN

The input is the temporal state tuple $s_{t-H:t}$ including the *history* state within a time window, with dimension $I_2 = B \times (N \times F_{dim} \times H)$, where H denotes the size of time window. Except the input size, the rest of the network structure is the same with iDNN, with two fully connected hidden layers and one output layer.

4.3.4.3 LSTM

Recurrent neural network (RNN) is applied to construct the LSTM grasping controller. The input state with dimension $I_3 = B \times (N \times F_{dim}) \times H$ is fed into two LSTM layers and one fully connected output layer.

4.3.5 Policy learning

With the collected training dataset, the aforementioned controllers are trained via supervised learning. The loss function is defined as mean squared errors between the ground truth and the output of control actions ($\dot{q}_{\mathrm{d}}$) plus L2 regularization:

$$loss = \sum_{i=1}^{n} (y_i - y_i^d)^2 / n + \lambda \sum_{i=1}^{k} \omega_i^2, \tag{4.2}$$

where y_i is the ground truth of finger actions at timestep i, and y_i^d denote the network's output actions. n is the number of samples. ω_i is the weight of the network, and λ is 0.001.

To evaluate the robustness of successful grasps against perturbations, we define γ as the force metric which is the resultant force generated by all the actuators of the fingers:

$$\gamma = \sqrt{\sum_{i=1}^{n} f_i^2}, \tag{4.3}$$

where f_i is the measured force of each actuator, n is the number of fingers. In this work, we only consider the grasping of general daily objects, excluding fragile objects like empty egg shells. Hence as long as the grasping forces do not damage the object, this metric can indicate the resistance against external disturbances. In the following experiments, the force metric γ is used to measure the robustness of each grasp of different objects. For robotic grasping, reward terms in Chapter 3 consider both grasping pose (Equation 3.2, 3.3) and grasping forces (Equation 3.5). In this chapter, since we focus on the generation of adaptive grasping forces, the grasping pose is determined by human demonstrator and assumed to be optimal. Hence, Equation 4.3 focuses on the resultant grasping forces.

Figure 4.6: Unseen objects with various physical properties for testing.

4.4 Experimental Results and Analysis

4.4.1 Hardware setup

An anthropomorphic multi-fingered hand, Inspire Robot Hand, is used for experiments. The hand has six degrees of freedom, two for the thumb and one for each of the remaining fingers. Each degree of freedom is driven by a linear actuator with a one-dimensional force sensor and a position sensor mounted at the output of the drive chain, detecting the real-time force and position of the motor. We have a customized control loop running at 50Hz to update the position references of an internal position-control loop to achieve velocity control.

4.4.2 Grasping experiments with unseen objects

To evaluate the robustness and generalization of the learned controllers, we conducted grasping experiments on 100 unseen objects with various sizes, shapes and stiffness, as shown in Figure 4.6. The wrist of robotic hand is positioned by

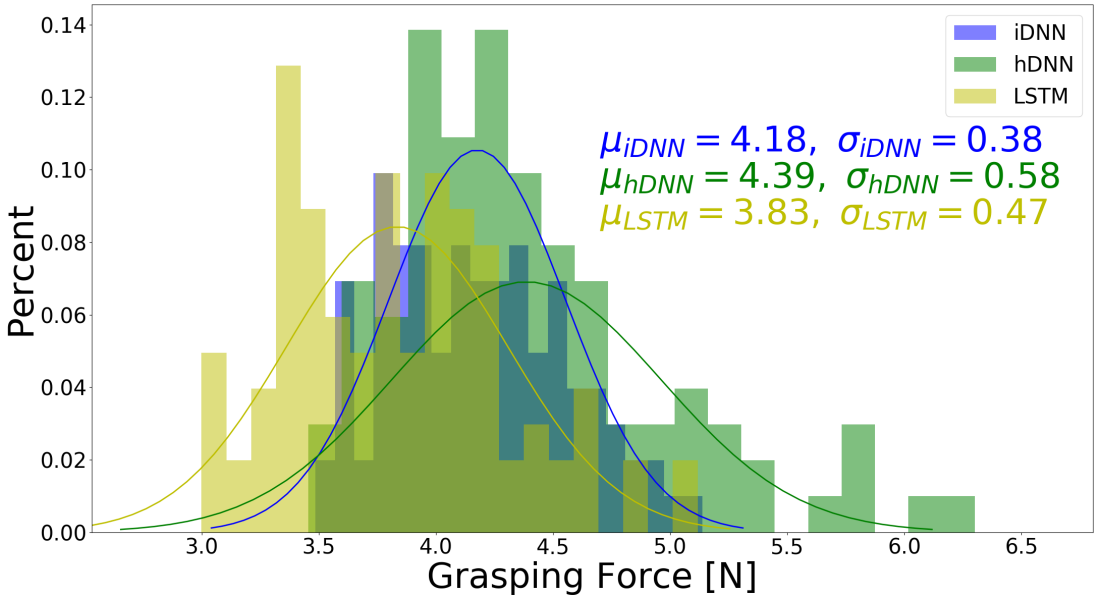


Figure 4.7: Distribution of grasping forces from different controllers over the testing objects, including the mean and standard deviation.

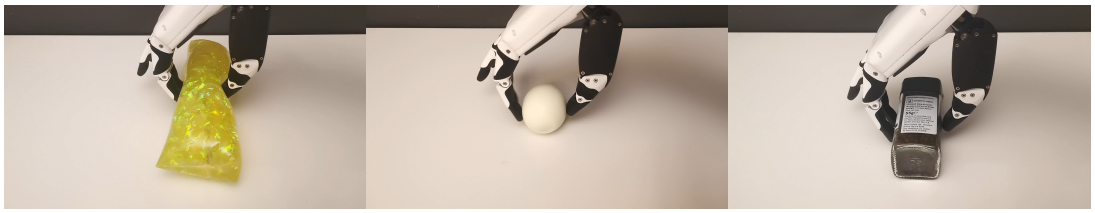
Table 4.2: Success rates for grasping experiments and effective state combinations for policy learning.

	iDNN	hDNN	LSTM
Success rates	93%	94%	88%
Valid combinations	None	$\dot{q}$, f+$\dot{q}$, $\dot{f}$+$\dot{q}$, f+$\dot{f}$+$\dot{q}$	f, f+$\dot{f}$, f+$\dot{q}$, f+$\dot{f}$+$\dot{q}$

an operator, who selects the grasping pose, and the rest of in-hand grasping is executed by the learned controllers. The success rates of three controllers are in Table 4.2, and hDNN controller has the best performance with the success rate of 94%.

Figure 4.7 shows the distribution of force metric γ over the grasping experiments of unseen objects with three proposed controllers. Despite of a high success rate, the iDNN controller has a smaller standard deviation of γ values, indicating a smaller range of adaptation of forces to different objects. LSTM controller has

(a) Small toy (b) Plastic model (c) Bottle



(d) Soft toy (e) Peeled boiled egg (f) Pepper bottle

Figure 4.8: Grasping objects with increasing sizes (top row) and increasing stiffness (bottom row).

higher standard deviation and better adaptability to objects with different stiffness, but has the lowest average γ value, which is less robust against uncertainties. Compared with iDNN and LSTM controllers, the hDNN controller is more versatile – on average, it generates larger grasping forces and also has a wide range of force adaptation.

4.4.3 Comparison study between three controllers

This section presents the results from grasping 6 distinct and representative objects to demonstrate the performances of learned controllers, as shown in Figure 4.8.

4.4.3.1 Grasping objects with similar stiffness and different sizes

As shown in Figure 4.9 (a), hDNN and LSTM controllers generate distinct output actions at different grasping phases, but iDNN controller generates relatively constant output during the grasping. As shown in Figure 4.9 (b), compared with iDNN, the time-series action curves of hDNN and LSTM are more distinct for

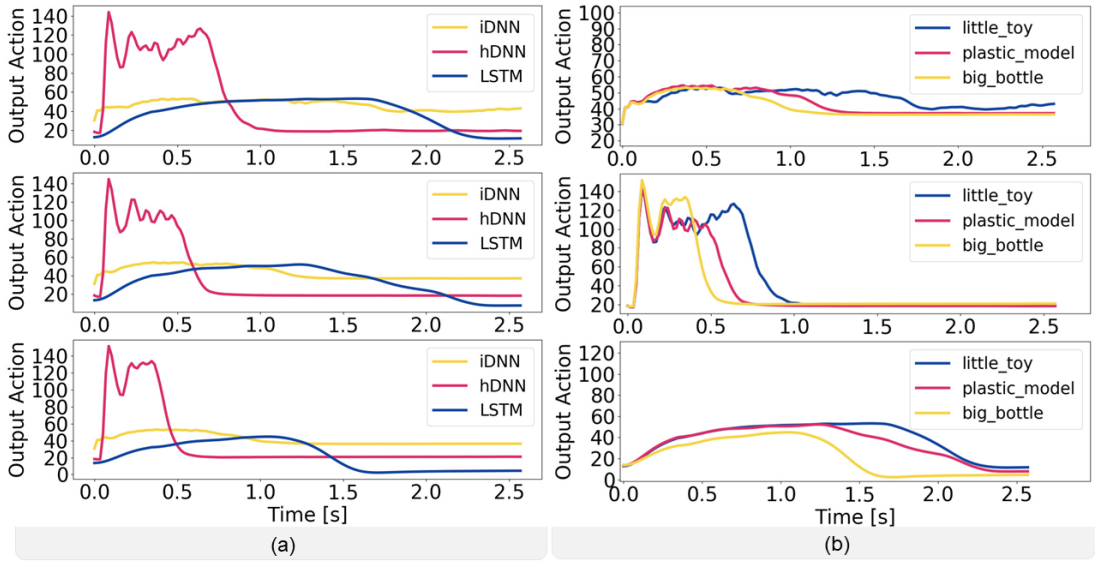


Figure 4.9: Comparison of three controllers. (a) Output actions (actuator velocity of index finger) for grasping a small toy, a plastic model, and a bottle respectively (top to bottom); (b) The same data sorted by iDNN, hDNN, LSTM controllers respectively (top to bottom).

different objects shown in Figure 4.8(a) - (c), indicating that the history data contributes to the disambiguation of object sizes, and can potentially lead to better adaptability.

4.4.3.2 Grasping objects with similar sizes and different stiffness

For grasping objects shown in Figure 4.8(d) - (f), though the finger joint configuration is similar once settled, the dynamic transitions are very different during the contact. With $s_{t-H:t} = [\dot{q},\ f,\ \dot{f}]$ capturing such transitional features within a time window, Figure 4.10 (b) shows that hDNN and LSTM controllers have distinct force adaptations, e.g. 3 different stable grasping forces for 3 different objects. The LSTM controller has smaller grasping forces γ than the other two controllers in general as can be seen in Figure 4.10 (a), resulting in softer grasps, which is consistent with the statistical results in Figure 4.7.

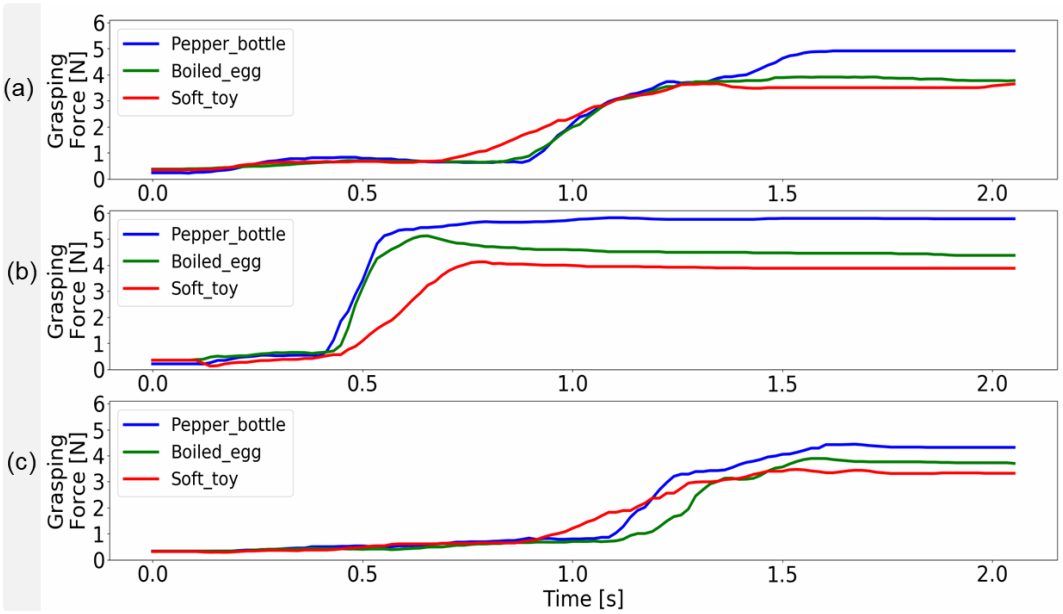


Figure 4.10: Comparison of three controllers. (a) Grasping forces during grasping a soft toy, a peeled boiled egg, and a pepper bottle respectively (top to bottom). (b) The same data sorted by iDNN, hDNN, and LSTM controllers respectively (top to bottom), and the transitions are in yellow highlight.

4.4.4 Ablation study

To evaluate the effect of each feedback as the controller input, the ablation study has been conducted. Figure 4.11 shows the profiles of output actions during grasping the pepper bottle using the three proposed controllers trained with all the possible state combinations. Empirically, the controllers that generate distinct output actions at different grasping stages are regarded as adaptive and reactive, and the corresponding state combinations are effective. As demonstrated in the Figure 4.11(a), none of the iDNN controllers are adaptive, generating relatively constant actions during the grasping; While in Figure 4.11(b) and (c), some state combinations are effective in training adaptive hDNN and LSTM controllers. The effective state combinations for three controllers are listed in Table 4.2.

The iDNN controller using instantaneous feedback s_t without the history data merely generated constant finger actions, which suggests that s_t does not capture

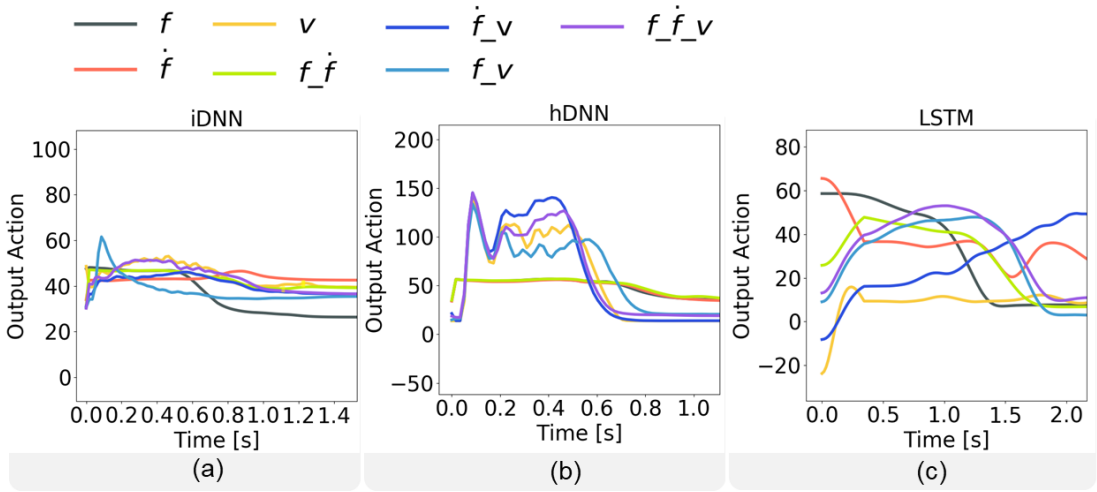


Figure 4.11: Output velocity of index finger during grasping the pepper bottle using three controllers trained with different state inputs.

sufficient information for encoding human grasp skills. Contrarily, with history information $s_{t-H:t}$, both hDNN and LSTM controllers achieve more human-like grasping, though the effective input combinations vary as shown in Table 4.2.

Further, Figure 4.12 compares grasping forces over rigid and deformable objects, suggesting that the combination of multi-sensory data is more effective in learning adaptive grasping. For hDNN and LSTM controllers, the learned policies using the complete state combination $[\dot{q}, f, \dot{f}]$ can generate the most distinct grasping forces for rigid and soft objects.

4.4.5 Comparison with the baseline controller

To evaluate the effectiveness and adaptiveness of the learned controllers, we compared them with a pre-programmed baseline controller, which generates constant joint velocities and has a threshold on the grasping force computed as in Equation 4.3. Once the grasping force exceeds the threshold, fingers will stop moving and maintain the current joint positions. The force threshold is pre-defined and fixed during the experiments. A paper card is chosen as the target object so that

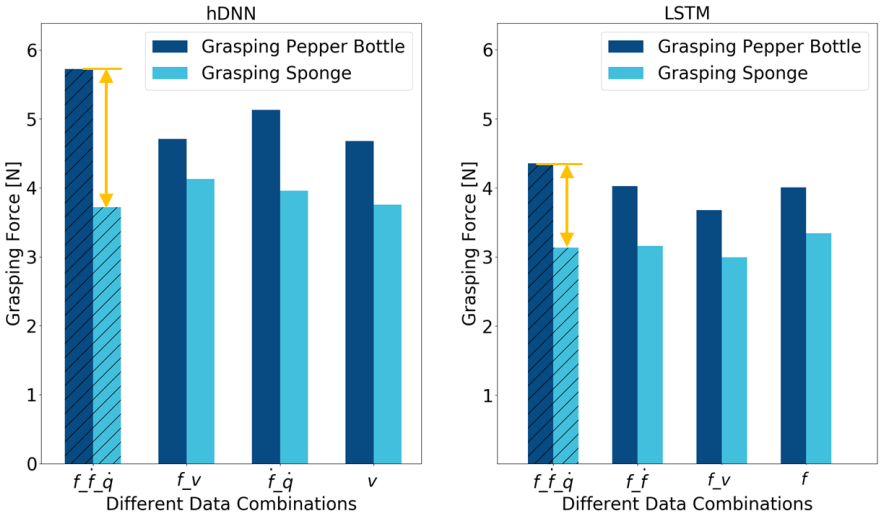


Figure 4.12: Settled grasping force using different state feedback from the hDNN (left) and LSTM (right) controller respectively. The combination in the left bar filled with slash has the best performance.

its deformation can be visually observed to evaluate the grasping performance.

Figure 4.13 (a) and (b) show the grasping motion of the baseline controller with a high (4N) and low (2N) force threshold respectively. The difference in the performances shows the importance of a proper force threshold, which requires the prior-knowledge of the object. The iDNN controller generated excessive forces and bent the card, leading to a failure and poor adaptation to low object stiffness during the interaction. In contrast, hDNN and LSTM controllers can hold the card stably without prior-knowledge of the card's physical properties, indicating that they have certain adaptability to the unknown object stiffness and can generate different grasping forces while interacting with different objects. On contrary, the iDNN controllers can only apply constant grasp forces with no self-adaptation, and the baseline controller requires a properly tuned force threshold.

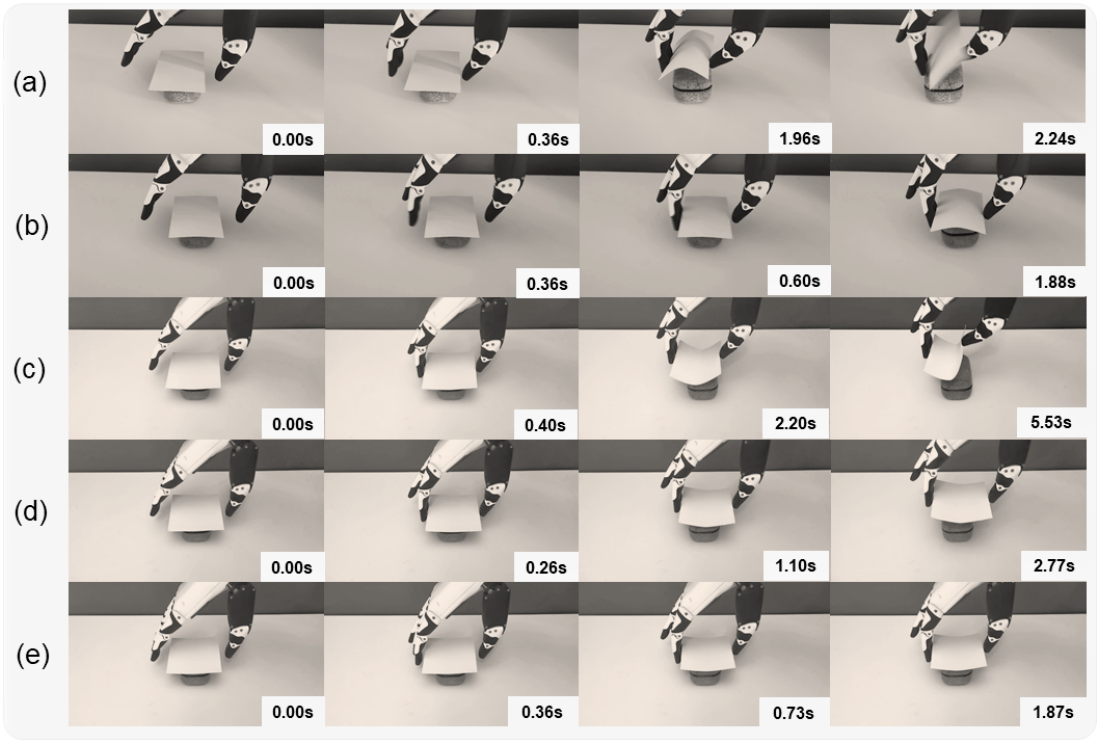


Figure 4.13: Comparison by grasping the unseen card: (a-b) Baseline controller with a high and low force thresholds, respectively; (c) iDNN controller; (d) hDNN controller; (e) LSTM controller.

