Dexterity in Robotic Grasping, Manipulation and Assembly

Abstract

Dexterity in Robotic Grasping, Manipulation and Assembly

by

Yongxiang Fan

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

Industrial manipulators are programmed and integrated into different systems to deliver various functions. Traditional industrial manipulators are highly efficient and precise in mass production but deficient in flexibility and dexterity in mass customization due to the heavy reprogramming efforts in limited product life cycle, variations of environments and uncertainties during robot-environment interactions.

This dissertation aims to address the aforementioned deficiencies by improving the dexterity of industrial manipulators. The manipulators with proposed algorithms are required to 1) reduce the hand-engineering in end-effector design, parameter tuning and system integration, and 2) exhibit robustness to uncertainties during the interaction with environments. The fulfillment of the requirements is decomposed into three aspects in this dissertation. The first aspect is to realize *kinematic dexterity* by developing a unified grasping framework with both customized end-effectors and general hands on objects of different categories. The second aspect is to achieve *dynamic dexterity* by constructing an in-hand manipulation and finger gaiting architecture to manipulate the grasped objects robustly and precisely. The third aspect is to attain *skill dexterity* by designing an intelligent assembly algorithm to learn assembly skills from uncertain environments.

The developed grasping framework actively avoids collision and is able to plan grasps and trajectories efficiently with different hands. The grasp planning with industrial customized grippers by surface fitting is introduced in Chapter 2, and the planning efficiency is improved by a learning-based grasp explorer in Chapter 3. The transferring of grasps from parallel grippers to multi-fingered hands by finger splitting is discussed in Chapter 4. Chapter 5 further presents an optimization model to directly plan precision grasps with multi-fingered hands. The final grasping framework is proposed in Chapter 6 by combining the optimization model with a multi-dimensional iterative surface fitting, to improve the grasp versatility and robustness of the optimization model. The constructed manipulation architecture achieves robust grasping and dexterous manipulation under uncertainties. A comprehensive architecture is introduced and verified with different physical multi-fingered hands in Chapter 7. Chapter 8 proposes a robust manipulation controller within the archi-

tecture to further increase the robustness under various uncertainties. To relocate fingers for long-range object motion, the manipulation architecture is augmented with a high-level finger gaits planner in Chapter 9. The designed assembly scheme learns automatic assembly skills with the proposed guided-deep deterministic policy gradient (guided-DDPG) in Chapter 10. By combining supervised learning and reinforcement learning, the proposed guided-DDPG is more efficient than reinforcement learning and achieves better stability and robustness compared with supervised learning.

The proposed grasping and manipulation strategies with customized/general-purposed grippers are able to reduce hand-engineering in mass customization and extend dexterities of automation systems in both kinematic and dynamic levels. The proposed learning assembly scheme increases the efficiency and stability of the assembly in contact-rich scenarios and extends the dexterity of automation systems in skill level. The effectiveness of the grasping, manipulation and assembly algorithms are verified by a series of simulations and experiments on different manipulators and hands.

To My Family and Friends

Contents

List of Figures

List of Tables

Acknowledgments

The past five years in Berkeley has been a wonderful journey for me. I would not have finished my PhD study without generous support from many people.

First of all, my deep and sincere gratitude goes to my advisor, Professor Masayoshi Tomizuka, who is a remarkable person because of his professionalism and dedication as a scholar, as well as the enthusiasm and patience as a mentor. His humor and optimism illuminate those hard days of mine. His unconditional support and complete trust equip me with strength to explore unknowns and overcome failures. The completeness of this dissertation would not have been possible without his support and encouragement.

I would also like to express my gratitude to my qualifying and dissertation committee members including Professor Roberto Horowitz, Professor Ruzena Bajcsy, Professor Shmuel Oren, Professor Kameshwar Poolla, and Professor Kenneth Goldberg for their insightful advices and invaluable guidance to the direction of my research and the completion of my dissertation.

Special thanks to Berkeley fellowship, J. K. Zee Fellowship, Graduate Division Block Grant, and FANUC Corporation for sponsoring my Ph.D. study. The generous financial support from UC Berkeley and FANUC makes me concentrate on my research. In particular, I would like to thank many experts in FANUC, including Dr. Wenjie Chen, Mr. Kaimeng Wang, Mr. Tetsuaki Kato, Mr. Hiroshi Nakagawa, and Ms. Weijia Li for their consistent discussion and tremendous support. Many parts of my dissertation would not have been possible without their dedication.

Furthermore, I would like to thank my colleagues during my internships, including Dr. George Wong, Dr. Hadi Akeel, Dr. Yaz Shehab in Brachium, Inc., and Dr. Yotto Koga, Dr. Jieliang Luo in Autodesk, Inc. It is fantastic to work with these brilliant colleagues who have sharpest thoughts and warmest hearts.

I am grateful to be a member of Mechanical Systems Control (MSC) laboratory over the past five years. I received invaluable advices and tremendous support from former and current members in robotic group: Professor Cong Wang, Dr. Chung-Yen Lin, Professor Changliu Liu, Dr. Te Tang, Dr. Hsien-Chung Lin, Dr. Yu Zhao, Yujiao Cheng, Shiyu Jin, Xinghao Zhu, Changhao Wang, and Ting Xu. I would also like to thank Professor Wenlong Zhang, Dr. Yizhou Wang, Dr. Raechel Tan, Professor Minghui Zheng, Dr. Chen-Yu Chan, Dr. Xiaowen Yu, Dr. Kevin Haninger, Dr. Junkai Lu, Dr. Shiying Zhou, Taohan Wang, Dennis Wai, Liting Sun, Shuyang Li, Cheng Peng, Wei Zhan, Daisuke Keneishi, Yu-Chu Huang, Zining Wang, Jianyu Chen, Kiwoo Shin, Chen Tang, Jiachen Li, Yeping Hu, Zhuo Xu, Hengbo Ma, Jessica Leu, Lingfeng Sun, Yiyang Zhou, Ge Zhang, and Huidong Gao. Thank you so much for your friendship, love and help throughout these years.

Last and foremost, my deepest gratitude goes to my parents and my beloved wife Qinqin Si. Your unconditional love, support and encouragement are the strongest motivation of my life. I could not have achieved these accomplishments without you. Thank you.

Chapter 1

Introduction

1.1 Background of Dexterity Research

Industrial manipulators have been applied in automation for decades. They are programmed and integrated into different systems to deliver different functions such as painting, welding, machining and assembly. Industrial manipulators are rigid, reliable, precise, and highly efficient in traditional mass production [8], since the human labor applied in system integration is neglectable in the long-lasting production lines. The recent popularity of mass customization [45], however, presents considerable challenges for traditional industrial manipulators. On one hand, the frequently changed supply chains require extensive hand-engineering to redesign end-effectors, reprogram manipulators and reintegrate production lines, which significantly increase the unit product cost. On the other hand, the personalized supply chains with numerous workpieces introduce considerable variations and uncertainties, which exponentially complicate the operation of workpieces and the assembly of products.

To address the aforementioned challenges in mass customization, industrial manipulators are desired to equip dexterity to operate workpieces of large variations and assemble products under uncertainties. Three levels of dexterity including kinematic dexterity, dynamic dexterity, and skill dexterity are defined and investigated in this dissertation. Kinematic dexterity describes the capacity of approaching and reaching valid contacts on a target object kinematically to move the object by a manipulator with desired end-effectors. Dynamic dexterity refers to the capacity of exerting force to manipulate the target object dynamically by an end-effector. Skill dexterity describes the ability to learn tasks efficiently and execute the learned tasks competently.

Following the levels of dexterity for the system composed of manipulators and end-effectors, this dissertation is divided into three parts including grasping, manipulation and assembly. To achieve the kinematic dexterity, a unified grasping framework with customized end-effectors/general hands is developed to reduce the redesigning and reprogramming efforts to operate workpieces. First, a surface fitting algorithm is proposed for grasp planning with customized industrial grippers. The reuse of customized grippers for grasping novel objects

of different categories extends the kinematic dexterity and minimizes the cost of hardware modification (Chapter 2). The planning algorithm is incorporated with a learning-based explorer in Chapter 3. The hierarchical learning structure improves the efficiency and keeps the interpretability and reliability of the planning algorithm. Grasp planning algorithm for multi-fingered hands is developed to further extend the kinematic dexterity on more general grasping tasks. A finger splitting algorithm is developed in Chapter 4 to transfer grasps from parallel grippers to multi-fingered hands. Parallel grasps are used to initialize the finger splitting process in order to localize well-performed optima more efficiently. A stronger optimization model is proposed in Chapter 5 to plan grasps with multi-fingered hands directly without the initialization of parallel grasps. The optimization model is able to converge to local optima from arbitrary infeasible configurations. A general grasping framework is finalized in Chapter 6 by incorporating the optimization model with a multi-dimensional surface fitting. The framework is able to produce versatile grasps robustly with both customized grippers/multi-fingered hands under various uncertainties.

Kinematic dexterity alone, however, may not be sufficient to enable manipulators to form stable grasps and manipulate workpieces dexterously. Dynamic dexterity is further investigated not only to operate workpieces in a safe and reliable manner, but also to reduce the cycle time of operation without placing and re-grasping the workpieces [61]. To achieve this, grasp forces in the post-grasping stage have to be optimized in order to lift the object or follow general object trajectories robustly and reliably. This type of force optimization is called in-hand manipulation. A comprehensive in-hand manipulation architecture is proposed in Chapter 7. The robustness is enhanced by a robust manipulation controller in Chapter 8. The proposed controller is robust to various uncertainties and undesired end-effector properties. A finger gaits planner is introduced in Chapter 9 to realize long-range object motion and reduce the cycle time of re-grasping.

The dexterity in kinematic and dynamic levels improves performance of industrial manipulators in operating the workpieces. To achieve full automation, industrial manipulators have to develop skill dexterity to assemble workpieces into personalized products. The skill dexterity is attained in Chapter 10 by designing an intelligent assembly algorithm called the guided deep deterministic policy gradient to learn the assembly skills from uncertain environments. The proposed algorithm reduces the effort in dedicated parameter tuning in mass customization.

1.2 Dissertation Outline

The overall objective of this dissertation is to develop algorithms to improve the dexterity of industrial manipulators. Three major levels of dexterity, including kinematic dexterity, dynamic dexterity and skill dexterity, is discussed and explored. A series of dexterous tasks including grasping, manipulation and assembly will be solved and evaluated by experiments on different manipulators and end-effectors. Figure 1.1 shows the structure of the dissertation. The chapter outlines are as follows. The experimental videos are available at [102].

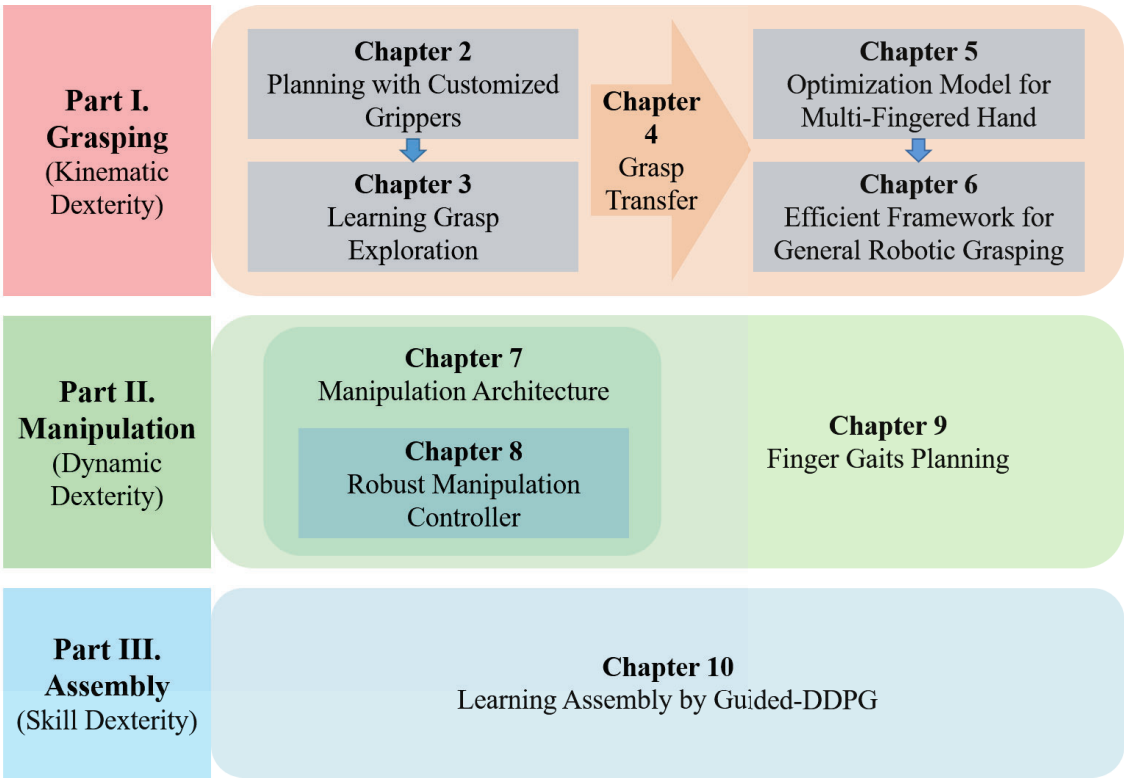


Figure 1.1: Structure of the dissertation.

Grasp Planning with Customized Industrial Grippers by Surface Fitting

Customized grippers have broad applications in industrial production lines. Compared with general parallel grippers, the customized grippers have specifically designed fingers to increase the contact area with the workpieces and improve the grasp robustness. However, grasp planning for customized grippers is challenging due to the object variations, surface contacts and structural constraints of the grippers. Chapter 2 proposes an iterative surface fitting (ISF) algorithm to plan grasps with customized grippers. ISF simultaneously searches for optimal gripper transformation and finger displacement by minimizing the surface fitting error. A guided sampling is introduced to avoid ISF getting stuck in local optima and improve the collision avoidance performance. The proposed algorithm is able to consider the structural constraints of the gripper and plan optimal grasps in real-time. The effectiveness of the algorithm is verified by both simulations and experiments. Part of this work was published in [20].

Learning Efficient Grasp Exploration with Customized Grippers

The grasp planning algorithm in Chapter 2 behaves well on single objects or light clutter environments but becomes slow in heavy clutter environments. To plan grasps in clutter environments with customized grippers, a grasp planner is desired to learn the graspability of different regions. Chapter 3 proposes a learning framework to plan robust grasps for customized grippers efficiently. The learning framework contains a low-level optimization-based planner with ISF to search for optimal grasps locally, and a high-level learning-based explorer to learn the grasp exploration based on previous grasp experiences. The high-level learning-based explorer trains a region-based convolutional neural network (R-CNN) to propose desired regions, which avoids ISF getting stuck in bad local optima and improves the collision avoidance performance. The proposed learning framework with RCNN-ISF is able to consider the structural constraints of the gripper, learn grasp exploration strategy from previous experiences, and plan optimal grasps in heavy clutter environments efficiently. The effectiveness of the algorithm is verified by experiments. Part of this work was published in [19].

Transferring Grasps from Parallel Grippers to Multi-Fingered Hands by Finger Splitting

Grasp planning for multi-fingered hands is computationally expensive due to the joint-contact coupling, surface nonlinearities and high dimensionality, thus is generally not affordable for real-time implementations. The planning method in Chapter 2 works well for parallel grippers but remains challenging for multi-fingered hands. Chapter 4 proposes a strategy called finger splitting, to plan precision grasps for multi-fingered hands starting from optimal parallel grasps. The finger splitting is optimized by a dual-stage iterative optimization including a contact point optimization (CPO) and a palm pose optimization (PPO), to gradually split fingers and adjust both contact points and palm pose. The dual-stage optimization is able to consider both object grasp quality and hand manipulability, address the nonlinearities and coupling, and achieve efficient convergence within one second. Simulation results demonstrate the effectiveness of the proposed approach. Part of this work was published in [24].

Optimization Model to Plan Grasps with Multi-Fingered Hands

The finger splitting in Chapter 4 transfers grasps from parallel grippers to multi-fingered hands. The transfer requires the initialization of optimal parallel grasps in order to converge to well-performed local optima. To simplify the initialization, Chapter 5 proposes an optimization model to directly search for precision grasps from arbitrary infeasible initial configurations. The model takes noisy point cloud of an object as input and optimizes the grasp quality by iteratively searching for the palm pose and finger joints positions. The collision between the hand and the object is approximated and penalized by sequential least-

squares. The collision approximation is able to handle the point cloud representation of the objects with complex shapes. The proposed optimization model is able to locate collision-free optimal precision grasps efficiently. The average computation time is 0.50 sec/grasp. The searching is robust to the incompleteness and noise of the point cloud. The effectiveness of the algorithm is demonstrated by experiments. Part of this work was published in [18].

Efficient Framework for General Robotic Grasping

In Chapter 6, the optimization model from Chapter 5 is further combined with a surface fitting in order to form a general framework to plan versatile grasps and trajectories under uncertainties with different types of hands. The framework includes a multi-dimensional iterative surface fitting (MDISF) for grasp planning and a grasp trajectory optimization (GTO) for grasp imagination. MDISF searches for optimal contact regions and hand configurations by minimizing the collision and surface fitting error, and the GTO algorithm generates optimal finger trajectories to reach the highly ranked grasp configurations and avoid collision with the environment. The proposed grasp planning and imagination framework plans grasps and trajectories of different categories efficiently with gradient-based methods using the captured point cloud. The framework is able to implement to both customized grippers and multi-fingered hands. The found grasps and trajectories are robust to sensing noises and underlying uncertainties. The effectiveness of the proposed framework is verified by both simulations and experiments. Part of this work was published in [17].

Object Manipulation Framework by Modified Impedance Control

Chapter 2 - 6 improve kinematic dexterity by developing grasping algorithms with both customized grippers and multi-fingered hands. However, kinematic dexterity itself is not sufficient for stable workpiece operation. Appropriate force is required to grasp workpieces robustly under uncertainties. We refer the capacity of end-effector to exert force for object manipulation as dynamic dexterity. Dynamic dexterity is essential to simplify the placing/regrasping procedures and reduce the cycle time. To achieve the dynamic dexterity, Chapter 7 introduces a comprehensive architecture for dexterous in-hand manipulation. The architecture includes a high-level robust controller for object Cartesian force generation, a mid-level manipulation controller for contact force optimization, and a low-level force tracking controller to track the optimized contact force. The architecture is able to run on hands with torque/force/velocity inputs and resist certain level of mass uncertainties, contact uncertainties and external disturbances. Part of this work was published in [23].

Robust Dexterous Manipulation under Various Uncertainties

To perform broad-scale manipulation tasks, it is desired that a multi-fingered robotic hand can robustly manipulate objects without knowing the exact objects dynamics (i.e. mass and inertia) in advance. However, realizing robust manipulation is challenging due to the

complex contact dynamics, the nonlinearities of the system, and the potential sliding during manipulation. Chapter 8 proposes a robust manipulation controller within the architecture in Chapter 7 to handle these challenges. In the first stage, feedback linearization is utilized to linearize the nonlinear uncertain system. Considering the structures of uncertainties, a robust controller is designed for the linearized system to obtain the desired Cartesian force on the object. In the second stage, a manipulation controller regulates the contact force based on the Cartesian force from the first stage. The robust manipulation controller is able to realize robust manipulation without modeling contact, prevent the slippage, and withstand 50% mass and 80% inertia uncertainties. Simulations and experiments verify the effectiveness of the proposed method. Part of this work was published in [27].

Finger Gaits Planning for Robust Dexterous Manipulation

To perform long-range manipulation tasks, a multi-fingered robotic hand sometimes has to sequentially adjust its grasping gestures, i.e. the finger gaits, to address the workspace limits and guarantee the object stability. However, realizing finger gaits planning in dexterous manipulation is challenging due to the complicated grasp quality metrics, uncertainties on object shapes and dynamics, and unexpected slippage under uncertain contact dynamics. With the architecture of Chapter 7 and robust manipulation controller from Chapter 8, Chapter 9 further proposes a high-level finger gaits planner to handle these challenges. The planner combines object grasp quality with hand manipulability. It is computationally efficient and realizes finger gaiting without 3D model of the object. Moreover, the planner is able to guarantee stability under unexpected slippage caused by uncertain contact dynamics. The proposed dual-stage optimization based planner is verified by simulations on Mujoco. Part of this work was published in [23] and [26].

Learning Industrial Assembly by Guided-DDPG

Skill dexterity is essential for industrial manipulators to achieve automatic assembly and realize full automation. Traditional assembly tasks use predefined trajectories or tuned force control parameters, which make the automatic assembly 1) time-consuming, 2) difficult to generalize, and 3) not robust to uncertainties. Chapter 10 proposes a learning framework for industrial assembly. The framework combines both the supervised learning and the reinforcement learning. The supervised learning utilizes trajectory optimization to provide the initial guidance to the policy, while the reinforcement learning utilizes actor-critic algorithm to establish the evaluation system even the supervisor is not accurate. The proposed learning framework is more efficient compared with the reinforcement learning and achieves better stability performance than the supervised learning. The effectiveness of the method is verified by both the simulation and experiment. Part of this work was published in [28].

Part I

Grasping

Chapter 2

Grasp Planning with Customized Industrial Grippers by Surface Fitting

2.1 Introduction

Grasping is an essential capability for robots to extend the functionality and execute complex tasks such as assembly, picking and packaging. Compared with general parallel grippers, the customized grippers are designed for particular class of grasping tasks by matching the contact surface of the gripper with the geometry of the workpieces. As a result, the customized grippers generally have larger gripping force and more robust grasp. For example, the food industry may design particular grippers to grasp cakes from conveyors for food packing, and the autonomous assembly line may require customized grippers to assure stable grasp and precise localization.

The grasp planning for customized grippers is challenging. The traditional point contact model or soft finger model [69] in grasp modeling is insufficient to describe the large surface contact between the gripper and the object. Thus, the quality evaluation is ambiguous for one particular grasp configuration. Therefore, it is desirable to exploit the surface information of the gripper and the object for grasp planning.

There are several related works proposed to reveal the importance of contact surfaces in grasp planning. A taxonomy of grasp is constructed to analyze grasp models and design grippers [14, 13]. The taxonomy observes that both power grasps and precision grasps tend to increase the contact surface to improve the stability and robustness of the grasp, where the object is either surrounded by the palm and fingers in power grasps, or by the soft fingertips in precision grasps.

In [37], a fingertip space is proposed to take into account the matching of the basic fingertip geometry to object surface. The grasps are searched based on several layers of fingertip space with different resolutions. This method assumes point contact between the finger and the object, which is not the case for customized grippers with large contact surfaces. To take advantage of the large contact surface on grasp stability and robustness,

the optimal grasps in [12] are searched by minimizing the distance between predefined points on the hand and the surface of the object. The resultant grasps match the object surface by enclosing the object with more contacts. However, the computation load is excessively heavy for online implementations. The grasp synthesis for human hands using shape matching algorithm is proposed in [52]. The algorithm matches hand shapes in a database to the query object by identifying collections of features with similar relative placements and surface normals. However, this approach requires lots of human demonstration to collect enough hand pose samples. Therefore, the searching for proper grasps requires considerable time with exhaustive sampling. The idea of matching the shapes of parallel grippers to the geometries of objects is utilized in [43, 73] to accelerate the grasp searching speed by filling the parallel gripper with the object. However, this method cannot be generalized to a gripper with complicated shape like customized grippers. In [49, 62], the grasp strategy is trained in an end-to-end manner from image to grasp policy with millions of grasping data. The resultant policy reflects the importance of matching the fingertip with objects, while changing grippers requires re-collecting data and re-training the policy.

In order to consider more complicated gripper shape, reduce the computation load, and retrieve reliable and secure grasps, it is desired to match the surfaces on the gripper to the object more precisely. The idea of shape registration using iterative closest point (ICP) is first proposed in [5] and then refined by [40, 59, 108]. However, these approaches only consider the alignment of one source object to one target object, while the surface matching in grasp planning for customized grippers can be regarded as registering multiple surfaces on different fingers to one target object. Moreover, the surfaces have to satisfy the constraints of the grippers such as the jaw width, allowable DOFs and the alignment of contact normals.

Considering the specific surface matching problem in grasp planning, this chapter proposes an iterative surface fitting (ISF) algorithm, where it fits the contact surface based on a metric measuring the distance of gripper-object surface as well as the misalignment of the contact normals. ISF takes the motions of both the palm and fingers into consideration by a proposed iterative palm-finger optimization (IPFO). A guided sampling is also introduced to encourage the exploration of the regions with high fitting scores.

The contributions of this chapter are as follows. First, the proposed algorithm achieves simultaneous surface fitting and gripper kinematic planning by considering both the desired grasp surfaces and the structural constraints of the gripper such as the jaw width and the degree of freedoms (DOFs). Second, the proposed guided sampling avoids getting stuck in the local optima by exploiting the previous grasping experience. The grasp planning by ISF and guided sampling achieves a real-time planning and the time to search for a collision-free grasp is less than 0.1 s in average for the objects in the simulation and experiment. Furthermore, by combining with the dedicated gripper design, the proposed surface fitting algorithm can deal with objects with complicated shapes and those in clutter environments with unsegmented point clouds. The experimental videos are available at [102].

The remainders of this chapter are as follows. Section 2.2 formulates the grasp planning of the customized grippers. Section 2.3 introduces the surface fitting method to solve the optimization for desired grasps. The simulations and experiments are presented in Section 2.4.

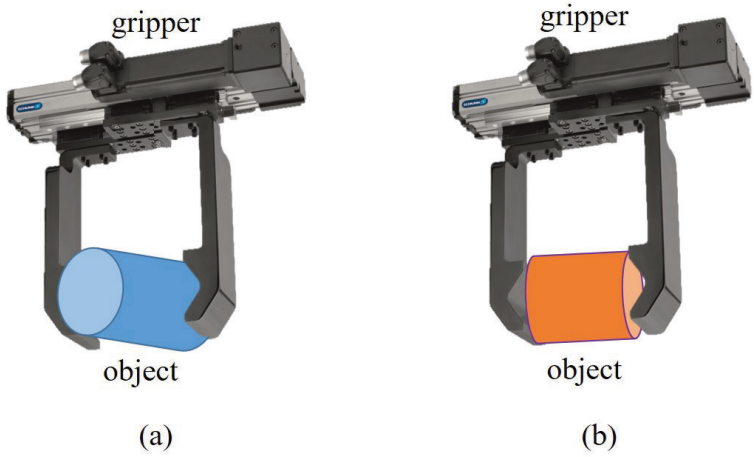


Figure 2.1: An example grasp of cylinder objects by a customized gripper.

Section 2.5 summarizes the chapter.

2.2 Contact Surface Optimization

A grasp planning example with a customized gripper is shown in Fig. 2.1. The customized gripper has parallel jaws with curved fingertip surfaces. The object to be grasped is a cylinder. The objective of the grasp planning is to search for the optimal pose of the gripper relative to the object by maximizing a quality metric considering the structural constraints of the gripper. More concretely, it is formulated as

$$\max_{R,t,\delta d,\mathcal{S}_j^f,\mathcal{S}_j^o} \quad Q(\mathcal{S}_1^f,\mathcal{S}_2^f,\mathcal{S}_1^o,\mathcal{S}_2^o) \tag{2.1a}$$

$$s.t. \quad \mathcal{S}_j^f \subset \mathcal{T}(\partial\mathcal{F}_j;R,t,\delta d), \quad j=1,2 \tag{2.1b}$$

$$\mathcal{S}_j^o = NN_{\partial\mathcal{O}}(\mathcal{S}_j^f), \quad j=1,2 \tag{2.1c}$$

$$(\mathcal{S}_1^f,\mathcal{S}_2^f) \in \mathcal{W}(d_0+\delta d) \tag{2.1d}$$

$$d_0+\delta d \in [d_{\min},d_{\max}] \tag{2.1e}$$

where $j \in \{1,2\}$ is the finger index, $R \in SO(3), t \in \mathbb{R}^3$ are the rotation and the translation of the gripper jaw from the original pose, $\delta d \in \mathbb{R}$ is the finger displacement from the original width d_0, and Q represents the grasp quality with respect to the finger contact surfaces $\mathcal{S}_j^f$ and the object contact surface $\mathcal{S}_j^o$. The finger contact surface $\mathcal{S}_j^f$ lies on the finger surface $\partial\mathcal{F}_j$ transformed by $\mathcal{T}$, as shown in (2.1b). The object contact surface $\mathcal{S}_i^o$ is determined by the nearest neighbor of the $\mathcal{S}_j^f$ on the object surface $\partial\mathcal{O}$, as shown in (2.1c). Constraint (2.1d) indicates that the finger contact surfaces should be in the workspace $\mathcal{W}$ parameterized by

the jaw width, and (2.1e) describes the constraint of the finger displacement. The optimization (2.1) searches for the optimal gripper transformation (R^*, t^*) and finger displacement d^* by maximizing the grasp quality Q.

Problem (2.1) is a standard grasp planning problem if the contact surfaces are degenerated into points. In general surface contact situation, the point contact model may not be able to incorporate the gripper surface directly into the planning. Problem (2.1) becomes challenging to solve by either gradient based methods or sampling based methods. On one hand, modeling the contact surfaces as decision variables is nontrivial in the gradient based methods. On the other hand, the sampling based methods require exploring the whole state space, which is not applicable for real-time implementation.

A natural surface-related quality can be constructed by matching the surfaces between the object and the gripper. Intuitively, the grasp with small surface matching error (Fig. 2.1(a)) is more stable and robust compared with the one with large error (Fig. 2.1(b)). Therefore, an iterative surface fitting (ISF) algorithm is proposed in this chapter to search for optimal grasps of the gripper relative to the object. The optimality is in the sense of surface fitting errors. The formulation is modified from the iterative closest point (ICP) by including the structural constraints of different fingers such as the width and DOFs. ISF is initialized by a guided sampling algorithm in order to avoid being trapped in local optima and achieve collision avoidance.

2.3 Gradient-based Searching by Iterative Surface Fitting

Iterative Closest Point

The ICP algorithm is first proposed in [5] for 3D shape registration. It searches for the optimal rigid transformation to align the source surface towards the target surface by minimizing the distance between them. The minimization is conducted by iteratively searching the correspondence points of the surfaces and minimizing the Euclidean distance of these correspondence points. More specifically, it is written as

$$\min_{R,t} \sum_{i=1}^{m} \|Rp_i + t - q_i\|_2^2, \tag{2.2}$$

where $p_i, q_i \in \mathbb{R}^3$ denote a correspondence pair on the source surface and the target surface, and m is the number of the correspondence pairs. The correspondence of (p_i, q_i) is searched by the nearest neighbor method. The optimal transformation (R, t) is calculated and applied to the source surface, after which the correspondence is updated by searching with the nearest neighbor again. These two steps are performed iteratively until convergence.

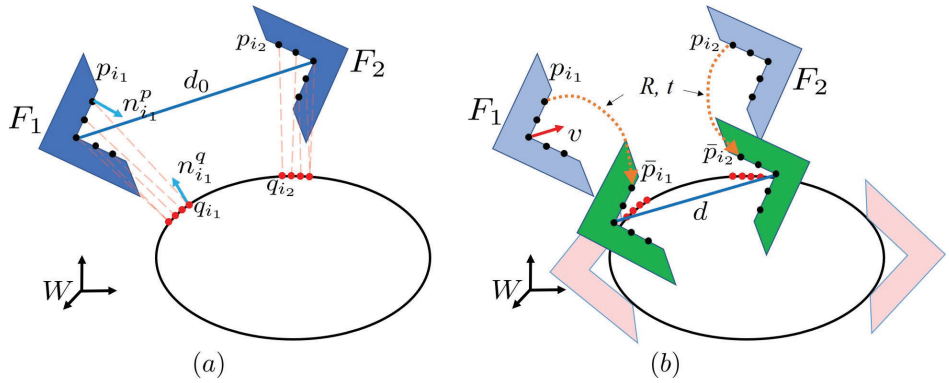


Figure 2.2: Illustration of the iterative surface fitting (ISF) including (a) correspondence matching and (b) surface fitting.

A relevant variation calculates the distance along the normal direction [11]:

$$\min_{R,t} \sum_{i=1}^{m} \left((Rp_i + t - q_i)^T n_i^q \right)^2, \tag{2.3}$$

where n_i^q indicates the normal of the target surface on point q_i. Instead of calculating the Euclidean distance between the transformed source point and the corresponding target point, the modification calculates the distance of the point to the plane by projecting the vector to normal direction n_i^q. Therefore, this modification allows the source surface to slide along the flat target surface.