4.4.6 Similarities of human and robot policies

The aforementioned results indicate that the history states play an important role in distinguishing the robot grasping phases, and encoding latent information of object shape and stiffness, which enables adaptive grasping of various objects.

Though hDNN and LSTM controllers have comparable performance, the former is better because it has a simpler network structure, larger grasping force and better adaptability to various objects. We constructed the nearest sample neighbours by t-distributed stochastic neighbor embedding (t-SNE) as shown in Figure 4.14, using actuator measurements from human tele-operated demonstrations and hDNN-based grasping. The visualisation of large overlapping areas suggest the underlying similarities between the human and learned policies, as well as

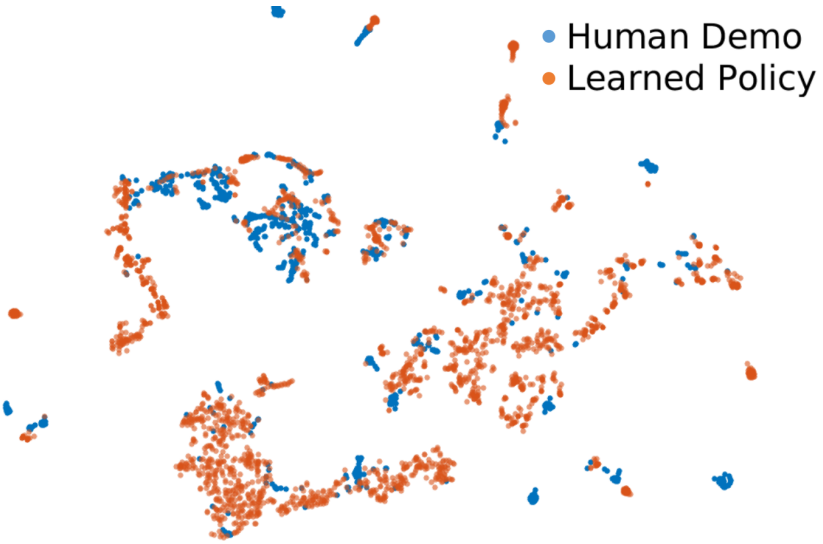


Figure 4.14: T-SNE analysis map of human and robotic grasping policies.

the effectiveness of history states in representing and extracting the state-action mapping from human grasp policies.

4.4.7 Investigation of failure cases

The success of grasping an object depends on the selection of contact points by the user, especially for the positioning the fingertips. Figure 4.15 and Figure 4.16 demonstrate both the success and failed grasping of representative objects using hDNN controller. Figure 4.16(a) and (b) show the failures caused by unbalanced and unstable contact points. Due to the characteristics of point contact, it is also difficult to grasp heavy and slippery objects, as shown in Figure 4.16(c).

4.5 Conclusion

In this work, we focused on the dexterous grasping and adaptive control of the robotic fingers, which is important while lacking of the prior-knowledge of the object's material and stiffness. We proposed a learning-based approach of adaptive grasping with an anthropomorphic robotic hand based on few real human

Figure 4.15: Grasping a variety of objects with different physical properties using the trained *hDNN* controller.

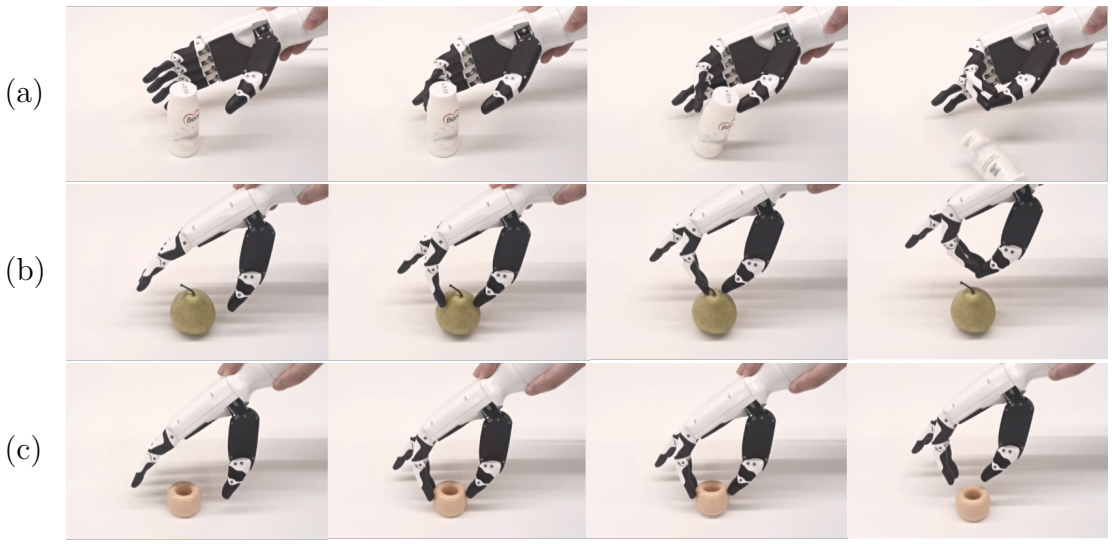


Figure 4.16: Representative failures of grasp attempts.

demonstrations. The proposed hDNN grasping controller is capable of generating adaptive forces to grasp objects with various sizes and stiffness, solely based on the robot proprioception data. The proposed grasping controller requires the minimum user input: merely a 0-1 activation to start and stop the grasp motion. Therefore the controller can be implemented in many scenarios, e.g. teleoperation system or prosthesis system where the arm motion is controlled by the user

and the grasping motion is executed by the proposed controller, which alleviates the operator's mental load from complex grasping control. Also, the grasping controller can be integrated with any off-the-shelf grasp planning algorithms.

We studied different multi-sensing state combinations to encode the state-action mapping of human grasping skills. Three different neural network structures are designed to compare the effectiveness of using the instantaneous and history states in the policy learning. The ablation study and data analysis showed the importance of history data in differentiating grasping phases and generating more robust and adaptive grasping actions. Finally, we extensively tested the learned *hDNN* controller with 100 unseen objects. The experimental results showed that the learned controller was capable of grasping objects with different shapes and stiffness, based on the transferred state-action mapping.

One future extension is to integrate the adaptive grasping controller with grasp planning algorithms which generate suitable pre-grasp poses given vision-guided object semantics. Therefore, more automatic "reach and grasp" motion can be integrated. Furthermore, we will study the usage of more sensory feedback, e.g. tactile and visual information, to improve environmental perception and enable the learning of more intelligent and versatile grasping policies.

Chapter 5

Learning to Catch Flying Objects

5.1 Introduction

We present a modular framework designed to enable a robot hand-arm system to learn how to catch flying objects, a task that requires fast, reactive, and accurately-timed robot motions. Our framework consists of five core modules: (i) an object state estimator that learns object trajectory prediction, (ii) a catching pose quality network that learns to score and rank object poses for catching, (iii) a reaching control policy trained to move the robot hand to pre-catch poses, (iv) a grasping control policy trained to perform soft catching motions for safe and robust grasping, and (v) a gating network trained to synthesize the actions given by the reaching and grasping policy. The former two modules are trained via supervised learning and the latter three use deep reinforcement learning in a simulated environment. We conduct extensive evaluations of our framework in simulation for each module and the integrated system, to demonstrate high success rates of in-flight catching and robustness to perturbations and sensory noise. Whilst only simple cylindrical and spherical objects are used for training, the integrated system shows successful generalization to a variety of household

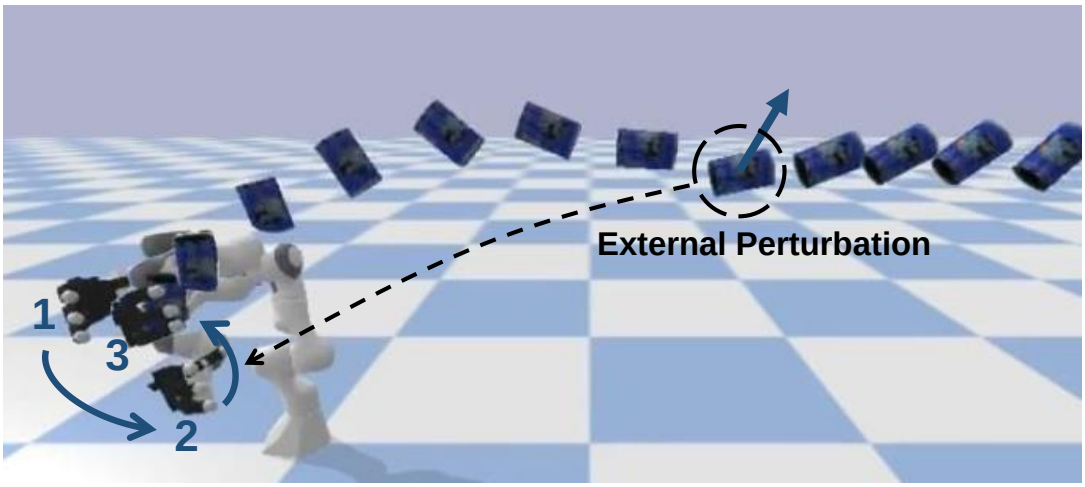


Figure 5.1: Catching a flying object under an unexpected external perturbation: labels 1-2-3 show reactive motions of the robot hand; the robot constantly re-estimates the flight trajectory, adjusts the motion rapidly, and catches the object successfully.

objects that are not used in training.

Humans are capable of interacting with flying objects in a variety of scenarios ranging from ball sports to brick-tossing in construction works. On the other end of the spectrum, the current use of robot manipulation is largely restricted to industrial environments with high regularity, and dynamic tasks with high variability such as catching flying objects still remain challenging to robots. To catch a flying object, within an extreme short duration (usually less than 1 second), the robot has to accomplish a sequence of sub-tasks including: accurate object trajectory prediction, catching pose determination and real-time motion generation. In this work we propose an end-to-end learning framework to address the problem of catching flying objects with a multi-joint robot hand-arm system. As shown in Figure 5.1, for a flying object under a sudden perturbation, our system quickly re-estimates the object trajectory, adapts the pre-catch pose, and successfully catches the object at a new pose.

Approaches to solving the problem of catching flying objects can be broadly classified in two categories: non-prehensile catching (Yu et al., 2021; Dong et al., 2020) and prehensile catching (Bäuml et al., 2010; 2011; Kim et al., 2014; Salehian et al., 2016). Most non-prehensile catch approaches focus on accurate prediction of the object trajectory, where the object is tossed from longer distance (usually 5 meters away), allowing more time for the non-prehensile end-effector, e.g. a net or a cup, to reach the interception pose.

For prehensile catching, the required robot motion is more demanding. Firstly, the determination of the catching pose should consider both the morphology and motion of the object. For example, a dexterous hand ought to grasp cylindrical objects from lateral directions, and the object velocity is better orthogonal with the palm instead of being parallel with it. The previous method uses human demonstrations to determine the pre-catch pose of target objects, neglecting the object motion when the catching happens Kim et al. (2014). While in this chapter we proposed a catching pose quality network to quantify the feasibility of the hand-object pose tuple for catching, considering both the object velocity and the required robot motion. Secondly, due to the typically fast motion of flying objects, the fingers need to close and safely grasp the object within a few milliseconds. Lastly, the motion of the hand and arm need to be coordinated. To extend the grasping duration and ease the required precision of grasping timing, a prescribed soft catching strategy has been proposed (Salehian et al., 2016), where the robot arm moves with the object for a short period of time. In this work, the self-emergent soft catching policy is learned via deep reinforcement learning (DRL).

Since the final catching configuration frequently changes as the object trajectory prediction updates, the robot movements need to be highly dynamic and responsive. It is difficult to re-plan the robot trajectory using nonlinear optimization approaches with a high frequency online (Lampariello et al., 2011; Bäuml et al.,

2011). Encoding the arm and hand as Dynamical Systems (DS) (Khansari-Zadeh and Billard, 2011a) offers an effective solution to catching tasks (Salehian et al., 2016; Kim et al., 2014). To train the models, kinematically feasible trajectories of robot arm and hand are collected as demonstrations.

Alternatively, DRL is a different approach to learn manipulation skills (Merzic et al., 2019b; Hu et al., 2021) without the dependence on human demonstrations. In this work, we train the motion generation policies with DRL: the trained policies output control commands based on current state observations. The state-action control scheme enables the policies to generate highly dynamic robot motion and react rapidly to environmental variations and sensory uncertainties.

Here we propose a systematic framework for catching flying objects, consisting of five different modules, as numerated in Figure 5.2: (i) an **object state predictor** for estimating the flying object trajectory; (ii) a **catching pose quality network** for choosing the best catch configuration; (iii) a **reaching policy** for moving the robot hand to the selected pre-catch pose; (iv) a **grasping policy** for performing a soft catching motion to reduce the contact impact between hand and object; and (v) a **gating network** to coordinate the reaching and grasping policies. Though the modules are designed for catching flying objects, they can easily be deployed to other tasks with minor modifications, e.g. grasping objects on a moving conveyor.

The contributions of our work are as follows:

(1) A catching pose quality network to evaluate and select the catching poses, which considers both the quality of the target object pose and the difficulty for the hand to approach it.

(2) A gating network to synthesize both the robot hand and arm motions, for seamlessly and smoothly coordinating the reaching and soft catching of

in-flight objects.

(3) An integrated framework with five learning-based modules to catch flying objects of various shapes, being robust in presence of sensory noise and perturbations.

In the following sections, we first review the literature related to robot catching in Section 5.2. Our methodology is elaborated in Section 5.3, and our proposed system is evaluated in Section 5.4. Finally, we summarize our work and propose future research directions in Section 5.5.

5.2 Related Work

5.2.1 Object trajectory prediction

Accurate motion prediction of in-flight objects is crucial for catching tasks. A ball is the most frequent target object in robot catching tasks, because its trajectory is relatively easy to predict. When the ball is small and has uniform mass distribution, the flying trajectory can be approximated as a parabola (Chen et al., 2017a). Apart from gravitational forces, aerodynamic drag is the most significant factor that needs to be considered in trajectory prediction. However, the air drag coefficient typically requires experimental calibration and is related to the object's shape (Bäuml et al., 2011). One mitigation strategy to the above problem is the use of the Extended Kalman Filter (EKF), which is a widely used estimator for the object's state and allows the consideration of air drag and other external factors (Bäuml et al., 2010; Müller et al., 2011; Dong et al., 2020). Developing explicit models of the object dynamics is another option for trajectory prediction (Jia et al., 2019). However, requirements on prior knowledge such as mass or moment of inertia limit the generalization ability.

Estimating the dynamics model with machine learning methods has achieved

promising results (Kim and Billard, 2012), but the learned models suffer from performance drop when generalizing to unseen objects due to limited data size. Learning-based models are widely used to approximate nonlinear dynamics. Yu et al. (2021) trained a neural network-based model and a differentiable Kalman filter to estimate acceleration of an uneven object by observing the previous detected trajectory. In this work, we learn the object trajectory prediction using a recurrent neural network with access to a short time history of the object trajectory.

5.2.2 Catching pose selection

When the object trajectory is predicted, a feasible catching pose intersecting the object trajectory and within the workspace of the robot arm should be selected. Moreover, the hand should be able to reach the selected target pose before the object arrives. For catching spherical objects, a common way to determine the interception pose is to find the nearest intersection between the object's flying trajectory and the robot's reachable space. The pose selection can be formulated as an optimization problem with nonlinear constraints (Bäuml et al., 2010), which can be solved by quadratic programming. For objects without central symmetry, the robot hand has to attain certain orientation before proceeding to catch. Kim et al. (2014) used a trained graspable space model of the specific target object to predict the hand pose. The aforementioned algorithms assume the robot can reach the determined interception pose before the object arrives, but that is not always the case. We propose a neural network-based scoring model, trained to evaluate and rank the candidate catch poses by the control effort required for the robot to move towards them from the current joint configuration.

5.2.3 Robot motion generation

Given a selected catching pose, to enable the robot to intercept the flying object at a particular time with a particular posture, Kim et al. (2014) used time-invariant dynamic systems to encode the robot's motion. Learning from demonstrations is also possible, where the motion dynamics can be modeled from the expert demonstrations, e.g. kinesthetic teaching (Chen et al., 2017b) . However, the generalization ability is limited by the scale and nature of the demonstration set. DRL offers an alternative approach for learning robot control (Zeng et al., 2020). Unlike aforementioned algorithms where the time-variant robot motion is planned in advance, we propose a DRL scheme, where the learned control action is generated based on the current state observation to maximize expected future reward. This state-action based control approach can react to the environmental changes rapidly, which is crucial for object catching tasks.

5.3 Method

Our proposed system consists of five distinct modules, as illustrated in Figure 5.2. The object state predictor estimates the flying trajectory, and the catching pose quality network evaluates the poses on the trajectory and selects the desired object pose when the catching motion happens. These two modules have strict demand of accuracy and hence we model them as neural networks and train them with supervised learning using large amount of synthetic data. The remaining three modules control the robot and they are trained with deep reinforcement learning, because the state-action control manner is effective for the highly dynamic tasks.

The object state predictor keeps predicting future poses $\{P^o_{t+1}, ..., P^o_{t+N}\}$ for a flying object, taking the object states within a period of time as input. Taking one candidate object pose P^o_k in the predicted trajectory, and the current robot

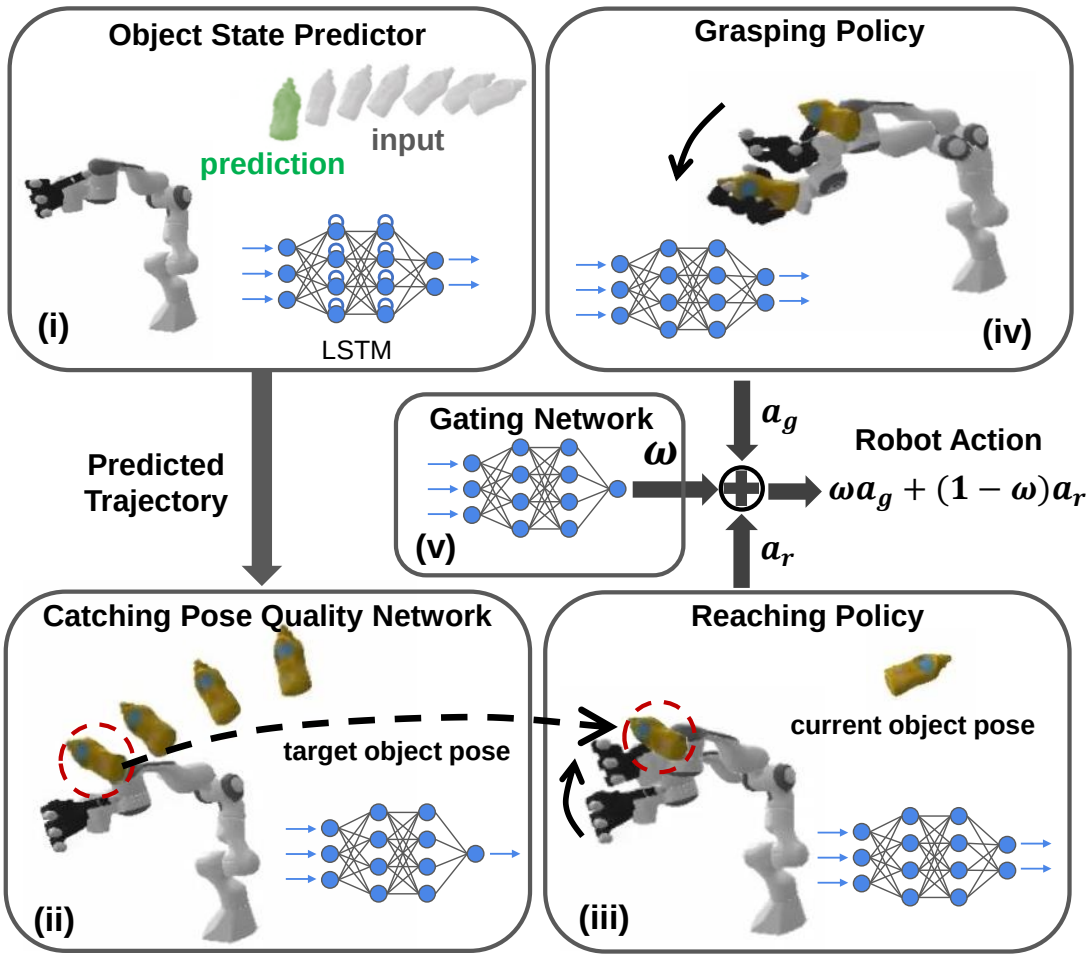


Figure 5.2: Integrated framework consisting of five proposed modules to learn to catch flying objects.

hand pose P_t^h as input, the catching pose quality network generates a score to evaluate the feasibility for the robot to reach and grasp the object. At every time-step, the catching pose quality network will evaluate each object poses on the predicted trajectory, and the one with highest score will be selected as the target object pose when the catching happens.

The reaching policy network and grasping policy network both output the robot control commands, while the former is responsible for moving the robot hand to the corresponding pre-catch pose, and the latter is responsible for accomplishing

the grasping motion. The gating network coordinates the reaching and grasping policies by updating the weights of the output actions of both policies at real time.

The first two modules work at 100 Hz, whereas the last three modules work at 50 Hz. The modules are trained and validated using PyBullet physics simulator (Coumans and Bai, 2016–2021). Based on the dependencies between the five modules, the order to train is: (1st) object state prediction network and reaching policy (separately); (2nd) catching pose quality network; (3rd) grasping policy; (4th) gating network. The training details of each module are discussed as follows.

5.3.1 Object Trajectory Prediction

An accurate prediction of the future motion trajectory is relevant to catching performance. Explicitly modeling the nonlinear dynamics system is difficult, so a practical approach is to estimate the object velocity and acceleration with a trained neural network. The Long Short-Term Memory (LSTM) network is widely used in processing and extracting intrinsic information from sequential data (Hochreiter and Schmidhuber, 1997). Yu et al. (2021) used LSTM as the backbone network to predict the flying trajectories of objects with uneven shapes. In this project, we use an LSTM network which takes a sequence of past object states with n time-steps $[X^o_{t-n+1} : X^o_t]$ as input, to predict the object state at next time-step X^o_{t+1}. The network has one LSTM cell with 100 features in the hidden state, followed by one fully connected hidden layer with 100 neurons.

The future object trajectory is then derived from the predicted object state. The object state vector X^o consists of positions, orientations, linear/angular velocities and linear/angular accelerations. To simulate air resistance in a simple manner, linear damping is introduced on the linear and angular velocity of the objects, which aerodynamically corresponds to a Stoke's drag assumption. The training

dataset is gathered in simulation with simple objects (see Section 5.4). However, our proposed predictor is compatible with using trajectory data obtained from real objects in flight, as it is learned in a supervised way prior to the training of other modules.

5.3.2 Catching Pose Quality Network

Given the current pose of robot hand, the success of a catching attempt is highly dependent on the choice of the catching timing, or equivalently, the choice of the object pose when the catching action occurs. The proposed catching pose quality network is used to evaluate the capability of the robot hand at current pose P^h to successfully catch the object at pose P^o. The network takes both current robot hand pose and candidate object pose as input: $[p^o, q^o, p^h, q^h]$, where p denotes Cartesian position and q denotes the orientation quaternion, and outputs a scalar quality score which quantifies the effectiveness of the hand to reach and grasp the object. The network has 2 fully-connected hidden layers, each with 100 neurons.

With the current hand pose, the trained network generates the scores of all the object poses on the predicted flight trajectory, and then the one with highest score will be selected as the target object pose when the catching happens, and the robot hand ought to reach the corresponding pre-catch hand pose before the object arrives. The pre-catch pose is determined by the trained reaching policy network, where the palm is close to the object, as described in Section 5.3.3. The computation of the score consists of two parts: the required time and movements for the hand to reach the pre-catch pose, and the effectiveness of the pre-catch pose in aiding the catching of the incoming object. The score is defined as:

$$s = e^{-\|J_d - J_{now}\|} \cdot e^{-\|p_d^h - p^o\|} \cdot \left(1 - \vec{u}_n \cdot \frac{\vec{v}}{||\vec{v}||}\right), \tag{5.1}$$

where J denotes the vector of robot arm joint positions; $\vec{u}_n$ denotes the normal unit vector of the palm, pointing outwards, and $\vec{v}$ denotes the object velocity vector.

The subscript d denotes the desired pre-catch pose of the hand. The first term evaluates the changes in the robot arm joint configuration during the reaching motion, indicating the required efforts for moving the hand from current pose to the desired pre-catch pose. The second term evaluates the distance between the pre-catch hand position p_d^h and the object position p^o. The third term relates to the orientation of the palm when it contacts the object. To increase the contact area during the catching motion, the plane of the palm should be perpendicular to the object's moving direction. The constant value 1 is added to this term, shifting the value range from $[-1,1]$ to $[0,2]$. The second and third term evaluate the effectiveness of the pre-catch hand pose in catching the moving object at the target pose. The object pose in flying trajectory which is most adequate for catching can be selected by the network. However, if the object flying trajectory is either too far from the robot hand or in an inappropriate pose, the object pose with highest score can be unfeasible for catching.

The catching pose quality network is trained in a supervised manner. To gather the training data, a learned reaching policy is necessary (see Section 5.3.3). The data gathering procedure consists of two steps: Firstly, N object flying trajectories $\tau_{1:N}$ with random initial object poses and velocities are sampled and logged. Then, for every trajectory, M object poses $P^o_{1:M}$ with fixed time interval are selected as the target object pose. K random hand poses $P^h_{1:K}$ within a certain range are also sampled and logged. Secondly, given the object fixed at one selected target pose P^o_i, the robot hand will reach it from the selected hand pose P^h_j, controlled by the learned reaching policy. After a fixed period of time, the corresponding score for the hand-object pose tuple $[P^o_i, P^h_j]$ is computed and logged, and the current robot hand pose is regarded as the desired pre-catch pose. In the data gathering procedure, $N * M * K$ data points are collected as training data.

5.3.3 Reaching Policy Network

Given the target object pose selected by the catching pose quality network, the reaching policy will control the robot hand to reach the corresponding pre-catch pose as soon as possible. The reaching policy is trained with DRL, in a trail-and-error manner, where the robot gradually updates the policy by gathering rewards through interaction with the environment. The learning algorithm is Proximal Policy Optimization (PPO), a widely used DRL algorithm for continuous control tasks (Schulman et al., 2017). To train the policy, the object is fixed at a random pose throughout the episode. To stabilize the physics simulator and accelerate learning, the contact check between the object and the robot is disabled.

The state space is $S_r = \{\tilde{p}, \tilde{q}, \vec{v}_a\}$, where $\tilde{p}$ and $\tilde{q}$ denote the position and quaternion of the selected target object pose relative to the robot hand, and $\vec{v}_a$ denotes the unit vector showing the desired direction for the hand to approach, pointing from the object to the outward, as shown in Figure 5.3. The action space A_r consists of the linear velocity and angular velocity of the robot hand. When the robot is controlled by the reaching policy, the finger joints maintain an open configuration, prepared for catching the object.

The reward function of the reaching policy is the linear combination of three different terms:

$$R_r = -\frac{1}{k}\sum_{i=1}^{k}\left\|p^{k_i} - p^o\right\| + (-\vec{u_n}\cdot\vec{v}_a) + \|\vec{u}_x \cdot \vec{v}_x\|\,. \tag{5.2}$$

The first term is the negative of mean distance between the target object position p^o and the hand key-points positions p^{k_i}. As shown in red color in Figure 5.3, the k key-points are equally distributed on the inner surface of the robot palm and fingers. This term will reward the hand to approach the object. In the second reward term, $\vec{u}_n$ denotes the unit normal vector of the palm. $\vec{v}_a$ denotes the approaching vector of the object, which is always orthogonal with the object's major axis. For the flying object, $\vec{v}_a$ is also coplanar with the gravity vector and

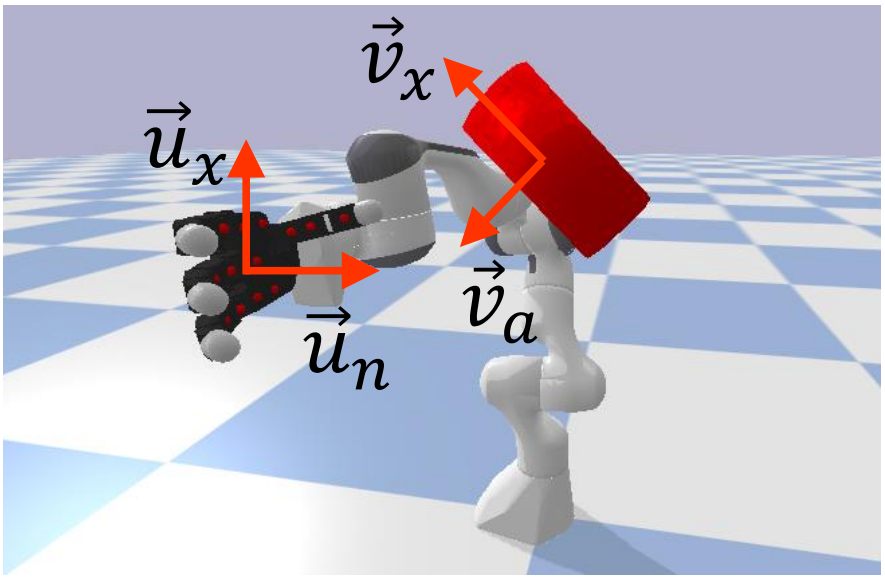


Figure 5.3: Illustration of the assignment of vectors and hand key-points for the reward function.

the object velocity vector. For small or spherical objects without major axis, the $\vec{v}_a$ is the unit linear velocity vector. This term will reward the hand to increase the contact area by adjusting the hand orientation. In the third reward term, $\vec{u}_x$ denotes one of the major axis of the hand, and $\vec{v}_x$ denotes the major axis of the cylindrical object. This term rewards the hand to align with the object, similar to the way that human grasp cylindrical objects. The four aforementioned vectors are illustrated in Figure 5.3. We also add early termination criteria where the robot arm approaches a singularity or the joints hit the position or torque limits, aiming to learn safe robot motion. The reaching policy network has 2 fully-connected hidden layers, each with 256 neurons, and it is trained with 4 million time steps.

5.3.4 Grasping Policy Network

The grasping policy is also trained with PPO to perform the final catching motion when the object is approaching. The learning of the grasping policy requires the trained three aforementioned modules. In every training episode, the object is

tossed from a random pose with a random velocity, flying towards the robot (see Section 5.4.1 for randomization range). The future object trajectory is predicted at 100 Hz. Then the catching pose quality network selects the target object pose, and the reaching policy moves the hand towards the corresponding pre-catch pose. Once the object is estimated to arrive at the selected target pose after time T, the grasping policy takes over the robot control to interact with the environment. We refer to T as the *preparation time* for the grasping policy. If T is too long, the reaching policy might not move the hand to the pre-catch pose yet, and if T is too short, the hand might not be able to reach the best catching speed and fingers might not be able to close in time. To increase the robustness of grasping policy, the preparation time is randomized during training: $T \in [0.05, 0.25]$s.

The state space of the grasping policy S_g consists of the object pose relative to the hand, linear velocity of the object, linear velocity of the hand and the joint positions of the fingers. The action space of the grasping agent A_g consists of the linear, angular velocities of the hand, and the target finger joint positions. The reward function used for training the grasping policy is the sum of three different terms:

$$R_g = -\frac{1}{k}\sum_{i=1}^{k}\left\|p^{k_i} - p^o\right\| + \frac{n_c}{k} + r_p. \tag{5.3}$$

The first term of Equation 5.3 is the same as the first term of Equation 5.2, both rewarding the hand to approach the object. However, here we use the real-time object pose, while in Equation 5.2, we use the target object pose selected by the catching pose quality network. In the second term, n_c denotes the number of the hand key-points that are in contact with the object. This term rewards the hand to grasp the object with larger contact area. The last term r_p is a piece-wise function defined as:

$$r_p = \begin{cases} e^{-\left\|p^h - p^o\right\|} \cdot \vec{v}_h \cdot \vec{v}_o & n_c = 0 \\ e^{-\|\vec{v}_h\|} & n_c > 0 \end{cases} \tag{5.4}$$

where $\vec{v}_h$ and $\vec{v}_o$ denote the linear velocity of the hand and object. When the

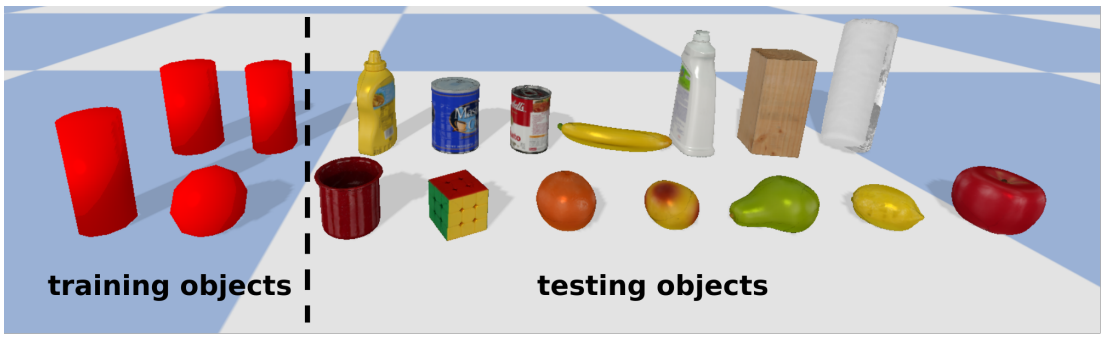


Figure 5.4: Objects used for training (red cylinders and spheres), and the various household objects (right section) used to test the generalization of the integrated catching system.

hand has no contact with the object, r_p promotes the hand to move towards the same direction as the object. This motion will reduce the impact when the object contacts the hand. The term $e^{-\|p^h - p^o\|}$ is a distance weight, motivating the soft catch motion only when the object is close enough. When the object is in contact with the hand, r_p will reward the hand for staying still, and aims to eliminate any undesired arm motion when the object is grasped. The grasping policy network has the same structure as the reaching policy network, and is also trained with 4 million time steps.

5.3.5 Gating Network

The reaching policy and the grasping policy play different roles in the catching task, and control the robot at different stages. Instead of switching the control with hard-coded criteria (e.g. the distance between hand and object), a gating network (Jacobs et al., 1991) is trained to synthesise the control commands generated from two policies. The state space of the gating network is the union set of the state spaces of two policies: $S_r \cup S_g$, and it outputs the action weight for the grasping policy $w_g \in [0, 1]$, and the corresponding weight for the reaching

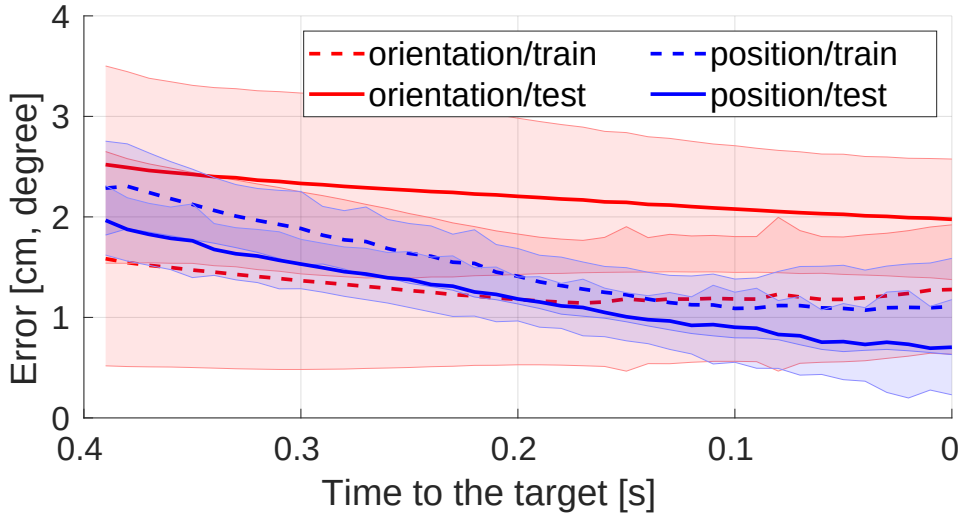


Figure 5.5: Decreasing errors of object pose prediction as time elapses. The horizontal axis denotes the remaining time to the target, and the vertical axis is the error between predicted object pose and the ground truth. Solid and dashed lines refer to testing and training objects respectively.

policy is $w_r = 1 - w_g$. The resultant robot action is blended from both policies: $w_g a_g + w_r a_r$, and the reward function is also the weighted sum of the rewards of both policies: $w_r R_r + w_g R_g$. To maximize the reward which is directly related to the robot action, the gating network ought to smoothly switch between two policies at the right time. With the trained all four aforementioned modules, the gating network is trained in full catching scenarios using PPO. The network has the same structure as the reaching and grasping policy network, and also trained with 4 million time-steps.

5.4 Validation

In this section, we first evaluate the performance of each learned module on their corresponding tasks. Then, the integrated system is tested and validated with

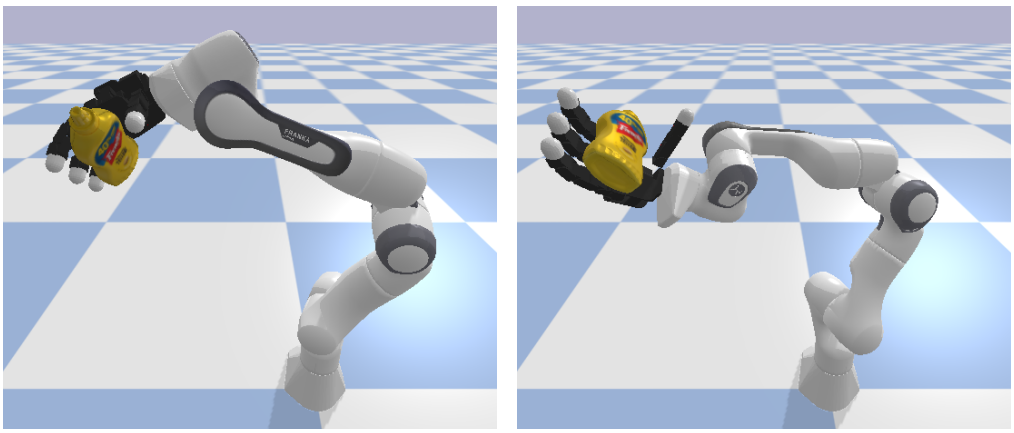


Figure 5.6: Illustration of how the reaching policy controls the robot to approach the pre-catch pose, based on the pose and moving direction of the object.

catching tasks using multiple unseen objects.

5.4.1 Validation of each module

Object state prediction network. To train the LSTM network, 10,000 flying trajectories of the training objects (as shown in Figure 5.4) with random starting poses and velocities are recorded. Trajectories are split as 90% for training and 10% for testing. The object poses are recorded at 100 Hz, and the according velocities and accelerations are computed by numerical differential. Figure 5.5 demonstrates the prediction performance of our LSTM model on the testing dataset. It suggests that the learned model can provide reliable prediction of the object trajectory for the other modules.