With the small rotation angle assumption, the rotation matrix R can be approximated by $I + \hat{r}$, where $\hat{r} \in so(3)$ represents the skew-symmetric matrix of the axis-angle vector $r \in \mathbb{R}^3$. With this approximation, (2.3) can be solved analytically by a standard least squares

$$\min_x \|\mathbf{A}x - \mathbf{b}\|_2^2, \tag{2.4}$$

where $\mathbf{A} = [a_1^T, ..., a_m^T]^T$, and $\mathbf{b} = [b_1, ..., b_m]^T$ with

$$a_i = \left[(p_i \times n_i^q)^T,\ (n_i^q)^T \right], \tag{2.5a}$$

$$b_i = (q_i - p_i)^T n_i^q. \tag{2.5b}$$

The optimal solution for the transformation is then given by $x = \left[r^T,\ t^T \right]^T = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ under the current correspondence.

The ICP algorithm, however, can only register one single surface to the target surface at a time. In the grasping applications, a gripper usually consists of two or more fingers, which are all expected to be fitted on the target workpiece. We propose an iterative surface fitting (ISF) algorithm here to register multiple finger surfaces to the target workpiece at the same time considering the allowable DOFs of fingers.

Iterative Surface Fitting with Customized Grippers

For ease of illustration, the ISF algorithm is introduced using a customized gripper with two fingers and one DOF, where the two fingers move in opposite directions to adjust the jaw width (Fig. 2.2). The ISF algorithm iteratively executes two modules: the correspondence matching and the surface fitting. The correspondence matching contains a nearest neighbor search and an outlier/duplicate filtering. As shown in Fig. 2.2(a), given the point p_{i_j} on the gripper surface (black dots) on the j-th finger, a KD-Tree is employed to search the nearest neighbor q_{i_j} on the object surface (red dots), where the dash lines represent their correspondence. The outlier filtering removes the pairs whose distances are excessively large and the duplicate filtering considers the case where multiple points are assigned to the same point on the object. In this case, only the pair with the minimum distance will be kept [108]. The outlier/duplicate filtering is particularly useful in our implementation since the gripper surface and object surface overlap only partially, and rejecting these point pairs greatly increases the robustness.

In Fig. 2.2(b), the surface fitting aims to search over the gripper transformation (R, t) as well as the relative motion $\delta d = d - d_0$ between the fingers, where d_0 and d indicate the jaw width before and during the optimization. The green gripper is the updated result by the optimization, and the pink one is the converged result after several iterations of ISF. To be more specific, the surface points are transformed to new locations by

$$\bar{p}_{i_j} = Rp_{i_j} + t + \frac{1}{2}(-1)^j Rv\delta d, \quad j = 1, 2 \tag{2.6}$$

where $v \in \mathbb{R}^3$ is a unit vector pointing from F_1 to F_2, as shown by the red arrow in Fig. 2.2(b). Then the matching of contact points can be quantified by an point matching error metric,

$$E_p(R, t, \delta d) = \sum_{i=1}^{m} \sum_{j=1}^{2} \left((\bar{p}_{i_j} - q_{i_j})^T n_{i_j}^q \right)^2. \tag{2.7}$$

An additional normal alignment error metric is

$$E_n(R) = \sum_{i=1}^{m} \left((Rn_i^p)^T n_i^q + 1 \right)^2, \tag{2.8}$$

where E_n represents the misalignment error between the normal of the gripper n_i^p and that of the object n_i^q. By minimizing E_n, the normals of the finger surface are forced to align towards the normals of the object surface.

The surface fitting is to minimize the overall error

$$\min_{R,t,\delta d} E(R, t, \delta d) \tag{2.9a}$$

$$s.t. \quad \delta d + d_0 \in [d_{\min}, d_{\max}] \tag{2.9b}$$

where $E(R,t,\delta d) = E_p(R,t,\delta d) + \alpha^2 E_n(R)$ represents the surface fitting error and $\alpha \in \mathbb{R}$ is the weighting factor that balances the the point matching and the normal alignment.

Problem (2.9) is nonlinear due to the coupling between the R and δd. In this chapter, the palm transformation R,t and the finger displacement δd are solved by an iterative palm-finger optimization (IPFO). The palm optimization optimizes for the optimal palm transformation R^*,t^* with fixed δd, while the finger optimization optimizes for the optimal finger displacement δd^* with fixed (R,t).

The palm optimization can be formulated as a least squares problem that is similar to (2.4) with an augmented matrix $\tilde{\mathbf{A}} = [\mathbf{A}_1^T, \mathbf{A}_2^T, \mathbf{A}_n^T]^T$ and an augmented vector $\tilde{\mathbf{b}} = [\mathbf{b}_1^T, \mathbf{b}_2^T, \mathbf{b}_n^T]^T$, where $\mathbf{A}_j$ and $\mathbf{b}_j$ are the point matching of each finger surface modified from (2.4), with p_i in (2.5) replaced by $\tilde{p}_{i_j} = p_{i_j} + 0.5(-1)^j v\delta d$ for $j = 1,2$ to consider the displacement of the finger. $\mathbf{A}_n$ and $\mathbf{b}_n$ are additional terms to align the contact normals. Derived from (2.8), we can get $\mathbf{A}_n = [a_{n,1}^T, \cdots, a_{n,m}^T]^T$ and $\mathbf{b}_n = [b_{n,1}, \cdots, b_{n,m}]^T$ with $a_{n,i} = [\alpha(n_i^p \times n_i^q)^T, 0_3^T]$ and $b_{n,i} = -\alpha(n_i^p)^T n_i^q - \alpha$. Therefore, the palm optimization has the closed form and can be represented as

$$x = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T\tilde{\mathbf{b}}. \tag{2.10}$$

The finger optimization with fixed (R,t) in (2.9) is a one-dimensional constrained quadratic programming,

$$\min_{\delta d} \sum_{i=1}^{m}\sum_{j=1}^{2}(b_{i_j} - a_{i_j}\delta d)^2 \tag{2.11a}$$

$$s.t. \quad \delta d + d_0 \in [d_{\min}, d_{\max}] \tag{2.11b}$$

where $a_{i_j} = 0.5(-1)^{j-1}(Rv)^T n_{i_j}^q$, and $b_{i_j} = (Rp_{i_j} + t - q_{i_j})^T n_{i_j}^q$. The optimal finger relative motion is given by

$$\delta d^* = \begin{cases} d_{\min} - d_0, & \text{if } \delta\hat{d} + d_0 < d_{\min} \\ \delta\hat{d}, & \text{if } d_{\min} \le \delta\hat{d} + d_0 \le d_{\max}, \\ d_{\max} - d_0, & \text{if } \delta\hat{d} + d_0 > d_{\max} \end{cases} \tag{2.12}$$

with

$$\delta\hat{d} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{2} a_{i_j} b_{i_j}}{\sum_{i=1}^{m}\sum_{j=1}^{2} a_{i_j}^2}. \tag{2.13}$$

The procedure of IPFO is summarized in Alg. (1). Given the correspondence (p_i, q_i) and the corresponding normals (n_i^p, n_i^q) as inputs, IPFO minimizes the error metric (2.9a) in an iterative manner (Line 5-6). The iteration stops when the error reduction is less than a threshold Δe (Line 3).

Inspired by [40], ISF is optimized hierarchically by searching with a multi-resolution pyramid, as shown in Alg. (2). The inputs to ISF include the initial gripper state R_c, t_c, d_0,

Algorithm 1 Iterative Palm-Finger Optimization (IPFO)

1: **Input:** $(p_i, q_i),\ (n_i^p, n_i^q),\ d_0$
2: **Init:** Initialize $\delta d^* = 0,\ R^* = I,\ t^* = 0_3,\ e_p = \infty$
3: **while** $e_p - E(R^*, t^*, \delta d^*) > \Delta e$ **do**
4: $\quad e_p \leftarrow E(R^*, t^*, \delta d^*)$
5: $\quad \{R^*, t^*\} \leftarrow \min_{R,t} E(R, t, \delta d^*)$ by (2.10)
6: $\quad \delta d^* \leftarrow \min_{\delta d} E(R^*, t^*, \delta d)$ by (2.12)
7: **end while**
8: **return** $\{R^*, t^*, \delta d^*, e_p\}$

Algorithm 2 Iterative Surface Fitting (ISF)

1: **Input:** Initial state R_c, t_c, d_0, $\partial\mathcal{O}$, $\partial\mathcal{F}$, L, I_0, ϵ_0
2: **Init:** $\partial\mathcal{F} = \mathcal{T}(\partial\mathcal{F}; R_c, t_c, d_0)$
3: **for** $l = L-1, \cdots, 0$ **do**
4: $\quad \mathcal{S}^f \leftarrow \texttt{downsample}(\partial\mathcal{F}, 2^l)$, $I_l = I_0/2^l$, $\epsilon_l = 2^l\epsilon_0$
5: $\quad \mathcal{S}_0^f \leftarrow \mathcal{S}^f$, $e_s \leftarrow \infty$, $\eta \leftarrow 0$, $it \leftarrow 0$
6: $\quad$ **while** $\eta \notin [1-\epsilon_l, 1+\epsilon_l]$ and it++ $< I_l$ **do**
7: $\quad\quad e_{s,p} \leftarrow e_s$
8: $\quad\quad \mathcal{S}^o \leftarrow NN_{\partial O}(\mathcal{S}^f)$
9: $\quad\quad \{\mathcal{S}^f, \mathcal{S}^o\} \leftarrow \texttt{filter}(\mathcal{S}^f, \mathcal{S}^o)$
10: $\quad\quad \{R^*, t^*, \delta d^*, error\} \leftarrow \textbf{IPFO}(\mathcal{S}^f, \mathcal{S}^o, d_0)$
11: $\quad\quad \mathcal{S}^f \leftarrow \mathcal{T}(\mathcal{S}^f; R^*, t^*, \delta d^*)$
12: $\quad\quad \partial\mathcal{F} \leftarrow \mathcal{T}(\partial\mathcal{F}; R^*, t^*, \delta d^*)$
13: $\quad\quad d_0 \leftarrow d_0 + \delta d^*$
14: $\quad\quad e_s \leftarrow \|\mathcal{S}^f - \mathcal{S}_0^f\|,\ \eta \leftarrow e_s/e_{s,p}$
15: $\quad$ **end while**
16: **end for**
17: **return** $\{error, \partial\mathcal{F}\}$

the surfaces $\partial\mathcal{O}$ and $\partial\mathcal{F}$, and the parameters for hierarchical searching (Line 1). L, I_0 and ϵ_0 denote the level number, maximum iteration and error bound for convergence, respectively. The gripper surface ∂F is first transformed to the specified initial state (Line 2). In each level of the pyramid, $\partial\mathcal{F}$ is downsampled adaptively to $\mathcal{S}^f$ with different resolutions (Line 4). The while loop iteratively searches correspondence and solves for desired gripper motion. The correspondence matching is conducted with the nearest neighbor search (Line 8) and with the outlier/duplicate filtering (Line 9). The desired palm transformation and finger displacement are optimized by IPFO (Line 10). Both the surface $\partial\mathcal{F}$ and its sample $\mathcal{S}^f$ are transformed by the optimized gripper motion (Line 11-12). The while loop is terminated if IPFO gives a similar transformation in adjacent iterations.

Algorithm 3 Grasp Planning Algorithm

1: **Input:** $\partial\mathcal{O}, \partial\mathcal{F}$, center# K, sample# K_s, d_0
2: **Init:** $C \leftarrow$ `k-means`$(\partial\mathcal{O}, K)$,$regret = 0_K$,$trial = 0_K$
3: **for** $It = 1, \cdots, K_s$ **do**
4: Guided sampling:
$k^* \leftarrow \text{argmin}_k\, regret(k)$
$t_c = C(k^*), R_c \leftarrow$ `randRot`$()$
5: ISF evaluation:
$\{error, \partial\bar{\mathcal{F}}\} \leftarrow \textbf{ISF}(R_c, t_c, d_0, \partial\mathcal{O}, \partial\mathcal{F})$
$col \leftarrow f_{\text{col}}(\partial\bar{\mathcal{F}}, \partial\mathcal{O})$
6: Regret update:
$regret(k^*) \leftarrow \frac{regret(k^*)\cdot trial(k^*)+error}{trial(k^*)+1}$
$regret(k^*) \leftarrow (1+\gamma\cdot col)regret(k^*)$
$trial(k^*) \leftarrow trial(k^*)+1$
7: **end for**

Initialization and Sampling

The initialization of ISF is important since ISF converges to local optima. Multiple initialization is desired for the algorithm to explore different regions of the object, so as to avoid getting trapped in bad local optima and achieve better collision-avoiding solutions.

In this chapter, the object is firstly partitioned into K clusters by k-means clustering. The center of each cluster is regarded as a candidate initial position of the gripper for ISF, and the initial orientation of the gripper is randomly sampled.

We build an empirical model to guide the sampling among K candidates. Similar to the multi-armed bandit model, we record the error of ISF evaluated in each cluster center and compute the average regret accordingly. The average regret of each cluster center is used to guide the succeeding sampling. The sampling process is summarized in Alg. (3).

The Alg. (3) is fed by the surfaces $\partial\mathcal{O}, \partial\mathcal{F}$ and the parameters including the cluster number K, the total sampling times K_s and initial jaw width d_0. The sample centers $C \in \mathbb{R}^K$ is generated by k-means clustering (Line 2). The system stores $trial \in \mathbb{R}^K$ and $regret \in \mathbb{R}^K$ to represent the previous sampling and evaluation experience. The sampling is guided by the average regret, and the cluster center with the minimum regret is chosen as the initial position of the gripper for the following ISF, while the initial orientation R_c is randomly sampled (Line 4). The gripper with the sampled initialization is evaluated by ISF and collision check (Line 5). The $f_{\text{col}}(\cdot,\cdot)$ is a boolean function that returns 1 when the inputs have collision, i.e. two object surfaces have intersection. In Line 6, the average regret of the k^*-th sample is updated by considering the surface fitting error and the collision penalty. The γ is a penalty factor that penalizes the average regret for the collided samples.

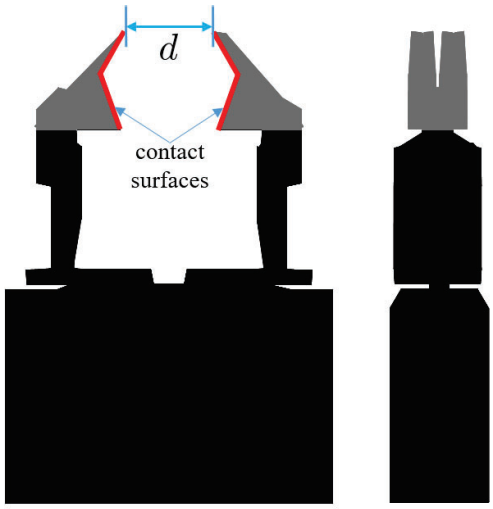


Figure 2.3: The parallel jaw gripper with curved fingertips.

2.4 Simulations and Experiments

In this section, both the simulation and the experiment results are presented to verify the effectiveness of grasp planning algorithm. The experimental videos are available at [102]. The computer we used was a desktop with 32 GB RAM and 4.0 GHz CPU. All the computations were conducted by Matlab. We used a SMC LEHF20K2-48-R36N3D parallel jaw gripper with the specialized fingers as shown in Fig. 2.3. The fingertips of the gripper and the contact surfaces are marked by gray and red colors, respectively. The allowed finger motion is shown by blue arrow. The desired contact surfaces are marked by red. The gripper width was constrained by $[d_{\min}, d_{\max}] = [1, 3]$ cm.

Parameter Lists

The initial gripper width was set as $d_0 = 2$ cm. The weight for normal penalty was $\alpha = 0.01$. The convergence threshold was selected as $\Delta e = 10^{-5}$ in Alg. (1). The pyramid level L, the maximum iteration I_0 and the tolerance ϵ_0 were set as 4, 200 and 0.008 respectively in Alg. (2). The k-means center number $K = 6$, the sample number $K_s = 60$, and the collision penalty $\gamma = 0.2$ in Alg. (3).

Simulation Study

The grasp planning optimized for grasps by guided sampling and ISF evaluation. The proposed guided sampling enabled more efficient exploration by minimizing the average regret based on the previous experience. The proposed ISF algorithm considered the surface fitting by optimizing both the palm transformation and finger displacement.

Figure 2.4 shows the grasp planning result on an Oscar model. The object and gripper surface are shown by the blue and red dots in Fig. 2.4(a). The vertices of the object were firstly processed by downsampling and normal estimation, after which the k-means clustering ran for centers of initialization, as shown by bold dots. Multiple grasps were generated (shown by red patches in Fig. 2.4(a)) and passed through the collision check function. The planned collision-free grasp in the figure is represented by the pose of the solid gripper, while the collided grasp is represented by the pose of the transparent one as shown in Fig. 2.4(b).

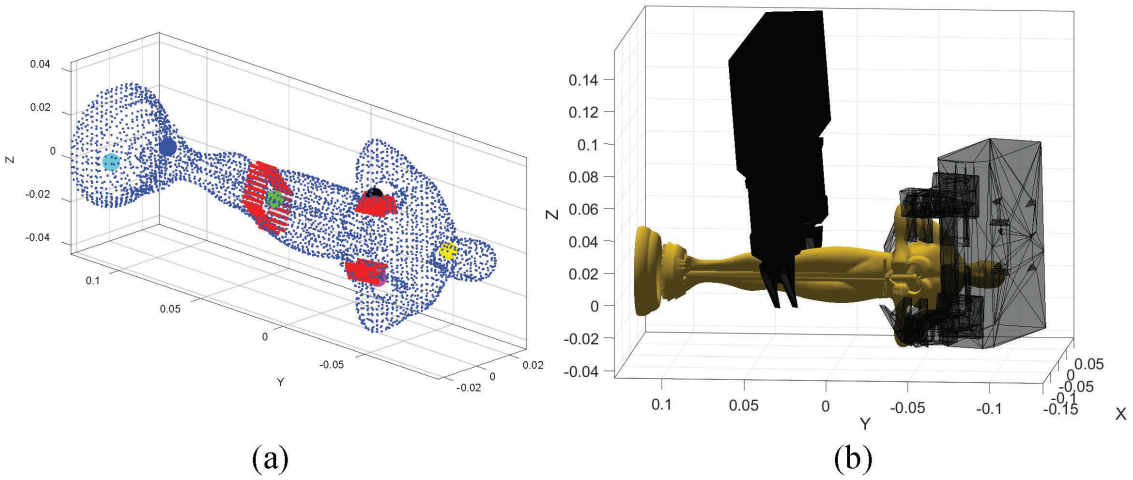


Figure 2.4: Illustration of grasp planning results on Oscar model.

Figure 2.5 shows the surface fitting errors of ISF and one execution of IPFO during the grasp planning on Oscar model. The average distance between $\partial\mathcal{F}$ and $\partial\mathcal{O}$ was dropped from 10 mm to 1 mm by running ISF as shown in Fig. 2.5(a), where the shaded area is magnified and shown in Fig. 2.5(b). The red line shows the finger displacement in different iterations and the blue line shows the fitting error. The IPFO could achieve efficient convergence within 7 iterations.

The simulation results on different objects are visualized in Fig. 2.6. Some of the objects (e.g. Doraemon and Bunny) were scaled to fit into the gripper range. The simulation details for these objects are shown in Table 2.1. The second column shows the number of collision-free optimal grasps over the total samples. The third column shows the total computation time (i.e. the time to run Alg. (3)) to generate these grasps. The last column shows the number of vertices for each object. The dragon was challenging to grasp due to collision caused by the complex geometry. In average, the grasp planning took 2.33 s to find 36.2 collision-free grasps in 60 times of samples. Thus, each collision-free grasp took 64.4 ms in average.

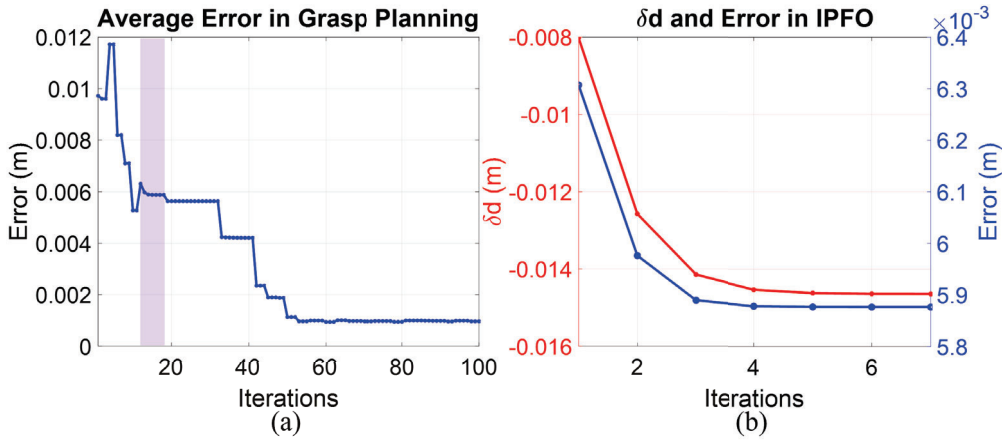


Figure 2.5: Surface fitting errors of (a) ISF and (b) one execution of IPFO.

Table 2.1: Numerical Results of Grasp Planning Simulation

Objects	Collision Free Grasps / Total Samples	t_{total} (sec.)	Number of Vertices
Hand	53/60	1.85	1939
Gun	36/60	2.45	2287
Car	18/60	1.68	2166
Oscar	31/60	2.03	3684
Dragon	3/60	5.61	3971
Bunny	33/60	1.56	740
Banana	54/60	3.82	1723
Screw Driver	58/60	1.71	930
Doraemon	40/60	1.84	769
Average	36.2/60	2.33	2022.1

Experiment Study

A series of experiments were further performed on a FANUC LR Mate 200-iD/7L industrial manipulator to verify the effectiveness of the proposed algorithm. Two IDS Ensenso N35 stereo camera sets were used to capture the point cloud of the object. Compared with simulation, the point cloud produced by Ensenso cameras was not able to reflect the object precisely due to occlusion and noise. The point cloud was smoothed and used to estimate the normals of the objects.

Figure 2.7 shows the grasp planning results on six different objects including three toy robot models and three tools in different sizes. The left side for each subfigure shows the observed point cloud and the optimized grasp, and the right side shows the grasp execution result in lifting the object by 10 cm. Although some of the objects had complicated shapes,

Figure 2.6: Simulation results on nine individual objects.

ISF could find a grasp that matched the fingertips to the object surface well. Therefore, the robot could firmly grasp the object and further increased the grasp robustness.

A picking task in a clutter environment was performed and shown in Fig. 2.8, where several objects were placed closely (Fig. 2.8(a)). Figure 2.8(b-f) show the consecutive grasps in the task. In this experiment, the proposed algorithm directly exploited grasp poses on the unsegmented point cloud by fitting the fingertip surface to the object sets. Even though the surface composed by the cluttered objects became more complicated than a single object, ISF filtered out the grasps with collision and found a suitable grasp to pick up objects sequentially.

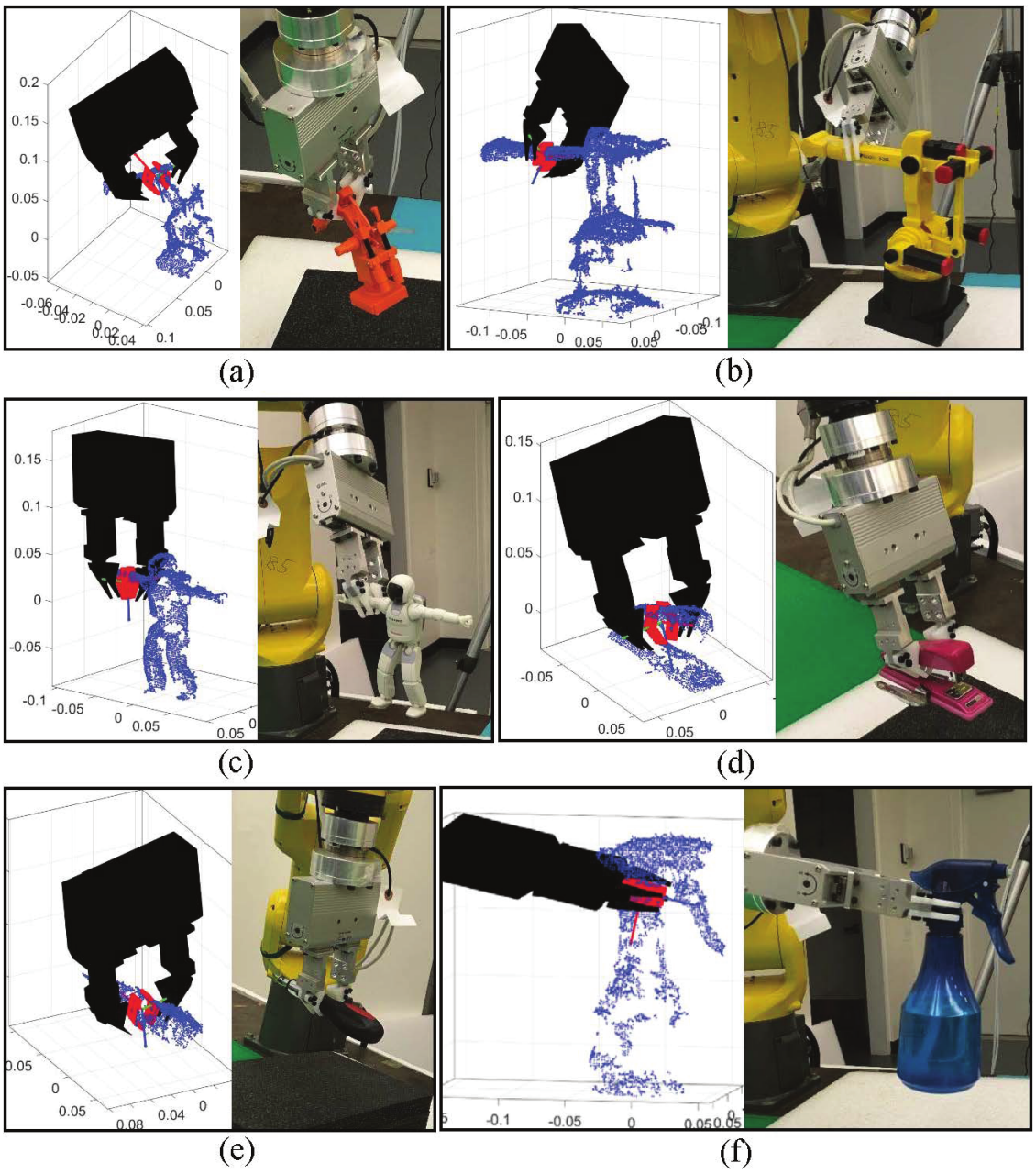


Figure 2.7: Experiment results on six individual objects.

2.5 Chapter Summary

This chapter proposed an iterative surface fitting (ISF) algorithm to plan grasps for customized grippers. ISF searches for optimal grasps by an iterative palm-finger optimization, which solves for the optimal palm pose and the finger displacement iteratively with closed-

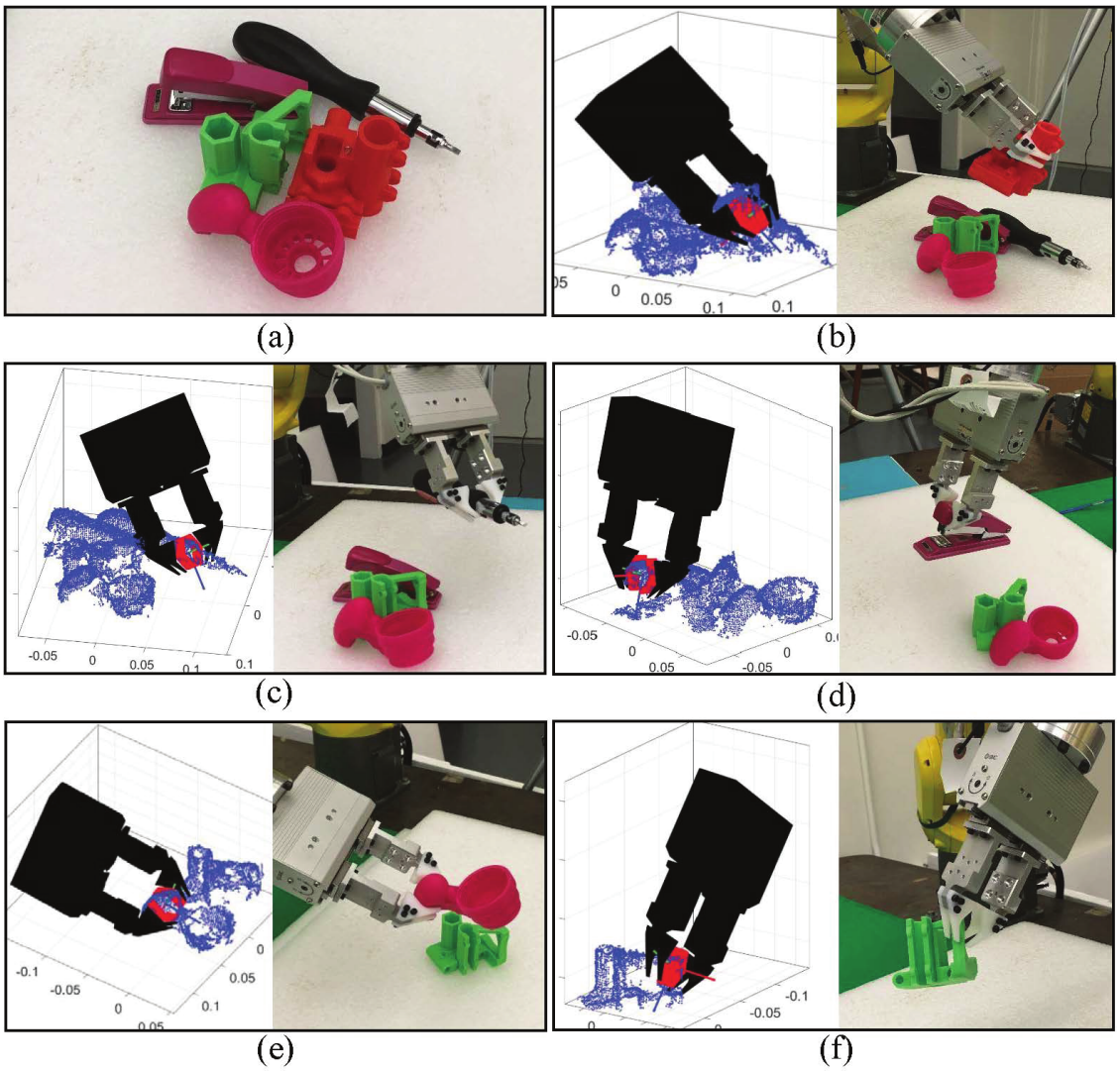


Figure 2.8: Grasp planning experiment in a clutter environment.

form solutions. To avoid bad local optima, a guided sampling was introduced to initialize ISF searching. The proposed grasp planning algorithm was applied to a series of simulations and experiments. ISF achieved 64.4 ms average searching time to find a collision-free grasp on the objects in simulations. The grasp planning was further implemented in clutter environments to grasp objects from unsegmented point clouds.

Chapter 3

Learning Efficient Grasp Exploration with Customized Grippers

3.1 Introduction

Chapter 2 presented an optimization-based planner called iterative surface fitting (ISF) to search grasps with customized grippers. Guided sampling was introduced to initialize and guide ISF in clutter environments to avoid ISF being trapped in ill-performed local optima. We refer this structure the baseline-ISF. Baseline-ISF behaves well in light clutter environments but becomes slow in heavy clutter environments.

To improve the efficiency of grasp planning in heavy clutter environments, we propose a topological learning approach using regions with convolutional neural networks (R-CNN) [34] to learn the essential features that affect the successful execution of grasps. The features include the spatial relationships between objects which are generally difficult to model. Therefore, the input to the CNN is the patches of the candidate regions. This learning-based planner is connected hierarchically with ISF in order to reduce the effect of object variations and plan reliable/precise grasps under data shortage. Therefore, the learning framework includes an optimization-based planner, which searches for precise grasp pose based on the fine details of the selected region using ISF, and a learning-based explorer, which learns the desired regions to start ISF searching.

Compared with end-to-end learning methods [62, 49], the proposed learning-based explorer ignores the details of grasp planning by detecting the desired regions within the image plane for potentially high fitting scores and better collision avoidance performance, thus the dimension of the learning module is lower than end-to-end learning. The optimization-based planner searches for optimal grasps precisely in the chosen region based on the object-specific features, which are generally not shared across objects and difficult to learn by end-to-end manner. Therefore, the proposed learning framework is able to improve the learning efficiency and performance at the same time.