Reaching policy network. For an object fixed at a random pose within the workspace of the robot, the learned reaching policy is capable of moving the robot hand to the proper pre-catch pose as soon as possible. As shown in **??**, for the cylindrical object, the policy can approach the object from the lateral direction, and align the robot hand with the object's major axis. For the spherical objects, or objects without major axis, the policy can approach them from any direction given by the approaching vector, as long as it is reachable by the robot.

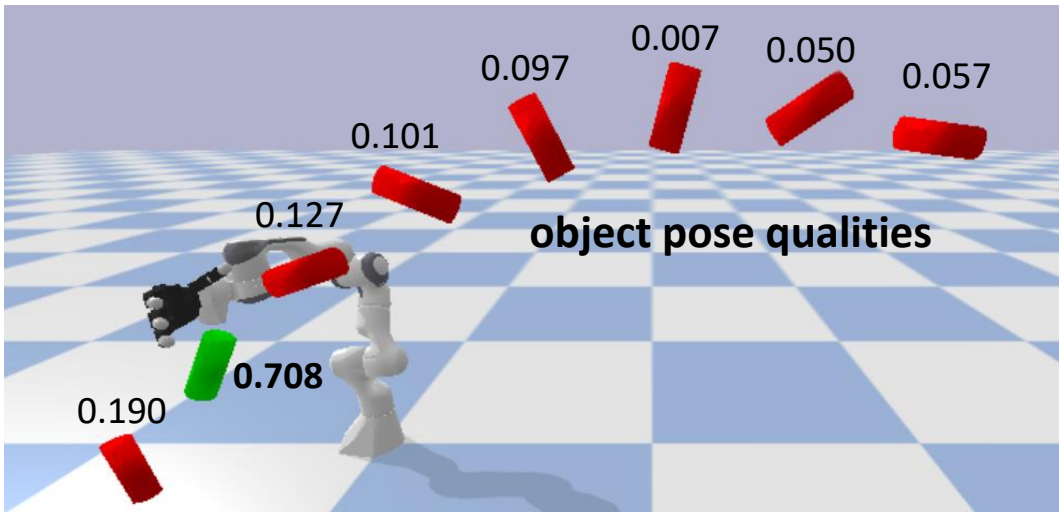


Figure 5.7: A representative case of the catching-pose scores (from the quality network) and various object poses. The pose in green is a good example of a high score selected as the target pose for the reaching policy.

Catching pose quality network. Taking the target object pose and current hand pose as input, the learned catching pose quality network outputs a score, assessing the effectiveness of the robot to catch the object in the specific condition. As shown in Figure 5.7, by evaluating every object pose on the predicted trajectory, we can select the one with highest score, as the target for the reaching policy to reach. Empirically, selecting the object pose with higher score as the catching target is more likely to success, because it is easier for the robot to approach the according adequate pre-catch pose.

Grasping policy network. When the hand is close to the pre-catch pose, as the object approaches, the trained grasping policy performs the coordinated motion of the arm and fingers to catch the object with a high speed. Figure 5.8 demonstrates the snapshots of one soft catching motion, and Figure 5.9 shows the corresponding velocities of hand and object, from which the motion can be divided into three phases: pre-catch phase, catching phase and the holding phase.

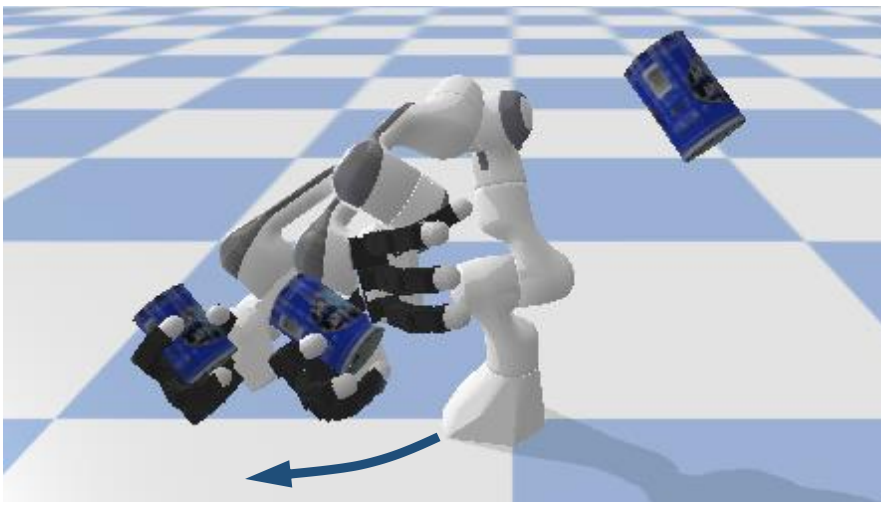


Figure 5.8: Active reactions of buffering impact during catching. The robot hand moves backwards before and after the initial contact with object, and finally holds the object in still.

In the pre-catch phase, the hand starts to move along with the object's moving direction, maximizing its translational velocity to reduce the contact impact with the object. In the catching phase, after the first contact between hand and object, the hand retains movement for a short period of time and then starts to decelerate. The grasping motion of the fingers is mainly completed in the catching phase. The soft catching motion provides more time for the fingers to close, and alleviates the bouncing of the object. In the holding phase, after the grasp is secured, the hand velocities converge to zero and the robot stays still. The reward term in Equation 5.4 penalizes unnecessary motion after the object is grasped. Due to the robot joint velocity limits, it is difficult for the end-effector to reach the object flying velocity before the contact happens, as shown in Figure 5.9. Therefore, the soft catching motion mitigates – but hardly completely eliminates – the impact force between hand and object. Hence the catching of fragile objects like raw eggs, or bouncy objects like rubber balls remains challenging for the proposed method.

Gating Network. The learned gating network synthesizes a blend of actions from both control policies by outputting a linear mixture weight. As shown in

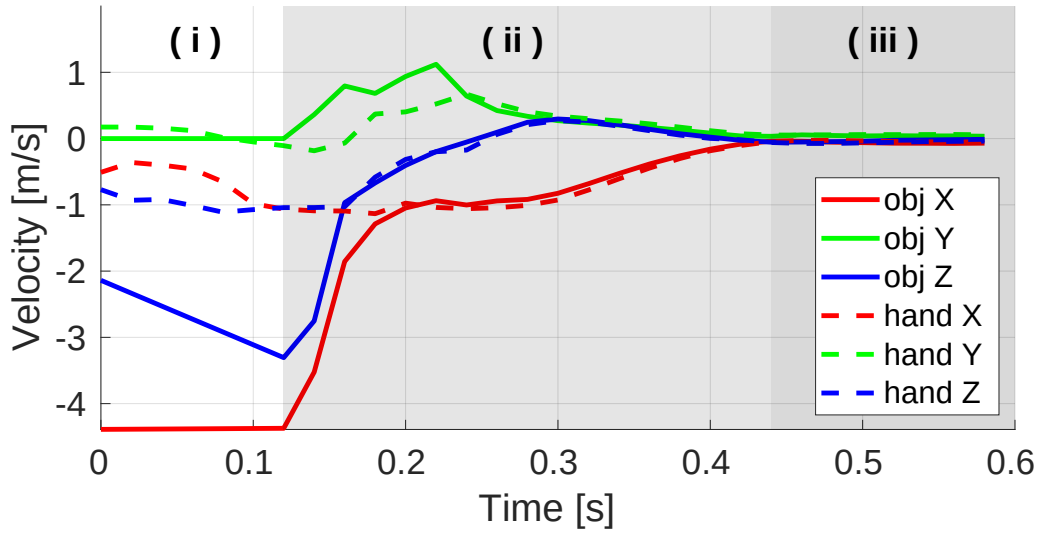


Figure 5.9: Linear velocities of the hand and object when the object is close to the robot. Empirically the process can be divided into three phases: (i) pre-catch phase, (ii) catching phase and (iii) holding phase.

Figure 5.10, the reaching policy controls the robot at the beginning. The switch starts when there is still some distance between hand and object, leaving enough time for the hand to accelerate to a proper velocity for soft catch. On average, the switch lasts for 0.15 seconds and then the grasping policy takes control for the rest of the time. To evaluate the effectiveness of gating network, we compare the performance of the integrated catching system with and without gating network. The hand-coded switch is: if the object is to arrive at the selected target pose after time T, the grasping policy controls the robot. Otherwise the reaching policy controls the robot. Based on the results demonstrated in the first three lines of Table 5.1, the learned gating network outperforms the manually designed switch criteria on catching both training and testing objects.

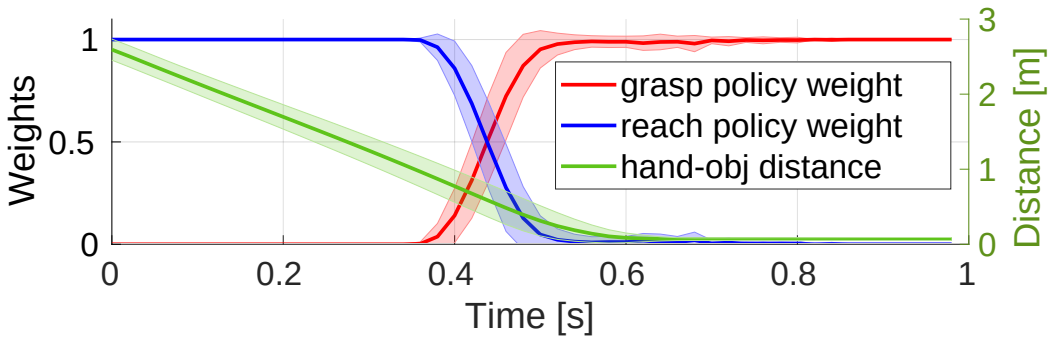


Figure 5.10: Action weights for grasping and reaching policies in multiple catching trials, generated by the gating network, and the corresponding hand-object distance. The switch finishes within 0.15 seconds on average.

5.4.2 Validation of the integrated system

The integrated system is evaluated by catching various objects in different scenarios. In the catching trials, the robot arm is fixed at the origin and starting from the same configuration. Considering the reachability and the max joint velocities of the robot arm, we randomize the object initial position between $[2.75 \pm 0.25, -0.50 \pm 0.10, 0.80 \pm 0.10]$m, orientation in Euler angle between $[0 \pm \frac{\pi}{4}, 0 \pm \pi, 0 \pm \frac{\pi}{4}]$rad, linear velocity between $[-4.5 \pm 0.5, 0, 3.0 \pm 0.5]$m/s, and angular velocity between $[0 \pm 2, 0 \pm 10, 0]$rad/s.

Catching Seen Objects. We first evaluate the catching performance of the integrated system using the four training objects. From Table 5.1, it can be seen that the integrated system performs better in catching the sphere than catching the cylinders with different diameters. For objects with major axes, such as cylinders, it is more challenging to get to the proper pre-catch pose in time, especially when the objects are spinning at high speed. In contrast, for spherical objects, the robot hand merely needs to move to the nearest pre-catch poses which interact with the flying trajectories and pointing the palm against the objects'

Table 5.1: Success rates of catching various objects in different scenarios, each stemming from a total of 100 trials.

		Training Objs†[%]				Testing Objects†[%]													
	avg.	cld0	cld1	cld2	ball	ybot.	bcan.	rcan.	bana.	wbot.	wood.	chip.	mug	cube	org.	peac.	pear	lemn.	app.
Integrated system	80	75	73	72	97	81	74	78	66	71	81	72	83	90	85	86	82	86	85
$T^* \in [0.05, 0.20]$ s	69	67	73	75	75	62	68	57	45	76	68	66	64	76	80	72	78	70	78
$T^* = 0.15$ s	72	60	72	66	90	63	61	62	69	78	64	64	58	73	84	89	80	79	86
$\sigma_{\text{noise}} = [5\text{cm}, 5°]$	64	60	63	63	73	48	56	52	38	54	59	56	64	77	75	72	77	77	84
$\sigma_{\text{noise}} = [15\text{cm}, 15°]$	37	29	40	42	38	31	35	29	23	39	31	29	27	50	47	43	44	43	48
Random perturbation	62	41	60	61	73	55	60	56	50	66	38	51	52	73	74	77	76	75	81

* Without using the gating network. T denotes the preparation time for grasping policy, as described in Section 5.3.4.

† Object list: cylinder0, cylinder1, cylinder2, ball, yellow bottle, blue can, red can, banana, white bottle, wood block, chips can, mug, rubik's cube, orange, peach, pear, lemon, apple.

moving directions, without considering the orientation of the objects.

Catching Unseen Objects. We then perform tests using objects never seen during the training. As shown in Figure 5.4, a set of household objects with various shapes and sizes is used to evaluate the generalization ability of our integrated system. Object models are from YCB dataset (Calli et al., 2017). For simplicity, the objects are rigid with uniformly distributed mass and have the same weight $m = 0.3$ kg. As demonstrated in Table 5.1, the success rates on catching training and testing objects are comparable, indicating that the learned modules and the integrated system have adequate generalization ability to various sizes and shapes of target objects. Since all the training objects in simulation are rigid and have the same stiffness, the proposed modules, especially the grasping policy network, have limited generalization ability to deformable and bouncy objects.

Adding noise and perturbations. Furthermore, we test the robustness of our

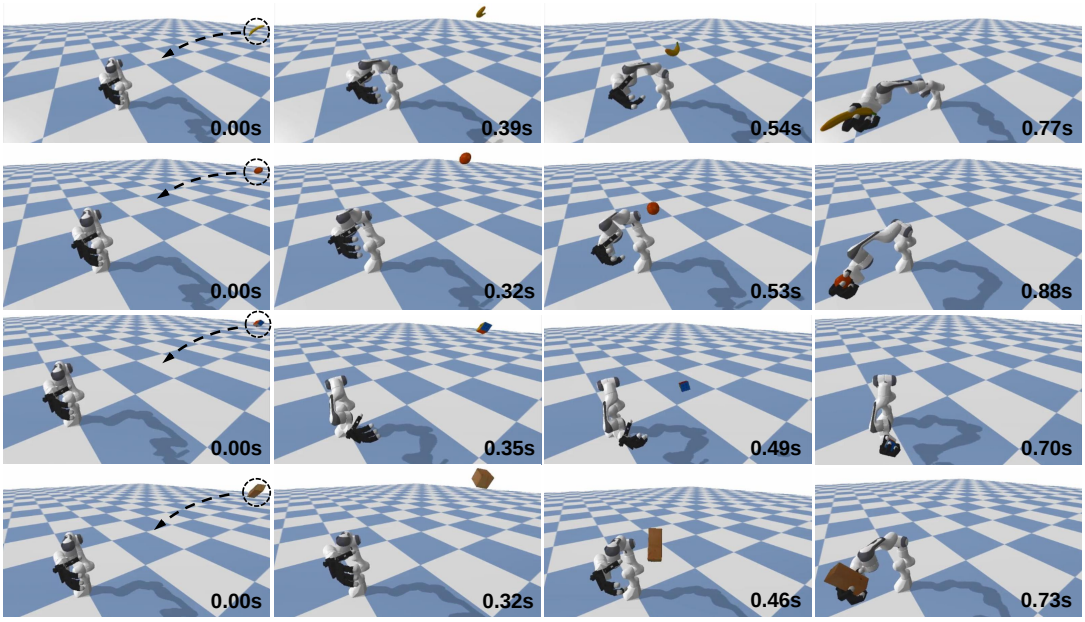


Figure 5.11: Validation of the integrated system by catching four new testing objects: a banana, an orange, a Rubik's cube, and a wood block. Initial poses and velocities of objects are randomized within a range.

system in the presence of noise in object pose observation, and by introducing a random perturbation to the object during flight.

Synthetic noise is added to the object position and orientation (Euler angles), as random three-dimensional vectors. The norms of the vectors are sampled from zero-mean Gaussian distributions with standard deviations σ_{noise}. Noisy object pose observations are processed by a low-pass second-order Butterworth filter with a cutoff frequency of 20 Hz before being fed into the trained modules. Compared with noise-free state observation, the catching performance with noisy observations drops by 16% and 43% with noise sets $\sigma_{\text{noise}} = [5\text{cm}, 5°]$ and $\sigma_{\text{noise}} = [15\text{cm}, 15°]$ respectively, indicating the integrated system is robust to sensory noise to some extent. These findings provide a guideline for the requirements of the state-estimation system in future implementations on real robot hardware.

To evaluate the system's reaction to sudden changes, we add a velocity pertur-

(a) Collision (b) Slipping (c) Imbalance

Figure 5.12: Representative examples of failure modes: (a) collision before catching, (b) slipping from the grasp or (c) imbalanced contact points.

bation to the in-flight object with direction $[1, 0, 1]$ and norm ranging from 0.5 m/s to 1.5 m/s, after the object flies for 0.1 s to 0.2 s. As demonstrated in Figure 5.1 and Table 5.1, in most cases the robot can react quickly to the perturbations and catches the object at the new pose.

5.4.3 Failure modes

The most common cause for catching failures is the collision before grasping, e.g. between the object and fingertips or the side of the palm. The smallest misalignment of the hand's pre-catch pose can lead to undesired contacts, and therefore this type of failure is more frequent when catching large rotating cylindrical objects, as shown in Figure 5.12a.

Another cause for catching failures is that the object slips from the hand, especially when the object is cylindrical, and its linear velocity is parallel to its major axis, as shown in Figure 5.12b. For small objects, balanced contact points are important for grasp quality. As shown in Figure 5.12c, the thumb does not contact the object, leading the object to be pushed towards the wrist, and falling out of the hand eventually.

5.5 Conclusion

In this work, we proposed a modular learning framework for catching objects in flight. We presented and discussed each module and its integration within the system. We employed a combination of supervised learning and deep reinforcement learning approaches to train these modules. Extensive tests were performed for each module and the integrated system in various scenarios, including tests with household objects that are never seen during training. Furthermore, we studied the robustness and generalization capabilities by performing tests with noisy observations and with random perturbations to the objects in flight.

Sim-to-real transfer and validation with real-world catching tasks remain to be completed as the future work, and our proposed framework needs to be expanded to enable sample-efficient learning from real-world trials. Currently, only the position and orientation of the object are passed as observations. As part of our future work, we aim to improve catching performance by including the approximate shape or size of the object into the observation space. Non-prehensile catching is another interesting and challenging task. For large objects that cannot be grasped by a robot hand, more dynamic manoeuvres are required by the robot arm to stop and capture them. Developing a unified controller for both prehensile and non-prehensile robot catching can be a promising extension.