The contributions of this chapter are as follows. The proposed hierarchical learning

framework provides a portable and reliable method to combine the precision and reliability of the optimization-based planner and the efficiency and intelligence of the learning-based explorer. On one hand, the proposed low-level optimization-based planner includes both the fingertip surfaces and the structural constraints of the gripper such as the jaw width and the allowable degree-of-freedoms (DOFs). It also achieves simultaneous surface fitting and gripper kinematic planning. On the other hand, the proposed grasp exploration strategy avoids getting stuck in the local optima by learning from the previous grasping experience. The grasp planning by ISF and R-CNN achieves an efficient planning and the time to search for a collision-free grasp is 1.5 secs for the objects in heavy clutter environments.

The remainder of this chapter first introduces the proposed learning framework in Section 3.2. After which the learning-based explorer based on R-CNN is presented in Section 3.3. Section 3.4 shows the experimental verification of the proposed framework with customized grippers. Section 3.5 summarizes the chapter. The experimental videos are available at [102].

3.2 The Hierarchical Learning Framework

In this chapter, we propose a hierarchical learning framework to extend the baseline-ISF in Chapter 2. To avoid being trapped into local optima and achieve better collision avoidance performance in clutter environments, we propose a learning-based explorer to detect the desired regions to initialize the low-level search. In summary, the learning framework includes both the low-level optimization-based planner to search for optimal grasp poses precisely and locally, and the high-level learning-based explorer to explore different grasp regions and avoid local optima.

Figure 3.1 shows the overall learning framework. The proposed learning framework decouples the end-to-end learning into a low-level optimization-based planner and a high-level learning-based explorer. The optimization-based planner searches for optimal gripper pose and finger configuration with the iterative surface fitting (ISF) from Chapter 2. The learning-based explorer detects the desired region to explore from the previous grasping experiences with R-CNN. Compared with end-to-end learning methods, the optimization-based planner is more precise and reliable when grasping on various unknown objects, and the learning-based explorer is more efficient with much less training data and lower learning dimension.

3.3 Learning-based Explorer

The optimization-based planner solved by baseline-ISF in Chapter 2 is inefficient and sub-optimal in heavy clutter environments. Meanwhile, we found that human tends to decouple the process of choosing the desired grasp region from that of searching specific grasp poses inside that region to increase the efficiency of the grasping. With this observation, we design a learning-based explorer to initialize the baseline-ISF search. The explorer selects the regions

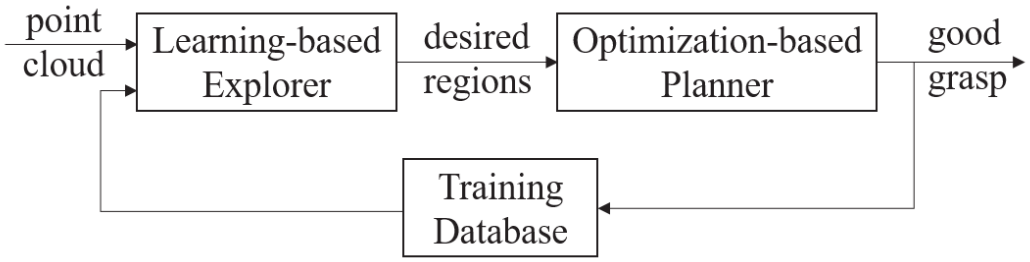


Figure 3.1: Block diagram of the hierarchical learning framework with customized grippers.

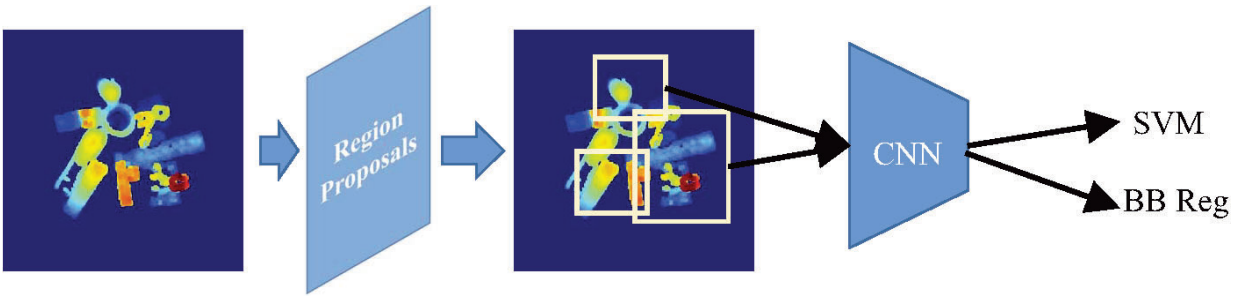


Figure 3.2: Illustration of R-CNN pipeline for learning-based exploration.

with potential low regret based on the previous grasp experiences. We use R-CNN to learn a classifier in order to detect the desired regions for initialization.

R-CNN Pipeline

The pipeline of R-CNN is shown in Fig. 3.2. R-CNN is first introduced in [34] for object detection. R-CNN contains a region proposal block to provide possible choices of regions. The region proposal selects 2K regions with different sizes using the method such as selective search. The regions are resized and fed into a CNN for feature extraction. The CNN can be pre-trained by AlexNet [46] or VGG-16/19 [87]. The outputs of CNN are used to represent the features of the region proposals. SVMs are applied to classify the regions and bounding box regression is applied to further correct the positions of the bounding boxes.

R-CNN Training

We use a grasping pool with 25 objects. We randomly choose some of the objects and place them in the workspace. The scene shown in Fig. 3.3(a) is observed by two stereo cameras and the collected point cloud (Fig. 3.3(b)) is rendered as depth images. The closer of the points to the camera, the more white they are in the depth image (Fig. 3.3(c)). The image is further rendered into jet colormap (Fig. 3.3(d)). R-CNN takes the rendered depth images as

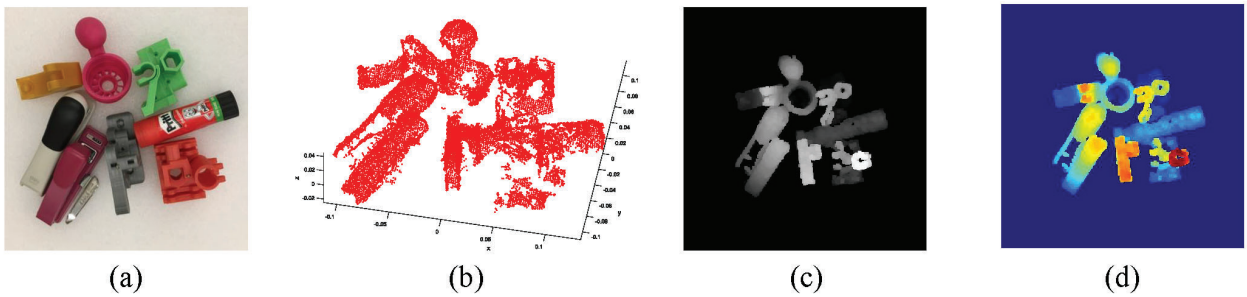


Figure 3.3: The depth rendering. (a) Original scene. (b) Point cloud observed by two stereo cameras. (c) Rendered depth images (d) Jet colormap.

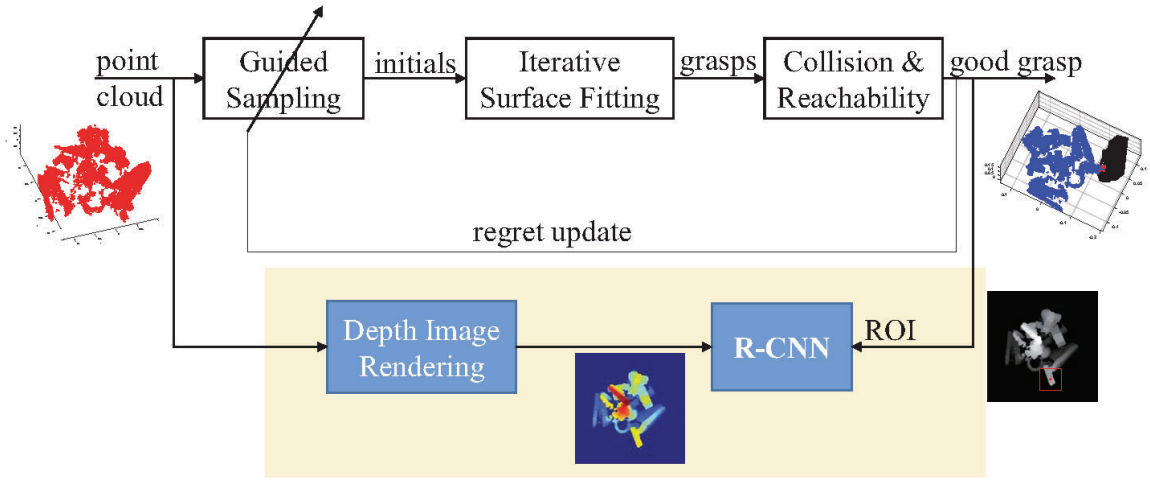


Figure 3.4: Illustration of the training framework.

inputs, and produces regions of interest (ROI) to initialize the ISF searching. The ROI used for training is generated based on the optimal grasps found from baseline-ISF. The training process is illustrated by Fig. 3.4. The R-CNN in this chapter is pre-trained by AlexNet and is fine-tuned by the data we collected. In this stage, we use 250 data pairs with data augmentation, producing 2000 data pairs to fine tune the network. Some of the data pairs are illustrated in Fig. 3.5.

R-CNN Testing

At the test time, the trained R-CNN is applied to generate the ROI on the colormap. The desired regions in the point cloud are then computed based on the box coordinates of ROI in the image plane as well as their depth values. The centers of these regions are regarded as the good initialization and ISF searches from these initial positions. The proposed learning ISF is called RCNN-ISF and the framework is illustrated in Fig. 3.6. The proposed learning framework with RCNN-ISF not only searches grasps locally within a small region, but also

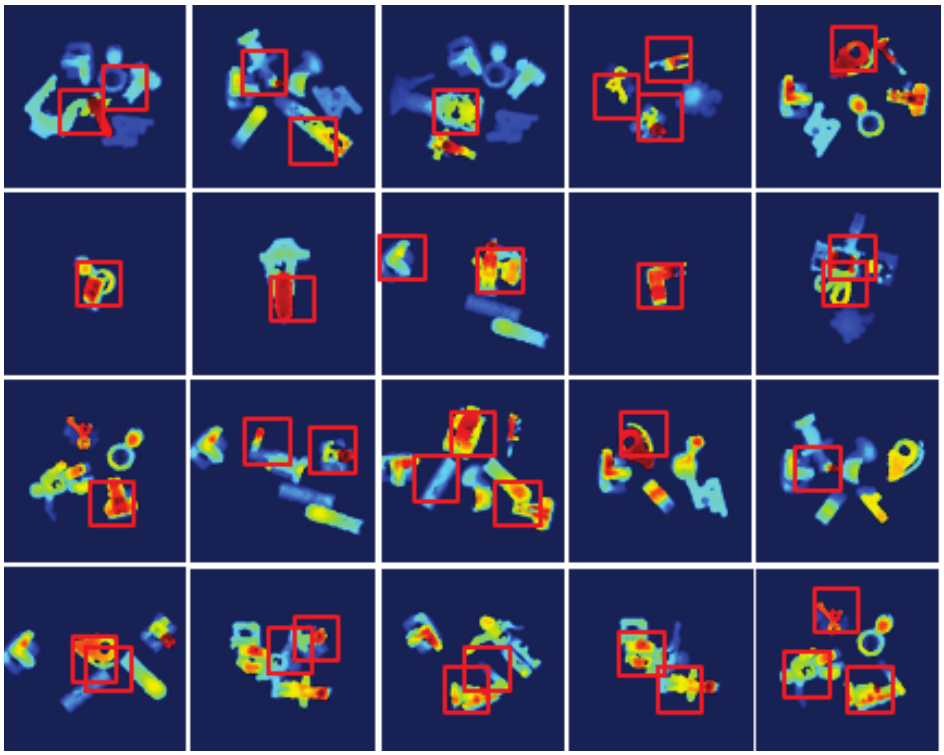


Figure 3.5: Database used to fine-tune the R-CNN for region detection.

learns the large-scale grasp exploration using R-CNN. Therefore, it tends to provide better initialization more efficiently than the baseline-ISF.

Compared with end-to-end learning [62, 49], the proposed RCNN-ISF method has the following advantages. First, the learning dimension of the RCNN-ISF is generally lower than that of the end-to-end learning. More specifically, RCNN-ISF searches ROI in the two-dimensional image plane, while the end-to-end learning searches over higher dimension depending on the grasps and grippers. For example, a grasp planning for a eight-DOF hand with three fingers has 32 dimensions [24]. Therefore, the end-to-end learning requires much more data than the proposed method. Secondly, ISF searches for optimal grasps based on object-specific features that are not shared cross objects, instead of learning the behavior end-to-end from millions of data. Consequently, ISF tends to generate more precise and robust grasps. Moreover, RCNN-ISF is able to produce versatile grasp as the experiment shows. On the contrary, the learned networks in [62, 49] produce top-down grasps with simple parallel grippers.

3.4 Experiment Study

This section presents the experimental results of the proposed learning framework with RCNN-ISF. The experimental videos are available at [102]. The learning framework includes

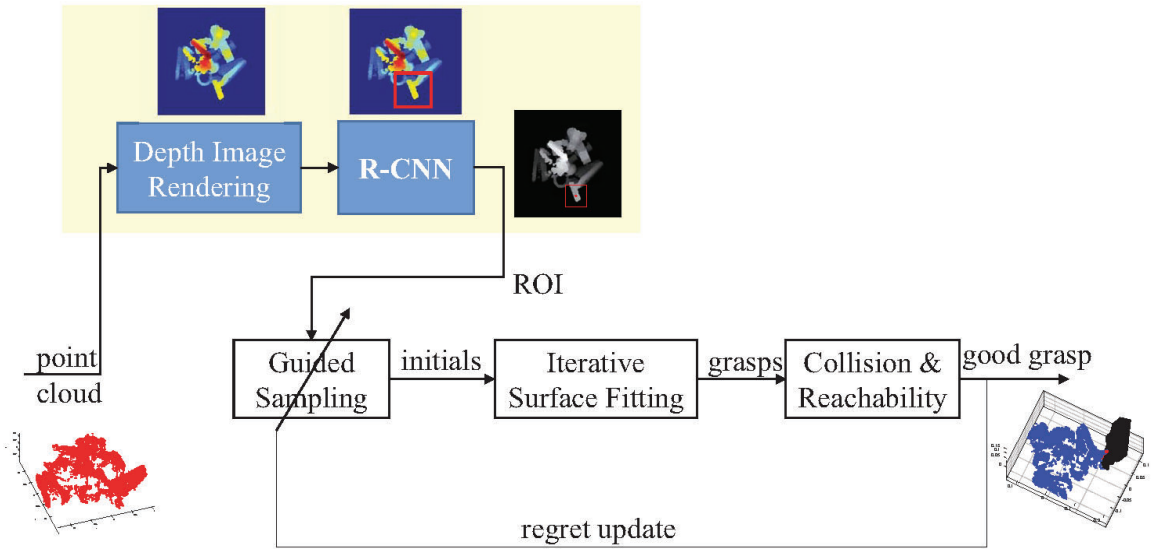


Figure 3.6: The learning framework with RCNN-ISF implementation.

the low-level optimization-based planner with baseline-ISF, and the high-level learning-based explorer with R-CNN for grasp exploration in heavy clutter environments. The training process of R-CNN is shown in Fig. 3.5.

Figure. 3.7 illustrates the performance of the RCNN-ISF in a light clutter environment. In this environment, RCNN-ISF achieved comparable performance with the baseline-ISF. The initial configuration of the object set is shown in Fig. 3.7(a). Figure 3.7(b)-(f) show the consecutive grasps in the task. The left side of each subfigure is the depth image and the R-CNN output of the confidence map for the desired regions, after which the baseline-ISF was performed on the chosen regions, and the best grasp was executed by the FANUC industrial manipulator, as shown in the right side of each subfigure. Even though the surface composed by the cluttered objects became more complicated than a single object, RCNN-ISF was able to successfully detect the desired regions for grasping and search on the selected regions for the optimal collision-free grasp.

Figure 3.8 shows the grasp planning results in heavy clutter environments by RCNN-ISF. The first row shows different clutter environments. R-CNN took rendered depth images as inputs and generated the desired regions to initialize ISF searching, as shown in Fig. 3.8 (Middle). The optimization-based planner produced optimal grasp pose by minimizing the fitting error and checking the collision/feasibility. The optimal grasps were executed by the FANUC manipulator, as shown in Fig. 3.8 (Bottom).

The comparison between the baseline-ISF and RCNN-ISF is shown in Table 3.1 for the clutter environments in Fig. 3.8. The searching in clutter environments is more difficult due to the collision with environments and the excessive search space for guided sampling. Consequently, the initialization becomes more important for efficient exploration. The baseline-ISF requires to initialize and execute ISF multiple times sequentially in order to explore broader

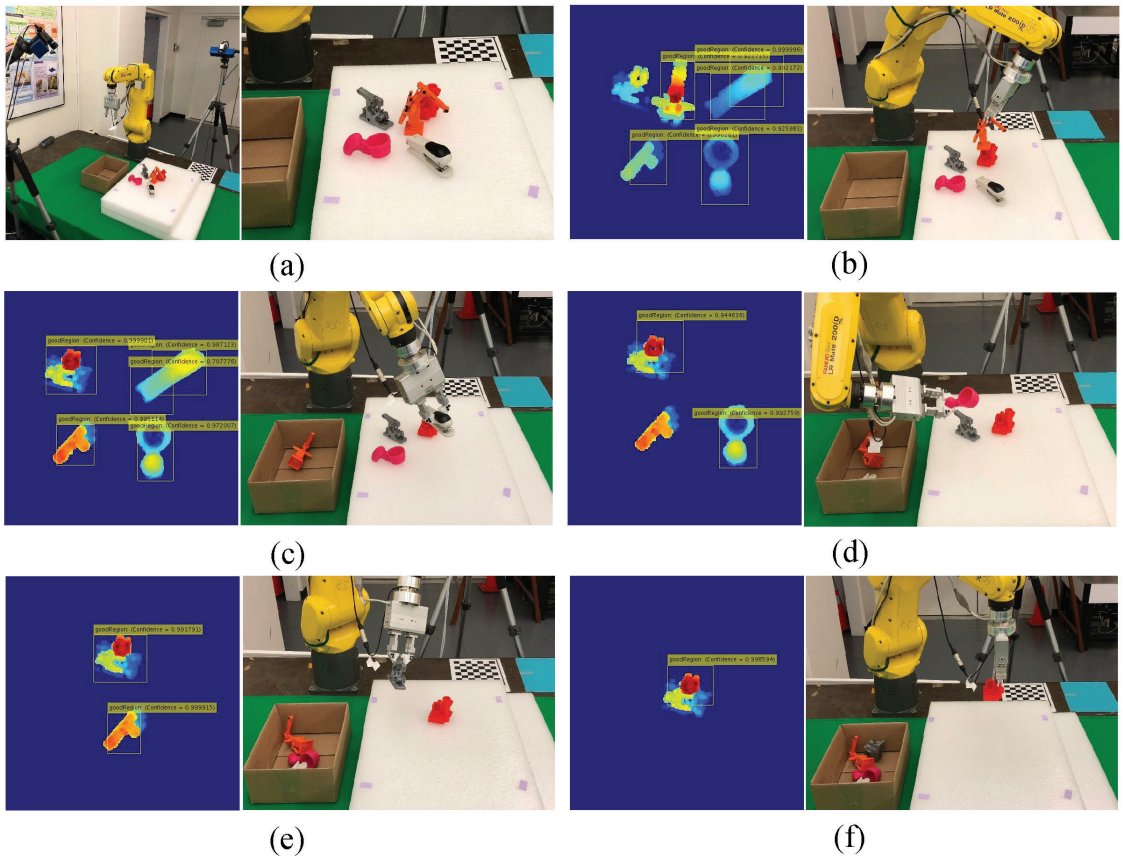


Figure 3.7: Grasp planning experiment in a clutter environment. (a) The initial object clutter. (b)-(f) The consecutive grasps in the task.

Table 3.1: Comparison of Baseline-ISF and RCNN-ISF

Methods	Baseline-ISF	RCNN-ISF
Search Time (s)	17.23	1.52
Found Grasps#	1	7

regions. On the contrary, RCNN-ISF employs the previous experience for initialization and generates only a few low-regret regions to start ISF searching. Therefore, RCNN-ISF is able to perform grasp planning more efficiently. In heavy clutter environments, the baseline-ISF spent 17.23 secs to find 1 optimal grasp, while the proposed RCNN-ISF spent 1.52 secs to find 7 optimal grasps.

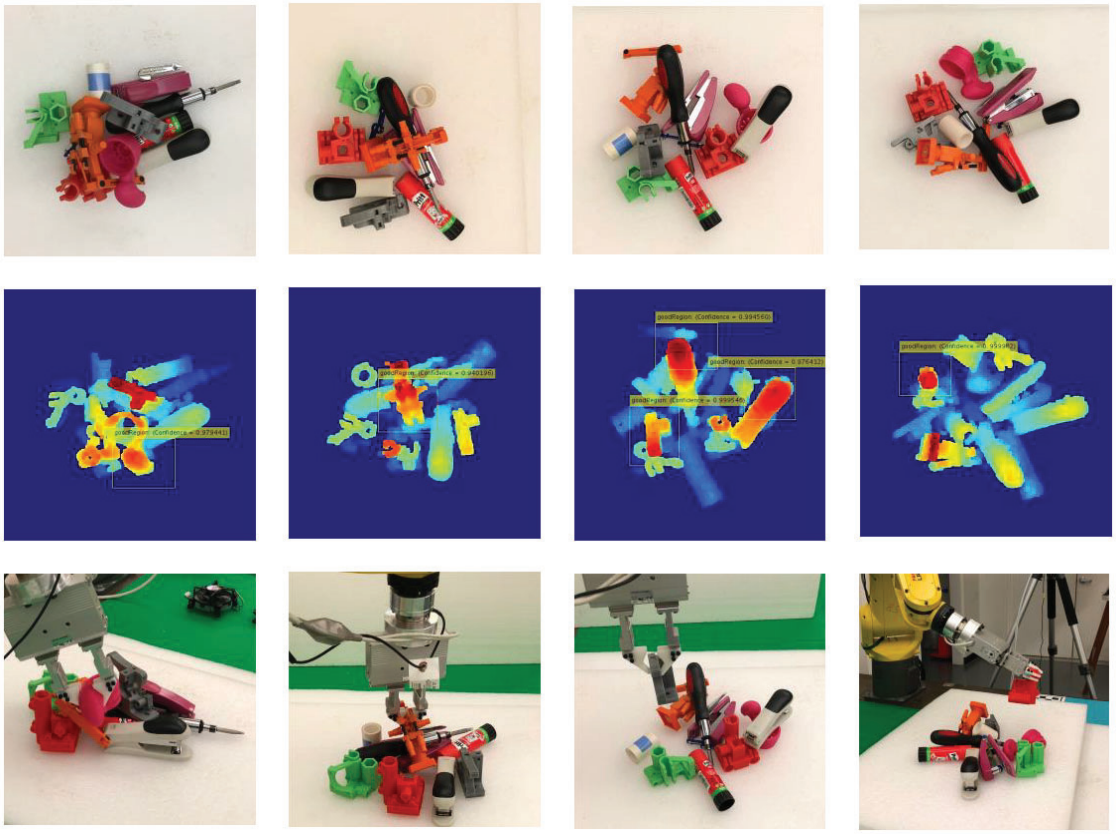


Figure 3.8: Grasp planning results in four different clutter environments by RCNN-ISF.

3.5 Chapter Summary

This chapter proposed a learning framework to plan robust grasps for customized grippers. The learning framework includes a low-level optimization-based planner and a high-level learning-based explorer. The optimization-based planner uses an iterative surface fitting (ISF) with guided sampling to search for optimal grasps by minimizing the surface fitting error. The performance of this low-level planner is locally effective and sensitive to initialization. Therefore, the learning-based explorer was introduced with a region-based convolutional neural network (R-CNN) to search for desired low-regret regions to initialize ISF search. A series of experiments on robotic bin picking were performed to evaluate the proposed method. Experimental results showed that the proposed learning framework with RCNN-ISF achieved a more efficient planning in heavy clutter environments, by significantly decreasing the average searching time from 17.23 secs to 1.52 secs.

Chapter 4

Transferring Grasps from Parallel Grippers to Multi-Fingered Hands by Finger Splitting

4.1 Introduction

Previous chapters introduced a grasp planning and exploration method with customized industrial grippers. Customized grippers behave reliable in mass production but usually require considerable time to design hardware and program motion sequences in mass customization. On the other hand, grasp planning for multi-fingered hands is important for robotic grasping and manipulation in order to increase the dexterity and collaborate with humans. First, the increase of joints offers more degree of freedoms (DOFs), which intrinsically provides more manipulability, makes the grasp more dexterous than parallel-jaw grippers. Moreover, by introducing more contact points, the grasp with multi-fingered hands is able to resist larger disturbances during the grasping and manipulation tasks. A general purposed multi-fingered hand will greatly simplify the procedure and improve the adaptability to new tasks. However, the grasp planning for general purposed multi-fingered hands is challenging due to the large variations of objects, coupling between the hand and objects, and high dimensionality of the hand-object system. More specifically, a multi-fingered hand for general purposes should be able to grasp different objects with various surfaces and sizes, from simple boxes to toys or tools with complicated shapes. Moreover, a stable grasp relies on proper contacts between the fingers and the object. The contacts that associate the hand and the object have to be on the object surface and also reachable by all the fingers attached to the palm. Considering the configuration of multiple fingers and aforementioned constraints, the searching for optimal grasp in the sense of maximizing the object grasp quality and the hand manipulability, becomes a high-dimensional problem, thus has strong limitation for real-time applications.

Due to the high dimensional state space, majority of the planning researches use sampling based methods [65, 99, 12, 83, 88] by employing the precise 3D mesh model of the object. To

overcome the curse of dimensionality, the hand poses in [99] were sampled around the object skeleton. To accelerate the inverse kinematics (IK) and collision detection during sampling, the object and fingertip workspace in [83] were approximated as tree structures. A part-based grasp planning method is proposed in [1] using Reeb graph, with the assumption that the grasping is constrained in single part of the object. Objects are represented by inscribing spheres in [78] and the grasp searching is conducted on qualified subset of spheres. Without the gradient information, the sampling-based methods essentially rely on dense sampling and heavy manually-designed heuristics. In [38], a hierarchical finger space is generated on object surface and the grasps are synthesized by a multi-level refinement strategy. The mapping from the contact pairs to hand configuration is obtained by an object-specific reachability table generated offline.

Compared with multi-fingered hands, the grasp planning for parallel grippers is conducted in much lower dimensional space due to the simpler gripper structures and fewer constraints. In [49], a deep reinforcement learning approach is proposed to directly learn the grasping policy from images. The grasping skills are learned from exploration measured by empirical success rate. Some others utilize databases to learn grasps for similar objects, these include the Columbia grasp database [35] and dexterity network (Dex-Net) [62].

In this chapter, we will study the knowledge transfer from grasps for parallel grippers to those for multi-fingered hands. We propose a strategy called finger splitting, to generate precision grasps for a multi-fingered hand from parallel ones. To be more specific, the multi-fingered hand is initialized by a parallel grasp with two contacts by assuming that the fingers are separated into two groups around contacts. The grasps for parallel grippers can be computed from database for parallel grippers [62] or planned by iterative surface fitting (ISF) in Chapter 2. Then the splitting algorithm gradually spreads all fingers from the original two contacts by optimizing both the object grasp quality and the hand manipulability. An optimal grasp with proper palm pose, joint angles and contacts will be generated after splitting.

The contributions of this chapter are as follows. First, a novel finger splitting strategy is proposed to transfer the knowledge from grasp databases for parallel grippers to planning of precision grasps for multi-fingered hands, where only fingertips are contacted with the object. The transferring leads to better feasibility guarantee and faster convergence. Moreover, a dual-stage iterative optimization algorithm is proposed to control the splitting behavior and search for optimal configurations to maximize both the object grasp quality and the hand manipulability. The dual-stage optimization effectively reduces the coupling between the palm and contacts and decomposes the high-dimensional optimization problem into two lower dimensional optimizations with fewer constraints. Furthermore, the decomposed optimizations are solved by customized gradient projections. The customization avoids modeling of the object surface and is computationally efficient. The average computation time is less than one second for the objects of different categories, and is appealing for real-time applications. The proposed approach is verified by simulations, and the simulation video is available at [102].

The remainder of this chapter is described as follows. First, the problem of a general grasp

planning for multi-fingered hands is stated in Section 4.2, followed by a detailed explanation of the proposed finger splitting in Section 4.3. Simulation results are presented in Section 4.4. Section 4.5 concludes this chapter and proposes future works.

4.2 Planning Multi-Fingered Grasps by Optimization

This section describes a general precision grasp planning problem with a three-fingered robotic hand illustrated in Fig. 4.1. The goal of the planning is to search for optimal contacts on the object and the associated hand configuration that maximize the object grasp quality and the hand manipulability. The contact position vector is denoted by $\boldsymbol{c} = \left[c_1^T, c_2^T, c_3^T\right]^T$, where $c_i \in \mathbb{R}^3$ is the contact for the i-th finger. The joint angle vector of the hand is denoted by $\boldsymbol{q} = \left[q_1^T, q_2^T, q_3^T\right]^T$, where $q_i \in \mathbb{R}^{N_{\text{jnt},i}}$ is the joint angle for the i-th finger and $N_{\text{jnt},i}$ is the number of joints for the i-th finger. The pose of the palm is represented by rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$. The contacts can be modeled as point contact with friction model [69] and would exhibit rotational freedoms. They can be represented as ball joints and characterized by Euler angles $\boldsymbol{E} = \left[E_1^T, E_2^T, E_3^T\right]^T$, where $E_i \in \mathbb{R}^3$ is the Euler angle for the i-th contact. Mathematically, the grasp planning problem can be formulated

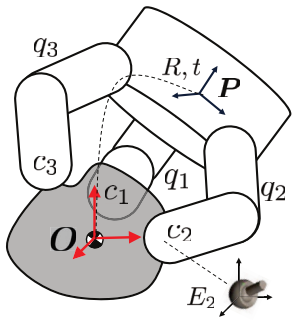


Figure 4.1: Illustration of grasp planning problem with a three-fingered hand.

as:

$$\max_{R,t,\boldsymbol{q},\boldsymbol{E},\boldsymbol{c}} \; Q(\boldsymbol{c},\boldsymbol{q}) \tag{4.1a}$$

$$s.t. \quad (R,t) = FK_{c2p}(\boldsymbol{q},\boldsymbol{E},\boldsymbol{c}) \tag{4.1b}$$

$$c_i \in \partial O \quad i = 1 \cdots 3 \tag{4.1c}$$

$$q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1 \cdots 3 \tag{4.1d}$$

where $Q(\boldsymbol{c},\boldsymbol{q})$ is the overall grasp quality containing both the object grasp quality and the hand manipulability, FK_{c2p} is the forward kinematics from contacts to the palm, and $q_{\min,i}, q_{\max,i} \in \mathbb{R}^{N_{\text{jnt},i}}$ are joint limits for the i-th finger. Constraint (4.1b) connects the palm and contacts through finger joints, (4.1c) constrains contacts on the object surface ∂O,

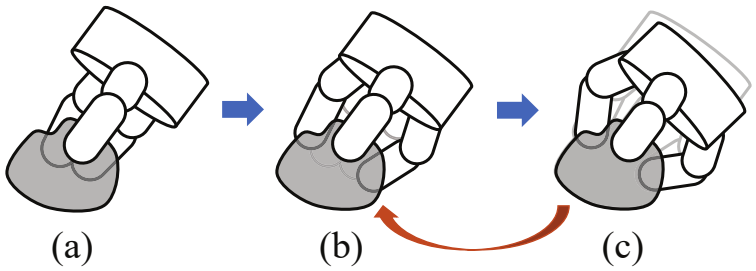


Figure 4.2: Finger splitting using dual-stage iterative optimization. (a) parallel grasp initialization, (b) contact point optimization, (c) palm pose optimization.

and (4.1d) shows the joint limits. With the palm pose, joints and contacts as optimization variables, the forward kinematics (4.1b) and complicated object surface (4.1c) as constraints, the problem (4.1) becomes a high-dimensional nonlinear programming. The optimization may become more challenging when considering collision (i.e. unexpected contact) between the hand and object.