Chapter 6

Tactile-based Dexterous In-hand Manipulation

6.1 Introduction

Continuous in-hand manipulation is an important physical interaction skill, where tactile sensing provides indispensable contact information to enable dexterous manipulation of small objects. This work proposed a framework for end-to-end policy learning with tactile feedback and sim-to-real transfer, which achieved fine in-hand manipulation that controls the pose of a thin cylindrical object, such as a long stick, to track various continuous trajectories — through multiple contacts of three fingertips of a dexterous robot hand with tactile sensor arrays. We estimated the central contact position between the stick and each fingertip from the high-dimensional tactile information and showed that the learned policies achieved effective manipulation performance with the processed tactile feedback. The policies were trained with deep reinforcement learning in simulation and successfully transferred to real-world experiments, using coordinated model calibration and domain randomization. We evaluated the effectiveness of tactile

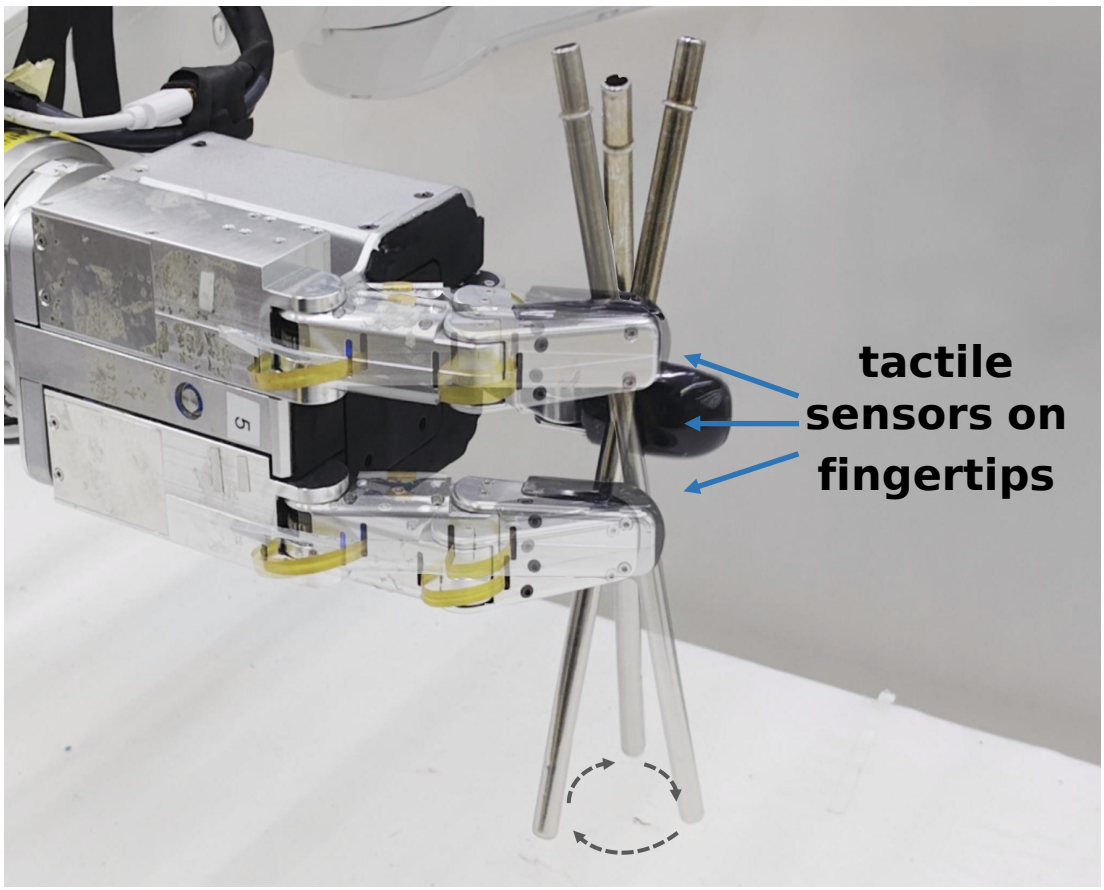


Figure 6.1: Rotating a stick with three position-controlled robotic fingers equipped with distributed tactile sensors on fingertips. The control policy is trained in simulation using deep reinforcement learning and directly transferred to the real robot.

information via comparative studies and validated the sim-to-real performance through real-world experiments.

The dexterous manipulation of in-hand objects is a skill that humans possess effortlessly. Such manipulation involves moving only our fingers, without excessive wrist or arm motion, to alter the pose of objects. The ability to endow robots with such athletic intelligence is highly desirable, particularly when working in a confined space or using certain tools, such as stirring rods. However, it still remains challenging for robots to maintain and secure a firm grasp while continuously changing the object pose.

Unlike manipulating large objects where the robot fingertips can be approximated as point contacts, manipulation of small objects (e.g. slender cylindrical objects) requires highly coordinated finger motion and precise perception of the contacts. As the feasible contact areas are small and narrow, in-hand manipulation is more sensitive to errors and uncertainties. In this chapter, we explored the in-hand manipulation ability of a three-fingered robot hand equipped with tactile sensors to address these challenges involved in manipulating cylindrical objects, as shown in Figure 6.1.

When humans manipulate objects, contact events are captured in exquisite detail by dense mechanoreceptors embedded in the skin, offering vital contact information such as the time, position and forces experienced. In robots, similar information can only be effectively captured by tactile sensors, since other sensors such as vision and proprioception may be occluded by fingers or overwhelmed by background noise (Li et al., 2020).

A variety of tactile sensors are available today, such as the vision-based tactile sensors (Gupta et al., 2022; Lambeta et al., 2020; Tian et al., 2019) and distributed tactile sensor arrays (Funabashi et al., 2022). Real-time spatial and kinetic relations between the hand and the manipulated object can be derived from the signals gathered by tactile sensors, providing a natural way to regulate the manipulation control loop (Rodriguez, 2021).

In this chapter, we analyzed different ways of exploiting the signals from distributed tactile sensors. We used the estimated positions of contact centers as tactile feedback instead of the raw data from the tactile sensors, because the low-dimensional tactile information alleviates the difficulties of sensor modeling and feature extraction. The comparative study showed that the policy trained with central contact positions outperforms baselines using other perception of the manipulated object.

Recently, many methods have been proposed to perform in-hand manipulation tasks. Model-based trajectory optimization achieved good performances on object reorientation tasks with both under-actuated hands (Calli and Dollar, 2017; Liarokapis and Dollar, 2016) and fully-actuated hands (Sundaralingam and Hermans, 2018; Kumar et al., 2014). However, the high-dimensional search space presented by multi-fingered robot hands results in optimization problems that are difficult to be solved in real time. The errors and uncertainties of the hand dynamics and contact model also limit the planning performance in real-world experiments.

Alternatively, model-free deep reinforcement learning (DRL) have been successfully implemented on various grasping and manipulation tasks (Khandate et al., 2022; Sievers et al., 2022; Andrychowicz et al., 2020), where the policies are learned from scratch in a trial-and-error manner. In this work, we utilized model-free DRL to train dexterous tactile-based manipulation skills. Policies trained with pure simulated data often suffer the performance decline when transferred to real-world experiments, and therefore we implemented model calibration of the finger joints and randomized the initial states to mitigate the sim-to-real gap.

During the manipulation of a slender cylindrical object, it is crucial for all three fingertips to remain in contact with it to avoid dropping. Therefore the finger-gaiting solution where the fingers are making and breaking contacts with the object by turns (Sievers et al., 2022) is not viable here. Instead, our robot manipulated the stick by continuously changing the contact locations, and sliding it across the curved fingertip surfaces without breaking any contact. Our proposed method is validated through the continuous stick pose tracking tasks, as shown in Figure 6.1.

The contributions of this work include:

(1) A proposed sim-to-real transfer method that accounts for mechanical back-

lash, characterizes finger joint dynamics, and uses domain randomization for initial training states to reduce the sim-to-real gap and enable successful real-world deployment of learning-based policies;

(2) Effective modeling and data processing for tactile sensors in physics simulation and real tactile feedback to enable dexterous manipulation of slender cylindrical objects, demonstrating the effectiveness of tactile sensing through a comparative study;

(3) Autonomous, continuous in-hand manipulation of slender cylindrical objects with controlled pose tracking using only 3 fingertips with tactile sensor arrays on a robot hand, demonstrating the potential for robotic dexterity in constrained spaces.

This chapter is organized as follows. In Section 6.2 we reviewed the related work. The real and simulated robot hand with tactile sensors were described in Section 6.3.1 and 6.3.2. We introduced the details of policy learning and sim-to-real techniques in 6.3.3 and Section 6.3.4. The evaluation of the learned policies in both simulation and real-world experiments were presented in Section 6.4. We concluded the chapter and proposed future research directions in Section 6.5.

6.2 Related Work

Tactile-based manipulation. Tactile provides important sensory information in dexterous grasping and manipulation tasks. For learning-based algorithms, tactile information improves both the performances and the sample-efficiency of policy training (Melnik et al., 2021). Vision-based tactile sensors like GelSight (Yuan et al., 2017a) and DIGIT (Lambeta et al., 2020) return the richest tactile data as high resolution images, which can be used to extract feature vectors (Calandra et al., 2018a; Lambeta et al., 2020) or predict future frames (Tian

et al., 2019). Simulation of such dense tactile data and the sim2real transfer is challenging (Lin et al., 2022; Wang et al., 2022). Distributed tactile sensors can cover more potential contact areas in hand. Graph convolutional network (GCN) is feasible for encoding the irregularly aligned tactile data while maintaining the positional relation of the sensors (Funabashi et al., 2022). Low-dimensional, processed tactile data such as contact position is also effective in grasping (Wu et al., 2019) and in-hand manipulation (Khandate et al., 2022). During the fingertip manipulation of the stick, in most cases only a small fraction of the distributed tactile sensors are in contact, because the stick is thin and the inner surfaces of fingertips are curved. Hence, the tactile information is sparse. Using the raw tactile data might require large data-set or complicated network structure. In this work we extracted the positions of contact center at each fingertip as the tactile feedback.

Deep reinforcement learning has been successfully implemented in dexterous in-hand manipulation (Andrychowicz et al., 2020). Veiga et al. (2020) proposed a hierarchical structure where a high-level DRL-based controller is combined with a low-level grip stabilizer. In-hand manipulation using fingertips can also be learned with DRL, e.g. finger-gaiting and finger-pivoting (Sievers et al., 2022; Khandate et al., 2022). In this work, we used DRL to learn a more challenging task: continuous fingertip manipulation of a stick where the contact areas are narrow and can be moving on the finger surfaces.

Sim2real transfer. By leveraging the physics simulator, the learning process can be sped up remarkably. However, the discrepancies between the simulated and real environments are the major cause of the performance drop when the convergent policies transferred to reality. System identification (SI) mitigates the sim2real gap by modeling the system dynamics and calibrating the parameters by experiments (Lowrey et al., 2018; Sievers et al., 2022). Domain randomization

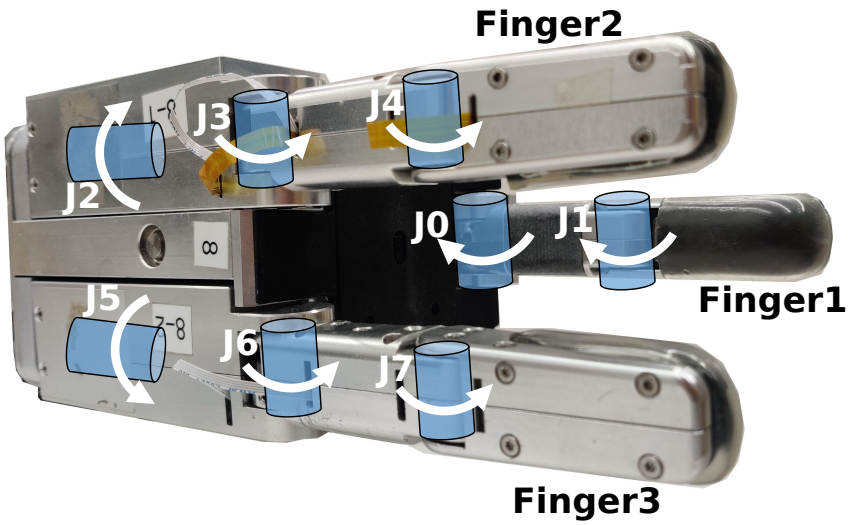


Figure 6.2: Kinematic diagram of the TRX Hand, showing all 8 joints.

(DR) trades the policy optimality for generalization ability, by randomizing the dynamics of simulated environment and perception space (Allshire et al., 2021; Andrychowicz et al., 2020). Human demonstrations can be used to identify the distribution of simulated dynamics parameters (Tsai et al., 2021). In this work, we used system identification to calibrate the model parameters, and domain randomization to enhance the policy resistance to errors and uncertainties. In this work, we utilized SI to calibrate the dynamic features of finger joints and tactile sensors, and DR to compensate model errors and randomize the physical properties that are difficult to model, e.g. friction coefficients between target object and fingers.

6.3 Method

6.3.1 Real Robot Hand with Tactile Sensors

This work used a custom build three-fingered robot hand. Named the TRX hand, it comprises of 8 fully-actuated joints, as shown in Figure 6.2. Each finger has two joints, while the base of fingers 1 and 2 are mounted on additional rotation

joints (J2 and J5) that allow them to spread independently about the palm up to 90 degrees. J0, J3 and J6 are transmitted by the worm gear mechanism and therefore have backlash and self-lock features. We simulated these features as described in Section 6.3.4.1. The maximum resultant fingertip force is 15N for each finger. Unlike many other robot hand designs using flat fingertips, the inner surfaces of TRX fingertips are deformable and curved, which are more human-like and conducive to dexterous in-hand manipulation.

All three fingertips are equipped with tactile sensors. These sensors are also developed in-house and feature an array of piezoresistive sensing elements (taxels) distributed across a continuous, curved surface. The sensor arrays are covered by an additional layer of silicone material to provide about 1 mm of mechanical compliance. There are 128 taxels per fingertip. Each tactile taxel returns a value that is proportional to the applied normal force on it.

6.3.2 Simulated Robot Hand and Tactile Sensors

We use MuJoCo physics engine (Todorov et al., 2012) to train the manipulation policies, because it is equipped with fast, accurate simulation and the tunable soft contact constraints. The simulation environment consists of the robotic hand and the target object, as shown in Figure 6.3. The wrist of the hand is fixed at a certain pose. At the beginning of each episode, the finger joints are set to initial positions and the object is placed at the initial pose, as shown in Figure 6.3, and the gravity applied on it is compensated for a short time (0.1s), allowing the fingers to close and establish contacts.

The virtual distributed contact sensors are simulated as the small, thin and almost massless cylinders located at the inner surface of fingers and palm, as demonstrated as the blue dots in Figure 6.3. Each sensor can return the binary signal of whether it is in contact with the target object. MuJoCo only supports point

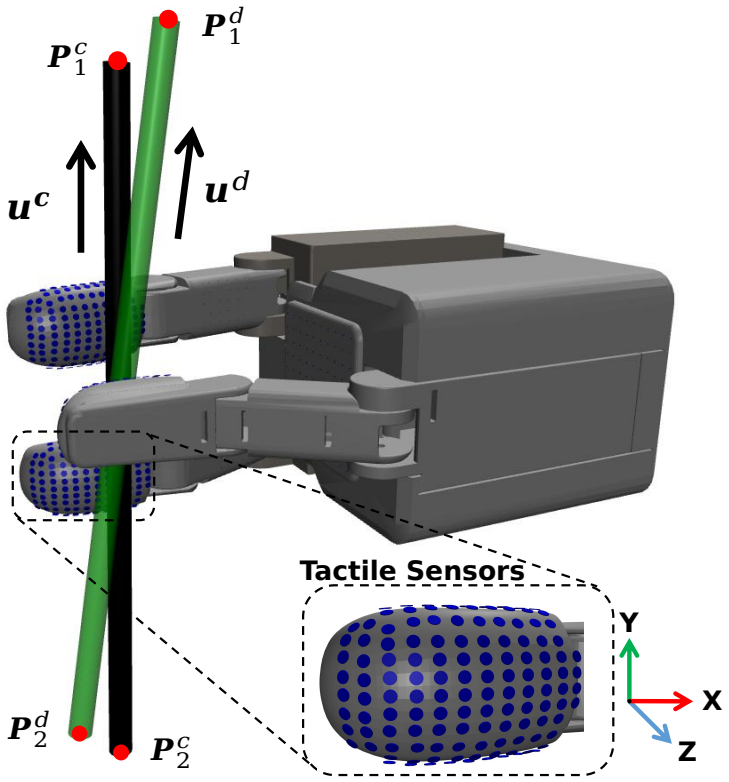


Figure 6.3: The real pose (black) and the desired pose (green) of the target stick. The learned policy aims to manipulate the stick to track the desired continuous pose in real time. $\mathbf{u}$ denotes the 3-dimensional unit vector of the major axis, and $\mathbf{P}_i$ denotes the key-point of the stick.

rigid body contact, but with the attached sensors we can approximate the line or plane contact with multi-point contact.

6.3.3 Policy Learning in Simulation

Task specification. The task is manipulating the stick with three fingers to follow the real-time target pose trajectories, as shown in Figure 6.3. The manipulation only involves the finger motion. We chose four different trajectories as the references, namely using the end of the stick to draw a line, a circle, a spiral and the number of eight, and trained four corresponding policies to follow them. We assumed that the stick is already grasped by the fingers when the trained policy

takes over the control, and the robot hand is fixed at a certain pose where the palm is perpendicular with the ground during the manipulation.

Deep Reinforcement Learning. We modeled the manipulation task as a finite-horizon discounted Markov decision process and used deep reinforcement learning algorithm to learn the control policy, consisting of an action space $\mathcal{A}$, a state space $\mathcal{S}$, the state transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ modeled by the physics simulator, and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. At every time-step, the policy π gets an observation of the state s_t and generates the action $a_t = \pi(s_t)$. After the robot interacts with the environment, it receives a reward $r(s_t, a_t)$. The target of the policy is to maximize the expected discounted sum of rewards $\mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right]$, where γ is the discounting factor. We chose Proximal Policy Optimization (PPO) as the DRL algorithm as it is stable and can be easily parallelized.

Observation and action space. The observation space consists of three parts: (a) the measured finger joint positions, (b) a unit vector $\vec{u}_d$ expressing the target stick's major axis direction, and (c) the position of contact center on each finger, which is computed as the mean position of all the tactile sensors in contact. The finger joint positions are normalized by the lower and upper limits. Since we ignored the stick rotation around its major axis, the unit vector is adequate in representing the stick orientation. The tactile sensors positions are in the local coordinate of the corresponding finger link, and normalized by a constant scale factor. The output actions are the desired displacements of finger joints.

Reward design. To alleviate the dependence on human prior knowledge, the reward function design is straight-forward, formulated as the weighted sum of four terms:

$$R = C - \omega_0 \left\|\mathbf{u}^d - \mathbf{u}^m\right\|_2 - \omega_1 \sum_{j=1}^{2} \left\|\mathbf{P}_j^d - \mathbf{P}_j^m\right\|_2 - \omega_2 \sum_{i=1}^{N} f_i. \tag{6.1}$$

C is a positive constant. Superscripts d, m denote the *desired* and *measured* values respectively. $\mathbf{u}$ denotes three-dimensional unit vector of the stick's major axis,

and $\mathbf{P}_j$ represents the positions of stick key-points, namely the two endpoints, as shown in Figure 6.3. f_i denotes the magnitude of contact force on each tactile sensor. $\{\omega_0, \omega_1, \omega_2\}$ are the scaling factors. We terminate the training episode if the height of the stick's geometry centre is below a threshold, indicating that it falls out of the grasp. Hence the constant term C promotes the learning of stable grasping by rewarding more active time-steps and longer episode length. The second term penalizing the orientation error, and the third term penalizing the position error between the desired and measured stick pose, where the errors are formulated as the L2-norm. These two terms guide the policy learning by direct quantification of the pose error. To avoid the damage on hardware caused by excessive forces, the last term regulates the sum of contact force detected by the tactile sensors, promoting the gentle manipulation skills.

6.3.4 Sim2real Transfer

For in-hand precision manipulation tasks, the discrepancy between simulation and reality is mainly introduced by inaccuracies in joint modelling, differences in sensory feedback and errors in physical parameters estimation. In this section, we introduced the methods to mitigate the sim2real gap and increase the generalization ability of trained policies for the direct sim2real transfer.

6.3.4.1 Joint model calibration

Since the robotic hand is position-controlled, in simulation the applied torque of each joint is calculated with proportional-derivative(PD) control:

$$\tau = K_P(q_d - q) - K_D\dot{q} \tag{6.2}$$

where K_P is the proportional gain and K_D is the derivative gain. q_d denotes the target joint position and q denotes the measured joint position. To calibrate the PD gains of each finger joint in simulation, we controlled the target finger joint

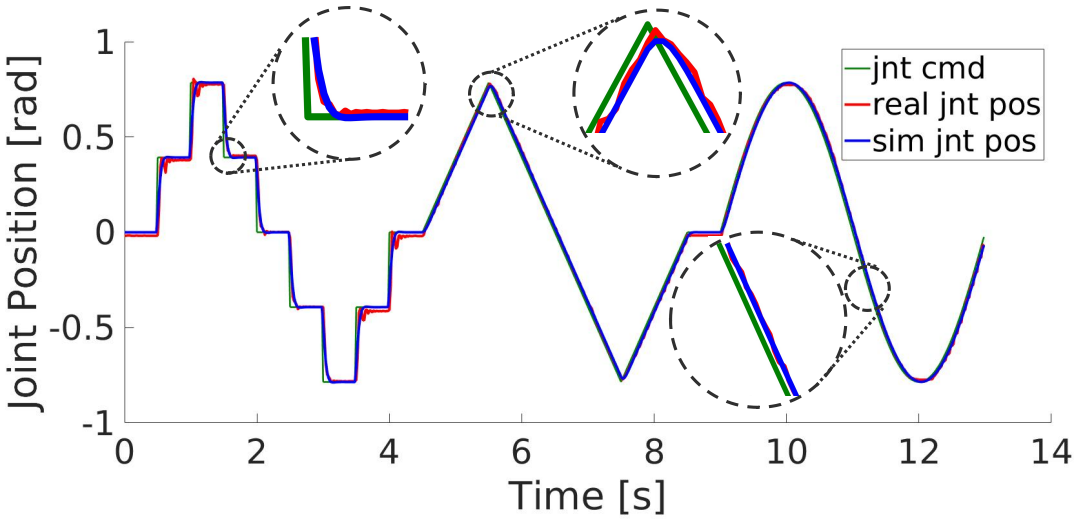


Figure 6.4: Measured joint positions in trajectory-following experiments in both simulated and real environments.

to follow the same reference trajectory in both simulation and real world, and then we compared the recorded joint position trajectories of the simulated robot and real robot, as shown in Figure 6.4. We optimized the simulated PD gains by minimizing the errors between the position trajectories of the simulated finger joint and real finger joint:

$$\begin{aligned} \min_{K_P,K_D} \quad & \sum_{t=0}^{T} \left\| q_t^{sim}(K_P,K_D) - q_t^{real} \right\|_2, \\ \text{s.t.} \quad & K_P, K_D > 0, \end{aligned} \tag{6.3}$$

where q_t^{sim}, q_t^{real} denote the joint position in both simulation and reality at time-step t, and T denotes the trajectory length. We repeated the process for each finger joint to optimize the PD gains respectively. We utilized the CMA-ES (Hansen and Ostermeier, 2001) optimization algorithm in this chapter. As shown in Figure 6.4, the calibrated simulated joints have similar dynamics response to step and continuous signals.