In this chapter, we propose a finger splitting strategy to solve (4.1). The finger splitting separates all the fingers into two groups and gradually spread them to maximize the object grasp quality and the hand manipulability. The idea of finger splitting is shown in Fig. 4.2. The initial parallel grasps can be generated by database (e.g. Dex-Net [62]) or computed by ISF in Chapter 2, as shown in Fig. 4.2(a), after which a dual-stage iterative optimization is introduced to search for new contacts and hand configuration. Stage one is named as contact point optimization (CPO). The objective of the CPO is to maximize the object grasp quality and hand manipulability by searching for contacts $\boldsymbol{c}$ and joints $\boldsymbol{q}$ while keeping palm pose R, t fixed, as shown in Fig. 4.2(b). Stage two is called palm pose optimization (PPO). The objective of PPO is to improve the hand manipulability by searching over the palm pose R, t and joints $\boldsymbol{q}$ while keeping the contacts $\boldsymbol{c}$ fixed, as shown in Fig. 4.2(c). The CPO and PPO will be iteratively executed until converge or termination conditions are reached.

4.3 Finger Splitting

Contact Point Optimization (CPO)

The CPO searches for desired contacts and joints to maximize the object grasp quality and the hand manipulability by assuming that the palm pose R, t is static. The CPO is

formulated as:

$$\max_{\boldsymbol{c},\boldsymbol{q}} \; Q(\boldsymbol{c},\boldsymbol{q}) \tag{4.2a}$$

$$s.t. \quad \boldsymbol{c} = FK_{c2p}(\boldsymbol{q}, R_0, t_0) \tag{4.2b}$$

$$c_i \in \partial O \quad i = 1 \cdots 3 \tag{4.2c}$$

$$q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1 \cdots 3 \tag{4.2d}$$

where $Q(\boldsymbol{c},\boldsymbol{q}) = w_1 Q_o(\boldsymbol{c}) + w_2 Q_h(\boldsymbol{q})$ indicates the overall grasp quality composed by the object grasp quality Q_o and the hand manipulability Q_h. The object grasp quality Q_o can be represented by the triangle area formed by the contacts: $Q_o(\boldsymbol{c}) = 2\text{Area}\left(\{c_i\}_{i=1\cdots3}\right)$ [91], and the hand manipulability Q_h can be represented by the deviation from the center of the joints: $Q_h(\boldsymbol{q}) = -0.5\sum_{i=1}^{3}\sum_{j=1}^{N_{\text{jnt},i}}\left((q_i^j - \bar{q}_i^j)/(q_{\max,i}^j - q_{\min,i}^j)\right)^2$ based on [54], where q_i^j is the j-th joint angle of the i-th finger, $q_{\min,i}^j$ and $q_{\max,i}^j$ are the limits of q_i^j, $\bar{q}_i^j = (q_{\max,i}^j + q_{\min,i}^j)/2$ is the middle position of the corresponding joint. FK_{q2c} is the forward kinematics from joint $\boldsymbol{q}$ to contact $\boldsymbol{c}$, and R_0, t_0 denote the fixed rotation and translation of the palm.

Optimization (4.2) remains a nonlinear programming due to the nonlinearities of (4.2b) and (4.2c). Moreover, the object surface has to be fitted and fed into the optimization before running the gradient based method. In [23, 26], we introduced a velocity-level finger gaits planner for dexterous manipulation. A similar approach is proposed in this chapter to solve (4.2). The idea is to iteratively search on tangent space and project to the nonlinear constraints.

Tangent Space Searching

The tangent space searching is to find the displacement vectors in order to maximize the quality in the next time step. More specifically,

$$\max_{\boldsymbol{d_c},\boldsymbol{d_q}} \; \nabla_{\boldsymbol{c}} Q(\boldsymbol{c},\boldsymbol{q})\boldsymbol{d_c} + \nabla_{\boldsymbol{q}} Q(\boldsymbol{c},\boldsymbol{q})\boldsymbol{d_q} \tag{4.3a}$$

$$s.t. \quad \boldsymbol{d_c} = J_{q2c}(\boldsymbol{q}, R_0, t_0)\boldsymbol{d_q} \tag{4.3b}$$

$$\boldsymbol{n}^T(\boldsymbol{c})\boldsymbol{d_c} = 0 \tag{4.3c}$$

$$\|\boldsymbol{d}\| \le \sigma_{cpo} \tag{4.3d}$$

where $\boldsymbol{d_c} = \boldsymbol{c}(t+T_s) - \boldsymbol{c}(t)$ and $\boldsymbol{d_q} = \boldsymbol{q}(t+T_s) - \boldsymbol{q}(t)$ are displacement vectors for contacts and joints, T_s is the simulation time step, and $\boldsymbol{d} = [\boldsymbol{d_c}^T, \boldsymbol{d_q}^T]^T$. $J_{q2c} = \text{diag}\left([J_{q2c,1}, J_{q2c,2}, J_{q2c,3}]\right)$ denotes a geometric Jacobian from $\boldsymbol{q}$ to $\boldsymbol{c}$, and $J_{q2c,i} \in \mathbb{R}^{3\times N_{\text{jnt},i}}$ is the translational Jacobian for the i-th finger. $\boldsymbol{n}(\boldsymbol{c}) = \text{diag}\left(\left[\nabla_{c_1}^T(\partial O), \nabla_{c_2}^T(\partial O), \nabla_{c_3}^T(\partial O)\right]\right)$ is the surface normal matrix of the contacts, and $\nabla_{c_i}(\partial O) \in \mathbb{R}^{1\times3}$ is the normalized surface gradient w.r.t. c_i. σ_{cpo} is step size constraint.

Optimization (4.3) is the tangent approximation of (4.2) and can be solved analytically [60] by:

$$\begin{aligned} \boldsymbol{d^*} &= \alpha^* \boldsymbol{d_0} \\ \boldsymbol{d_0} &= (I - A^T(AA^T)^{-1}A)\nabla_{\boldsymbol{x}} Q^T(\boldsymbol{x}) \end{aligned} \tag{4.4}$$

where $\boldsymbol{d^*} = [\boldsymbol{d_c^{*T}}; \boldsymbol{d_q^{*T}}]^T$ is optimal tangent displacement vector and $A = [n^T(\boldsymbol{c}), 0; -I_9, J_{q2c}]$, $\boldsymbol{x} = [\boldsymbol{c}^T, \boldsymbol{q}^T]^T$. α^* is the optimal step size obtained by:

$$\alpha^* = \begin{cases} \sigma_{cpo}/\|\boldsymbol{d_0}\|, & \text{if } F_{ls} = 0 \\ \underset{0 \le \alpha \le \sigma_{cpo}/\|\boldsymbol{d_0}\|}{\arg\max} \left(Q\left(Proj\left(\boldsymbol{x} + \alpha \boldsymbol{d_0}\right)\right)\right), & \text{if } F_{ls} = 1 \end{cases}$$

where F_{ls} is the option for line search, and the $Proj$ represents projection operation introduced below. The line search can be disabled (i.e. $F_{ls} = 0$) empirically to accelerate the computation, though it would introduce certain oscillation.

Nonlinear Constraints Projection

Several steps are taken in order to project the tangent state $\boldsymbol{x} + \alpha \boldsymbol{d_0}$ back to the nonlinear constraints $h(\boldsymbol{c}, \boldsymbol{q}) = \{\boldsymbol{c} \in \partial O, \boldsymbol{c} = FK_{q2c}(\boldsymbol{q})\}$. First, the reference nearest neighbor $\boldsymbol{c}_{\text{ref}}$ of the optimal tangent contacts $\boldsymbol{c} + \boldsymbol{d_c^*}$ is searched by KD-Tree on the object surface. Second, the found nearest neighbor $\boldsymbol{c}_{\text{ref}} \in \partial O$ is tracked by a simple stiffness controller $\dot{\boldsymbol{q}}_{\text{des}} = J_{q2c}^{-1}(\boldsymbol{c}) K_{cpo}(\boldsymbol{c}_{\text{ref}} - \boldsymbol{f})$, where $\boldsymbol{f}$ is the current fingertip position vector. Last, the joint state in simulator is updated by $\boldsymbol{q}_{\text{des}} \leftarrow \boldsymbol{q} + \dot{\boldsymbol{q}}_{\text{des}} T_s$ if the termination condition is not satisfied.

Collision Detection

Unexpected contact between the object and the hand is called collision in this chapter since it might effect the execution of the optimized grasps. A simple collision detection algorithm is introduced to stop the finger splitting upon collision. The algorithm takes supervoxel representation of the object [72] as input, and check the inclusion relation of each supervoxel in finger links. The supervoxel representation usually only includes hundreds of points and the finger links are approximated by boxes or cylinders, thus the collision detection can be extremely fast. The collision detection is implemented in both CPO and PPO.

Finally, the overall CPO algorithm is summarized in Alg. (4). The CPO will be terminated once the quality increment is less or equal than δ_c, or the angle between the surface normal $\boldsymbol{n}_{\text{des}}$ and a predefined artificial fingertip normal $\boldsymbol{n_f}(\boldsymbol{q}_{\text{des}})$ is larger than γ, as shown in Line (7). A graphical illustration of the CPO algorithm is shown in Fig. 4.3. Figure 4.3(a) describes the tangent space searching (blue arrow line) and the nearest neighbor search (red arrow line) of the reference contact corresponding to Line (3-4), and Fig. 4.3(b) shows the tracking of the reference contact (purple arrow line) and estimation of the desired contact (red arrow line) corresponding to Line (6-7). The CPO is able to handle the case where the initial fingertip positions are not on the object surface by using the tracking control.

Palm Pose Optimization (PPO)

The CPO optimizes contact points by assuming that the palm pose is constant. However, the palm might not be in the best pose, thus the CPO can only find a local optimum in

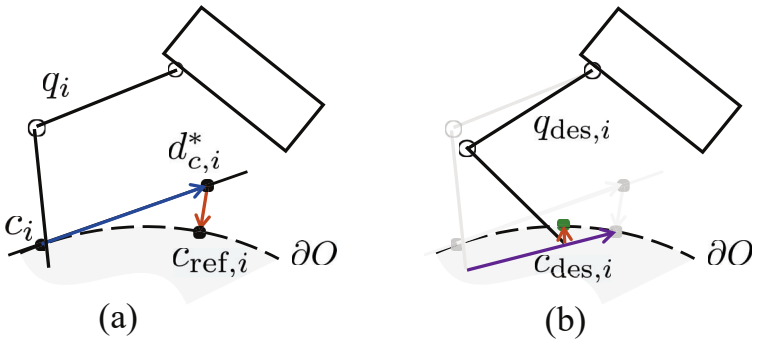


Figure 4.3: Illustration of the CPO algorithm. (a) tangent space searching and (b) nonlinear projection by reference tracking.

Algorithm 4 Contact Point Optimization (CPO)

Input: Static palm pose R_0, t_0, initial states $\boldsymbol{c}, \boldsymbol{q}$
while it_{cpo}++ $< M$ **do**
  Search tangent motion $\boldsymbol{d}^*_{\boldsymbol{c}}$ in (4.3) by solving (4.4)
  Search reference contact $\boldsymbol{c}_{\text{ref}} \leftarrow NN_{\partial O}(\boldsymbol{c} + \boldsymbol{d}^*_{\boldsymbol{c}})$
  Track reference contact $\dot{\boldsymbol{q}}_{\text{des}} \leftarrow J^{-1}_{q2c} K_{cpo}(\boldsymbol{c}_{\text{ref}} - \boldsymbol{f})$
  Compute desired state $\boldsymbol{q}_{\text{des}}, \boldsymbol{c}_{\text{des}}, \boldsymbol{n}_{\text{des}}$ by:
    $\boldsymbol{q}_{\text{des}} \leftarrow \dot{\boldsymbol{q}}_{\text{des}} T_s + \boldsymbol{q}$
    $(\boldsymbol{c}_{\text{des}}, \boldsymbol{n}_{\text{des}}) \leftarrow NN_{\partial O}(FK_{q2c}(\boldsymbol{q}_{\text{des}}, R_0, t_0))$
  Test termination: $\Delta Q_{\text{des}} = Q_{\text{des}}(\boldsymbol{c}_{\text{des}}, \boldsymbol{q}_{\text{des}}) - Q(\boldsymbol{c}, \boldsymbol{q})$
    $stop \leftarrow (\Delta Q_{\text{des}} \leq \delta_c) \parallel (|\boldsymbol{n}_{\text{des}}{}^T \boldsymbol{n}_{\boldsymbol{f}}(\boldsymbol{q}_{\text{des}})| \geq \gamma)$
    $collide = col\_detect(\boldsymbol{q}_{\text{des}}, R_0, t_0)$
  if $stop \parallel collide$ **then break**
  end if
  Set state $\boldsymbol{q} \leftarrow \boldsymbol{q}_{\text{des}},\ \boldsymbol{c} \leftarrow \boldsymbol{c}_{\text{des}},\ \boldsymbol{n} \leftarrow \boldsymbol{n}_{\text{des}}$
end while

subspace $\mathcal{S} = \{\boldsymbol{c} \in \partial O, \boldsymbol{q} \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}] \mid R_0, t_0\}$. In this section, the palm pose optimization will be introduced to maximize the overall grasp quality assuming that the contact points are static (i.e. object grasp quality Q_o is constant). PPO can be formulated as:

$$\max_{R \in SO(3), t, \boldsymbol{q}, \boldsymbol{E}} \ Q(\boldsymbol{c}_0, \boldsymbol{q}) \tag{4.5a}$$

$$s.t. \quad (R, t) = FK_{c2p}(\boldsymbol{c}_0, \boldsymbol{q}, \boldsymbol{E}) \tag{4.5b}$$

$$q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1...3 \tag{4.5c}$$

where $\boldsymbol{c}_0$ is the current static contacts on the object. The optimization (4.5) is nonlinear because of the constraint (4.5b) and $R \in SO(3)$. Similar to CPO, PPO in this section is

solved by linearization and projection.

Tangent Space Searching

Considering the relativity between the object and the hand motion, the hand palm is assumed to be static and object pose is optimized in PPO, since it is easier to implement and would increase the robustness to numerical errors. Rather than taking derivative of (4.5b), we notice that the linearization of (4.5b) is closely related to the fundamental grasping constraint [69]:

$$G(x_{po}, \boldsymbol{q})^T V_{po}^b = J_h(x_{po}, \boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.6}$$

where $x_{po} = [t_{po}^T, E_{po}^T]^T \in \mathbb{R}^6$ is a local parameterization of the object pose in palm frame $\boldsymbol{P}$, with t_{po} and E_{po} denoting the translation and orientation components, respectively. $V_{po}^b = [v_{po}^{b\ T}, \omega_{po}^{b\ T}]^T \in \mathbb{R}^6$ is the body velocity of the object, with v_{po}^b and ω_{po}^b denoting the translational and rotational velocities, respectively. $G(x_{po}, \boldsymbol{q}) \in \mathbb{R}^{6\times 9}$ and $J_h(x_{po}, \boldsymbol{q}) \in \mathbb{R}^{9\times N_{\text{jnt}}}$ represent grasp map and hand Jacobian [69]. Equation (4.6) connects joint velocity and object velocity under the static contact condition. The optimization (4.5) can be solved by utilizing this connection and searching on the tangent space. If V_{po}^b and $\dot{\boldsymbol{q}}$ are treated as the tangent displacements in one time step, then the tangent space searching of (4.5) becomes:

$$\max_{V_{po}^b, \dot{\boldsymbol{q}}} \ \nabla_{\boldsymbol{q}} Q(\boldsymbol{c_0}, \boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.7a}$$

$$s.t. \quad G(x_{po}, \boldsymbol{q})^T V_{po}^b = J_h(x_{po}, \boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.7b}$$

$$\|\dot{\boldsymbol{q}}\| \le \sigma_{ppo} \tag{4.7c}$$

where σ_{ppo} is a trust region of the joint motion. Similar to (4.3), (4.7) can be solved analytically as:

$$\begin{aligned} V_{po,\text{des}}^b &= \sigma_{ppo}\boldsymbol{d_c}/\|\boldsymbol{d_q}\| \\ \boldsymbol{d_c} &= G\left(G^TG + J_hJ_h^T\right)^{-1} J_h\nabla_{\boldsymbol{q}}^T Q \\ \boldsymbol{d_q} &= \nabla_{\boldsymbol{q}}^T Q - J_h^T\left(G^TG + J_hJ_h^T\right)^{-1} J_h\nabla_{\boldsymbol{q}}^T Q \end{aligned} \tag{4.8}$$

The desired tangent displacement of the object is represented as $V_{po,\text{des}}^b T_s$.

Nonlinear Projection

With the desired object tangent displacement in one step $V_{po,\text{des}}^b T_s$, the desired object pose $g_{po,\text{des}} = (R_{po,\text{des}}, t_{po,\text{des}})$ can be obtained by

$$g_{po,\text{des}} = g_{po} e^{\hat{V}_{po,\text{des}}^b T_s} \tag{4.9}$$

where $\hat{V}_{po,\text{des}}^b \in se(3)$ is the matrix representation of $V_{po,\text{des}}^b$ shown in [69] and $g_{po} \in SE(3)$ denotes the current object pose in palm frame. The joint $\boldsymbol{q}_{\text{des}}$ is computed as follows. First, the position $\boldsymbol{c}_{\text{des}}^{\boldsymbol{p}}$ and translational velocity $\boldsymbol{v}_{\boldsymbol{c},\text{des}}^{\boldsymbol{p}}$ of the static contact in palm frame is

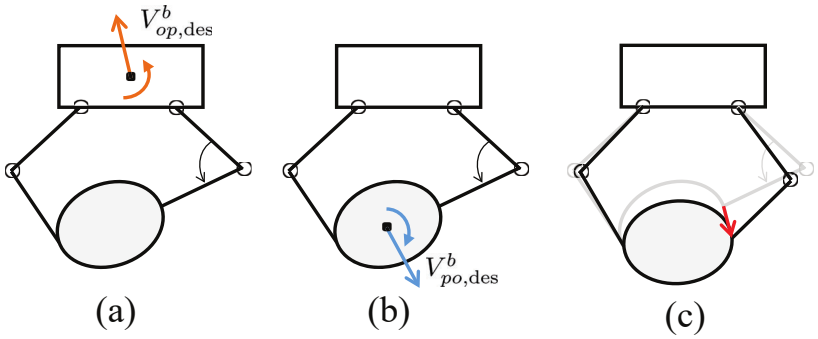


Figure 4.4: Illustration of the PPO algorithm.

obtained by the desired object motion $V^b_{po,\text{des}}$, after which a tracking controller is applied to compute the projected joint velocity $\dot{\boldsymbol{q}}_{\text{des}}$ by tracking both the position and velocity of the reference contact:

$$\dot{\boldsymbol{q}}_{\text{des}} = J^{-1}_{q2c}\left(\boldsymbol{v}^{\boldsymbol{p}}_{\boldsymbol{c},\text{des}} + K_{ppo}(\boldsymbol{c}^{\boldsymbol{p}}_{\text{des}} - \boldsymbol{f}^{p})\right) \tag{4.10}$$

where $\boldsymbol{f}^{\boldsymbol{p}} \in \mathbb{R}^9$ is the current fingertip position vector in palm frame, and K_{ppo} is the tracking gain used to reduce the misalignment between $\boldsymbol{c}^{\boldsymbol{p}}$ and $\boldsymbol{f}^{\boldsymbol{p}}$ during projection. With the projected displacement $\dot{\boldsymbol{q}}_{\text{des}}T_s$, the desired finger joints can be computed as $\boldsymbol{q}_{\text{des}} = \boldsymbol{q} + \dot{\boldsymbol{q}}_{\text{des}}T_s$.

Finally, the overall PPO algorithm is summarized in Alg. (5). PPO is terminated once the quality increment is less than δ_p, or angle between the surface normal $\boldsymbol{n}_{\text{des}}$ and the artificial fingertip normal $\boldsymbol{n}_{\boldsymbol{f}}(\boldsymbol{q}_{\text{des}})$ is larger than γ, as shown in Line (7). A graphical illustration of the PPO algorithm is shown in Fig. 4.4. Figure 4.4(a) shows the desired tangent space motion of the palm w.r.t. the object (orange arrow lines), and Fig. 4.4(b) shows the equivalent tangent space motion of the object w.r.t. the palm (blue arrow lines) corresponding to Line (3) based on the relativity of motion. Fig. 4.4(c) shows the computing and tracking of the desired contacts (red arrow line) corresponding to Line (4-6).

Iterative CPO-PPO

The proposed finger splitting decouples the optimization of palm from contacts, thus is capable to accelerate the computation and implement in real-time applications. CPO and PPO are iteratively optimized to achieve the finger splitting, as shown in Alg. (6).

The iterative CPO-PPO takes a parallel grasp and object mesh as inputs, as shown in Line (1). A simple function Map can be designed based on the structure of the hands to convert the parallel grasp parameterized by contacts c_1, c_2 and approach vector v_{ap} into a grasp for the multi-fingered hand parameterized by $\{\boldsymbol{c}, \boldsymbol{q}, R, t\}$, as shown in Line (2). The mapped initial grasp is adjusted by the proposed CPO/PPO. The CPO and PPO are optimized sequentially in the loop, as shown in Line (4-5). The iterative CPO-PPO will be terminated if both the CPO and PPO are close to convergence or reaching the constraint

Algorithm 5 Palm Pose Optimization (PPO)

Input: Static contact $\boldsymbol{c_0}$ on object, initial joint $\boldsymbol{q}$
while it_{ppo}++ $< M$ **do**
  Compute object $V^b_{po,\text{des}}, g_{po,\text{des}}$ by (4.8) and (4.9)
  Compute desired contact $\boldsymbol{c^p_{\text{des}}}, \boldsymbol{v^p_{c,\text{des}}}$ by:
    $c^p_{\text{des},i} \leftarrow R_{po} c_i + t_{po}$
    $v^p_{c_i,\text{des}} \leftarrow R_{po}(v^b_{po} - c_i \times \omega^b_{po})$
  Track desired contact by (4.10)
  Compute desired joint $\boldsymbol{q}_{\text{des}}$ by $\boldsymbol{q}_{\text{des}} \leftarrow \dot{\boldsymbol{q}}_{\text{des}} T_s + \boldsymbol{q}$
  Test termination $\Delta Q_{\text{des}} = Q(\boldsymbol{c_0}, \boldsymbol{q}_{\text{des}}) - Q(\boldsymbol{c_0}, \boldsymbol{q})$
    $stop = (\Delta Q_{\text{des}} \leq \delta_p) \parallel (|\boldsymbol{n}_{\text{des}}{}^T \boldsymbol{n_f}(\boldsymbol{q}_{\text{des}})| \geq \gamma)$
    $collide = col\_detect(\boldsymbol{q}_{\text{des}}, g_{po,\text{des}})$
  if $stop \parallel collide$ **then break**
  end if
  Set state $\boldsymbol{q} \leftarrow \boldsymbol{q}_{\text{des}},\ g_{po} \leftarrow g_{po,\text{des}}$
end while

Algorithm 6 Iterative CPO-PPO

Input: Parallel grasp $\{c_1, c_2, v_{ap}\}$, object mesh ∂O
Init: Initialize state $\{\boldsymbol{c}, \boldsymbol{q}, R, t\} \leftarrow Map(c_1, c_2, v_{ap}, \partial O)$
    $it_{cpo} = 0;\ it_{ppo} = 0$
while $True$ **do**
  Optimize contacts by Algorithm (4):
    $\{\boldsymbol{c}, \boldsymbol{q}, it_{cpo}\} \leftarrow CPO(R, t, \boldsymbol{c}, \boldsymbol{q})$
  Optimize palm pose by Algorithm (5):
    $\{(R, t)^{-1}, \boldsymbol{q}, it_{ppo}\} \leftarrow PPO(\boldsymbol{c}, \boldsymbol{q})$
  if $it_{cpo} < m$ **and** $it_{ppo} < m$ **then break**
  end if
end while

boundaries, as shown in Line (6-7). $m \in \mathbb{Z}^+$ denotes an iteration threshold to stop the finger splitting. A graphical illustration of the iterative CPO-PPO is shown in Fig. 4.2.

Convergence of the Iterative CPO-PPO

The convergence of the proposed iterative CPO-PPO is proved in this section using the global convergence theorem [60]. Firstly, we show that CPO converges to a local optimum if solved by Alg. (4). Based on the global convergence theorem, the convergence of the CPO requires 1) compact domain, 2) existence of a continuous decent function, and 3) closeness of the algorithmic mapping outside of the solution set. The domain $D = \partial O \times SE(3) \times R^{nq}$ is apparently compact. As for the continuous decent function, we use $-Q(x)$ as the decent function, it is continuous and decent at the outside of the solution set. Furthermore, the algorithmic mapping composited by the tangent space searching $\mathcal{T}$ and nonlinear projection $\mathcal{P}$ is closed in the absence of inequality constraints, since $\mathcal{T}$ is continuous and point-to-point, and $\mathcal{P}$ is closed in $\mathcal{T}(x)$. Therefore, the CPO solved by Alg. (4) converges to a local optimum. Similarly, PPO converges to a local optimum if solved by Alg. (5).

Secondly, we prove that the iterative CPO-PPO converges to a local optimum of (4.1). The composite mapping $\mathcal{A}_{\text{dual}} = \mathcal{A}_{\text{cpo}} \circ \mathcal{A}_{\text{ppo}}$ is closed since $\mathcal{A}_{\text{ppo}}$ is a continuous point-to-point mapping and $\mathcal{A}_{\text{cpo}}$ is closed on $\mathcal{A}_{\text{ppo}}(x)$, where $\mathcal{A}_{\text{cpo}}$ is Alg. (4) and $\mathcal{A}_{\text{ppo}}$ is Alg. (5). Therefore, we conclude that (4.1) solved by Alg. (6) converges to a local optimum.

4.4 Simulation Study

Simulation results are introduced in this section to verify the effectiveness of the iterative CPO-PPO. The simulation video is available at [102]. The grasp planning process was computed in Matlab, visualized in V-REP [80], and tested in Mujoco physics engine [96] on a Windows PC with 4.0GHz CPU and 32GB RAM. We used a build-in Barrett hand model but removed joint coupling and enabled all eight degree of freedoms (DOFs), as shown in Fig. 4.5.

Parameter Lists

Simulation time step $T_s = 0.05$ sec. The weights of grasp quality $w_1 = 1, w_2 = -0.01$ in (4.2). The maximum iteration $M = 50$ and termination condition $\gamma = 0.6$ in Alg. (4) and (5). The minimum iteration bound $m = 2$ in Alg. (6). The tracking gain $K_{cpo} = K_{ppo} = 2I_9$ and termination condition $\delta_c = \delta p = 0$. The σ_{cpo} and σ_{ppo} represent the trust regions of the search to guarantee the accuracy of the approximation of the cost and constraints, and are determined by the smoothness of object surface and kinematics of joints. Empirically $\sigma_{cpo} = 0.15$ and $\sigma_{ppo} = 0.5$.

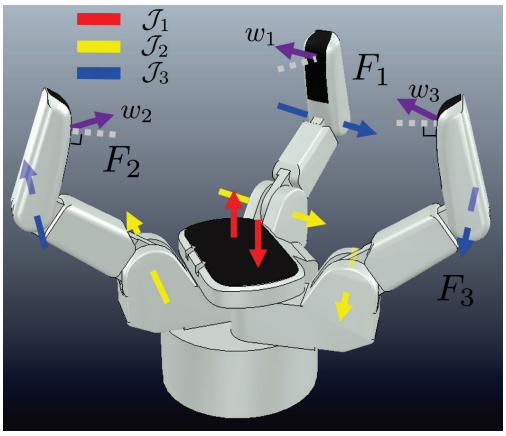


Figure 4.5: Illustration of the hand structure.

Finger Splitting Results

Figure 4.6 shows ten grasping examples by the proposed algorithm. The tested objects cover several categories including fruits (apple), toys (Doraemon, Hello Kitty, Oscar), animal (bunny), and tools (mug, screwdriver) with different number of vertices (100 - 150,000). To test the adaptability to different initial conditions, we also tested the grasping of the same object with different poses (e.g. Hello Kitty(H) shows the horizontal grasp of Hello Kitty, and bunny(R) denotes grasping of bunny with reverse pose). Each object was first smoothed by [15] then the surface normals were estimated by supervoxel clustering [72]. The optimization was able to find all the optimal grasps even the initial parallel grasps were infeasible, as shown in apple and bunny grasps of Fig. 4.6.

Table 4.1 shows the details of grasp synthesis for all ten objects. The 2-5th column shows the number of iterations, the total number of the tangent space searching (and projections), the total optimization time, and the number of vertices of the objects, respectively. In average, for an object with 64K vertices, the iterative CPO-PPO ran for 4.2 outer iterations (84.9/29.2 inner iterations for CPO and PPO, respectively) in total with 0.58 sec in order to find an optimal precision grasp for a three-fingered hand with 8 DOFs.

The average time distribution of the optimization for ten grasps is shown in Table 4.2. The tangent space searching (and projection) iterated for 84.9 and 29.2 times for CPO and PPO, respectively. The projection (403.2 ms) took longer time than tangent space searching (139.9 ms) because of the nearest neighbor computation.

Figure 4.7 shows the visualization of the iterative CPO-PPO on grasping the screwdriver. The initial grasp generated by a parallel grasp is shown in Fig. 4.7(a). Then CPO was enabled to solve for optimal contacts by maximizing the overall grasp quality. Due to the successive tangent space searching and projection, the fingers behaved as sliding on the object surface, as shown in Fig. 4.7(b)-(d). The CPO stopped when reaching the termination conditions in Alg. 4. Then PPO was enabled, and the pose of the palm was optimized. In practice,

Figure 4.6: Grasp planning examples for different objects with the multi-fingered hand.

the object pose was optimized due to the relativity of motion, as shown in Fig. 4.7(e)-(g). PPO and CPO could iterate for several times before converging or reaching the constraint boundaries (e.g. collision or $|\boldsymbol{n}_{\text{des}}{}^T \boldsymbol{n}_{\boldsymbol{f}}(\boldsymbol{q}_{\text{des}})| \geq \gamma$). The corresponding video is available in [102].

Figure 4.8 shows the quality profile of the grasp planning on the same object. The active regions for CPO and PPO are shown by yellow and blue shaded areas, and the quality profile is shown by red solid curve. The algorithm started from a low-quality parallel grasp and optimized for joints, contacts and palm by running the iterative CPO-PPO algorithm. PPO was enabled and terminated in the first iteration, since the J_1 was on the joint limit and there was insufficient joint space to improve the overall quality. The CPO was enabled from the second iteration and the contacts were searched by optimizing the overall grasp quality. The CPO ran for 28 iterations, after which PPO took over and optimized for object motion. CPO and PPO iterated until convergence.

Figure 4.9 shows the measurements of the finger splitting by several commonly adopted quality metrics. The measurements for ten grasps were resized to be the same length. The mean and standard deviation (SD) of the measurements for ten simulated grasps are represented by red sold lines and blue vertical bars, respectively. Figure 4.9(a) shows the

Table 4.1: Optimization Details for Grasp Generation

Objects	#Iters	#CPO/PPO	Time (ms)	#Vertices
Mug	5	90/22	1257.3	185,511
Oscar	2	69/3	719.7	148,616
Screwdriver	3	40/53	380.6	46,721
Bunny	4	106/28	465.5	43,318
Bunny(R)	4	62/44	403.7	43,318
Doraemon	6	125/12	484.0	42,551
Hello Kitty(H)	5	108/38	467.1	29,659
Hello Kitty	7	69/56	406.6	29,659
Apple	1	43/2	183.0	22,487
Polygons	5	96/21	1053.1	100
Average	4.20	84.9/29.2	582.1	63,516

Table 4.2: Average Time Distribution for Grasp Generation

Time (ms)	Tangent search	Projection & Test	Collision	Total
CPO	37.0	326.9	29.0	392.9
PPO	102.9	76.3	10.0	189.2
Total	139.9	403.2	38.9	582.1

profile of the quality metric optimized in this chapter. The average quality was increased monotonically during the finger splitting process. Figure 4.9(b) shows the profile of the grasp isotropy index $Q_{iso} = \sigma_{\min}(G)/\sigma_{\max}(G)$ [42]. Figure 4.9(c) shows the profile of the wrench space volume metric $Q_{vol} = \sqrt{\det(GG^T)}$ [53]. Figure 4.9(d) shows the profile of the Ferrari-Canny metric [30]. All the qualities were normalized in order to display the trend during grasp planning of different objects. Figure 4.9(b)(c)(d) indicate that the finger splitting solved by the iterative CPO-PPO improves grasping performance for all tested grasp metrics.