Backlash is a common characteristics for gear driven joints, mainly caused by the small gaps between gears and gearbox. With the existence of backlash, at some certain positions, the joint will be out of control within a small range. To model

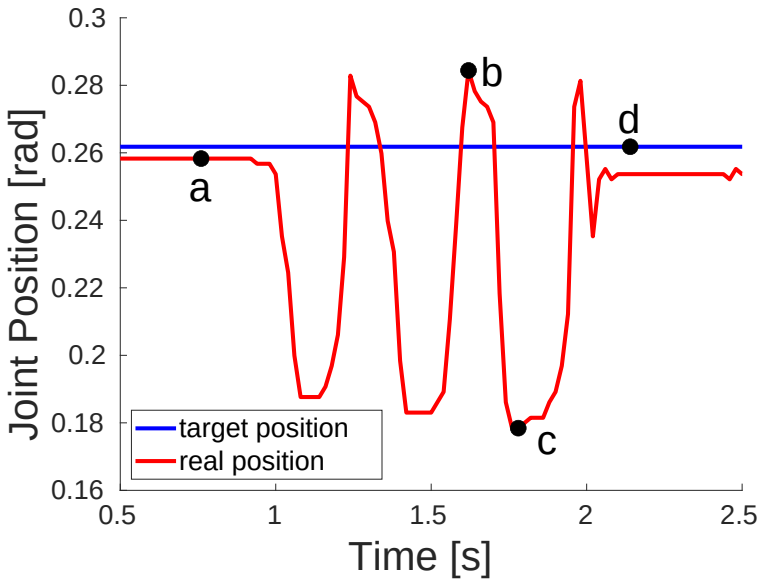


Figure 6.5: Calibration of the joint backlash feature. The joint is controlled to maintain a target position q_d, and then driven by external torques. Errors between the static position q_a and extreme positions q_b and q_c are regarded as the magnitude of the backlash at the target position q_d.

such feature we added the uncontrolled backlash joints alongside the actuated joints in simulation, and the actual rotation between the parent and child link is the sum of both joint positions. The ranges of the backlash joints are calibrated by the experiments: We controlled the target joint to maintain a certain position q_d, and then applied external torques to rotate it as much as possible. As shown in Figure 6.5, the static joint position q_a and extreme joint positions under external torques $[q_c, q_b]$ were recorded, and $[q_c - q_a, q_b - q_a]$ was the range of backlash at target position q_d. We repeated the experiments at different joint positions and then used the mean values as the backlash joint range.

Except for the dynamics and backlash, another feature that needs to be model is the self-lock of the proximal finger joints. Since the proximal joints use worm gear mechanism as motion transmission, the back driving is not allowed, which means they cannot move against the actuating direction. We simulate such feature by

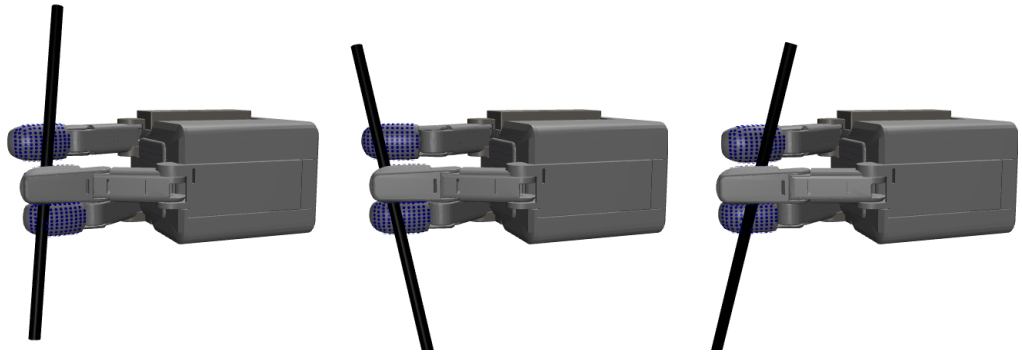


Figure 6.6: Examples of the generated initial states for policy training. The stick pose is randomized and the fingers are in contact with the stick.

adjusting the corresponding joint positions and velocities.

6.3.4.2 Tactile sim2real

In reality, each tactile sensor returns a value which is positive correlated to the normal force applied to it. Directly utilizing the raw signals can exploit the full potential of the tactile sensors. However, the mapping from tactile signal to experienced normal force is nonlinear and unique for each sensor. It is difficult to precisely calibrate such the model due to sensory noises. Moreover, training the end-to-end policies require large amount of real tactile data. Therefore, we binarized the tactile signal with a threshold. Sensors that return values greater than the threshold are regarded as in contact, while the values below the threshold are filtered as noises.

Unlike the rigid-body simulation, the real tactile sensors are wrapped in the deformable finger pulps. When the finger is in contact, the sensors around the actual contact regions often return positive signals due to pressures from deformation. We set the threshold of the tactile readings manually based on grasp experiments, which has to filter out such signals and sensory noises, but pass any signal from the sensors in contact.

6.3.4.3 Domain randomization

Besides the mean values of optimized dynamics parameters, the CMA-ES algorithm also provides their standard deviations. At the beginning of each training episode, we sampled the parameters, namely the PD gains of actuated joints and passive stiffness/damping of the backlash joints, from the optimized normal distributions. We also randomized other domain parameters to increase policy generalization ability, namely the stick weight, radius and the friction co-efficient between it and the fingers.

When training the precision manipulation skills with reinforcement learning, a common challenge is sufficient exploration of the state space, because the object is likely to be dropped at the early stage of training, and as the policy converges, the exploration becomes more limited. Policies trained with deficient data space tend to be sensitive to disturbances and uncertainties. To tackle such problems, we generated a data-set of random finger joint positions and object poses, and the training environment was initialized by sampling from the data-set at the beginning of each episode.

For better policy convergence and maneuverability, in candidate initial configurations, all fingers are in contact with the stick. The formed grasp does not have to be stable, because (a) it is an instantaneous state during a dynamic motion, and (b) the policy should learn to recover from the unstableness and continue the manipulation task. The proposed initial state generation method is described in Algorithm 2. We first sampled the finger joint positions and stick pose from uniform distributions, where the distribution ranges ensure the stick is between the fingers. Then we fixed the stick at the sampled pose, which cannot be moved by any external force. We closed the fingers until they contacted with the stick, and the current joint positions and stick pose were stored as one candidate initial state for policy training, as demonstrated in Figure 6.6.

Algorithm 2: Generating initial states for policy training.

Input: Joint positions distribution ρ_{jnt}, stick pose distribution ρ_{obj}

Output: A set for finger joint positions and stick poses: $\{\boldsymbol{q}_k, \boldsymbol{p}_k\}, k \in \{1, ..., N\}$

while $k \leq N$ **do**

- Sample from distributions: $\boldsymbol{q}_s \sim \rho_{\text{jnt}}, \boldsymbol{p}_s \sim \rho_{\text{obj}}$;
- Set starting finger joint positions $\boldsymbol{q}_s$;
- Fix the stick at the sampled pose $\boldsymbol{p}_s$;
- Close the finger joints until contacts detected;
- Add current data pair $\{\boldsymbol{q}_t, \boldsymbol{p}_t\}$ into the state set;

6.4 Experiments

In this section we presented the performances of learned policies in both simulated and real environments. We validated the effectiveness of tactile feedback by comparing the policies trained with different observation on the object.

6.4.1 Simulation Performance

To validate the learning algorithm, we trained four policies with different reference stick pose trajectories, namely drawing the straight line, the circle, the number eight and the spiral with stick end-point. Figure 6.7 demonstrates reference and real trajectories of the stick lower end-point positions, indicating that the trained policies are able to manipulate the stick to follow the real-time pose references. The spiral trajectories are most challenging to the robot as it requires delicate maneuvers especially at the inner laps, thus the resultant trajectories have more errors.

6.4.2 Sim-to-real Transfer

To validate the trained policies in simulation, we deployed them to the real robot hand with tactile sensors. The high level policy loop runs at 50 Hz, taking the measured joint positions and tactile information as feedback, and sending

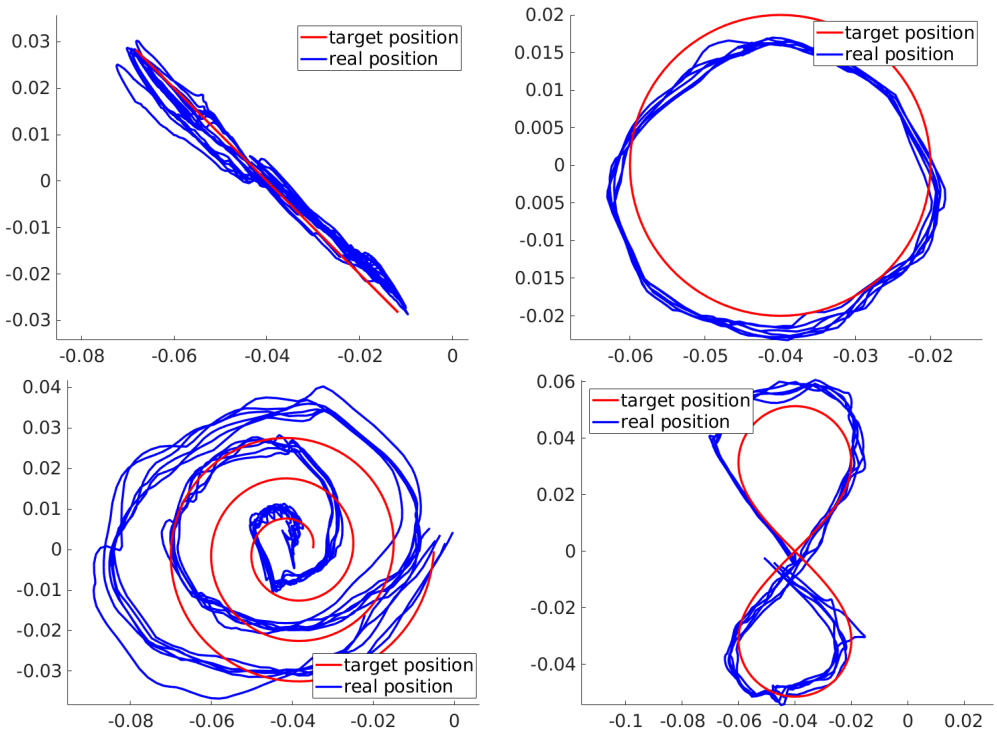


Figure 6.7: The reference (red) and actual (blue) trajectories of the stick lower endpoint on the X-Y plane. Four types of reference trajectories with increasing difficulties: the straight line, the circle, the spiral and the number eight. Each plot includes 10 trajectories and each trajectory starts from the center. In the 'circle' subplot we choose the second lap for clearness.

the commands to the low level joint position controller which runs at 1 kHz. Each trial starts from a configuration where the stick is already grasped by the three fingertips as illustrated in Figure 6.6. During the manipulation the robot hand is fixed at a certain pose by the robot arm. Figure 6.8 demonstrates the snapshots of running the trained policies in the real-world experiments, showing that the proposed DRL-based algorithm is able to acquire dexterous motor skills of manipulating the stick to follow different trajectories.

Due to the deformation of the materials wrapping the tactile sensors, the readings of some sensors might drift, returning positive values even when there is no contact. Since we binarize the tactile readings with a fixed threshold, such

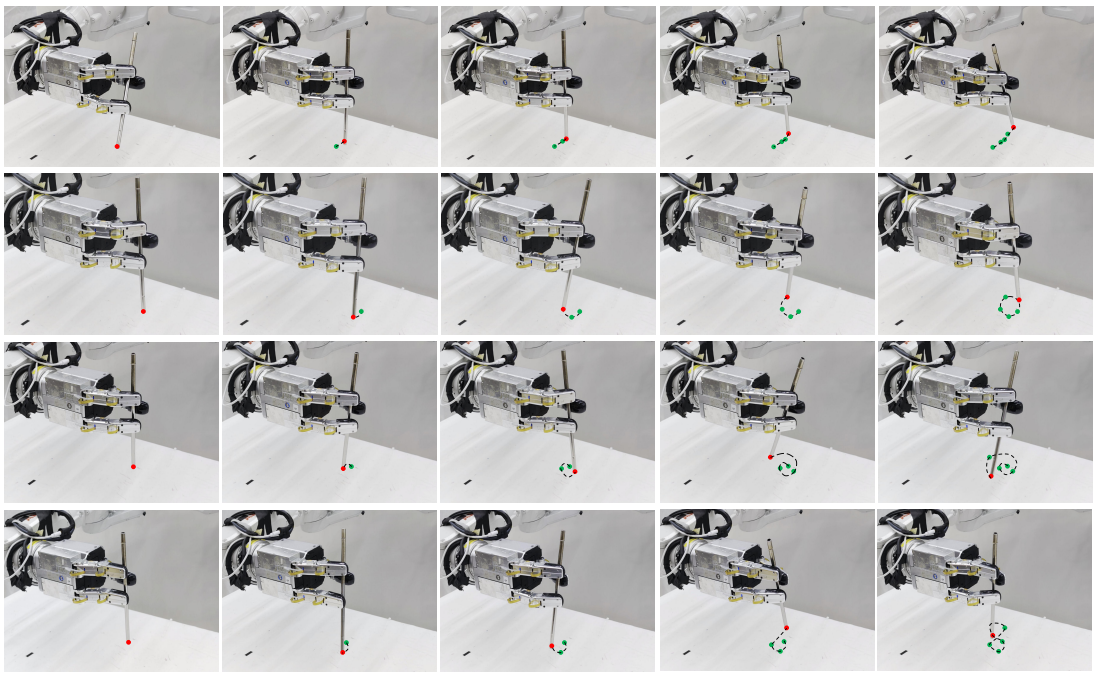


Figure 6.8: Snapshots of the real robot experiments, where the stick is manipulated to follow four different trajectories. Note that the stick is not contacted with the table. The red dots denote the current position of stick lower endpoint, while the green dots denote previous positions. The trajectories of the endpoint are illustrated as black dashed lines.

temporary sensor drifts lead to errors in discrimination of contact states. Hence we calibrate the tactile sensors before every manipulation trial. We first log the sensor readings without any contact in a period of time, and then use the mean values $\overline{s'}$ as the offsets. The calibrated tactile readings are:

$$s(t) = \max(s^*(t) - \overline{s'},\ 0), \tag{6.4}$$

where s^* denotes the raw tactile readings. We regard the negative values as the invalid and set them to 0.

We compared the tactile feedback on one fingertip during the circle-drawing stick manipulation task in both simulation and reality, as shown in Figure 6.9. The detected contact position trajectory is less continuous in real-world experiments. We set the contact position to $[0,0,0]$ when no tactile sensor is activated (see the

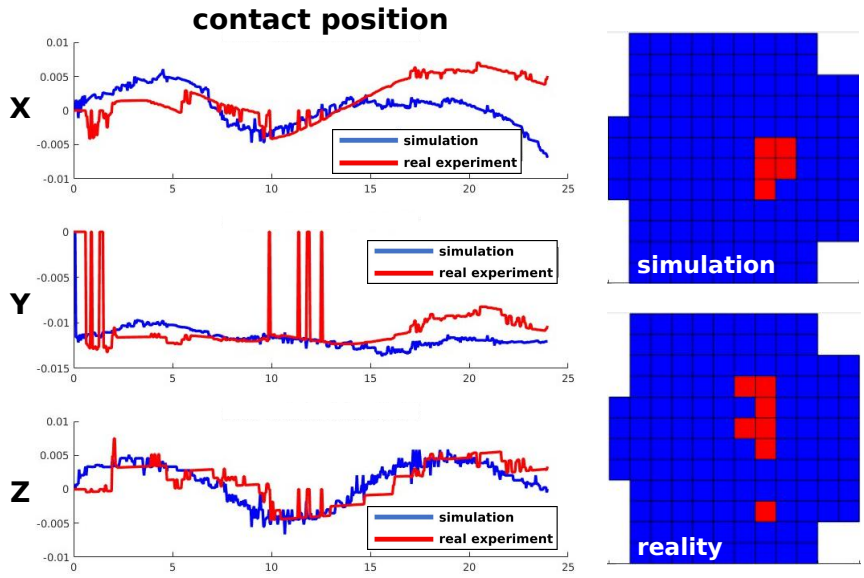


Figure 6.9: Left column: Contact position on a fingertip during the manipulation in both simulation (blue) and real-world experiment (red), which is averaged from the positions of activated tactile sensors, in fingertip local coordination. Right column: Examples of activated tactile sensors (red blocks) in simulation and reality during the stick manipulation. The fingertip is pointing towards the right.

spikes in subplots). This is because the real tactile sensors are less sensitive than the simulated ones. Unlike the simulated tactile sensors which are placed on the finger surfaces, the real sensors are beneath the deformable surfaces. During the fingertip-stick interaction, if the contact force fails to produce enough deformation to the finger surface, the tactile sensors cannot generate readings over the threshold and thus regarded as inactivated.

Moreover, the finger deformation lags behind the changes of actual contact positions which exacerbated the discontinuity. The right column of Figure 6.9 illustrates the examples of activated tactile sensors. Compared with simulation, the detected contact region in real-world experiment has more irregular margins, due to the uncertainties in fingertip deformation and inconsistent sensitivities of different sensor. Though trained with pure simulated tactile information, the

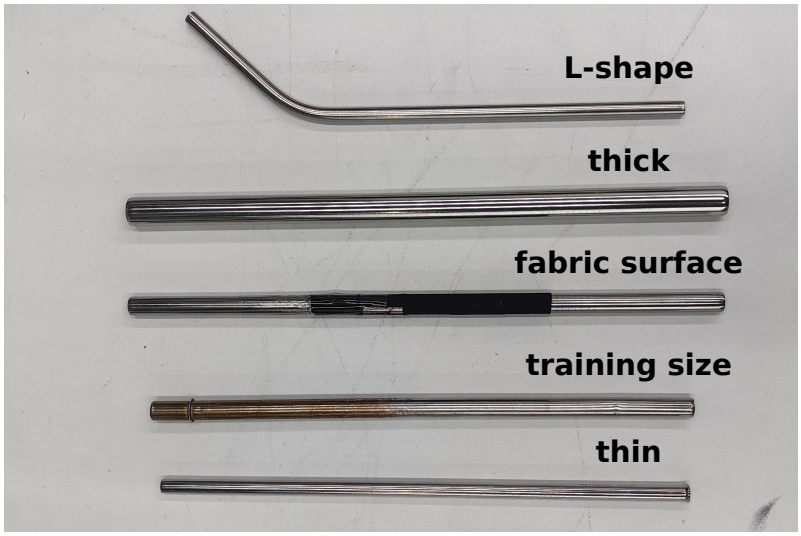


Figure 6.10: Sticks with different radiuses, weights and surfaces (stainless steel and fabric) are used to validate the policy in real-world experiments.

policies can adapt to the tactile sim2real gap and perform well in real-world experiments.

We evaluated the generalization ability of the trained policies over different objects, as shown in Figure 6.10. With real-time tactile perception, though trained with a single stick in simulation, the policies can adapt to diversities in the stick radiuses, weights and surfaces.

6.4.3 Comparative Study

To validate the effectiveness of the tactile information in learning dexterous in-hand manipulation, we conducted the comparison study where policies are trained with different observation spaces. In comparison to (a) contact center positions on each finger, we trained the policies with (b) object pose, (c) object pose and contact center positions, (d) raw tactile information, (e) object pose and the binary information of whether each finger is in contact. For each setting we trained the policy for three runs with random seed and the average learning curve is shown in Figure 6.11. Policies using contact center positions achieved the best perfor-

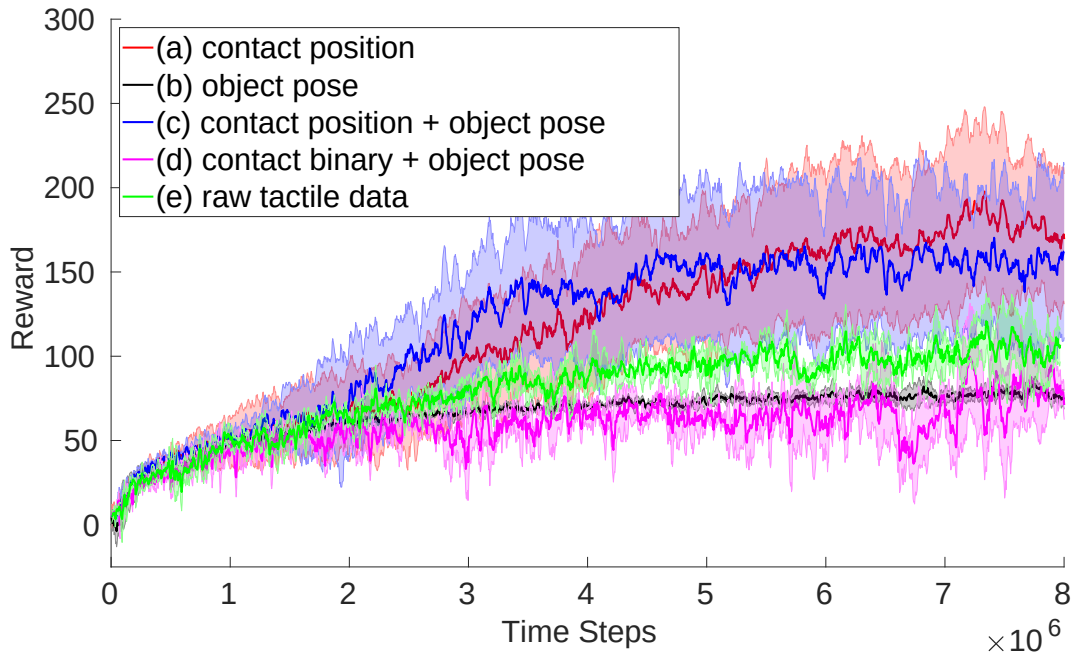


Figure 6.11: Learning curves of policies using different perception on the object. Training of each setting is repeated for five times with random seed. Policies using contact positions outperforms others, and adding object pose does not further improve the performance.

mance, and adding the ground truth object pose into observation space does not provide further improvement. Moreover, policies trained with object pose solely cannot achieve comparable performance, showing that the tactile information is more essential than visual information.

The tactile information of binary contact checks for three fingers (1×3 dimensional) is too sparse for learning feasible manipulation policies, while the raw tactile data (128×3 dimensional) requires more complex network structure (e.g. graph convolutional networks) to distill the spatial information. Compared with them, the contact center positions (3×3 dimensional) provide more direct and useful information, and achieves better performance in stick rotation tasks with the same size of training data and network structure.

6.5 Conclusion

In this work, we achieved the tactile-based dexterous in-hand manipulation via deep reinforcement learning. The trained policies can manipulate the stick to follow different real-time reference trajectories with robot fingers equipped with distributed tactile sensors. Gathering training data with real robot is time-consuming and dangerous, thus we utilize physics simulator to train the polices and transfer the policies to real-world robot directly. To mitigate the sim2real gap, we modeled and calibrated the finger joints features, binarized the tactile signals and introduced domain randomization during the training. The trained policies perform well in real-world experiments and can generalize to novel objects and tasks. To evaluate the effectiveness of tactile feedback in dexterous in-hand manipulation, we conducted the comparison study, training policies with different state spaces. The policies trained with perception of contact positions outperform the others, indicating that the tactile feedback is essential in such tasks.

Although the binarized tactile information is feasible for dexterous in-hand manipulation tasks, the performance can be further improved with denser information. Continuous tactile readings provide not only the regions of contact but also the pressure distribution, implying the deformation of fingers and the exerted force on the object. In the future work, we will model the non-linear relation between tactile readings and surface pressure and utilize the continuous signals as tactile feedback instead of Boolean ones.

Chapter 7

Discussions

In this thesis we study the learning of robotic motor skills of grasping and manipulation. We focus on the implementation of model-free deep reinforcement learning and supervised learning from human demonstration data. The trained control policies are fully autonomous and reactive, generating the control actions to the robot based on the real-time perception of the environment.

Three major topics are researched in the thesis, covering the pre-grasp phase, grasping phase and post-grasp phase respectively: dynamic reaching and grasping of moving objects (Chapter 3, 5), adaptive grasping of property-unknown objects (Chapter 4) and dexterous in-hand manipulation of slender objects (Chapter 6). Chapter 3 presents the learning of a single controller for dynamic reaching, grasping and re-grasping of objects sliding on the ground with coordinated hand-arm motion. In Chapter 5 we extend the policy training method into a multi-modular framework. A gating network is trained to merge the sub-modules, and the close cross-module coordination enables the hand-arm system to react quickly to the uncertainties, and catch the flying objects with mitigated impact force. We further implement model-free DRL to in-hand manipulation in Chapter 6, focusing on the application of tactile sensors and sim-to-real transfer. When the robot

is difficult to model, training the control policy with DRL in simulation is no longer an appealing option. For example in Chapter 4, the robotic fingers are cable driven and the force sensors are mounted at the actuator-side. Therefore we study the supervised learning from human demonstration data, to achieve the adaptive grasping of objects with unknown physical properties. For training data acquisition, we record human hand motion with a camera and teleoperate the robotic hand to grasp various objects via kinematic mapping. The control policy is encoded as a neural network, and three different structure designs are evaluated and compared. History proprioceptive data is proved to be more effective than the instantaneous data in the adaptive grasping.

In Chapter 3, 5 and 6 we study the implementation of model-free DRL in robotic grasping and manipulation, where five different policies are trained in the trial-and-error manner. Through the implementation of different tasks, we summarize a general pipeline of policy acquisition, from setting up simulation environment to deployment in real robot. The performances of trained policies proves the effectiveness of the pipeline, and the diversity amongst the accomplished tasks indicates that the pipeline can be extended to a broad range of manipulation robotic tasks. Moreover the modular structure in Chapter 5 shows that a long-sequential complex task can be divided into several sub-tasks, and each sub-task can be trained with the proposed pipeline. The pipeline can be divided into four major steps: DRL algorithm selection, simulation set-up, reward design and sim-to-real transfer. In the subsequent paragraphs we discuss our experiences and the empirical principles in designing each step.

DRL algorithm selection should considers the robotic task, and the way to collect training data. For tasks with discrete action space, a group of algorithms developed from deep Q-learning (DQN) are available, e.g. Double DQN (Van Hasselt et al., 2016) and dueling double DQN (Wang et al., 2016). However,

robotic grasping and manipulation usually require operations with continuous action space. Deep deterministic policy gradients (DDPG) (Lillicrap et al., 2015) is firstly extended from DQN to continuous actions. To solve the overestimation of Q-values of DDPG, Haarnoja et al. (2018) proposed soft actor critic (SAC) algorithm, which combines Q-learning with maximum entropy RL. Twin delayed DDPG (TD3) also addresses the problem of sensitivity to hyper-parameters of DDPG by introducing some critical tricks (Fujimoto et al., 2018). Aforementioned off-policy algorithms are more sample efficient than on-policy ones, and are compatible with human demonstrations. For tasks that can be trained in pure simulation, on-policy methods are generally more stable in policy learning. TRPO is one of the most famous on-policy DRL algorithm in recent decade. Developed from TRPO, PPO methods mitigate the implementation difficulties while maintain the reliable performances. Amongst the different algorithms, TD3, SAC and PPO are generally considered better than others with respect to the learning performances, but it is inconclusive that one specific algorithm significantly outperforms the rest on most robotic tasks. Based on the successful implementation of PPO in several robotic domains, in this thesis we use it as the backbone algorithm to train the grasping and manipulation policies.

Simulation set-up is essential for policy learning. First priority is to make sure that the simulation settings are well-matched with reality. Then the task settings should promote the policy exploration and the emergent of desired behaviours. For example in Chapter 3, in order to let the policy learn to keep the fingers open while approaching the object, the finger joint positions are randomized at the beginning of each training episode. For the states that are critical for policy learning but difficult to be visited via random exploration, we can manually set them as the initial states of training episodes to provide enough training data, e.g. the two special states for learning re-grasp motion in Section 3.3.2. At the early stage of development, several rounds of policy training and simulation

modifications may be applied in turns before final simulation set-up is fixed. Such *testing* training rounds – usually starting from simplified tasks and gradually increasing the task complexity – helps the developers to detect any defects in simulation set-up and reward function design. For example when learning the in-hand manipulation skills of stirring with a stick in Chapter 6, in the initial *testing* training rounds, the target stick pose is fixed. In other words we firstly let the policy learn to hold the stick still. When the simulation set-up is determined and fixed, proper randomization in the task specifications is necessary in acquisition of policy generalization ability. The training task randomization should distribute within the potential task space that the robot might encounter. For example in Chapter 3 and Chapter 5 the object moving velocity and trajectory are sampled from pre-defined ranges. In Chapter 6 the initial finger joint positions and stick pose of each training episode are randomized, to enhance the motor skills of manipulating the stick from undesired states.

Reward design is the core component of RL, guiding the trained policy towards desired behaviours by evaluating the current state. Shaping the reward function is equivalent to implicitly encoding the human prior knowledge and understanding of the task into mathematical formula. The policy that can maximize the accumulated reward naturally accords with the user expectation. Sparse rewards are hardly feasible for learning complex continuous motor skills, and hence we use dense reward functions throughout the thesis.

A proper reward function should accurately represent the desired behaviours, and if necessary, prevent the unexpected behaviours. Since the optimal policy can have different behaviour in terms of different situation, the reward function is conventionally a linear combination of multiple terms where each term explicitly describes one specific behaviour. Taking the five-term reward function of Equation 3.1 to learn dynamic reaching and grasping as an example, two terms guide

the reaching motion, and two terms promote the power grasp motion, and one term penalize the object translational velocity to prevent post-grasp movements. The weights for reward terms are tuned via the performance validation after each round of policy training. Plotting the time-reward curve of each reward term can be used inspect the status of behaviour learning. We find that the variation magnitude and absolute value of reward terms does not necessarily represent their importance level. For example in Figure 3.5, the absolute values of r_3 and r_5 are smaller than others, and the variation of r_2 is not obvious. However they are all necessary in learning the desired behaviours, as shown in the ablation study in Section 3.4.4. Reward function can be different in different situations. For example the piece-wise reward term defined as Equation 5.4 guides the hand velocity before and after the contact with object.

We summarize some empirical principles of reward design: (a) Each reward term should be normalized and bounded, to prevent the policy from over-exploitation of certain reward terms, and to facilitate the weight tuning. (b) With the premise of policy performances, the reward function is the simple the better. The difficulty of reward shaping and chances of conflict between different reward terms increase as the complexity of reward function increases. Over-complicated reward function might introduce excessive prior knowledge and restricts the policy exploration. (c) Reward terms promoting desired behaviours are more important than penalty terms preventing undesired behaviours. The emergence of desired behaviours should be guaranteed first, and the penalty terms should not impair learned desired behaviours. (d) Policy learning of DRL has randomness, and hence it is better to train the policy under same settings for multiple times with random seeds before modifying the reward.

Sim-to-real transfer is the last but not least barrier for policy deployment. In Chapter 6 the problem is addressed from two aspects: (a) mitigating the

discrepancies between simulated and real environments via model calibration, and (b) increasing the robustness of trained policies to environmental variations and uncertainties via domain randomization.

Full cognition of the robot mechanical structure is essential for simulation set-up, since the URDF files may not describe complete robot features. For robotic grasping and manipulation, we find that the major sim-to-real gap comes from the *joint model* and *friction coefficient*. Taking the robotic hand in Chapter 6 as an example, the joints have uncontrollable backlashes due to the gaps between gears, and back driving self-lock feature due to the worm gear mechanism. The aforementioned properties are not presented in the URDF files and need to be modeled and calibrated explicitly. To make sure the simulated and real robots have same response with same action input, we calibrate the dynamics feature of the joint control via the trajectory following experiments. The simulated friction coefficient can be determined by the real surface material, but we find that coefficient is not even on the real robot surfaces with same material. Such feature is hard to precisely modeled and hence we randomize the simulated friction within a reasonable range. Domain randomization trades the policy optimality for generalization ability. For the domains with large variations, e.g. the object physical properties and tactile sensor readings, the parameters can be sampled from a wider range. While for the domains with more certainty, e.g. the robot surface friction and calibrated joint damping, the magnitude of randomization can be smaller.

This thesis study the implementation of model-free DRL and learning from demonstrations, presenting four case studies of different robotic grasping and manipulation tasks. For future research based on the ideas and contents of this thesis, we propose three possible exploration directions:

Combine multiple sub-policies for complex tasks. Daily tasks are usually

composites of several primitive sub-tasks. For example, “taking a cup of milk out of the microwave oven” consists of opening the oven door, picking up the cup, placing the cup on the table and closing the oven door. Each task is unique and it is difficult for a single model to acquire the multi-modal skills, due to the catastrophic forgetting problem of DRL. Hence a promising way is to construct a hierarchical structure consisting of multiple expert policies and one central control policy. Each expert policy possess the motor skills for one specific task, and the central control policy works as the *brain*, invoking and coordinating the expert policies. Chapter 5 provides a primitive demo of the hierarchical structure, where the reaching and grasping networks are expert policies, and the gating network serves as the central control policy. The trained expert policies should be scalable and reusable, to form a *skill library*. The proposed DRL-based framework can be an option for developing the expert policies, though the efficiency needs to be improved. The central control policy should be capable of continuous learning (also known as incremental or lifelong learning), acquiring new knowledge efficiently when new expert policies are available and when new tasks arise.

Combine learning with control methods. Learning-based algorithms like DRL are adept at decision making in high-dimensional state space, while conventional control methods are generally more stable and interpretable in low-level motor skills. In this thesis we have not used advanced control methods since the tasks do not involve precise force regulation. However, for tasks like glass polishing and chip installation that have force/position constraints, pure end-to-end learning approaches struggles to achieve comparable performances with control approaches using explicit models. Combining the generalization ability of learning and stability of control methods shows good results in bipedal or quadrupedal walking, and hence it is promising to explore along this direction in the field of robotic grasping and manipulation.

Combine DRL with demonstration data. Adding demonstrations into DRL algorithms exploits both the sample efficiency property of expert data and the exploration capability of DRL. In Chapter 4 we collect training data from hundreds of grasps which is time-consuming. Humans can learn new skills by observing only a few demonstrations and constantly practicing. Endowing the robot with comparable learning capability is worth for exploration. Reward shaping in DRL algorithms is non-trivial. *Automatically extracting the reward signals from demonstrations* is a possible solution. Some related research has been done following this concept, but a general algorithm for learning complex motor skills, e.g. the stirring motion in Chapter 6, is still missing. Currently the demonstrations are usually provided by a single expert, and here comes the questions: What if there are several different demonstrators? What if the demonstrators are not experts? *Addressing inconsistent or non-optimal demonstration data* remains vacant in research field. The desired method should extract the key requirements and constraints from the problematic data, and learn a policy that outperforms the demonstrations.

Bibliography

A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg. Transferring dexterous manipulation from GPU simulation to a remote real-world trifinger. *CoRR*, abs/2108.09779, 2021.

S. Amiri, M. Shokrolah Shirazi, and S. Zhang. Learning and reasoning for robot sequential decision making under uncertainty. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:2726–2733, 04 2020. doi: 10.1609/aaai.v34i03.5659.

O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1): 3–20, 2020. doi: 10.1177/0278364919887447.

P. Ardon, M. Dragone, and M. S. Erden. Reaching and grasping of objects by humanoid robots through visual servoing. In D. Prattichizzo, H. Shinoda, H. Z. Tan, E. Ruffaldi, and A. Frisoli, editors, *Haptics: Science, Technology, and Applications*, 2018.

A. Bicchi. Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16

(6):652–662, 2000. doi: 10.1109/70.897777.

A. Billard and D. Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019. doi: 10.1126/science.aat8414. URL https://www.science.org/doi/abs/10.1126/science.aat8414.

J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on robotics*, 30(2):289–309, 2013.

J. Bohg, A. Morales, T. Asfour, and D. Kragic. Data-driven grasp synthesis-A survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014. ISSN 15523098. doi: 10.1109/TRO.2013.2289018.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

B. Bäuml, T. Wimböck, and G. Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2592–2599, 2010. doi: 10.1109/IROS.2010.5651175.

B. Bäuml, O. Birbach, T. Wimböck, U. Frese, A. Dietrich, and G. Hirzinger. Catching flying balls with a mobile humanoid: System overview and design considerations. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 513–520, 2011. doi: 10.1109/Humanoids.2011.6100837.

R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. Adelson, and S. Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *IEEE Robotics and Automation Letters*, PP:1–1, 07 2018a. doi: 10.1109/LRA.2018.2852779.

R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *CoRR*, 2018b.

B. Calli and A. M. Dollar. Vision-based model predictive control for within-hand precision manipulation with underactuated grippers. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2839–2845, 2017. doi: 10.1109/ICRA.2017.7989331.

B. Çalli, A. Walsman, A. Singh, S. S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols. *CoRR*, abs/1502.03143, 2015.

B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017. doi: 10.1177/0278364917700714.

I.-M. Chen and J. Burdick. Finding antipodal point grasps on irregularly shaped objects. *IEEE Transactions on Robotics and Automation*, 9(4):507–512, 1993. doi: 10.1109/70.246063.

J. Chen, S. Shen, and H. Y. K. Lau. Hitting flying objects with learning from demonstration. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 55–60, 2017a. doi: 10.1109/ICAR.2017.8023496.

J. Chen, S. Shen, and H. Y. K. Lau. Hitting flying objects with learning from demonstration. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 55–60, 2017b. doi: 10.1109/ICAR.2017.8023496.

T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 297–307. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/chen22a.html.

H. Choi, C. Crump, C. Duriez, A. Elmquist, G. D. Hager, D. Han, F. Hearl,

J. K. Hodgins, A. Jain, F. A. Leve, C. Li, F. Meier, D. Negrut, L. Righetti, A. Rodriguez, J. Tan, and J. C. Trinkle. On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences of the United States of America*, 118, 2020.

A. Church, J. W. James, L. Cramphorn, and N. F. Lepora. Tactile model O: fabrication and testing of a 3d-printed, three-fingered tactile robot hand. *CoRR*, abs/1907.07535, 2019.

E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

E. De Coninck, T. Verbelen, P. Van Molle, P. Simoens, and B. Dhoedt. Learning robots to grasp by demonstration. *Robotics and Autonomous Systems*, 127, 2020. ISSN 09218890. doi: 10.1016/j.robot.2020.103474.

Z. Deng, Y. Jonetzko, L. Zhang, and J. Zhang. Grasping Force Control of Multi-Fingered Robotic Hands through Tactile Sensing for Object Stabilization. *Sensors*, 20(4):1050, feb 2020. ISSN 1424-8220. doi: 10.3390/s20041050.

M. Deniša, A. Gams, A. Ude, and T. Petrič. Learning compliant movement primitives through demonstration and statistical generalization. *IEEE/ASME transactions on mechatronics*, 21(5):2581–2594, 2015.

R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.

K. Dong, K. Pereida, F. Shkurti, and A. P. Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. *CoRR*, abs/2003.07489, 2020.

H. Duan, P. Wang, Y. Huang, G. Xu, W. Wei, and X. Shen. Robotics dexterous grasping: The methods based on point cloud and deep learning. *Frontiers in Neurorobotics*, 15, 2021. ISSN 1662-5218. doi: 10.3389/fnbot.2021.