Verification with Physical Simulations

The quality improvement of the grasp after finger splitting was verified by the Mujoco physics engine. Given the desired palm pose and contact positions, the fingers were controlled using the virtual frame method [38]. To simulate the nonconvex object for contact and collision, the object was decomposed into convex shapes by v-hacd. Figure 4.10 shows the simulation results of the initial optimal parallel grasp and the finger splitting result. The initial parallel grasp could not lift the object successfully due to the gravitational force and acceleration as shown in Fig. 4.10 (Top), while the optimized grasp by finger splitting was able to hold the object steadily during lifting as shown in Fig. 4.10 (Bottom). The contact status and magnitude of the contact force are represented by the yellow cylinder and the gray arrow,

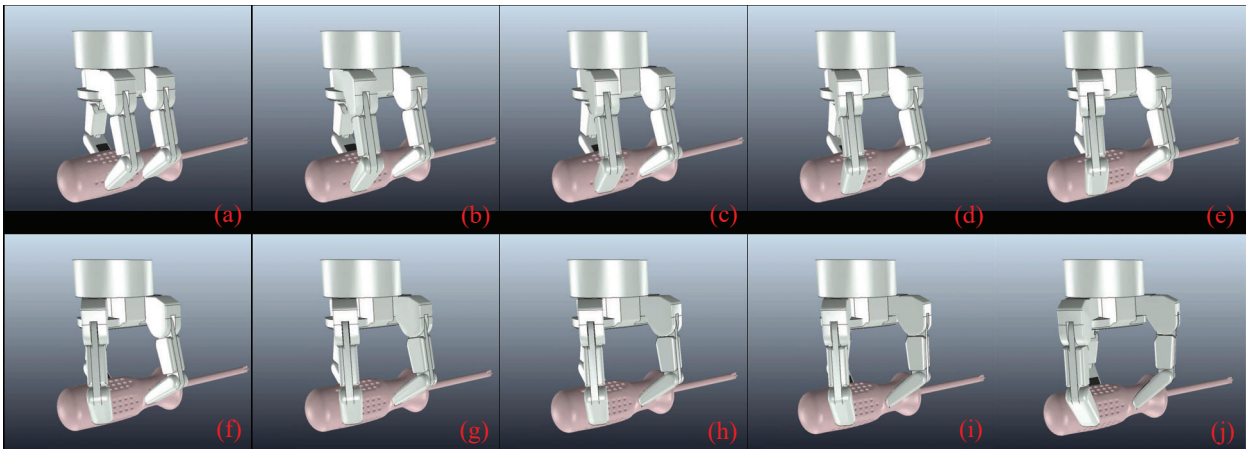


Figure 4.7: Snapshots of finger splitting on a screwdriver.

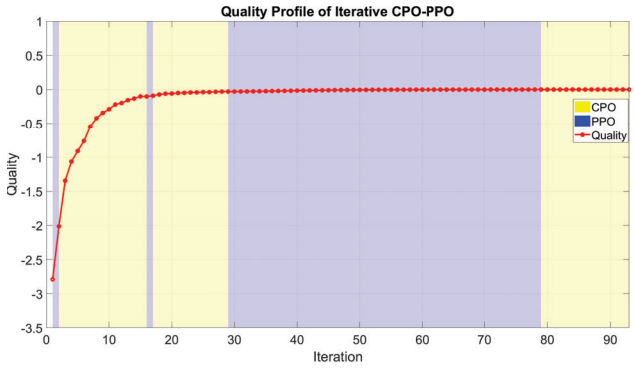


Figure 4.8: Quality improvement during iterative CPO-PPO on a screwdriver.

respectively.

Comparisons

The efficiency of the grasp planning by finger splitting was compared with the methods in [50] and [37] on bunny object, as shown in Table 4.3. The object surface in method [50] had to be fitted analytically (took 8.36 secs), after which an optimal grasp was searched by AMPL using IPOPT solver (took 15.32 secs). The hand reachability was not considered in the algorithm. The HFTS planner used a hierarchical representation of the object. The preprocessing time for the representation was 0.764 sec for the object with 1002 vertices and 13.41 secs for a point cloud with 20,586 vertices. The average time for grasp planning on Bunny object was 16.26 secs [37]. In comparison, the proposed method achieves the most efficient computation by initializing parallel grasps with ISF in Chapter 2 (took 0.15 sec),

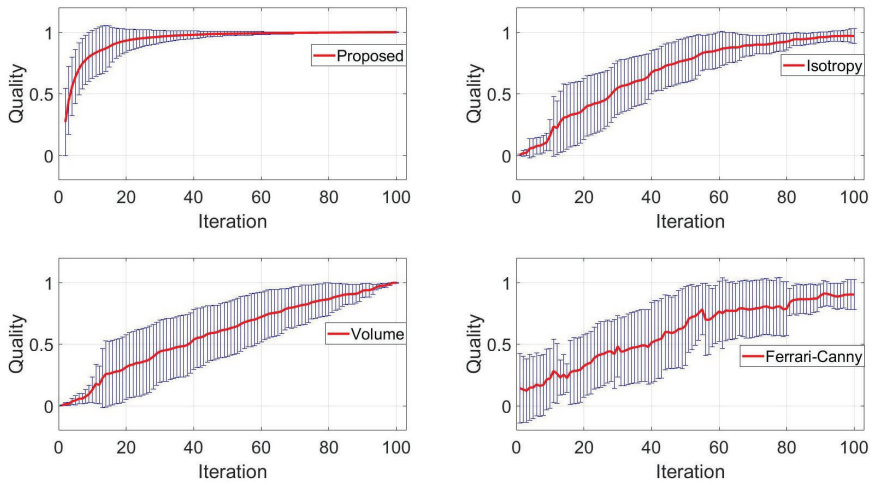


Figure 4.9: Normalized quality measurements including (a) the proposed quality metric. (b) grasp isotropy, (c) wrench volume, and (d) Ferrari-Canny metrics.

Figure 4.10: Comparison of the initial optimal parallel grasp (Top) and the finger splitting result (Bottom) in a physical simulator.

and searching for optimal grasps for multi-fingered hand by finger splitting (took 0.43 sec)[1].

[1]It is worth noting that different methods might converge to different local optima.

Table 4.3: Computation Time (Seconds) for Different Methods

Methods	Preprocessing	Optimization	Total Time
Li et. al. [50]	8.36	15.32	23.68
HFTS [37]	0.76 - 13.41	16.26	17.02 - 29.67
Proposed	0.15	0.43	**0.58**

4.5 Chapter Summary

This chapter proposed a finger splitting strategy for grasp planning with multi-fingered hands by transferring the knowledge from grasp databases of parallel grippers. The splitting was initialized by the planning result of the parallel gripper, and was optimized continuously by a novel iterative CPO-PPO algorithm. The CPO optimizes for contact points by assuming that the palm is static while PPO optimizes for palm pose by assuming that the contacts on object are static. CPO and PPO were both solved by consecutive tangent space searching and nonlinear projection. The iterative CPO-PPO algorithm was able to find a local optimal collision-free grasp within one second in average for the objects studied in simulations.

Chapter 5

Optimization Model to Plan Grasps with Multi-Fingered Hands

5.1 Introduction

Chapter 4 introduced a strategy called finger splitting to address the learning complexity and plan grasps for multi-fingered hands. The algorithm splits fingers by alternatively optimizing the palm pose and the contact positions starting from a parallel grasp. The algorithm is able to converge to a local optimum efficiently. However, instead of searching for new contacts to avoid the collision or reduce misalignment, the algorithm terminates if there is a collision between hand and object or large misalignment between contact normal and fingertip normal. Moreover, with fixed contacts, the palm may have less freedom to adjust if the hand is under-actuated or has low degree of freedoms (DOFs), thus the searching range is limited to small regions.

The objective of this chapter is to plan precision grasps with a general multi-fingered hand and remove the dependency on initial parallel grasps. As one category of grasps with multi-fingered hands, precision grasp has great importance in grasping the small/flat objects or executing the high-precision in-hand manipulation tasks. In these tasks, a robotic hand uses the fingertips to contact with the object and manipulate the object with the force/torque produced from these contacts. With the maximum joints in the kinematic tree, the fingertips provide enough dexterity and generate forces in different directions for object manipulation and disturbance rejection. The realization of precision grasps, however, is challenging due to the complex shape of the objects, high-dimensionality of the planning problem, collision avoidance requirements during grasp searching and execution, and the robustness to various uncertainties.

The idea of iterative contact point optimization (CPO) - palm pose optimization (PPO) in Chapter 4 provides a sustainable way to solve the nonlinear optimization in grasp planning. In this chapter, we further propose an optimization model to search for optimal grasps while avoiding collision with multi-fingered hands. The optimization model formulates the

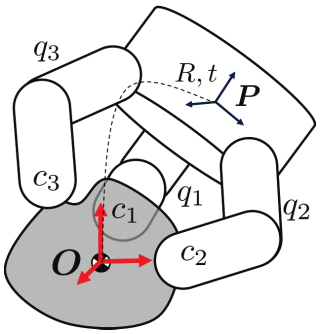


Figure 5.1: Illustration of grasp planning problem with a three-fingered hand.

planning problem into a gradient-based optimization and searches grasps from scratch. The optimization is solved by iterating between a palm pose optimization (PPO) and a joint position optimization (JPO). Instead of fixing contact points as Chapter 4, PPO in this chapter searches for desired palm pose to optimize the geometrical quality metrics with fixed finger joints. In contrast, JPO searches for desired finger joint positions to optimize both the object quality and hand manipulability with fixed palm pose.

The contributions of this chapter are as follows. First, the planning problem is formulated as a gradient-based optimization with a random start. Moreover, the collision is actively penalized and avoided within the optimization instead of pruning the collided grasps after the optimization. With the customized penalization, the iterative PPO-JPO changes the non-convex planning problem into a sequence of least-squares, and the average computation time is 0.5 sec/grasp. Third, the algorithm is robust to sensing uncertainties and noises by employing contact patches rather than individual contact points. The effectiveness of the algorithm is verified by both the simulation and the experiment. The experimental videos are available at [102].

The remainder of this chapter is as follows. The formulation of the planning problem and the optimization model are stated in Section 5.2. Section 5.3 presents an efficient iterative PPO-JPO to solve the modeled optimization. Simulation and experiment are introduced in Section 5.4. Section 5.5 summarizes the chapter.

5.2 Optimization Model for Precision Grasps

We consider the planning of precision grasps with multi-fingered hands. In the precision grasp mode, the hand contacts with the object by fingertips to gain most dexterity and increase the grasping manipulability. Precision grasps are necessary if the object to be grasped is flat or requires further precision operations.

Problem Statement

An example of precision grasp with a three-fingered hand is restated in Fig. 5.1. The contacts that connect the hand $\mathcal{F}$ and the object $\partial\mathcal{O}$ are denoted as $\boldsymbol{c} = [c_1, ..., c_{N_c}]$, where N_c is the number of contacts. Different contacts are associated by the hand configuration characterized by the palm pose ($R \in SO(3), \boldsymbol{t} \in \mathbb{R}^3$) and the joint angles $\boldsymbol{q} = [q_1, ..., q_{N_c}]$, where $q_i \in \mathbb{R}^{N_{jnt,i}}$ is joint angles of the i-th finger and $N_{jnt,i}$ is the number of joints for this finger.

Given the object $\partial\mathcal{O}$, the grasp planning is to determine the grasp $\mathcal{G} = \{\boldsymbol{c}, (R, \boldsymbol{t}), \boldsymbol{q}\}$ to successfully lift the object. To be more specific,

$$\max_{R,\boldsymbol{t},\boldsymbol{q},\boldsymbol{c}} \; Q(\boldsymbol{c}, \boldsymbol{q}) \tag{5.1a}$$

$$s.t. \quad \boldsymbol{c} \in FK(\mathcal{F}_t; R, \boldsymbol{t}, \boldsymbol{q}) \tag{5.1b}$$

$$\boldsymbol{c} \in \partial\mathcal{O} \tag{5.1c}$$

$$dist(FK(\mathcal{F}; R, \boldsymbol{t}, \boldsymbol{q}), \partial\mathcal{O}) \geq 0 \tag{5.1d}$$

$$q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1 \cdots N_c \tag{5.1e}$$

where $Q(\boldsymbol{c}, \boldsymbol{q})$ denotes the grasp quality including the $Q_{com}(\boldsymbol{c}, \partial\mathcal{O})$, $Q_{jc}(\boldsymbol{q})$ and $Q_{align}(n_c, n_f)$. Q_{com} is the distance between the centroid of the contact polygon and the objects center of mass. With minimal distance, the moment of gravitational force is reduced due to the small moment arm. Q_{jc} is the derivation from the center of the joints. Q_{align} is the misalignment between the contact normal n_c and fingertip normal n_f. A large misalignment error implies the exerted force may be outside of the friction cone.

Constraint (5.1b) is the kinematic constraint that connects the hand configuration with the contacts, where $\mathcal{F}_t$ represents the fingertip surfaces and $FK(\mathcal{F}_t; R, \boldsymbol{t}, \boldsymbol{q})$ denotes the forward kinematics parameterized by hand configuration. Constraint (5.1c) restricts the contacts to object surface ∂O, (5.1d) denotes that the hand surface $\mathcal{F}$ parameterized by $(R, \boldsymbol{t}, \boldsymbol{q})$ should not collide with the object, and (5.1e) shows the joint limits. With the palm pose, joints and contacts as optimization variables, forward kinematics (5.1b), complex object surface (5.1c) and collision (5.1d) as constraints, Problem (5.1) becomes a high-dimensional non-convex programming.

Optimization Model Structure

The structure of the proposed grasp planning algorithm is shown in Fig. 5.2. It contains two loops. the inner loop iterates the palm pose optimization (PPO) and the joint position optimization (JPO) and is called the iterative PPO-JPO. The outer loop samples hand configuration and restarts the iterative PPO-JPO. The PPO algorithm fixes joints, $\boldsymbol{q} = \boldsymbol{q}_0$, and searches for the palm pose $(R, \boldsymbol{t})$ by optimizing the grasp quality $Q_{com}(\boldsymbol{c}(R, \boldsymbol{t}, \boldsymbol{q}_0), \partial\mathcal{O}) + Q_{align}(n_c, n_f)$ while minimizing the collision with the object and the ground. The JPO algorithm fixes the palm pose $(R, \boldsymbol{t}) = (R_0, \boldsymbol{t}_0)$ while searches for the joints $\boldsymbol{q}$ by optimizing the $Q_{com}(\boldsymbol{c}(R_0, \boldsymbol{t}_0, \boldsymbol{q}), \partial\mathcal{O}) + Q_{jc}(\boldsymbol{q}) + Q_{align}(n_c, n_f)$ and minimizing the collision. The iterative PPO-JPO finds a local optimum of the optimization (5.1).

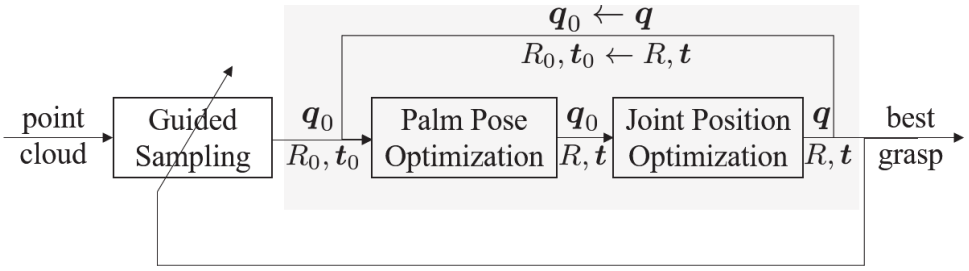


Figure 5.2: Structure of the iterative PPO-JPO.

The guided sampling in the outer loop is introduced from Chapter 2 to avoid the iterative PPO-JPO being trapped in poor-performed local optima. It employs the K-means clustering of the object surface and places the hand onto the cluster centers. The success rate of locating a high-quality collision-free grasp is different due to the varying local geometries. The guided sampling ranks different clustering centers based on the success rate and the center with higher success rate will be sampled more often. The details of the guided sampling are neglected in this chapter for simplicity.

5.3 Iterative PPO-JPO for Precision Grasp Planning

Constraint Relaxation

Optimization (5.1) is an abstract formulation of the grasp planning problem. We relax some of the constraints to locate the grasps smoothly in the space occupied by the object, including the surface constraint (5.1c) and the collision constraint (5.1d). More specifically,

$$\max_{R,\boldsymbol{t},\boldsymbol{q},\boldsymbol{c}} \; Q(\boldsymbol{c},\boldsymbol{q}) - w(E_{col}(R,\boldsymbol{t},\boldsymbol{q}) + E_{cls}(\boldsymbol{c}, FK(\mathcal{F}_t; R,\boldsymbol{t},\boldsymbol{q}))) \tag{5.2a}$$

$$s.t. \quad \boldsymbol{c} = NN_{\partial\mathcal{O}}(FK(\mathcal{F}_t; R,\boldsymbol{t},\boldsymbol{q})) \tag{5.2b}$$

$$q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1 \cdots N_f \tag{5.2c}$$

where $NN(\bullet)_{\partial\mathcal{O}}$ in (5.2b) denotes the nearest neighbor of $\bullet$ on object surface. $E_{col}(R,\boldsymbol{t},\boldsymbol{q})$ corresponds to (5.1d) and penalizes the collision violation. $E_{cls}(\boldsymbol{c}, FK(\mathcal{F}_t; R,\boldsymbol{t},\boldsymbol{q}))$ and the constraint (5.2b) together correspond to (5.1b,5.1c) and penalize the distance between the contact and the fingertip, and w is an increasing penalty weight for constraint violation.

Optimization (5.2) is a relaxed formulation of (5.1) to reduce the nonlinearities introduced by (5.1b) and (5.1d). Despite the relaxation, the direct optimization of $R \in SO(3), \boldsymbol{t}, \boldsymbol{q}$ is challenging in the following aspects: 1) the palm orientation is constrained in special orthogonal group $SO(3)$, 2) $R, \boldsymbol{t}, \boldsymbol{q}$ are searched based on the quality determined by $FK(R,\boldsymbol{t},\boldsymbol{q})$ and contacts in complex $\partial\mathcal{O}$, and 3) $\mathcal{F}_t$ is in surface form and is hard to optimize by gradient-based methods.

Incremental Search and Point Representation

To enable the gradient-based search on the object space, we instead search incrementally on $\delta R, \delta \boldsymbol{t}, \delta \boldsymbol{q}$, where $\delta R, \delta \boldsymbol{t}$ denote the *transformation* of palm and $\delta \boldsymbol{q}$ denotes the joint displacement. The problem is further simplified by sampling the *current* hand surface $\mathcal{F}$ into points $\{p_k, n_k^p\}_{k=1}^{N_p}$, where $p_k \in \mathbb{R}^3, n_k^p \in \mathbb{S}^2$ denote the k-th points and normals on hand surface pointing outwards. Similarly, the object surface $\partial\mathcal{O}$ is sampled into points $\{o_k, n_k^o\}_{k=1}^{N_o}$. We retrieve all those $\{p_k\}_{p_k \in \mathcal{F}_t}$ within $\mathcal{F}_t$ and then search the nearest neighbor on the object surface $NN_{\partial\mathcal{O}}(\{p_k\}_{p_k \in \mathcal{F}_t^i})$ to find corresponding points $\{o_k\}_{o_k \in \mathcal{I}^i}$, where $\mathcal{F}_t^i$ is the hand surface of the i-th finger. The mean values of $\{p_k\}_{p_k \in \mathcal{F}_t^i}$ and $\{o_k\}_{o_k \in \mathcal{I}^i}$ are denoted as p_{f_i} and c_i. The contact points on hand and object are $\boldsymbol{p}_f = \left[p_{f_1}, ..., p_{f_{N_c}}\right]$ and $\boldsymbol{c} = [c_1, ..., c_{N_c}]$.

With the point representation, $Q_{com}(\boldsymbol{c}, \partial\mathcal{O})$ becomes:

$$Q_{com}(\boldsymbol{c}, \partial\mathcal{O}) = -\sum_{i=1}^{N_c}((\bar{p}_{f_i} - p_{com})^T n_\perp)^2,$$

where $\bar{p}_{f_i} = \delta R p_{f_i} + \delta \boldsymbol{t} + \delta R J_{f_i}^v(q_i)\delta q_i$ is the fingertip position for finger i after transformation, $J_{f_i}^v \in \mathbb{R}^{3\times N_{jnt,i}}$ is the translational Jacobian matrix at q_i with $N_{jnt,i}$ denoting the number of joints in the i-th finger, p_{com} is the object center point, and $n_\perp$ is the normal vector of the polygon formed by fingertips. Fingertip positions are used to replace the contacts to avoid searching on object surface. This replacement is reasonable under the assumption that $\boldsymbol{c} \approx \boldsymbol{p}_f$.

The quality $Q_{jc}(\boldsymbol{q})$ becomes:

$$Q_{jc}(\boldsymbol{q}) = -\sum_{i=1}^{N_c}\sum_{j=1}^{N_{jnt,i}}(\alpha_i^j \frac{q_i^j - \bar{q}_i^j}{q_{\max,i}^j - q_{\min,i}^j})^2,$$

where $\bar{q}_i^j, q_{max,i}^j, q_{min,i}^j$ are the mean and limit values of the j-th joint in the i-th finger. α_i^j is the weights for the j-th joint of the i-th finger.

Quality $Q_{align}(n_c, n_f)$ becomes:

$$Q_{align}(n_c, n_f) = -\beta^2\sum_{i=1}^{N_c}(n_{c_i} \cdot \delta R e^{(J_i^w \delta q_i)\hat{}} n_{f_i} + 1)^2,$$

where n_{c_i}, n_{f_i} are the normals of the i-th contact and fingertips, respectively, and $J_i^w \in \mathbb{R}^{3\times N_{jnt,i}}$ is the rotational Jacobian matrix for finger i. β is the scaling factor of Q_{align}.

The formulation of collision penalty E_{col} is approximated in this chapter to accelerate the computation (Fig. 5.3), though a rigorous formulation can be found in [84]. Figure 5.3(ab) show hand-object collision. Collided points are obtained by transforming object points to bounding boxes of fingers and check the inclusion of object points in bounding boxes. Collision types are determined by the sign of $\sum n_l^p \cdot o_l$. The inner side contact and outer side

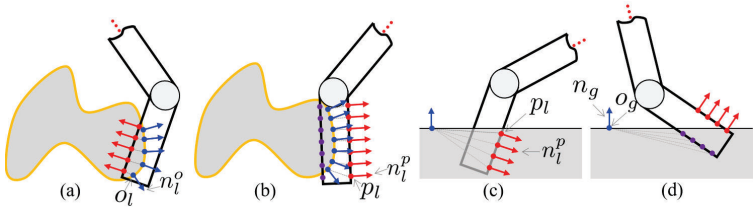


Figure 5.3: Illustration of collision detection.

contact are shown in Fig. 5.3(ab). For outer side contact, p_l is replaced by purple points. With the approximation shown in Fig. 5.3, collision penalty E_{col} becomes:

$$E_{col} = \sum_{l=1}^{N_{col}} \|\bar{p}_l - o_l\|^2 + \sum_{l=1}^{N_{col,g}} ((\bar{p}_l - o_g)^T n_g)^2,$$

where $N_{col}, N_{col,g}$ are numbers of collided points with the object and ground, respectively. $\bar{p}_l = \delta R p_l + \delta \boldsymbol{t} + \delta R \mathcal{J}_l^v(\boldsymbol{q})\delta \boldsymbol{q}$, where $\mathcal{J}_l^v \in \mathbb{R}^{3\times N_{jnt}}$ denoting the translational hand Jacobian matrix, $o_g \in \mathbb{R}^3, n_g \in \mathbb{S}^2$ are sampled points and normal of the ground.

The penalty E_{cls} is $\sum_{i=1}^{N_c}(\bar{p}_{f_i} - c_i)^T n_{c_i}$.

Problem (5.2) with the incremental search and point representation becomes:

$$\min_{\delta R, \delta \boldsymbol{t}, \delta \boldsymbol{q}} \quad -Q_{com} - Q_{jc} - Q_{align} + w(E_{col} + E_{cls}) \tag{5.3a}$$

$$s.t. \quad q_i \in [q_{\min,i}, q_{\max,i}] \quad i = 1 \cdots N_f \tag{5.3b}$$

Problem (5.3) remains a nonlinear programming due to the coupling between δR and $\boldsymbol{q}$. We solve it with the iterative PPO-JPO algorithm. The details of PPO and JPO are described below.

Palm Pose Optimization (PPO)

The PPO algorithm optimizes for $\delta R, \delta \boldsymbol{t}$ by fixing the finger joints (i.e. $\delta \boldsymbol{q} = 0$). To search on the trivial Euclidean space, the hand rotation δR is parameterized by the axis-angle representation and approximated by $\delta R \approx I_{3\times 3} + \hat{r}$, where $\hat{\bullet}$ is the matrix representation of cross product and $r \in \mathbb{R}^3$ is the angle-axis vector.

With the fixed joint and approximation of rotation, (5.3) becomes a least-squares problem:

$$\min_x \|Ax - b\|_2^2 \tag{5.4}$$

where $x = [r^T, \delta \boldsymbol{t}^T]^T \in \mathbb{R}^6$. $A = [a_{com,i}^T ... a_{align,i}^T ... a_{col,l}^T ... a_{cls,i}^T]^T \in \mathbb{R}^{(3N_c+3|\mathcal{L}_o|+|\mathcal{L}_g|)\times 6}$, with $\mathcal{L}_o$ denoting the indexes of hand-object collision and $\mathcal{L}_g$ denoting the indexes of hand-ground collision. $a_{com,i} = [(p_{f_i} \times n_\perp)^T, n_\perp^T]$, $a_{align,i} = \beta[(n_{f_i} \times n_{c_i})^T, 0_3^T]$, $a_{col,l}$ includes hand-object

collision $a_{obj,l} = w[-\hat{p}_l, I_3]$ and hand-ground collision $a_{gnd,l} = w[(p_l \times n_g)^T, (n_g)^T]$, and $a_{cls,i} = w[(p_{f_i} \times n_{c_i})^T, n_{c_i}^T]$.

Similarly, $b = [b_{com,i}...b_{align,i}...b_{col,l}...b_{cls,i}]^T$, $b_{com,i} = n_\perp^T(p_{com} - p_{f_i})$, $b_{align,i} = -\beta(n_{f_i}^T n_{c_i} + 1)$, $b_{col,l}$ includes $b_{obj,l} = w(p_l - o_l)$ and $b_{gnd,l} = w(p_l - o_g)^T n_g$, and $b_{cls,i} = w(c_i - p_{f_i})^T n_{c_i}$.

The PPO (5.4) is a least-squares problem and can be solved analytically by

$$x^* = (A^T A)^{-1} A^T b.$$

The optimal palm transformation is $\delta R = e^{(x^*_{1:3})\hat{}}$, $\delta \boldsymbol{t} = x^*_{4:6}$. We update the hand configuration by $(R, \boldsymbol{t}) \leftarrow (\delta R, \delta \boldsymbol{t}) * (R, \boldsymbol{t})$ and start JPO.

Joint Position Optimization (JPO)

The JPO algorithm optimizes for $\delta \boldsymbol{q}$ by fixing the palm pose. With the fixed palm pose, JPO becomes a least-squares with constraint problem:

$$\min_{\delta \boldsymbol{q}} \ \|C\delta \boldsymbol{q} - d\|_2^2 \tag{5.5a}$$

$$s.t. \quad \delta \boldsymbol{q} + \boldsymbol{q} \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{5.5b}$$

where $C = [C_{com}^T...C_{jc}^T...C_{align}^T...c_{col,l}^T...C_{cls}^T]^T \in \mathbb{R}^{(3N_c+N_{jnt}+3|\mathcal{L}_o|+|\mathcal{L}_g|)\times N_{jnt}}$, $C_{com} = \texttt{diag}(n_\perp^T J_i^v)$, $C_{jc} = \texttt{diag}(\alpha_i^j/(q^j_{\max,i} - q^j_{\min,i}))$, $C_{align} = \texttt{diag}(n_{c_i}^T n_{f_i}\hat{} J_i^w)$, $c_{col,l}$ includes hand-object collision $c_{obj,l} = w\mathcal{J}_l(\boldsymbol{q})$ and hand-ground collision $c_{gnd,l} = (n_g)^T \mathcal{J}_l(\boldsymbol{q})$, and $C_{cls} = w\texttt{diag}(n_{c_i}^T J_i^v)$. Similarly, $d = [d_{com,i}...d_{jc,i}...d_{align,i}...d_{col,l}...d_{cls,i}]^T \in \mathbb{R}^{3N_c+N_{jnt}+3|\mathcal{L}_o|+|\mathcal{L}_g|}$, with $d_{com,i} = (p_{com} - p_{f_i})^T n_\perp$, $d_{jc,i} = \alpha_i^j(\bar{q}_i^j - q_i^j)/(q^j_{\max,i} - q^j_{\min,i})$, $d_{align} = -(n_{c_i}^T n_{f_i} + 1)$, $d_{col,l}$ includes $d_{obj,l} = w(p_l - o_l)$ and $d_{gnd,l} = w(p_l - o_g)^T n_g$, and $d_{cls,i} = (c_i - p_{f_i})^T n_{c_i}$.

Problem (5.5) is a least-squares with box constraints and can be solved by either a solver or by initializing $\delta \boldsymbol{q}_0 = (C^T C)^{-1} C^T d$ and iterating between

$$\delta \boldsymbol{q}_{\bar{m}} = \delta \boldsymbol{q}_m - \gamma C^T (C\delta \boldsymbol{q}_m - d) \tag{5.6a}$$

$$\delta \boldsymbol{q}_{m+1} = \max(\min(\delta \boldsymbol{q}_{\bar{m}}, \boldsymbol{q}_{\max} - \boldsymbol{q}), \boldsymbol{q}_{\min} - \boldsymbol{q}) \tag{5.6b}$$

Iterative PPO-JPO Summary

The Iterative PPO-JPO algorithm is summarized in Alg. (7). The algorithm is fed with the sampled hand configuration from guided sampling and the hand/object geometries (Line 1). In each iteration, we first search the contacts by the forward kinematics and nearest neighbor (Line 4), and run the PPO algorithm to optimize for hand pose (Line 5-6). The contacts are refreshed accordingly and fed into the JPO algorithm for optimized $\delta \boldsymbol{q}^*$ (Line 7-8). The iteration terminates after $T_{\max}$ iterations.

Algorithm 7 Iterative PPO-JPO Algorithm

Input: Initial $R_s, \boldsymbol{t}_s, \delta\boldsymbol{q}_s$, $\partial\mathcal{O}$, $\mathcal{F}$, $T_{\max}$
Init: $(R, \boldsymbol{t}, \boldsymbol{q}) \leftarrow (R_s, \boldsymbol{t}_s, \delta\boldsymbol{q}_s)$
for $t = 0, \cdots, T_{\max}$ **do**
  $(\boldsymbol{c}, \boldsymbol{p}_f) \leftarrow \texttt{update}(FK(\mathcal{F}, R, \boldsymbol{t}, \boldsymbol{q}), \partial\mathcal{O})$
  $\delta R^*, \delta\boldsymbol{t}^* \leftarrow \texttt{PPO}(\boldsymbol{c}, \boldsymbol{p}_f)$
  $(R, \boldsymbol{t}) \leftarrow (\delta R^*, \delta\boldsymbol{t}^*) * (R, \boldsymbol{t})$
  $(\boldsymbol{c}, \boldsymbol{p}_f) \leftarrow \texttt{update}(FK(\mathcal{F}, R, \boldsymbol{t}, \boldsymbol{q}), \partial\mathcal{O})$
  $\delta\boldsymbol{q}^* \leftarrow \texttt{JPO}(\boldsymbol{c}, \boldsymbol{p}_f)$
  $\boldsymbol{q} \leftarrow \boldsymbol{q} + \delta\boldsymbol{q}^*$
end for
return $\{R, \boldsymbol{t}, \boldsymbol{q}\}$

5.4 Simulations and Experiments

This section shows the simulation and experiment results. The simulation ran on a desktop with 32GB RAM and 4.0GHz CPU. The computation was conducted in Matlab and visualized in VREP. For the experiment, we used a BarrettHand BH8-282 multi-fingered hand attached to a FANUC LRMate 200iD/7L industrial manipulator for grasping. Two Ensenso N35 cameras were used to capture the point cloud of the scene.

Parameter Lists

The hand surface was discretized into 1798 points and each fingertip had 216 points sampled. The scaling factor $\alpha_{1:2}^{1:3} = 1$ and $\alpha_3^{1:2} = \sqrt{2}$ in Q_{jc} to balance the gradients in different sides of the hand. The scaling factor $\beta = 0.03$ in Q_{align}. The collision penalty w was initialized as 1.0 and increased exponentially with factor 1.1. The maximum iteration $T_{\max} = 40$.

Simulation Results

Figure 5.4 shows 5 out of 7 collision-free grasps found with 10 samples in simulation using the proposed iterative PPO-JPO method. The object to be grasped was a *robot* model with a complex shape. The proposed algorithm was able to find the versatile grasps without colliding with the object and the ground. The majority of the grasps found were precision grasps with fingertip contacts.