658280. URL https://www.frontiersin.org/articles/10.3389/fnbot.2021.658280.

J. C. Dunn†. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104, 1974. doi: 10.1080/01969727408546059.

S. Ekvall and D. Kragić. Grasp recognition for programming by demonstration. *Proceedings - IEEE International Conference on Robotics and Automation*, 2005:748–753, 2005. doi: 10.1109/ROBOT.2005.1570207.

J. Eschmann. *Reward Function Design in Reinforcement Learning*, pages 25–33. Springer International Publishing, Cham, 2021. ISBN 978-3-030-41188-6. doi: 10.1007/978-3-030-41188-6_3. URL https://doi.org/10.1007/978-3-030-41188-6_3.

F. Ficuciello, A. Migliozzi, G. Laudante, P. Falco, and B. Siciliano. Vision-based grasp learning of an anthropomorphic hand-arm system in a synergy-based control framework. *Science Robotics*, 4(26), jan 2019. ISSN 24709476. doi: 10.1126/SCIROBOTICS.AAO4900.

C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 49–58, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/finn16.html.

E. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

S. Funabashi, T. Isobe, F. Hongyi, A. Hiramoto, A. Schmitz, S. Sugano, and T. Ogata. Multi-fingered in-hand manipulation with various object properties using graph convolutional networks and distributed tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):2102–2109, 2022. doi: 10.1109/LRA.2022.3142417.

M. Gou, H. Fang, Z. Zhu, S. Xu, C. Wang, and C. Lu. RGB matters: Learning 7-dof grasp poses on monocular RGBD images. *CoRR*, abs/2103.02184, 2021.

A. L. Gunderman, J. A. Collins, A. L. Myers, R. T. Threlfall, and Y. Chen. Tendon-driven soft robotic gripper for blackberry harvesting. *IEEE Robotics and Automation Letters*, 7(2):2652–2659, 2022. doi: 10.1109/LRA.2022.3143891.

A. K. Gupta, L. Aitchison, and N. F. Lepora. Tactile image-to-image disentanglement of contact geometry from motion-induced shear. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 14–23. PMLR, 08–11 Nov 2022.

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

K. Hang, J. A. Stork, N. S. Pollard, and D. Kragic. A framework for optimal grasp contact planning. *IEEE Robotics and Automation Letters*, 2017.

S. Hangl, E. Ugur, S. Szedmak, J. Piater, and A. Ude. Reactive, task-specific

object manipulation by metric reinforcement learning. In *2015 International Conference on Advanced Robotics (ICAR)*, 2015.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398.

Y. Hao, Z. Gong, Z. Xie, S. Guan, X. Yang, Z. Ren, T. Wang, and L. Wen. Universal soft pneumatic robotic gripper with variable effective length. In *2016 35th Chinese Control Conference (CCC)*, pages 6109–6114, 2016. doi: 10.1109/ChiCC.2016.7554316.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

W. Hong and J.-J. E. Slotine. Experiments in hand-eye coordination using active vision. In O. Khatib and J. K. Salisbury, editors, *Experimental Robotics IV*, 1997.

W. Hu, C. Yang, K. Yuan, and Z. Li. Learning motor skills of reactive reaching and grasping of objects. In *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

J.Bohg, A.Morales, T.Asfour, and D.Kragic. Data-driven grasp synthesis: A survey. *IEEE Transactions on Robotics*, 2014.

Y.-B. Jia, M. Gardner, and X. Mu. Batting an in-flight object to the target. *The International Journal of Robotics Research*, 38(4):451–485, 2019. doi: 10.1177/0278364918817116.

E. K. Hashimoto. *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*. World Scientific, 1993.

M. Kaboli, K. Yao, and G. Cheng. Tactile-based manipulation of deformable objects with dynamic center of mass. In *IEEE-RAS International Conference on Humanoid Robots*, pages 752–757. IEEE Computer Society, dec 2016. ISBN 9781509047185. doi: 10.1109/HUMANOIDS.2016.7803358.

D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673, 29–31 Oct 2018.

D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation*, 2015.

G. Khandate, M. Haas-Heger, and M. Ciocarlie. On the feasibility of learning finger-gaiting in-hand manipulation with intrinsic sensing. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2752–2758, 2022. doi: 10.1109/ICRA46639.2022.9812212.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5): 943–957, 2011a. doi: 10.1109/TRO.2011.2159412.

S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical sys-

tems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5): 943–957, 2011b.

S. Kim and A. Billard. Estimating the non-linear dynamics of free-flying objects. *Robotics and Autonomous Systems*, 60(9):1108–1122, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.05.022.

S. Kim, A. Shukla, and A. Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014. doi: 10.1109/TRO.2014.2316022.

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt. One-shot learning and generation of dexterous grasps for novel objects. *The International Journal of Robotics Research*, 35(8):959–976, 2016. doi: 10.1177/0278364915594244.

M. Körber, J. Lange, S. Rediske, S. Steinmann, and R. Glück. Comparing popular simulation environments in the scope of robotics and reinforcement learning. *arXiv preprint arXiv:2103.04616*, 2021.

O. Kroemer, R. Detry, J. Piater, and J. Peters. Combining active learning and reactive control for robot grasping. *Robotics and Autonomous Systems*, 2010.

O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *The Journal of Machine Learning Research*, 22(1):1395–1476, 2021.

K. Kronander, M. Khansari, and A. Billard. Incremental motion learning with locally modulated dynamical systems. *Robotics and Autonomous Systems*, 70: 52–62, 2015.

V. Kumar, Y. Tassa, T. Erez, and E. Todorov. Real-time behaviour synthesis for dynamic hand-manipulation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6808–6815, 2014. doi: 10.1109/ICRA.2014.6907864.

M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, D. Jayaraman, and R. Calandra. Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation. *IEEE Robotics and Automation Letters*, 5 (3):3838–3845, 2020. doi: 10.1109/LRA.2020.2977257.

R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters. Trajectory planning for optimal robot catching in real-time. In *2011 IEEE International Conference on Robotics and Automation*, pages 3719–3726, 2011. doi: 10.1109/ICRA.2011.5980114.

T. Lampe and M. Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013.

J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47): eabc5986, 2020.

M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019. doi: 10.1109/ICRA.2019.8793485.

S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills

with guided policy search. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015, 01 2015. doi: 10.1109/ICRA.2015.7138994.

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *CoRR*, 2016.

Q. Li, O. Kroemer, Z. Su, F. F. Veiga, M. Kaboli, and H. J. Ritter. A review of tactile information: Perception and action through touch. *IEEE Transactions on Robotics*, 36(6):1619–1634, 2020. doi: 10.1109/TRO.2020.3003230.

M. V. Liarokapis and A. M. Dollar. Learning task-specific models for dexterous, in-hand manipulation with simple, adaptive robot hands. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2534–2541, 2016. doi: 10.1109/IROS.2016.7759394.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Y. Lin, S. Ren, M. Clevenger, and Y. Sun. Learning grasping force from demonstration. In *2012 IEEE International Conference on Robotics and Automation*, pages 1526–1531, 2012a. doi: 10.1109/ICRA.2012.6225222.

Y. Lin, S. Ren, M. Clevenger, and Y. Sun. Learning grasping force from demonstration. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1526–1531, 2012b. doi: 10.1109/ICRA.2012.6225222.

Y. Lin, J. Lloyd, A. Church, and N. F. Lepora. Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch. *IEEE Robotics and Automation Letters*, 7(4):10754–10761, 2022. doi: 10.1109/LRA.2022.3195195.

C. K. Liu and D. Negrut. The role of physics-based simulators in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:35–58, 2021.

K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 35–42, 2018.

Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. *CoRR*, 2018.

Z. Lu, N. Wang, and C. Yang. A constrained dmps framework for robot skills learning and generalization from human demonstrations. *IEEE/ASME Transactions on Mechatronics*, pages 1–1, 2021. doi: 10.1109/TMECH.2021.3057022.

J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, 2017a.

J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017b.

T. B. Martin, R. O. Ambrose, M. A. Diftler, R. Platt, and M. J. Butzer. Tactile gloves for autonomous grasping with the NASA/DARPA Robonaut. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2004, pages 1713–1718. Institute of Electrical and Electronics Engineers Inc., 2004. doi: 10.1109/robot.2004.1308071.

N. Marturi, M. Kopicki, A. Rastegarpanah, V. Rajasekaran, M. Adjigble, R. Stolkin, A. Leonardis, and Y. Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Autonomous Robots*, 2019.

M. T. Mason, A. Rodriguez, S. S. Srinivasa, and A. S. Vazquez. Autonomous manipulation with a general-purpose simple hand. *The International Journal of Robotics Research*, 31(5):688–703, apr 2012. ISSN 0278-3649. doi: 10.1177/0278364911429978.

A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter. Using tactile sensing to improve the sample efficiency and performance of deep deterministic policy gradients for simulated in-hand manipulation tasks. *Frontiers in Robotics and AI*, 8:538773, 06 2021. doi: 10.3389/frobt.2021.538773.

H. Merzic, M. Bogdanovic, D. Kappler, L. Righetti, and J. Bohg. Leveraging contact forces for learning to grasp. In *2019 IEEE International Conference on Robotics and Automation*, 2019a.

H. Merzic, M. Bogdanovic, D. Kappler, L. Righetti, and J. Bohg. Leveraging contact forces for learning to grasp. In *2019 IEEE International Conference on Robotics and Automation*, 2019b.

H. Merzic, M. Bogdanovic, D. Kappler, L. Righetti, and J. Bohg. Leveraging contact forces for learning to grasp. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 2019-May, 2019c. ISBN 9781538660263. doi: 10.1109/ICRA.2019.8793733.

E. Misimi, A. Olofsson, A. Eilertsen, E. R. Øye, and J. R. Mathiassen. Robotic Handling of Compliant Food Objects by Robust Learning from Demonstration. In *IEEE International Conference on Intelligent Robots and Systems*, 2018a. doi: 10.1109/IROS.2018.8594368.

E. Misimi, A. Olofsson, A. Eilertsen, E. R. Øye, and J. R. Mathiassen. Robotic handling of compliant food objects by robust learning from demonstration. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6972–6979, 2018b. doi: 10.1109/IROS.2018.8594368.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

A. Mohseni-Kabir, C. Rich, S. Chernova, C. L. Sidner, and D. Miller. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 205–212, 2015.

D. Morrison, P. Corke, and J. Leitner. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *CoRR*, 2018a.

D. Morrison, J. Leitner, and P. Corke. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. In *Robotics: Science and System(RSS)*. Robotics: Science and Systems Foundation, aug 2018b. doi: 10.15607/rss.2018.xiv.021.

M. Müller, S. Lupashin, and R. D'Andrea. Quadrocopter ball juggling. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5113–5120, 2011. doi: 10.1109/IROS.2011.6094506.

A. Namiki, K. Hashimoto, and M. Ishikawa. A hierarchical control architecture for high-speed visual servoing. *The International Journal of Robotics Research*, 2003.

T. Narita, S. Nagakari, W. Conus, T. Tsuboi, and K. Nagasaka. Theoretical derivation and realization of adaptive grasping based on rotational incipient slip detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 531–537, 2020. doi: 10.1109/ICRA40945.2020.9196615.

R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg,

A. Morales, T. Asfour, D. Kragic, et al. Deep learning approaches to grasp synthesis: A review. *arXiv preprint arXiv:2207.02556*, 2022.

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287. Citeseer, 1999.

H. Nguyen and H. La. Review of deep reinforcement learning for robot manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 590–595, 2019. doi: 10.1109/IRC.2019.00120.

OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *CoRR*, 2018.

X. Pan, Y. You, Z. Wang, and C. Lu. Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*, 2017.

P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

T. Patten, K. Park, and M. Vincze. Dgcm-net: Dense geometrical correspondence matching network for incremental experience-based robotic grasping. *Frontiers in Robotics and AI*, 7, 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.00120. URL https://www.frontiersin.org/articles/10.3389/frobt.2020.00120.

X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

M. Pfanne, M. Chalon, F. Stulp, and A. Albu-Schaffer. Fusing joint measurements

and visual features for In-Hand object pose estimation. *IEEE Robotics and Automation Letters*, 3(4):3497–3504, oct 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2853652.

M. Pfanne, M. Chalon, F. Stulp, H. Ritter, and A. Albu-Schäffer. Object-Level Impedance Control for Dexterous In-Hand Manipulation. *IEEE Robotics and Automation Letters*, 5(2):2987–2994, apr 2020. ISSN 23773766. doi: 10.1109/LRA.2020.2974702.

L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, 2015.

E. Psomopoulou, D. Karashima, Z. Doulgeri, and K. Tahara. Stable pinching by controlling finger relative orientation of robotic fingers with rolling soft tips. *Robotica*, 36(2):204–224, feb 2018. ISSN 14698668. doi: 10.1017/S0263574717000303.

D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6284–6291, 2018. doi: 10.1109/ICRA.2018.8461039.

A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.049.

N. Ratliff, J. A. Bagnell, and S. S. Srinivasa. Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, 2007.

H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent ad-

vances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3, 2020. ISSN 2573-5144. doi: 10.1146/annurev-control-100819-063206.

A. Rodriguez. The unstable queen: Uncertainty, mechanics, and tactile feedback. *Science Robotics*, 6(54):eabi4667, 2021. doi: 10.1126/scirobotics.abi4667.

J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079, 2011. ISSN 15523098. doi: 10.1109/TRO.2011.2162271.

J. Romero, T. Feix, H. Kjellström, and D. Kragic. Spatio-temporal modeling of grasping actions. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 2103–2108, 2010. doi: 10.1109/IROS.2010.5650701.

N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *CoRR*, abs/2109.11978, 2021. URL https://arxiv.org/abs/2109.11978.

S. S. M. Salehian, M. Khoramshahi, and A. Billard. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics*, 32(2):462–471, 2016. doi: 10.1109/TRO.2016.2536749.

C. Schenck and D. Fox. Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR, 2018.

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

A. A. Shahid, L. Roveda, D. Piga, and F. Braghin. Learning Continuous Control Actions for Robotic Grasping with Reinforcement Learning. In *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, volume 2020-October, 2020. doi: 10.1109/SMC42975.2020.9282951.

L. Sievers, J. Pitz, and B. Bäuml. Learning purely tactile in-hand manipulation with a torque-controlled hand. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2745–2751, 2022. doi: 10.1109/ICRA46639.2022.9812093.

D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016. doi: 10.1038/nature16961.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

G. Solak and L. Jamone. Learning by demonstration and robust control of dexterous in-hand robotic manipulation skills. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8246–8251, 2019. doi: 10.1109/IROS40897.2019.8967567.

S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the Wild: Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, jul 2020. ISSN 23773766. doi: 10.1109/LRA.2020.3004787.

M. W. Spong, S. Hutchinson, M. Vidyasagar, et al. *Robot modeling and control*,

volume 3. Wiley New York, 2006.

W. Sun, A. Venkatraman, G. J. Gordon, B. Boots, and J. A. Bagnell. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *International conference on machine learning*, pages 3309–3318. PMLR, 2017.

B. Sundaralingam and T. Hermans. Relaxed-rigidity constraints: Kinematic trajectory optimization and collision avoidance for in-grasp manipulation. *CoRR*, abs/1806.00942, 2018.

Y. Suzuki, K. Koyama, A. Ming, and M. Shimojo. Grasping strategy for moving object using net-structure proximity sensor and vision sensor. In *2015 IEEE International Conference on Robotics and Automation*, 2015.

T. Takahashi, T. Tsuboi, T. Kishida, Y. Kawanami, S. Shimizu, M. Iribe, T. Fukushima, and M. Fujita. Adaptive grasping by multi fingered hand with tactile sensor based on robust force and position control. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 264–271, 2008. ISBN 9781424416479. doi: 10.1109/ROBOT.2008.4543219.

J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

S. Tian, F. Ebert, D. Jayaraman, M. Mudigonda, C. Finn, R. Calandra, and S. Levine. Manipulation by feel: Touch-based control with deep predictive models. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 818–824, 2019. doi: 10.1109/ICRA.2019.8794219.

J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Y.-Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang. Droid: Minimizing the reality gap using single-shot human demonstration. *IEEE Robotics and Automation Letters*, 6:3168–3175, 2021.

H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

H. van Hoof, T. Hermans, G. Neumann, and J. Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127, 2015. doi: 10.1109/HUMANOIDS.2015.7363524.

F. Veiga, R. Akrour, and J. Peters. Hierarchical tactile-based control decomposition of dexterous in-hand manipulation tasks. *Frontiers in Robotics and AI*, 7, 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.521448.

S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra. Tacto: A fast, flexible, and open-source simulator for high-resolution vision-based tactile sensors. *IEEE Robotics and Automation Letters*, 7(2):3930–3937, 2022. doi: 10.1109/LRA.2022.3146945.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8: 279–292, 05 1992. doi: 10.1007/BF00992698.

W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong. Dvgg: Deep variational grasp generation for dextrous manipulation. *IEEE Robotics and Automation Letters*, 7:1659–1666, 2022.

R. Wen, K. Yuan, Q. Wang, S. Heng, and Z. Li. Force-Guided High-Precision Grasping Control of Fragile and Deformable Objects Using sEMG-Based Force Prediction. *IEEE Robotics and Automation Letters*, 5(2):2762–2769, apr 2020. ISSN 23773766. doi: 10.1109/LRA.2020.2974439.

T. Wimböck, C. Ott, A. Albu-Schäffer, and G. Hirzinger. Comparison of object-level grasp controllers for dynamic dexterous manipulation. *The International Journal of Robotics Research*, 31(1):3–23, jan 2012. ISSN 0278-3649. doi: 10.1177/0278364911416526.

B. Wu, I. Akinola, J. Varley, and P. K. Allen. MAT: multi-fingered adaptive tactile grasping via deep reinforcement learning. In *3rd Annual Conference on Robot Learning*, volume 100, pages 142–161, 2019.

C. Xiong, M. Wang, Y. Tang, and Y. Xiong. Compliant grasping with passive forces. *Journal of Field Robotics*, 22(5):271–285, May 2005. ISSN 1556-4959. doi: 10.1002/rob.20064.

C. Yang, K. Yuan, W. Merkt, T. Komura, S. Vijayakumar, and Z. Li. Learning whole-body motor skills for humanoids. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 2018.

C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li. Multi-expert learning of adaptive legged locomotion. *Science Robotics*, 5(49), 2020.

T. Yoshikawa. Multifingered robot hands: Control for grasping and manipulation.

Annual Reviews in Control, 34(2):199–208, dec 2010. ISSN 13675788. doi: 10.1016/j.arcontrol.2010.09.001.

H. Yousef, M. Boukallel, and K. Althoefer. Tactile sensing for dexterous in-hand manipulation in robotics—a review. *Sensors and Actuators A: Physical*, 167(2):171–187, 2011. ISSN 0924-4247. doi: https://doi.org/10.1016/j.sna.2011.02.038. URL https://www.sciencedirect.com/science/article/pii/S0924424711001105. Solid-State Sensors, Actuators and Microsystems Workshop.

C. Yu and P. Wang. Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: A review. *Frontiers in Neurorobotics*, 16, 2022. ISSN 1662-5218. doi: 10.3389/fnbot.2022.861825. URL https://www.frontiersin.org/articles/10.3389/fnbot.2022.861825.

H. Yu, D. Guo, H. Yin, A. Chen, K. Xu, Y. Wang, and R. Xiong. Neural motion prediction for in-flight uneven object catching. *CoRR*, abs/2103.08368, 2021.

W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12), 2017a. ISSN 1424-8220. doi: 10.3390/s17122762.

W. Yuan, S. Dong, and E. H. Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12), 2017b. ISSN 1424-8220. doi: 10.3390/s17122762.

B. S. Zapata-Impata, P. Gil, and F. Torres. Learning Spatio temporal tactile features with a convLSTM for the direction of slip detection. *Sensors (Switzerland)*, 19(3), feb 2019. ISSN 14248220. doi: 10.3390/s19030523.

K. Zeng, R. Mottaghi, L. Weihs, and A. Farhadi. Visual reaction: Learning to play catch with your drone. In *2020 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR), pages 11570–11579, 2020. doi: 10.1109/CVPR42600.2020.01159.

J. Zhou, X. Chen, J. Li, Y. Tian, and Z. Wang. A soft robotic approach to robust and dexterous grasping. In *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, pages 412–417, 2018. doi: 10.1109/ROBOSOFT.2018.8404954.