Figure 5.5 shows the animation of a grasp search on the *robot* object. The iterative PPO-JPO started searching from a vertical grasp with fully opened hand (Fig. 5.5(a)) and gradually adjusted the palm pose/joints angles to maximize the quality and avoid the collision, as shown in Fig. 5.5(b-h).

The grasp results on 12 different objects are shown in Figure 5.6. The iterative PPO-JPO was able to find collision-free precision grasps for a) thin objects close to the ground

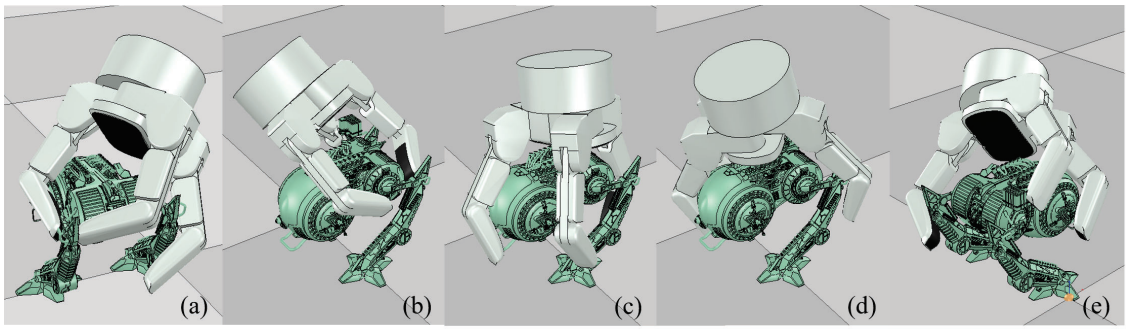


Figure 5.4: Visualization of 5 out of 7 grasps found on Robot object.

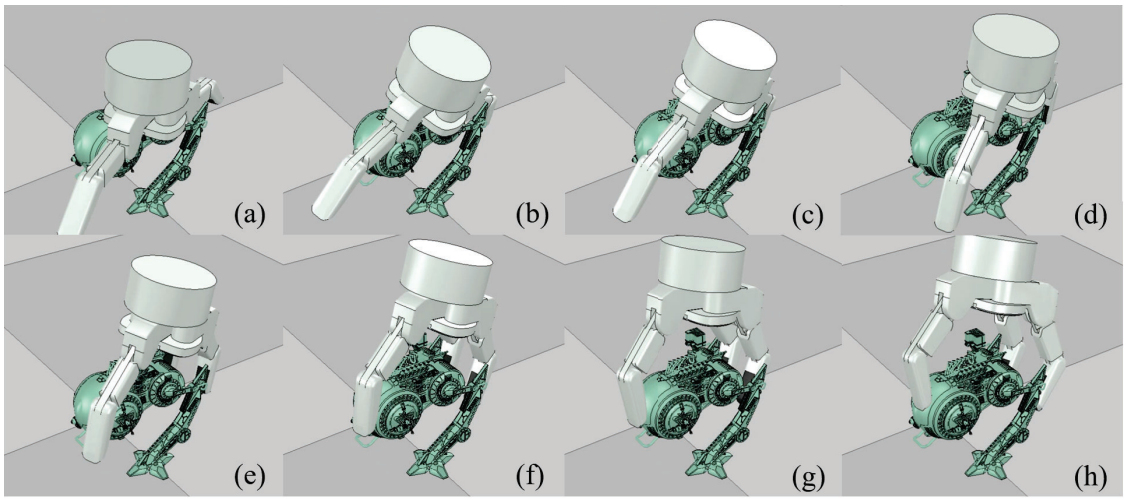


Figure 5.5: Simulation result of the iterative PPO-JPO on Robot object.

(Fig. 5.6(1,5)), b) objects with complex surfaces (Fig. 5.6(2,6,9,12)), or c) the objects with sharp edges (Fig. 5.6(3,7,12)).

The numerical results of the iterative PPO-JPO on these objects are shown in Table 5.1. Among the 12 different objects, the *iPhone X*, *Gun*, *Hand* and *Mouse* are among the most challenging objects to grasp. The proposed algorithm was not sensitive to object complex surfaces but rather suffered from the low-height properties of the objects due to the collisions caused by the narrow space between the feasible grasp region and the ground.

In average, the iterative PPO-JPO were able to find 6.58 collision-free grasps out of 10 samples within 3.26 secs (0.496 sec/grasp).

Figure 5.7 shows the error reduction profile running the iterative PPO-JPO algirthm on Bunny object. The algorithm ran for 50 trials, and generated 46 collision-free grasps. We recorded the negative quality $E_{quality} = -Q_{com} - Q_{jc} - Q_{alig}$ and the penalty error $E_{penalty} = E_{col} + E_{cls}$. In average, the grasp quality $E_{quality}$ reduced from 1.36 ± 0.0036 m to

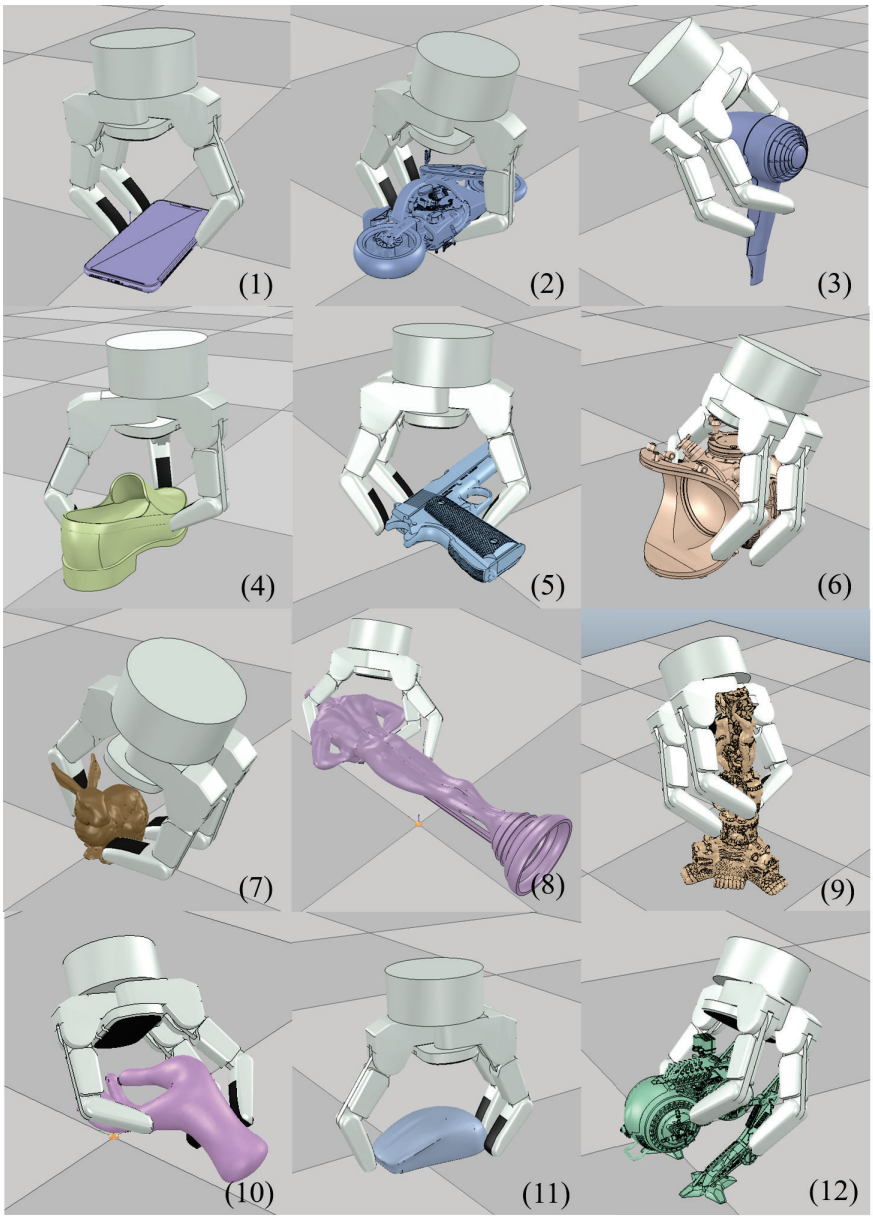


Figure 5.6: Simulation results of iterative PPO-JPO on 12 different objects.

0.29 ± 0.093 m (Fig. 5.7(Middle)), and the penalty error $E_{penalty}$ reduced from 0.31 ± 0.27 m to 0.031 ± 0.0050 m (Fig. 5.7(Right)). The red (purple) and blue (yellow) plots show the mean and standard deviation for all (collision-free) grasps. The total error $E_{overall} = E_{quality} + E_{penalty}$ Fig. 5.7(Left).

Table 5.1: Numerical Results of the iterative PPO-JPO

ID	Object	$\frac{\text{collision-free}\#}{\text{total samples}}$	Time (s)
1	iPhone X	5/10	2.48
2	Motorbike	6/10	3.72
3	Hairdryer	10/10	3.44
4	Shoe	6/10	3.45
5	Gun	4/10	2.89
6	Diving helmet	9/10	3.85
7	Bunny	8/10	3.15
8	Oscar	8/10	3.60
9	Catcam	7/10	3.67
10	Hand	5/10	3.15
11	Mouse	4/10	2.48
12	Robot	7/10	3.27
1-12	Average	6.58/10	3.263

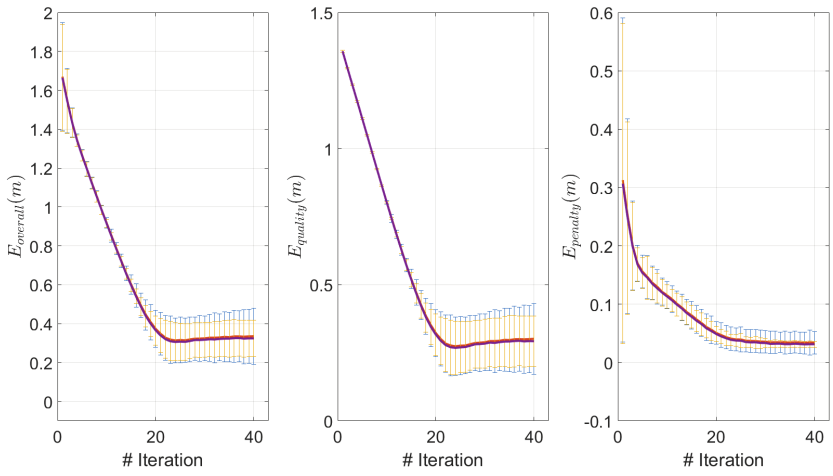


Figure 5.7: Error profiles of iterative PPO-JPO on Bunny object running 50 samples.

Experiment Results

This section shows the experimental results with the BarrettHand BH8-282. The whole experiment could be separated into two phases: 1) the grasp planning by iterative PPO-JPO, and 2) the object clamping by simply continuing finger motion for certain time until the tactile sensor reading reaching the target value ($1.0N/cm^2$). The focus of this chapter is the grasp planning using the proposed iterative PPO-JPO algorithm.

Figure 5.8 shows the experimental setup (Fig. 5.8(1)), the grasp planning and execution results (Fig. 5.8(2-18)) on 15 different objects. The perceived point cloud and the located grasp are shown on the left side of each subfigure. The physical grasp pose and the execution result of the planned grasp are shown in the middle and right, respectively.

To address the noise and incompleteness of the perceived point cloud, the contact points were regarded as a region close to the fingertip, instead of a single point. Therefore, the system exhibits certain robustness to the noise and incompleteness of the object point cloud, as shown in Fig. 5.8(2-14).

While the algorithm shows certain robustness to noise and incompleteness of the point cloud, the resistance to uncertainties remains a challenge. These uncertainties were 1) positioning uncertainties including calibration error ($\sim$ 3 mm for robot-camera frame alignment), installation error ($\sim 1^\circ$ TCP-palm alignment), actuation error ($\sim 2.0^\circ$ finger joint tracking error), 2) communication uncertainties including synchronization error ($\sim$ 0.1 sec ROS-Matlab transmission misalignment), non-real time error ($\sim$ 0.1 sec ROS latency on different fingers), and dynamics uncertainties including the mass uncertainties, friction uncertainties and softness uncertainties.

Figure 5.8(15-18) shows four failure cases caused by these uncertainties. More specifically, Figure 5.8(15) was failed from the unsynchronized contacts of different fingers. The finger which contacted with the object first would keep pushing the object, introducing extra disturbance and perturbing the object. Consequently, the hand contacted with the object on undesired positions and caused the object slipped. Figure 5.8(16) slipped off since the object was heavy and two fingers contacted with soft parts of the object (plug and wire). However, the quality metric cannot take these factors into account. Similar failure appears in Fig. 5.8(17,18).

5.5 Chapter Summary

This chapter proposed an efficient optimization model for precision grasp planning. To optimize the quality and avoid the collision, the planning problem was formulated into an optimization with penalties and solved by iterating between the palm pose optimization (PPO) and joint position optimization (JPO). The iterative PPO-JPO algorithm was able to locate a collision-free grasp within 0.50 sec (based on 120 grasps on 12 objects in different categories). Experiments on a BarrettHand BH8-282 further demonstrated the effectiveness of the algorithm. The experimental videos are available at [102].

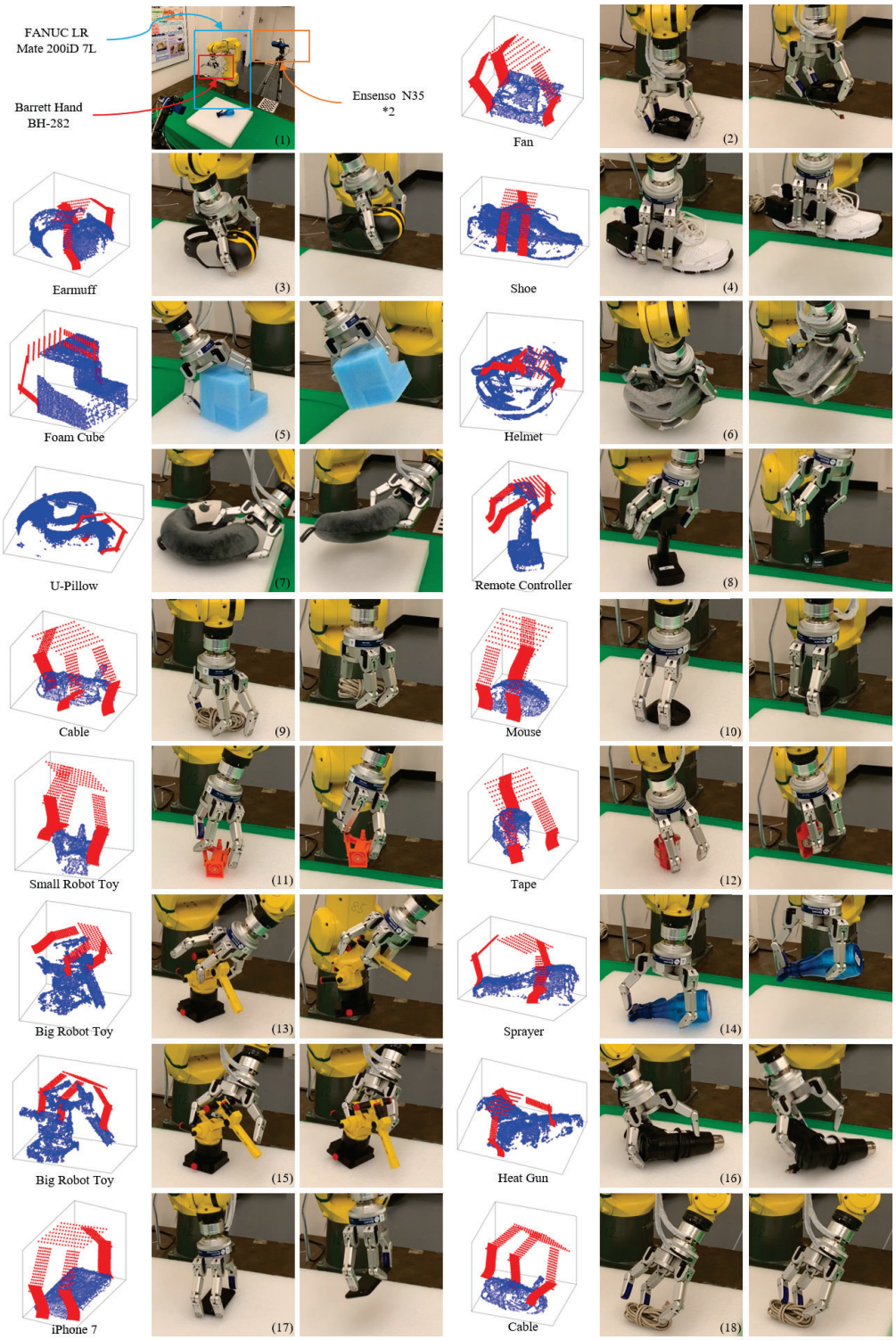


Figure 5.8: (1) Experimental setup and (2-18) planning and execution results on 15 objects.

Chapter 6

Efficient Framework for General Robotic Grasping

6.1 Introduction

Chapter 5 introduced an optimization model to plan precision grasps for multi-fingered hands. In this chapter, we further improve the versatility of the produced grasps and the robustness of the optimization model to imperfect sensing, calibration and actuation. In Chapter 2, the robustness to sensing and actuation uncertainties is improved by fitting the contact surface between the customized grippers and workpieces to generate powerful, robust grasps. The surface fitting produces grasps with larger contact surfaces and provides force/torque from various positions, thus the resultant wrench space can resist larger disturbances. However, the previous surface fitting has several limitations. First, it can only handle the grippers with one degree of freedom (DOF). Secondly, the grasps with collisions were detected and pruned after the optimization. The optimize-then-prune operation produced sub-optimal grasps.

Besides planning the desired grasp configurations, the robot should generate proper robot-finger trajectories to execute the grasps. The trajectory planning can be extremely expensive in this high dimensional space. Majority of the current grasp planning methods ignore possible collisions during the execution and simply close the fingers to execute the grasps [99, 83, 88, 86]. A method based on rapidly-exploring random tree (RRT) was introduced in [98] to plan motion and grasp simultaneously. With the manually designed heuristics, the RRT dimension was reduced to three. These heuristics defined a potentially narrow and suboptimal subspace for RRT search. A general trajectory optimization (TrajOpt) algorithm was presented in [84] using the sequential quadratic programming (SQP) [7]. Both the RRT and TrajOpt require object mesh model during the optimization, which is generally absent in the online grasp planning scenario.

In this chapter, we combine the surface fitting in Chapter 2 and the optimization model in Chapter 5, and propose a general framework for efficient robot grasping with different

types of grippers. The framework includes both the grasp planning and grasp imagination. With the consideration of surface fitting, the grasp planning searches for optimal grasps by deforming the hand surfaces along its feasible kinematic directions and matching towards the surface of the workpieces, given the assumption that the large matching area produces more stable and powerful grasp. The grasp planning is able to handle the hands with multi-DOFs. With the planned grasp configurations, we further propose a grasp imagination method to optimize the robot-finger trajectories to reach the target grasps given the point cloud representation of the objects.

The contributions of this chapter are as follows. First, the grasp planning is able to find grasps with plausible surface fitting performance efficiently. The average optimization time is 0.40 sec/grasp using the raw point cloud captured by stereo cameras. By optimizing the palm pose and finger joints iteratively, the planning algorithm is able to implement on the hands with multiple DOFs. Secondly, the collision is penalized by the gradient-based methods directly, instead of being pruned after the optimization as [20, 38]. Furthermore, the proposed method can generate both the power grasps and precision grasps by adjusting the fitting weights of fingertips. Finally, the proposed grasp imagination is able to plan collision-free finger trajectories in 0.61 sec/grasp with the imperfect point cloud and underlying uncertainties.

The remainder of the chapter is as follows. Section 6.2 describes the problem formulation, followed by the proposed grasping framework in Section 6.3. The experimental results on a multi-fingered hand are introduced in Section 6.4. Section 6.5 concludes the chapter and describes the future work. The experimental videos are available at [102].

6.2 General Optimization Model for Grasping

With the surface contact, the grasp planning for a multi-fingered hand problem can be formulated as:

$$\max_{R,\boldsymbol{t},\delta\boldsymbol{q},\boldsymbol{\mathcal{S}^f},\boldsymbol{\mathcal{S}^o}} \quad Q(\boldsymbol{\mathcal{S}^f},\boldsymbol{\mathcal{S}^o}) \tag{6.1a}$$

$$s.t. \quad \boldsymbol{\mathcal{S}^f} \subset \mathcal{T}(\boldsymbol{\partial\mathcal{F}}; R, \boldsymbol{t}, \delta\boldsymbol{q}), \tag{6.1b}$$

$$\boldsymbol{\mathcal{S}^o} = NN_{\partial\mathcal{O}}(\boldsymbol{\mathcal{S}^f}), \tag{6.1c}$$

$$dist(\mathcal{T}(\boldsymbol{\partial\mathcal{F}}; R, \boldsymbol{t}, \delta\boldsymbol{q}), \partial\mathcal{O}|\mathcal{G}) \geq 0 \tag{6.1d}$$

$$\boldsymbol{q}_0 + \delta\boldsymbol{q} \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{6.1e}$$

where $R \in SO(3), \boldsymbol{t} \in \mathbb{R}^3$ denote the rotation and translation of the hand palm, $\boldsymbol{q} \in \mathbb{R}^{N_{jnt}}$ denotes the joint angle, with N_{jnt} representing the number of joints, and $\boldsymbol{q}_0$ and $\delta\boldsymbol{q}$ represent the original and displacement of $\boldsymbol{q}$. $\boldsymbol{\mathcal{S}^f} = [\mathcal{S}_1^f, ..., \mathcal{S}_{N_{cnt}}^f], \boldsymbol{\mathcal{S}^o} = [\mathcal{S}_1^o, ..., \mathcal{S}_{N_{cnt}}^o]$ are contact surfaces for all fingers/palms and objects, with $\mathcal{S}_i^f$ and $\mathcal{S}_i^o$ representing the i-th contact surface on the finger/palm and object, and N_{cnt} denoting the number of contact surfaces. $Q \in \mathbb{R}$ represents the grasp quality related to $\boldsymbol{\mathcal{S}^f}, \boldsymbol{\mathcal{S}^o}$. Constraints (6.1b) shows that $\boldsymbol{\mathcal{S}^f}$ is a

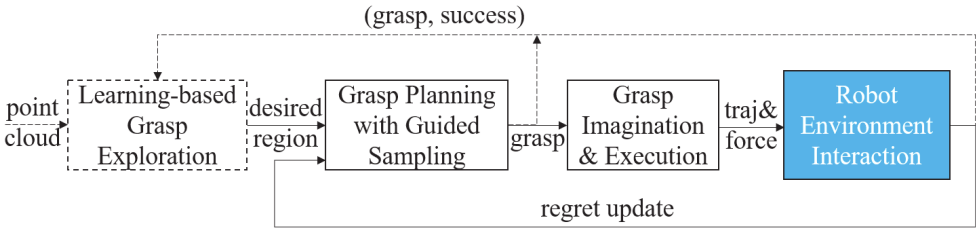


Figure 6.1: Illustration of the general grasping framework.

subset of the surface transformed from the hand surface $\boldsymbol{\partial\mathcal{F}}$ by $(R, \boldsymbol{t}, \delta\boldsymbol{q})$. Constraint (6.1c) denotes $\boldsymbol{\mathcal{S}^o}$ is computed from the nearest neighbor (NN) of $\boldsymbol{\mathcal{S}^f}$ on object surface $\partial\mathcal{O}$. Constraint (6.1d) denotes that the transformed hand surface $\boldsymbol{\partial\mathcal{F}}$ should not collide with the object $\partial\mathcal{O}$ and ground $\mathcal{G}$, and (6.1e) indicates that $\boldsymbol{q}$ stays in $[\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}]$.

Problem (6.1) would be a standard grasp planning problem if all contact surfaces were degenerated into contact points. In general case, however, the problem is challenging to solve by either the sampling based methods or gradient based methods considering the inverse kinematics (IK) and collision detection with the objects of complex shapes.

We observe that human tends to match the contact surfaces during grasping in order to increase the force exerted on the object and improve the robustness to uncertainties. Therefore, the grasp quality Q is chosen as the surface fitting error between the hand contact surface $\boldsymbol{\mathcal{S}^f}$ and object contact surface $\boldsymbol{\mathcal{S}^o}$. More concretely,

$$Q(\boldsymbol{\mathcal{S}^f}, \boldsymbol{\mathcal{S}^o}) = -dist(\boldsymbol{\mathcal{S}^f}, \boldsymbol{\mathcal{S}^o}). \tag{6.2}$$

Based on this quality formulation, this chapter introduces a framework to plan and execute the grasps. The technical details are explained in the following sections.

6.3 Grasp Planning and Imagination by MDISF-GTO

Framework Architecture

Figure 6.1 introduces the architecture of the grasping framework. It consists of three main blocks: grasp planning by a multi-dimensional iterative surface fitting (MDISF), the grasp imagination and execution by a grasp trajectory optimization (GTO), and an optional learning-based grasp explorer by a region-based convolutional neural network (R-CNN).

Starting with an initial configuration, MDISF optimizes the palm transformation $(R, \boldsymbol{t})$ and joint displacements $\delta\boldsymbol{q}$ by minimizing the surface fitting error between the hand and object. The optimization actively avoids the collision between the hand and the object as well as the surrounding environment, and is able to deform in feasible hand directions. Compared with the ISF algorithm in Chapter 2, MDISF is collision-aware and can plan grasps for the hands with multiple DOFs. Compared with the optimization model in Chapter 5,

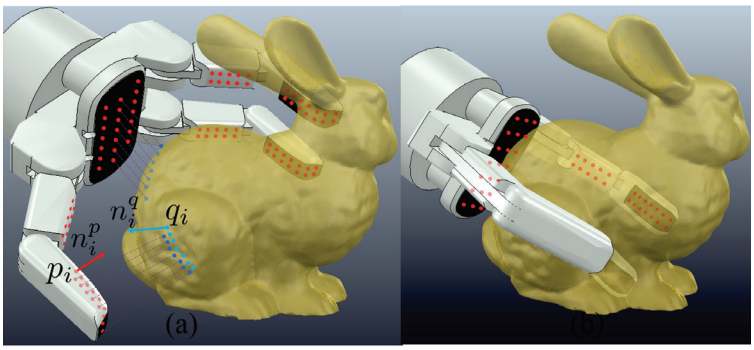


Figure 6.2: Illustration of the multi-dimensional iterative surface fitting (MDISF) algorithm.

MDISF in this chapter increase the contact surface to improve the robustness to sensing and positioning uncertainties. Guided sampling is introduced to avoid trapped in bad-performed local optima by prioritizing different initial configurations, so that the regions with better performance are sampled more often. The introduction of guided sampling is in Chapter 2 and the detailed introduction is neglected in this chapter.

Grasp imagination is to evaluate the planned grasps robustly and generate robot-finger trajectories to approach the highly ranked grasps. The GTO algorithm is proposed to avoid collision with the environment and plan optimal finger trajectories using the incomplete point cloud under various types of uncertainties.

With the capacity to grasp objects in clutter environments, we also introduce a learning-based grasp explorer to accelerate the initialization of grasp planning. The grasp explorer learns from previous grasp experience on the grasp affordance and collision avoidance. The technical details have been introduced in Chapter 3 and are ignored here for brevity.

Multi-Dimensional Iterative Surface Fitting

With the surface fitting score (6.2) as the quality, Problem (6.1) can be solved with the proposed multi-dimensional iterative surface fitting (MDISF) algorithm. Similar to iterative closest point (ICP) [5], MDISF iterates between the correspondence matching (Fig. 6.2(a)) and surface fitting (Fig. 6.2(b)). To employ the gradient in improving the searching efficiency, the hand surface $\boldsymbol{\partial\mathcal{F}}$ is discretized into points $\{p_i, n_i^p\}_{i=1}^{N_p}$ and bounding boxes $\{\mathcal{B}_k\}_{k=1}^{N_b}$, where $p_i \in \mathbb{R}^3, n_i^p \in \mathbb{S}^2$ represent the point position and normal vector pointing outward, and N_p, N_b are the total number of hand surface points and boxes to cover the surface, as shown in Fig. 6.2(a). Only the front surface is sampled for simplification. Similarly, the object surface $\partial\mathcal{O}$ is discretized into points $\{q_i, n_i^q\}_{i=1}^{N_q}$, where $q_i \in \mathbb{R}^3, n_i^q \in \mathbb{S}^2$ represent the point position and normal vector pointing outward, and N_q is the total number of points on the object point cloud.

The correspondence matching finds the paired points $\{q_i \in \mathbb{R}^3, n_i^q \in \mathbb{S}^2\}_{i\in\mathcal{I}}$ on object

point cloud by the nearest neighbor search with duplicate/outlier removal [108]. The surface fitting minimizes the distance between the point pairs $\{p_i, n_i^p\}_{i\in\mathcal{I}}$ and $\{q_i, n_i^q\}_{i\in\mathcal{I}}$, as shown in Fig. 6.2(b). With the point representation of the surfaces, the surface fitting error E_{fit} is re-formulated as

$$E_{fit}(R, \boldsymbol{t}, \delta\boldsymbol{q}) = \sum_{i\in\mathcal{I}} \left((\bar{p}_i - q_i)^T n_i^q\right)^2 + \alpha^2((Re^{(\mathcal{J}_i^w \delta\boldsymbol{q})\hat{}} n_i^p) \cdot n_i^q + 1)^2 \tag{6.3}$$

where $\bar{p}_i = Rp_i + \boldsymbol{t} + R\mathcal{J}_i^v(\boldsymbol{q})\delta\boldsymbol{q}$ describes the hand surface point after the palm transformation and finger displacement, and $\mathcal{J}_i^v(\boldsymbol{q}), \mathcal{J}_i^w(\boldsymbol{q})$ are translational and rotational Jacobian matrices at the point p_i with the joint $\boldsymbol{q}$. The first term describes the point distance projected to the object surface normal direction. This point-to-plane distance is broadly used in ICP [82] to allow sliding on flat surface, so that the algorithm is not sensitive to incomplete point cloud. The second term describes the alignment of the normal vectors. α is to balance the scale of normal alignment.

With the current correspondence matching and fitting error representation in (6.3), Problem (6.1) becomes:

$$\min_{R,\boldsymbol{t},\delta\boldsymbol{q}} \; E_{fit}(R, \boldsymbol{t}, \delta\boldsymbol{q}) \tag{6.4a}$$

$$s.t. \quad dist\left(\mathcal{T}(\{\mathcal{B}_k\}_{k=1}^{N_b}; R, \boldsymbol{t}, \delta\boldsymbol{q}), \partial\mathcal{O}\right) \geq 0, \tag{6.4b}$$

$$dist\left(\mathcal{T}(\boldsymbol{\partial\mathcal{F}}; R, \boldsymbol{t}, \delta\boldsymbol{q}), \mathcal{G}\right) \geq 0, \tag{6.4c}$$

$$\delta\boldsymbol{q} + \boldsymbol{q}_0 \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{6.4d}$$

where (6.4b) represents the collision between the object $\partial\mathcal{O}$ and the bounding boxes of the hand $\{\mathcal{B}_k\}_{k=1}^{N_b}$, and (6.4c) denotes the collision between the hand surface and the ground. Equation (6.4) is a non-convex programming due to the coupling term $R\mathcal{J}_i^v(\boldsymbol{q})\delta\boldsymbol{q}$ in (6.4a) and the collision constraints (6.4b, 6.4c).

Collision Handling

The collision avoidance has been introduced in Chapter 5, we restate here for clarity. To address the collision term, we employ the point representation of the object $\{q_i, n_i^q\}_{i=1}^{N_q}$ and check the inclusion of the points in $\{\mathcal{B}_k\}_{k=1}^{N_b}$. As for the hand-ground collision, we represent the ground by a point on ground q_g and normal vector n_g, and check the ground collision by the sign of $(p_i - q_g)^T n_g$. Penalty method [60] is introduced to avoid collision and ensure that the hand can move smoothly in the space occupied by the object. More concretely, the collision error is formulated as:

$$E_{col}(R, \boldsymbol{t}, \delta\boldsymbol{q}) = \sum_{l\in\mathcal{L}_o} \|\bar{p}_l - q_l\|_2^2 + \sum_{l\in\mathcal{L}_g} \left((\bar{p}_l - q_g)^T n_g\right)^2 \tag{6.5}$$

where $\{q_l\}_{l\in\mathcal{L}_o}$ denotes the object points that are in collision with the bounding boxes. $\{p_l\}_{l\in\mathcal{L}_o}$ denotes the corresponding points on the box front or back surfaces, and $\bar{p}_l = Rp_l +$

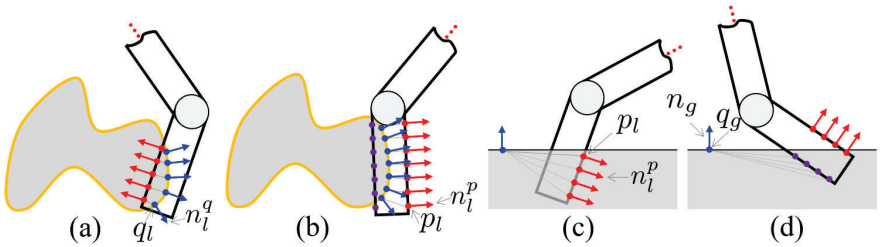


Figure 6.3: Illustration of different collision types.

$\boldsymbol{t} + R\mathcal{J}_l^v(\boldsymbol{q})\delta\boldsymbol{q}$. To ensure that (6.5) reduces all types of collisions, we choose the front or the back surfaces that q_l paired with, as shown in Fig. 6.3. Figure 6.3(ac) show the inner side collision while Fig. 6.3(bd) show the outer side collision. The algorithm first searches the correspondence pair $\{p_l, q_l\}_{l\in\mathcal{L}}$, as shown by red and blue points. The sign of $\sum_{l\in\mathcal{L}} n_l^p \cdot n_l^q$ is used for detect the collision type, and $\sum_{l\in\mathcal{L}} n_l^p \cdot n_l^q \leq 0$ means inner side contact. For the outer side contact, p_l is replaced by the points on the back, as shown by purple dots in Fig. 6.3(bd).

With the penalty method, the surface fitting (6.4) becomes

$$\min_{R,\boldsymbol{t},\delta\boldsymbol{q}} \; E(R,\boldsymbol{t},\delta\boldsymbol{q}) \tag{6.6a}$$

$$s.t. \quad \delta\boldsymbol{q} + \boldsymbol{q}_0 \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{6.6b}$$

where $E(R,\boldsymbol{t},\delta\boldsymbol{q}) = E_{fit}(R,\boldsymbol{t},\delta\boldsymbol{q}) + w^2 E_{col}(R,\boldsymbol{t},\delta\boldsymbol{q})$ represents the overall error during the surface fitting of the current correspondence. w denotes the penalty weight of the collision.

Iterative Palm Finger Optimization (IPFO)

Problem (6.6) is a discretization of (6.1) under the current correspondence and collision penalty, and is solved by the iterative palm finger optimization (IPFO) revised from [20]. The IPFO algorithm iteratively optimizes the palm transformation $(R,\boldsymbol{t})$ and the finger displacements $\delta\boldsymbol{q}$.

Palm Optimization The palm optimization searches for optimal $(R,\boldsymbol{t})$ by fixing the finger joint configuration:

$$\min_{R,\boldsymbol{t}} E(R,\boldsymbol{t},0) = \min_{x} \|Ax - b\|^2 \tag{6.7}$$

where $x = [r^T, \boldsymbol{t}^T]^T \in \mathbb{R}^6$ is a local parameterization of the palm transformation, and $r \in \mathbb{R}^3$ is the axis-angle vector to approximate R in small rotation angle assumption, i.e. $R \approx I + \hat{r}$, where $\hat{\bullet}$ is a skew-symmetric representation of cross product. The matrix $A = [a_{p,i}^T ... a_{n,i}^T ... a_{col,l}^T]^T \in \mathbb{R}^{(2|\mathcal{I}|+3|\mathcal{L}_o|+|\mathcal{L}_g|)\times 6}$, with $a_{p,i} = [(p_i \times n_i^q)^T, (n_i^q)^T]$ as the point-to-plane fitting error, and $a_{n,i} = \alpha[(n_i^p \times n_i^q)^T, 0_3^T]$ as the normal alignment error. $a_{col,l}$ includes

hand-object collision $a_{obj,l} = w[-\hat{p}_l, I_3]$ and hand-ground collision $a_{gnd,l} = w[(p_l \times n_g)^T, (n_g)^T]$. Similarly, $b = -[b_{p,i}^T ... b_{n,i}^T ... b_{col,l}^T]^T \in \mathbb{R}^{2|\mathcal{I}|+3|\mathcal{L}_o|+|\mathcal{L}_g|}$, with $b_{p,i} = (p_i - q_i)^T n_i^q$ and $b_{n,i} = \alpha((n_i^p)^T n_i^q + 1)$. $b_{col,l}$ includes $b_{obj,l} = w(p_l - q_l)$ and $b_{gnd,l} = w(p_l - q_g)^T n_g$.

Equation (6.7) is a least squares problem and is solved analytically by:

$$x^* = (A^T A)^{-1} A^T b \tag{6.8}$$

Finger Optimization The finger optimization fixes the palm transformation $(R^*, \boldsymbol{t}^*)$ and searches for optimal finger displacements $\delta \boldsymbol{q}$:

$$\min_{\delta \boldsymbol{q}} \; E(R^*, \boldsymbol{t}^*, \delta \boldsymbol{q}) = \min_{\delta \boldsymbol{q}} \|C\delta \boldsymbol{q} - d\|^2 \tag{6.9a}$$

$$s.t. \quad \delta \boldsymbol{q} + \boldsymbol{q} \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \tag{6.9b}$$

where $C = [c_{p,i}^T ... c_{n,i}^T ... c_{col,l}^T]^T \in \mathbb{R}^{(2|\mathcal{I}|+3|\mathcal{L}_o|+|\mathcal{L}_g|) \times N_{jnt}}$, with $c_{p,i} = (n_i^q)^T \mathcal{J}_i^v(\boldsymbol{q})$ as the point-to-plane fitting error, $c_{n,i} = \alpha(n_i^p \times n_i^q)^T \mathcal{J}_i^w$ as the normal alignment error, $c_{col,l}$ includes hand-object collision $c_{obj,l} = wR^* \mathcal{J}_l^v(\boldsymbol{q})$ and hand-ground collision $c_{gnd,l} = (n_g)^T \mathcal{J}_l^v(\boldsymbol{q})$. Similarly, $d = -[d_{p,i}^T ... d_{n,i}^T ... d_{col,l}^T]^T \in \mathbb{R}^{2|\mathcal{I}|+3|\mathcal{L}_o|+|\mathcal{L}_g|}$, with $d_{p,i} = R^* p_i + \boldsymbol{t}^* - q_i$ and $d_{n,i} = \alpha(n_i^p \cdot n_i^q + 1)$. $d_{col,l}$ includes $d_{obj,l} = w(R^* p_l + \boldsymbol{t}^* - q_l)$ and $d_{gnd,l} = R^* p_l + \boldsymbol{t}^* - q_g$. Equation (6.9) is a least-squares with box constraints, and is solved by initializing $\delta \boldsymbol{q}_0 = (C^T C)^{-1} C^T d$ and iterating between

$$\delta \boldsymbol{q}_{\bar{m}} = \delta \boldsymbol{q}_m - \gamma C^T (C \delta \boldsymbol{q}_m - d) \tag{6.10a}$$

$$\delta \boldsymbol{q}_{m+1} = \max(\min(\delta \boldsymbol{q}_{\bar{m}}, \boldsymbol{q}_{\max} - \boldsymbol{q}), \boldsymbol{q}_{\min} - \boldsymbol{q}) \tag{6.10b}$$

until converge, where γ is the step size for gradient decent and is set as $0.1 N_{jnt} / trace(C^T C)$. Equation (6.10) is able to converge around $10 \sim 50$ iterations[1].

IPFO is summarized in Alg. (8). The Alg. (8) feeds as inputs $\boldsymbol{\partial \mathcal{F}}$ represented by $\{p_i, n_i^p\}_{i=1}^{N_p}$, $\partial \mathcal{O}$ represented by $\{q_i, n_i^q\}_{i=1}^{N_q}$, the surface fitting indices $\mathcal{I}$ and collision avoidance indices $\mathcal{L}$. The corresponding points for fitting and collision are then sampled in Line (4-5). The palm optimization and finger optimization are shown in Line (6-7). The hand surface and hand configuration are updated in Line (8-9). IPFO terminates once the error reduction is less than threshold Δ, as shown in Line (10-13). IPFO returns the optimal transformation $R, \boldsymbol{t}$, finger displacement $\delta \boldsymbol{q}$ and the updated hand surface $\boldsymbol{\partial \mathcal{F}}$.

Theorem 1 *The IPFO algorithm in Alg. (8) converges to a local optimum of Problem (6.6).*

proof: *The convergence of IPFO is proved based on the global convergence theorem [60]. First, The $R, \boldsymbol{t}, \delta \boldsymbol{q} \in D = SE(3) \times \mathbb{R}^{N_{jnt}}$ is a compact set. Second, the function $E(R, \boldsymbol{t}, \delta \boldsymbol{q})$ in (6.6) is a continuous function. With the construction of IPFO in Line (6-7) of Alg. (8), we claim that the function $E(R, \boldsymbol{t}, \delta \boldsymbol{q})$ is decent since $E(R^*, \boldsymbol{t}^*, \delta \boldsymbol{q}^*) < E(R, \boldsymbol{t}, \delta \boldsymbol{q})$ when outside*

[1] Due to the simple form of the constraints, the iteration is more efficient than calling a general constrained least squares solver.

Algorithm 8 Iterative Palm-Finger Optimization (IPFO)

1: **Input:** $\boldsymbol{\partial\mathcal{F}}, \partial\mathcal{O}, \mathcal{I}, \mathcal{L}$
2: **Init:** $(R, \boldsymbol{t}) \leftarrow (I, 0), \delta\boldsymbol{q} = 0, e_{prev} = \infty$
3: **for** $t = 0 : T_{max}$ **do**
4: $\quad \{p_i, n_i^p, \mathcal{J}_i^v\}_{i\in\mathcal{I}}, \{q_i, n_i^q\}_{i\in\mathcal{I}} \leftarrow sample(\boldsymbol{\partial\mathcal{F}}, \partial\mathcal{O}, \mathcal{I})$
5: $\quad \{p_l, q_l\}_{l\in\mathcal{L}} \leftarrow sample\_collision(\boldsymbol{\partial\mathcal{F}}, \partial\mathcal{O}, \mathcal{L})$
6: $\quad \{R^*, \boldsymbol{t}^*\} \leftarrow \min_{R,\boldsymbol{t}} E(R, \boldsymbol{t}, 0)$ by (6.8)
7: $\quad (\delta\boldsymbol{q}^*, e) \leftarrow \min_{\delta\boldsymbol{q}} E(R^*, \boldsymbol{t}^*, \delta\boldsymbol{q})$ by (6.10)
8: $\quad \boldsymbol{\partial\mathcal{F}} \leftarrow \mathcal{T}(\boldsymbol{\partial\mathcal{F}}; R^*, \boldsymbol{t}^*, \delta\boldsymbol{q}^*)$
9: $\quad \delta\boldsymbol{q} \leftarrow \delta\boldsymbol{q} + \delta\boldsymbol{q}^*, \quad (R, \boldsymbol{t}) \leftarrow (R^*, \boldsymbol{t}^*) * (R, \boldsymbol{t})$
10: $\quad$ **if** $e_{prev} - e < \Delta$ **then**
11: $\quad\quad (R, \boldsymbol{t}, \delta\boldsymbol{q}, e) \leftarrow (R_{prev}, \boldsymbol{t}_{prev}, \delta\boldsymbol{q}_{prev}, e_{prev})$
12: $\quad\quad$ **break**
13: $\quad$ **end if**
14: $\quad (R_{prev}, \boldsymbol{t}_{prev}, \delta\boldsymbol{q}_{prev}, e_{prev}) \leftarrow (R, \boldsymbol{t}, \delta\boldsymbol{q}, e)$
15: **end for**
16: **return** $\{R, \boldsymbol{t}, \delta\boldsymbol{q}, \boldsymbol{\partial\mathcal{F}}, e\}$

of the solution set. Lastly, the IPFO algorithm composited by the palm optimization $\mathcal{PO}$ (Line 6) and finger optimization $\mathcal{FO}$ (Line 7) is a closed mapping, since $\mathcal{PO}$ is continuous and point-to-point, and $\mathcal{FO}$ is closed in $\mathcal{PO}(R, \boldsymbol{t}, \delta\boldsymbol{q})$. Therefore, IPFO described by Alg. (8) converges to a local optimum under the current correspondence. [***End of Proof***]

Fitting Weights Reshaping

The current MDISF algorithm assumes that all points have equivalent importance. With this assumption, MDISF may 1) produce unsatisfying power grasps which either prevent the hand from closing fingers if it matches to the region close to hinge, or 2) easily collide with the ground if the object is flat. To generate natural power grasps, we shape the weights of points on different regions of the hand surface with Gaussian, as shown in Fig. 6.4(a). With this shaping, the central regions of palms and links are emphasized since these regions have better robustness to uncertainties and allow large-scale joint motion. To produce precision grasps for flat objects, we emphasize the fitting of points on fingertips, as shown in Fig. 6.4(b). The surface fitting with weight shaping is similar the original one and can be solved by IPFO. The details are neglected for simplicity.

With the IPFO algorithm in Alg. (8), MDISF searches optimal hand configuration hierarchically using the multi-resolution pyramid, as shown in Alg. (9). MDISF iterates between matching the correspondence (Line 6-7) and searching for optimal transformation and finger displacements $R, \boldsymbol{t}, \delta\boldsymbol{q}$ with IPFO (Line 8).

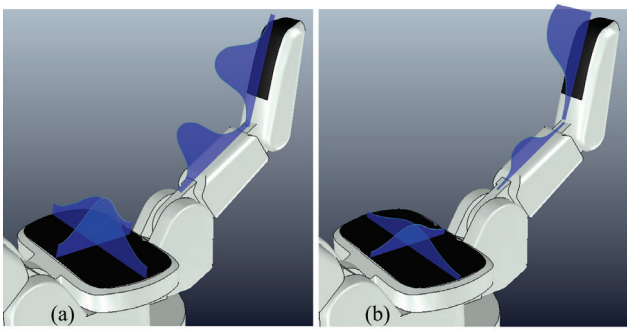


Figure 6.4: Weights shaping for (a) power grasp and (b) precision grasp generation.

Algorithm 9 Multi-Dimensional Iterative Surface Fitting

Input: Initial $R_s, \boldsymbol{t}_s, \delta\boldsymbol{q}_s$, $\partial\mathcal{O}$, $\boldsymbol{\partial\mathcal{F}}$, L, I_0, ϵ_0
Init: $\boldsymbol{\partial\mathcal{F}} = \mathcal{T}(\boldsymbol{\partial\mathcal{F}}; R_s, \boldsymbol{t}_s, \delta\boldsymbol{q}_s)$
for $l = L-1, \cdots, 0$ **do**
    $I_l = I_0/2^l$, $\epsilon_l = 2^l\epsilon_0$, $e_{prev} \leftarrow \infty$, $\eta \leftarrow 0$, $it \leftarrow 0$
    while $\eta \notin [1-\epsilon_l, 1+\epsilon_l]$ and it++ $< I_l$ **do**
        $\mathcal{I} \leftarrow \texttt{filter}(NN_{\partial O}(\texttt{downsample}(\boldsymbol{\partial\mathcal{F}}, 2^l)))$
        $\mathcal{L} \leftarrow \texttt{collisioncheck}(\boldsymbol{\partial\mathcal{F}}, \partial\mathcal{O})$
        $\{R, \boldsymbol{t}, \delta\boldsymbol{q}, \boldsymbol{\partial\mathcal{F}}, e\} \leftarrow \texttt{IPFO}(\boldsymbol{\partial\mathcal{F}}, \partial\mathcal{O}, \mathcal{I}, \mathcal{L})$
        $\eta \leftarrow e/e_{prev}, e_{prev} \leftarrow e$, **Confs** $\leftarrow \{R, \boldsymbol{t}, \delta\boldsymbol{q}\}$
    end while
end for
return $\{e, \boldsymbol{\partial\mathcal{F}}, \textbf{Confs}\}$

Additional Regulation Terms

In this section, we introduce two additional grasp regulation terms to specify the characteristics of desired grasps. First, to produce a feasible grasp that satisfy the manipulability of the manipulator, we may regulate the orientation of the palm. Second, to avoid excessive large diameter grasps, we may regulate the volume of the desired grasp.

Palm Orientation Regulation

Manipulability of manipulators is crucial for grasp execution and automation. A planned grasp with nontrivial pose (Fig. 6.11(10)) may pose a challenge for motion planning of the robot-hand system. On the other hand, grasping in a top-down manner is beneficial to maintain the manipulability of the manipulator, despite the fact it constrains the grasp to a subspace of all feasible grasps. This section introduces a palm orientation regulation term in order to maintain grasp versatility and prioritize the manipulable grasps. The palm

regulation term E_{pr} has the following form:

$$E_{pr} = \left[(Rn_p)^T e_z - 1\right]^2 \tag{6.11}$$

where R is the palm rotation matrix, n_p is the current normal direction of the palm pointing towards the finger workspace, $e_z = [0, 0, -1]^T$ is the target palm normal direction corresponding to a top-down grasp.

The palm orientation term can be integrated if the manipulability of the manipulator needs to be regulated.

Grasp Volume Regulation

MDISF searches grasps based on the surface fitting error. Minimizing surface fitting error, however, may produce unfavorable grasps in some special cases. For instance, the desired grasp on a book in the sense of surface fitting is a large diameter grasp, and the contacts might only lie on the top surface without satisfying the force closure conditions. Similar behavior can be observed for the surface fitting in clutter environments, where the point cloud is spanned across the scene. To avoid large diameter non force-closure grasping, we regulate the volume of the object that contained inside of the convex hull formed by the contact polygon and inner hand surface. The volume, however, is challenging to connect with the optimization variables $\{R, \boldsymbol{t}, \delta\boldsymbol{q}\}$. To simplify the formulation, the volume regulation term E_{vr} is replaced by the hand manipulability measure in Chapter 5:

$$E_{vr} = -\sum_{i=1}^{N_f} \sum_{j=1}^{N_{jnt,i}} (\alpha_i^j \frac{q_i^j - \bar{q}_i^j}{q_{\max,i}^j - q_{\min,i}^j})^2, \tag{6.12}$$

where $N_f, N_{jnt,i}$ are the number of fingers and number of joints in the i-th finger, and $\bar{q}_i^j, q_{max,i}^j, q_{min,i}^j$ are the mean and limit values of the j-th joint in the i-th finger. α_i^j is the weights for the j-th joint of the i-th finger. To obtain versatile grasps, the spread joints are not regulated, i.e., $\alpha_1^1 = \alpha_2^1 = 0$.

The grasp volume term can be integrated if the algorithm is running in clutter environments.

Grasping Imagination

In this section, the found grasps are first ranked based on the proposed quality metric, after which the grasp trajectories are planned to reach the highly ranked grasps.

Grasp Quality Evaluation

The grasp quality is evaluated based on the grasp wrench space (GWS) [79]. In this chapter, GWS $\mathcal{P}$ is constructed by 1) finding contact points by the nearest neighbor of the final hand surface on the object, 2) removing the contacts with large normal alignment error, 3)

extracting the center points and the average normals by K-means, and 4) building $\mathcal{P}$ based on the extracted grasp points and normals using the soft finger model [69]. With the GWS $\mathcal{P}$, three quality features are calculated. The first feature Q_{in} is a bool type variable indicates the ability to resist arbitrary small disturbance by checking the inclusion of origin in $\mathcal{P}$. The second feature Q_{vol} indicates the magnitude of disturbance resistance by computing the volume of $\mathcal{P}$. The third feature Q_{cond} indicates the isotropy of the disturbance resistance by the condition number of the WW^T, where $W \in \mathbb{R}^{6\times N_p}$ is the vertex matrix of convex hull $\mathcal{P}$.

The final grasp quality metric Q_{gsp} is represented as:

$$Q_{gsp} = Q_{vol} + \frac{3}{Q_{cond}} + 11Q_{in} \tag{6.13}$$

The parameters of (6.13) is obtained by regression using the standard Ferrari-Canny metric [30] on 200 grasps from 10 objects. Equation (6.13) is able to rank the found collision-free grasps more efficiently than the Ferrari-Canny metric with comparable accuracy. Compared with Ferrari-Canny metric, the computation time of (6.13) reduced by 98.77% from 2.43 secs/grasp to 0.034 sec/grasp. The top-1 score is 70% and top-3 score is 90%, out of 20 classes to be ranked.

To enhance the robustness of MDISF, the candidate grasps are randomly sampled from a Gaussian distribution to mimic the uncertainties. More concretely, for a candidate grasp with final palm pose $R_{\text{final}}, \boldsymbol{t}_{\text{final}}, \boldsymbol{q}_{\text{final,m}}$, we randomly sample N_t times:

$$(R_t, \boldsymbol{t}_t, \boldsymbol{q}_{t,m}) \sim \mathcal{N}((R_{\text{final}}, \boldsymbol{t}_{\text{final}}, \boldsymbol{q}_{\text{final,m}}), diag(\Sigma_r, \Sigma_t, \Sigma_q))$$

where $\boldsymbol{q}_m \in \mathbb{R}^4$ (different with joint angle $\boldsymbol{q} \in \mathbb{R}^8$) denotes the motor value of hand for BarrettHand, and $\Sigma_r \in \mathbb{R}^{3\times3}, \Sigma_t \in \mathbb{R}^{3\times3}, \Sigma_{q,m} \in \mathbb{R}^{4\times4}$ represent the covariance matrices of the Euler angle of the rotation matrix, translation vector, and joint angles. The grasp qualities of different grasp samples are averaged, and the highly ranked grasps are fed for trajectory generation.

Grasp Trajectory Optimization

This chapter presents a two-step procedure to plan the robot-finger trajectories. First, the hand keeps half-closed and approaches the pre-grasp pose with [56]. The object is represented by its bounding box in this step. The pre-grasp position is defined by lifting the hand 0.3 m from the final grasp, and the rotation is defined as the closest canonical orientation from the final grasp pose. Second, we optimize for the finger trajectories while predefining the palm

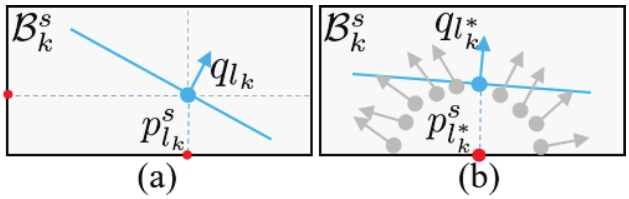


Figure 6.5: (a) Point-box distance calculation. (b) Cloud-box distance calculation.

trajectory by interpolation. The grasp trajectory optimization (GTO) becomes:

$$\min_{\boldsymbol{q}_1,...,\boldsymbol{q}_S} \sum_{s=1}^{S-1} \|\boldsymbol{q}_{s+1} - \boldsymbol{q}_s\|^2 \tag{6.14a}$$

$$s.t. \quad dist\left(\mathcal{T}(\boldsymbol{\partial\mathcal{F}_s}; \boldsymbol{q}_s - \boldsymbol{q}_s^0), \partial\mathcal{O}|\mathcal{G}\right) \geq 0, \tag{6.14b}$$

$$|\boldsymbol{q}_{s+1} - \boldsymbol{q}_s| \leq \Delta_q, \quad s = 1, ..., S-1 \tag{6.14c}$$

$$\boldsymbol{q}_s \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \quad s = 1, ..., S \tag{6.14d}$$

$$\boldsymbol{q}_1 = \boldsymbol{q}_{\text{pregrasp}}, \boldsymbol{q}_S = \boldsymbol{q}_{\text{final}}. \tag{6.14e}$$

where s is the sample index, S is the number of samples on the trajectory, and $\mathcal{T}(\boldsymbol{\partial\mathcal{F}_s}; \boldsymbol{q}_s - \boldsymbol{q}_s^0)$ denotes the transformed hand surfaces after the joint displacement at the s-th sample. Optimization (6.14) is to minimize the total length of the trajectory (6.14a) from the pre-grasp to final grasp (6.14e) and avoid collision with both the object and ground (6.14b).

Similar to MDISF, the collision constraints are penalized in the cost. We adopt the formulation in TrajOpt [84]:

$$col\_term = |d_{safe} - sd(\partial\mathcal{O}, \mathcal{T}(\boldsymbol{\partial\mathcal{F}_s}; \boldsymbol{q}_s - \boldsymbol{q}_s^0)|^+ \tag{6.15}$$

where d_{safe} denotes the safety distance, $sd(A, B)$ denotes the signed distance between A and B, and $|x|^+ = \max(x, 0)$.

We propose an approach to compute the signed distance $sd(\partial\mathcal{O}, \mathcal{T}(\boldsymbol{\partial\mathcal{F}_s}; \boldsymbol{q}_s - \boldsymbol{q}_s^0))$ in absence of the 3D mesh and convex decomposition of the object, as shown in Fig. 6.5. We first inflate the bounding boxes $\{\mathcal{B}_k^s\}_{k=1}^{N_b}$ at sample s by d_{check} and check the inclusion of object points. For each interior point q_{l_k}, we calculate the signed distance by projecting q_{l_k} to surfaces of $\mathcal{B}_k^s$ and filtering out the points with $(q_{l_k} - p_j)^T n_l^q < 0$ for those $q_{l_k} \in \mathcal{B}_k^s$. The closest point to q_{l_k} is denoted as $p_{l_k}^s$, as shown in Fig. 6.5(a). The point-box signed distance $sd(\mathcal{B}_k^s, q_{l_k}) = (p_{l_k}^s - q_{l_k})^T n_{l_k}^s$ and $n_{l_k}^s$ is a normal vector with direction $q_{l_k} - p_{l_k}^s$ if $q_{l_k} \in \mathcal{B}_k^s$ or reverse otherwise. Therefore, $sd(\mathcal{B}_k^s, \partial\mathcal{O}) = \min_{l_k} sd(\mathcal{B}_k^s, q_{l_k})$ with the critical index $l_k^* = \text{argmin}_{l_k} sd(\mathcal{B}_k^s, q_{l_k})$, as shown in Fig. 6.5(b). The hand-object collision indexes $\mathcal{L}_{s,o} = \{l_k^*\}_{k=1}^{N_b}$. Similarly, the hand-ground collision indexes $\mathcal{L}_{s,g}$ includes all the points that have potential collision with ground.

With $\mathcal{L}_{s,o}, \mathcal{L}_{s,g}$, the collision for the s-th sample is penalized as:

$$E_{col,s} = \sum_{l_k^* \in \mathcal{L}_{s,o}} \left(|d_{safe} - (\bar{p}_{l_k^*}^s - q_{l_k^*})^T n_{l_k^*}^s|^+ \right)^2 + \sum_{l_k^* \in \mathcal{L}_{s,g}} \left(|d_{safe} - (\bar{p}_{l_k^*}^s - q_g)^T n_g|^+ \right)^2 \tag{6.16}$$

where $\bar{p}_{l_k^*}^s = p_{l_k^*}^s + \mathcal{J}_{l_k^*}^v(\boldsymbol{q}_s^0)\delta\boldsymbol{q}_s$. Therefore, GTO (6.14) can be reformulated as:

$$\min_{\delta\boldsymbol{q}_1,...,\delta\boldsymbol{q}_S} \sum_{s=1}^{S-1} \|\boldsymbol{q}_{s+1}^0 + \delta\boldsymbol{q}_{s+1} - \boldsymbol{q}_s^0 - \delta\boldsymbol{q}_s\|^2 + cE_{col,s} \tag{6.17a}$$

$$s.t. \quad |\boldsymbol{q}_{s+1}^0 + \delta\boldsymbol{q}_{s+1} - \boldsymbol{q}_s^0 - \delta\boldsymbol{q}_s|_\infty \leq \Delta_q, \tag{6.17b}$$

$$\boldsymbol{q}_s^0 + \delta\boldsymbol{q}_s \in [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}], \quad s = 1, ..., S \tag{6.17c}$$

$$\delta\boldsymbol{q}_1 = 0, \delta\boldsymbol{q}_S = 0, \tag{6.17d}$$

$$|\delta\boldsymbol{q}_s| < \Delta_{\delta q}. \quad s = 1, ..., S \tag{6.17e}$$

Optimization (6.17) solves for optimal joint displacements $\{\delta\boldsymbol{q}_s^*\}_{s=1}^S$ using the current joint samples and collided points. The $\{\delta\boldsymbol{q}_s^*\}_{s=1}^S$ then updates joint samples $\boldsymbol{q}_s^0 \leftarrow \boldsymbol{q}_s^0 + \delta\boldsymbol{q}_s^*$, hand surfaces $\boldsymbol{\partial\mathcal{F}_s} \leftarrow \mathcal{T}(\boldsymbol{\partial\mathcal{F}_s}; \delta\boldsymbol{q}_s^*)$, collision penalty $c \leftarrow \mu c$ and indexes $\mathcal{L}_{s,o}, \mathcal{L}_{s,g}$. Optimization (6.17) iterates until no collision or reaching the maximum iterations.

6.4 Simulations and Experiments

This section presents the simulations and experiments. The experimental videos are available at [102].

Parameter Lists

For MDISF, $\alpha = 0.03$. $N_p = 450, N_b = 7$, $L = 4, I_0 = 200, \epsilon_0 = 0.02$. IPFO used $\Delta = 1e-5, T_{max} = 20$. The power grasp used Gaussian $\Sigma = diag([l/2, w/0.1])$ with mean at the link center, where l, w are the link length and width, and the base weights for palm, proximal and distal link were $0.1, 0.1, 1$. The precision grasp used Gaussian $\Sigma = diag([l/5, w/0.1])$ with mean at the fingertip and base weights $0.01, 0.01, 1$. For robust quality analysis, $N_t = 9$, $\Sigma_r = 0.03^2 diag([1,1,1])$, $\Sigma_t = 0.003^2 diag([1,1,1])$, $\Sigma_q = 0.05^2 diag([1,1,1,1])$.

As for GTO, $d_{check} = 0.03$ m and $d_{safe} = 0.01$ m, $S = 30$, $\Delta_{\delta q} = [0.2, 0.2, 0.2, 0.4]$, $\Delta_q = [0.4, 0.4, 0.4, 0.4]$, The maximum iterations for (6.17) was 20. The starting collision penalty $c_0 = 1$ and $\mu = 2$.

Simulation and Experiment Results

The simulation was conducted on a desktop with 32 GB RAM and 4.0 GHz CPU. The grasps were computed by Matlab and visualized by V-REP. The BarrettHand BH8-282 was used to

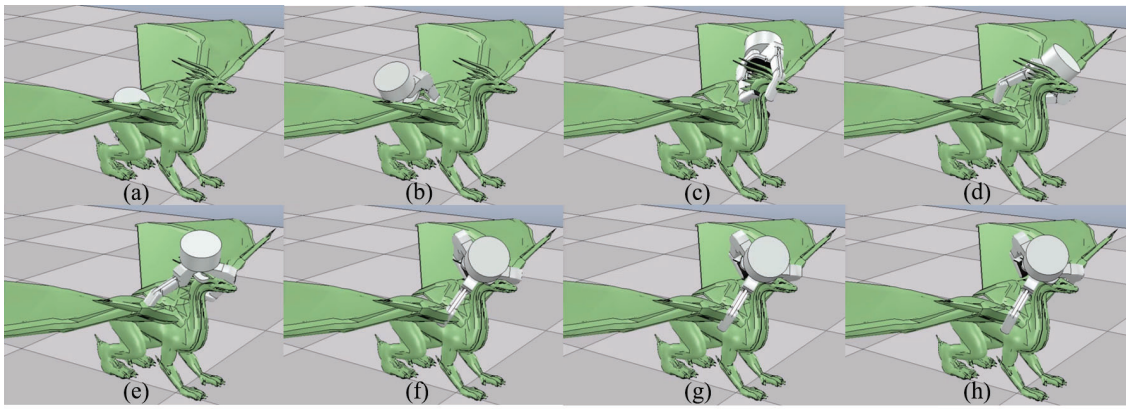


Figure 6.6: Visualization of MDISF iterations on Dragon object.

test the effectiveness of the algorithm. The visualization of the MDISF iterations is shown in Fig. 6.6. MDISF considers both the collision avoidance and the surface fitting in each IPFO iteration. MDISF started from a random pose around the object (Fig. 6.6(a)) and optimized for palm pose and joint displacements to reduce the fitting error and penalize the collision (Fig. 6.6(b-h)).

Figure 6.7 shows the error reduction profile to validate that both the average surface fitting error $E_{fit}/|\mathcal{I}|$ and collision cost E_{col} are reduced during MDISF. Figure 6.7(Left) shows the overall error, Fig. 6.7(Middle) shows the average surface fitting error after outliers/duplicate removal, and Fig. 6.7(Right) shows the collision error without multiplying the penalty weight. The red and blue plots show the mean errors and the deviations for all samples, while the purple and yellow plots show those for collision-free grasps. In average, it took 25.2 ± 6.3 IPFOs and 100.2 ± 34.4 PFOs to converge. The average fitting error $E_{fit}/|\mathcal{I}|$ reduced from 0.0072 ± 0.0031 m to 0.0027 ± 0.0012 m, and the absolute E_{col} reduced from 0.5952 ± 0.4342 m to 0.0287 ± 0.0410 m. All statistics were computed based on 50 samples on bunny object shown in Fig. 6.2.

Figure 6.8 and Table 6.1 show the visualization and quantitative results of the proposed method on ten different objects. The MDISF algorithm sampled 10 times for each object and returns 6.2 collision free grasps in 2.45 secs. The highest 5 (or the maximum number of collision free grasps found by MDISF) grasps were selected and fed to GTO for trajectory optimization. GTO returned 3.3 collision-free trajectories out of 4.0 grasps in 2.02 secs. The surface fitting error provided a reliable metric for grasp searching, since the majority (6.1/6.2) grasps found by MDISF were force closure (FC).

The Q_{gsp} reflects the graspability (difficulty of being grasped) of the objects with the Barrett hand. Goblet and screwdriver were the top-2 objects with the highest graspability due to the proper size and simple structure. Hand model and gun were the top-2 objects with the lowest graspability since they are flat and close to ground, and Barrett hand can easily collide with ground/object and get trapped in the infeasible local optima.

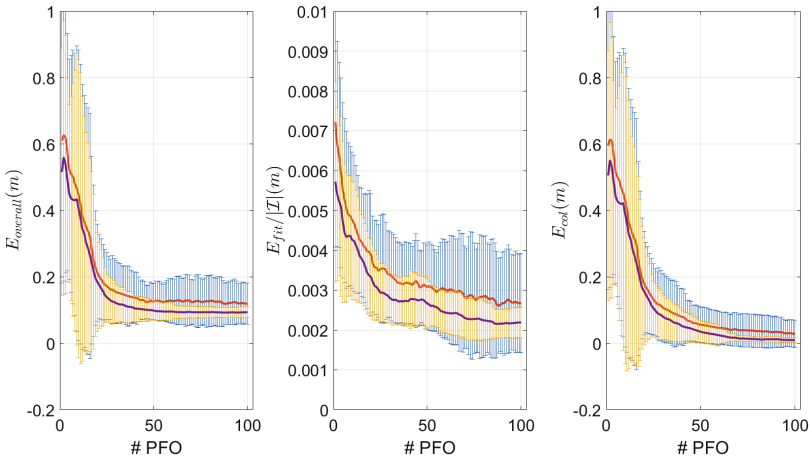


Figure 6.7: Profile of the error reduction during MDISF.

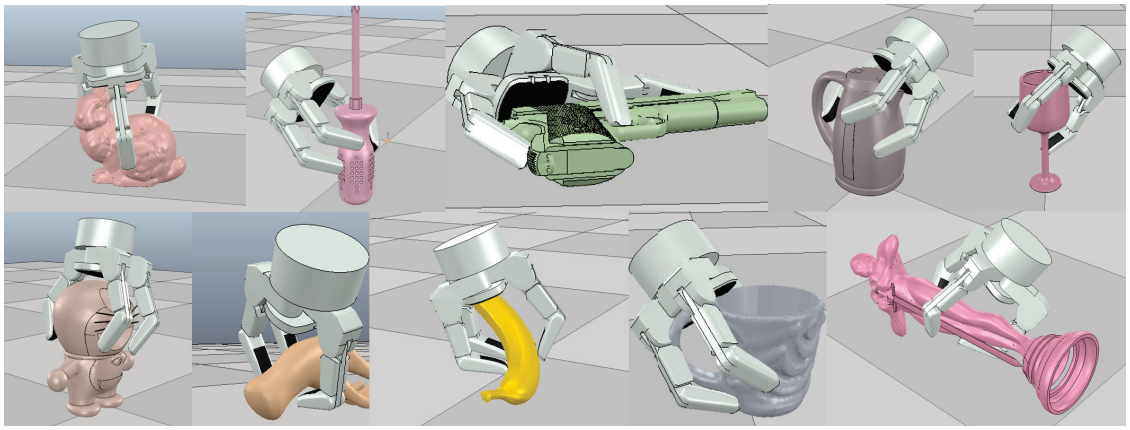


Figure 6.8: Simulation results of MDISF on ten objects.

Figure 6.9 compares the precision grasps and power grasps generated by MDISF. The precision mode and power mode generated 8 and 5 collision-free grasps out of 10 samples on Bunny object, respectively. The hand tended to collide with the object in power grasp mode since the fitting of palm/proximal links were emphasized and hand stayed closer to object, as shown in Fig. 6.9(b).

Figure 6.10 shows the result of GTO on Kettle object. The trajectory started from the top and reached the desired grasp with one finger in narrow space. Green spots indicate the collided regions. The fingers collided with the object when grasping with the predefined finger motion, as shown in Fig. 6.10(Top). GTO planed finger trajectories to avoid collision and reached the target grasp in narrow space, as shown in Fig. 6.10(Bottom).

Figure 6.11 shows the experimental results using the FANUC LRMate 200iD/7L ma-

Table 6.1: Numerical Results of the Grasping Framework

Object	collision-free# / total samples		Time (secs)			Qualities	
	ISF	GTO	ISF	GTO	Sum	FC	Q_{gsp}
Bunny	8/10	4/5	2.8	3.2	6.0	7/8	6.86
Screwdriver	9/10	1/5	2.2	3.0	5.2	9/9	9.56
Gun	2/10	1/2	2.5	1.8	4.3	2/2	-9.41
Kettle	8/10	5/5	2.1	2.2	4.3	8/8	3.39
Goblet	10/10	5/5	2.4	1.7	4.1	10/10	13.66
Doraemon	9/10	5/5	2.1	1.6	3.7	9/9	9.27
Hand	1/10	1/1	2.3	0.6	2.9	1/1	-12.32
Banana	4/10	3/4	2.2	2.5	4.7	4/4	-1.71
Mug	8/10	5/5	3.0	2.2	5.2	8/8	8.18
Oscar	3/10	3/3	2.9	1.4	4.3	3/3	-6.80
Average	$\frac{6.2}{10}$	$\frac{3.3}{4.0}$	2.45	2.02	4.47	$\frac{6.1}{6.2}$	2.068

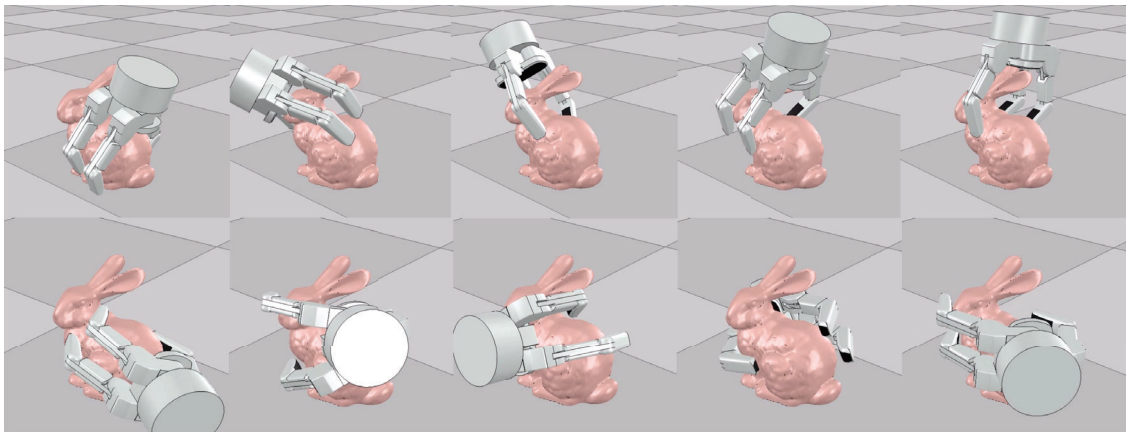


Figure 6.9: Comparison of (Top) precision grasp mode and (Bottom) power grasp mode of MDISF on Bunny object.

nipulator and BarrettHand BH8-282 on ten objects. The scene was captured by two IDS Ensenso N35 stereo cameras. The observed point cloud and the optimized grasp are shown in the left, and the executed grasp after GTO is shown in the middle. Extra amount of finger motion was executed in order to provide necessary force to clamp the object, as shown in the right. The observed point cloud was incomplete and noisy, and the system also contained uncertainties in calibration (∼3 mm for robot-camera frame alignment), positioning (∼1° TCP-palm alignment, ∼2.0° finger joint tracking error) and communication (∼0.1 sec robot-hand command synchronization error). The system exhibited certain robustness and was able to plan and execute grasps under the unsatisfying point cloud and various types of

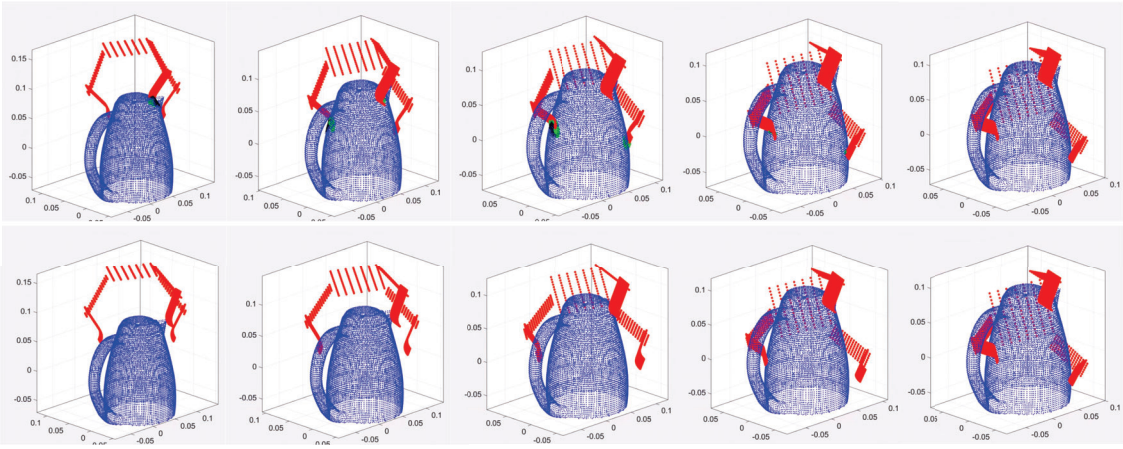


Figure 6.10: Trajectory snapshots for grasp execution including (Top) predefined finger motion, and (Bottom) optimized trajectory by GTO.

uncertainties, as shown in Fig. 6.11(1-9).

We also include three failed grasps to reflect three failure modes, as shown in Fig. 6.11(10-12). The first failure mode was raised from slippage under non-static contacts with the under-actuated hand. Without quasi-static pressing within the actual friction cone, the equilibrium of the object will be destroyed, as shown in Fig. 6.11(10). The second failure mode was raised from the asymmetric distribution of contact forces during clamping, as shown in Fig. 6.11(11). The third failure mode was raised from the internal disturbance. The proximal link accidentally collided with object during the clamp stage and introduced a large disturbance, as shown in Fig. 6.11(12).

Finally, we demonstrate the grasp planning and execution of the proposed framework in clutter environments. The BarrettHand can form a large diameter grasp by stretching fingers. To avoid BarrettHand grasping multiple objects in a single grasp, we first segmented the current scene into individual clusters by Density-based spatial clustering of applications with noise (DBSCAN) [16]. The largest cluster was then selected for surface fitting by MDISF. The remaining part of the point cloud was regarded as obstacles in the grasp planning and grasp trajectory optimization. DBSCAN might group several objects into a single cluster, especially when the objects are in heavy clutter environments. To avoid MDISF forming excessively large diameter grasps, the maximum span of the grasp was controlled by a volume regulation term in Section 6.3. To produce reliable and manipulable grasps, we also added a palm orientation regulation term in Section 6.3.

Figure 6.12 shows the snapshots of execution. The images with green background show a successful picking sequence. A grasp test on 6 objects by 97 grasps shows a 82.47% success rate. The images with red background show failed grasps. The failure is caused by two main reasons. The first is the unexpected perturbation from unsynchronized contacts, as shown in Fig. 6.12(f0-f2). The second reason is the slippage of contacts during the object clamping

stage, as shown in Fig. 6.12(g0-g2).

6.5 Chapter Summary

This chapter proposed an efficient framework for grasp generation and execution by combining the surface fitting in Chapter 2 and optimization modeling in Chapter 5. The framework includes a multi-dimensional iterative surface fitting (MDISF) and a grasp trajectory optimization (GTO). The MDISF algorithm searches for optimal grasps by minimizing the hand-object fitting error and penalizing the collision, and the GTO algorithm plans finger trajectories for grasp execution with the point cloud representation of the object. The MDISF-GTO exhibits certain robustness to the incomplete/noisy point cloud and various underlying uncertainties. In average, it took 0.40 sec for MDISF to find a collision-free grasp, and took 0.61 sec for GTO to optimize the trajectory to reach the grasp. The proposed framework is able to be implemented in both customized grippers and multi-fingered hands, and plan grasps on individual objects or in clutter environments. Simulations and Experiments with BarrettHand BH8-282 verified the effectiveness of the framework.

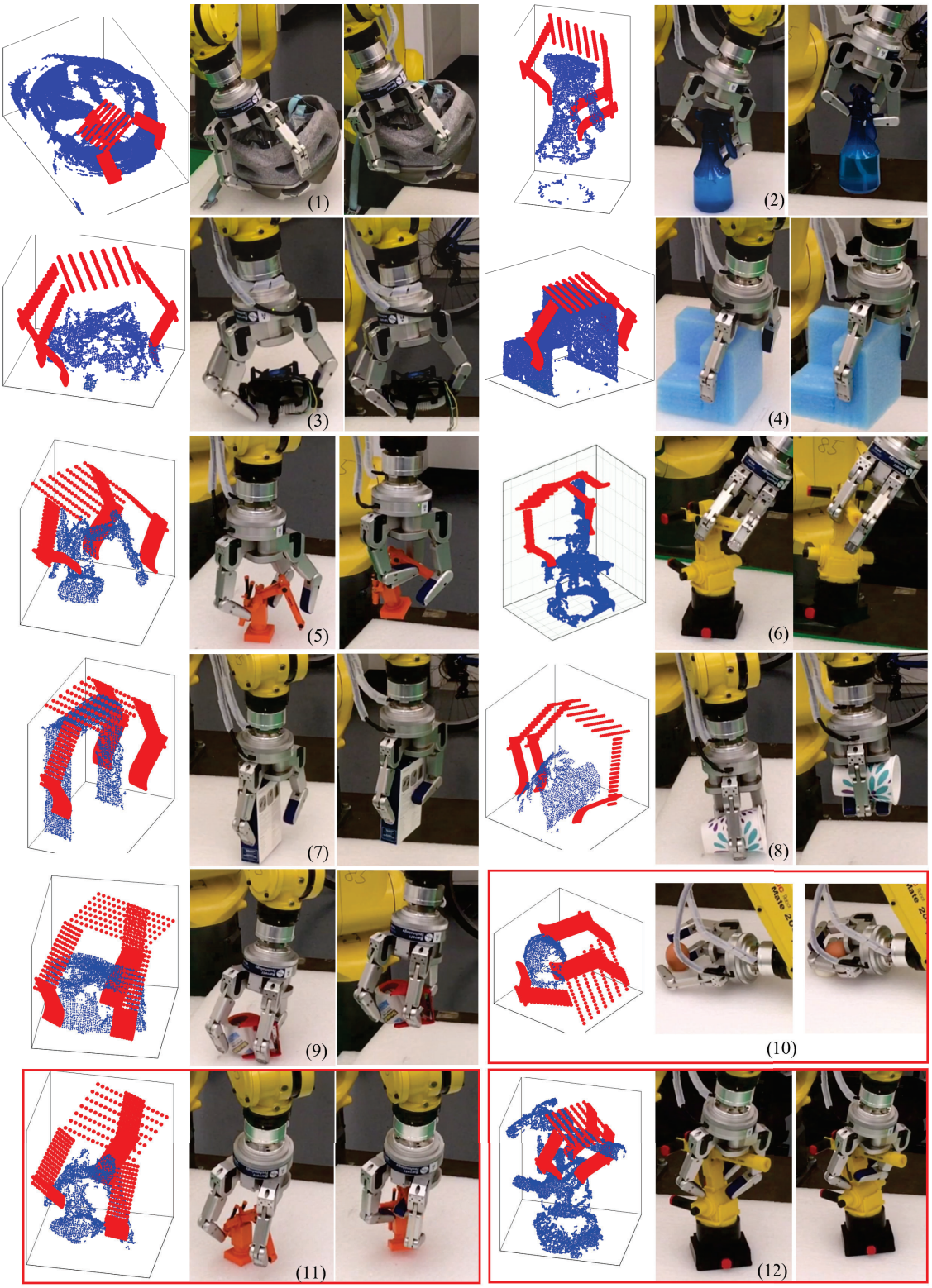


Figure 6.11: Illustration of the grasp experiments on 10 objects.

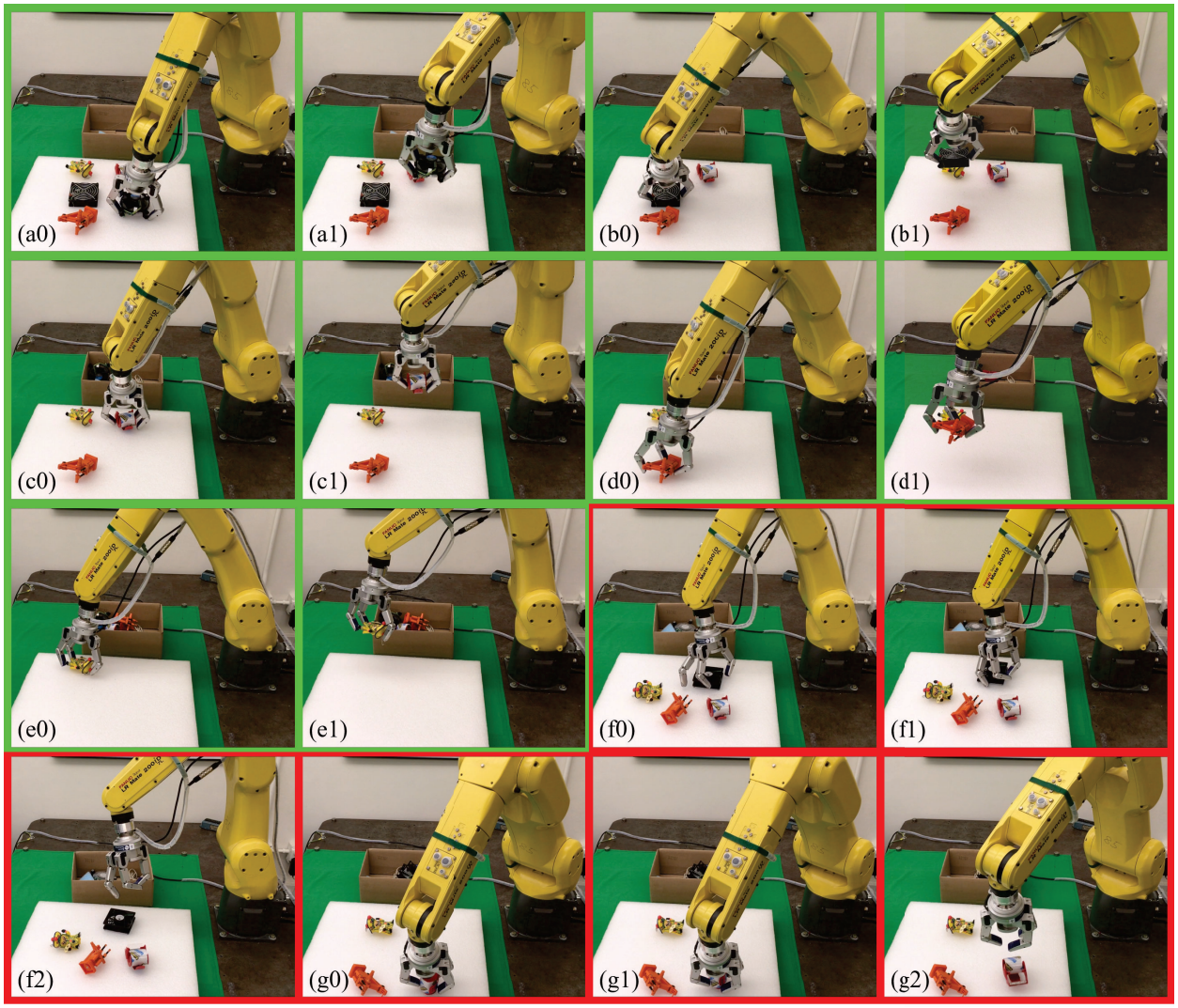


Figure 6.12: Snapshots of grasp planning in clutter environments with success rate 80/97.

Part II

Manipulation

Chapter 7

Object Manipulation Architecture by Modified Impedance Control

7.1 Introduction

In Chapter 6, we introduced a general framework for efficient grasp planning and execution. In order to lift the object or follow more general object trajectories robustly and reliably, we have to optimize the grasp forces after building contacts, instead of clamping the object by simply closing fingers. This type of force optimization in post-grasping stage is called in-hand manipulation. In general, in-hand manipulation is to produce desired time-varying forces from fingertips to object, in order to manipulate the object to follow reference trajectories. However, realizing dexterous in-hand manipulation is challenging regarding the manipulative dexterity and the grasping robustness. In particular, the fingers and the manipulated objects are required to achieve desired dexterous object motion in the presence of uncertainties such as uncertain 3D shapes or imprecise dynamic models.

In-hand manipulation has broad potential applications in industry and agriculture. Figure 7.1(a) shows an industrial circuit board assembly task, where a circuit board is picked and placed onto a platform before it is re-grasped from a different orientation in order to perform the final insertion procedure. Figure 7.1(b) shows a fruit packaging task, where each fruit has to be rotated and arranged in certain orientation. By properly manipulate the object in hand, the in-hand manipulation can simplify the procedure and reduce cycle time.

This chapter introduces a comprehensive architecture for dexterous in-hand manipulation. The architecture includes a high-level robust controller for object Cartesian force generation, a mid-level manipulation controller for contact force optimization, and an optional low-level force tracking controller to track the optimized contact force. The objective of this chapter is to build the manipulation architecture and validate the architecture with different types of physical multi-fingered hands. The architecture should be able to adapt to hands with different structures (sensor types, drivetrains, and control inputs, etc.), and achieve basic level of robustness.

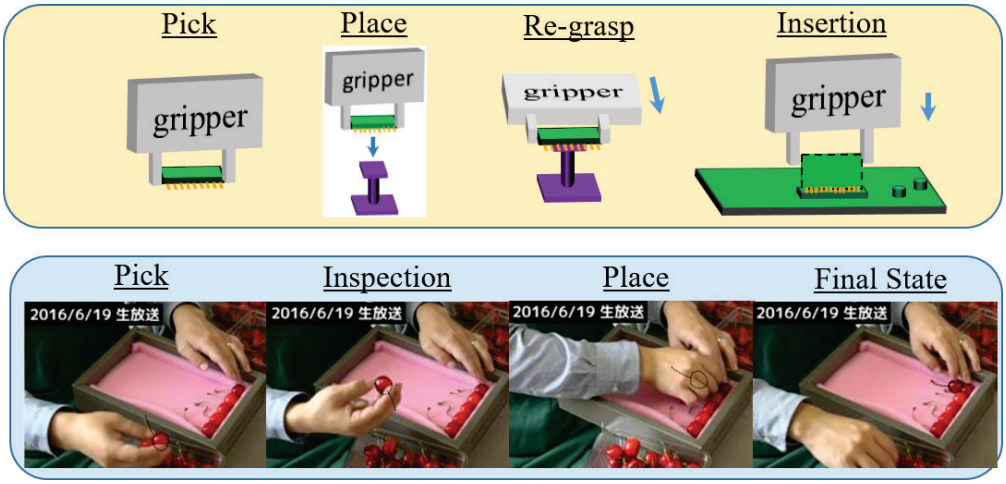


Figure 7.1: (Top) Circuit board assembly and (Bottom) fruit packaging.

The presented manipulation architecture exhibits both flexibility and certain robustness. With the hierarchical structure, the architecture is able to run on hands with torque, force or velocity inputs and resist certain level of mass uncertainties, contact uncertainties and external disturbances. The robustness is achieved by the proposed modified impedance control (MIC). A more effective model-based robust manipulation controller (RMC) will be introduced in Chapter 8. The manipulation architecture can be served as a low-level controller for a higher level finger gaits planning. The details of the connection will be introduced in Chapter 9.

The remainders of this chapter are as follows. Section 7.2 introduces the object manipulation architecture, including a high-level modified impedance controller, a mid-level manipulation controller, and a low-level tracking controller for command execution. The simulation and experimental details are presented in Section 7.3 and Section 7.4. Section 7.5 concludes this chapter and introduces the future work.

7.2 In-Hand Manipulation Architecture

Overall Structure

Figure 7.2 describes the overall object manipulation architecture. In this figure, r, y, n and e denote the reference pose, the actual pose, the measurement noise and the pose error of the object, respectively. The signal u denotes the control input to hand motors. The signal F_{dis} is the external disturbance to the plant. F is the Cartesian space force command on the object. The signal f is the contact force command to the object in order to realize F. The objective is to: 1) track the desired pose r of the object, 2) achieve basic robustness to

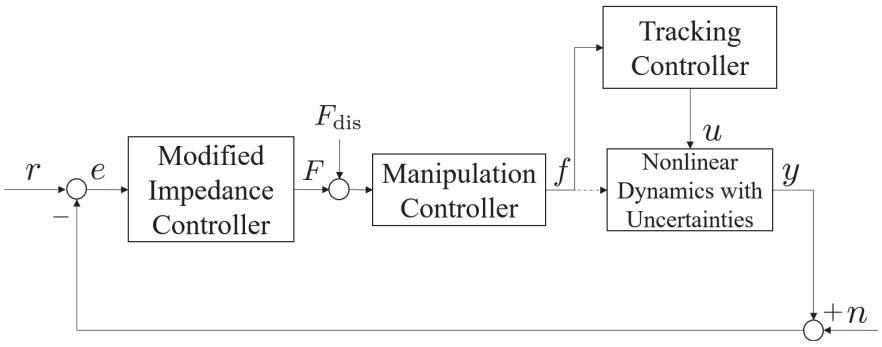


Figure 7.2: Illustration of the object manipulation architecture.

object mass uncertainty, contact force tracking errors and external disturbances F_{dis}, and 3) realize firm contact without violating the friction cone constraints.

To achieve the objectives, the manipulation architecture is designed to include a high-level modified impedance controller (MIC), a mid-level manipulation controller and a low-level tracking controller. The high-level MIC takes the object trajectory error as input and generates object Cartesian force command F to track the desired object trajectory and suppress external disturbances. The mid-level manipulation controller transforms the Cartesian force command F into contact force command f on contact frames to satisfy the requirements including slippage avoidance, effort minimization and force smoothness. The low-level tracking controller converts the contact force command f to hand inputs u, where u can be the torque, velocity or other commands. The hand takes u and interacts with the object to change the object pose y. The object pose is observed by a vision sensor with noise n and feedback to high-level MIC to close the control loop. Depending on the equipped sensors in each finger, the tracking controller may have different structures. These variances will be introduced below.

High-Level Modified Impedance Control (MIC)

The hand and object dynamics are described by:

$$\begin{aligned} &M_h(q)\ddot{q} + C_h(q,\dot{q})\dot{q} + N_h(q,\dot{q}) + J_h^T(q,x_o)f_c = \tau \\ &M_o(x_o)\ddot{x}_o + C_o(x_o,\dot{x}_o)\dot{x}_o + N_o = G(q,x_o)f_c \end{aligned} \tag{7.1}$$

where $M_{h/o}, C_{h/o}$ and $N_{h/o}$ are inertia matrices, Coriolis matrices and gravities for the hand/object. $q, \dot{q}$ and $\ddot{q} \in \mathbb{R}^{n_q}$ are joint angle, velocity and acceleration, with n_q as the total degree of freedoms (DOFs) of the hand. $x_o, \dot{x}_o$ and $\ddot{x}_o \in \mathbb{R}^{n_x}$ are a local parameterization of object position, velocity and acceleration, where n_x is the dimension of the pose of the object, with $n_x = 6$ for 3D manipulation ($n_x = 3$ for 2D manipulation). $f_c \in \mathbb{R}^{d_c n_c}$ and $\tau \in \mathbb{R}^{n_q}$ are contact force vector and joint torque vector, where d_c is the dimension

of each contact, and n_c is the contact number. $J_h \in \mathbb{R}^{(d_c n_c)\times n_q}$ is the hand Jacobian and $G \in \mathbb{R}^{n_x \times (d_c n_c)}$ is the grasp map [69].

If the contacts are fixed w.r.t both object and fingertips, then

$$J_h(q, x_o)\dot{q} = G^T(q, x_o)\dot{x}_o \tag{7.2}$$

holds. Equation (7.2) assumes the contact forces remain in the friction cone.

The object and hand dynamics in (7.1) can be connected by (7.2):

$$M(q, x_o)\ddot{x}_o + C(q, \dot{q}, x_o, \dot{x}_o)\dot{x}_o + N(q, x_o) = GJ_h^{-T}\tau \tag{7.3}$$

where:

$$\begin{aligned} M &= M_o + GJ_h^{-T}M_hJ_h^{-1}G^T \\ C &= C_o + GJ_h^{-T}C_hJ_h^{-1}G^T + GJ_h^{-T}M_h\frac{d(J_h^{-1}G^T)}{dt} \\ N &= N_o + GJ_h^{-T}N_h \end{aligned} \tag{7.4}$$

The torque command τ can be related to the object-centered force F:

$$\tau = J_h^T(G^\dagger F + N_G\lambda) \tag{7.5}$$

where N_G is the matrix composed by the basis of the null space of G, and λ is a free variable to control the magnitude and direction of the contact force.

The state space equation can be derived by plugging (7.5) into (7.3):

$$M(q, x_o)\ddot{x}_o + C(q, \dot{q}, x_o, \dot{x}_o)\dot{x}_o + N(q, x_o) = F \tag{7.6}$$

The desired motion of the object is transformed into desired forces on the object through impedance control, which is similar to [51, 104]. In this section, an additional integral term is added to impedance controller to address object mass uncertainty:

$$\begin{aligned} F_{\text{imp}} =& M^d\ddot{x}_o + B^d(\dot{x}_o - \dot{r}) + K^d(x_o - r) + \\ & I^d\left(\int_0^t (x_o - r)dt\right) \end{aligned} \tag{7.7}$$

where $\dot{r}$ is the desired velocity of the object, and M^d, B^d and K^d are the desired inertia, damping and stiffness, respectively. I^d is a gain matrix for additional integral term.

With the impedance force F_{imp} as a virtual force, the full system dynamics become:

$$M(q, x_o)\ddot{x}_o + C(q, \dot{q}, x_o, \dot{x}_o)\dot{x}_o + N(q, x_o) = F + F_{\text{imp}} \tag{7.8}$$

The desired inertia M^d is set as M in order to remove the inertia term and acceleration measurement [51]. In general 3D manipulation, the Coriolis term is typically ignored due to the low-speed operation condition, as shown in [51]. By plugging (7.7) into (7.8), the Cartesian force command becomes:

$$F = C(q, \dot{q}, x_o, \dot{x}_o)\dot{x}_o + N(q, x_o) + B^d(\dot{r} - \dot{x}_o) + K^d(r - x_o) + I^d\int^t (r - x_o)dt \tag{7.9}$$

Mid-Level Manipulation Controller Design

The manipulation controller is utilized to generate force commands for the hand to track the desired force generated by the modified impedance controller in Section 7.2. The manipulation controller consists of a force optimizer to search desired contact force f on fingertips from the desired force F on the object. The force optimizer is formulated into a quadratic programming (QP):

$$\min_{\beta} \quad \alpha_1\|f\|_2^2 + \alpha_2\|f - f_{\text{prev}}\|_2^2 + \alpha_3\|\Psi\|_2^2 \tag{7.10a}$$

$$s.t. \quad \Psi = F - G(q, x_o)f \tag{7.10b}$$

$$f = B\beta \tag{7.10c}$$

$$\beta \geq 0 \tag{7.10d}$$

$$f_{\min} \leq f \leq f_{\max} \tag{7.10e}$$

where $f = [f_1^T, ..., f_{n_c}^T]^T$ is a concatenated contact force vector in contact frame. f_{prev} is the contact force of the previous time step. $B = \text{diag}\{B_1, ..., B_{n_c}\}$ and B_i is a conservative pyramid approximation of friction cone [57]. $\beta \geq 0$ is the non-negative coefficients of columns of B. A slack variable Ψ is introduced to relax the hard constraint $F = Gf$, since $F = Gf$ might result in an infeasible solution, and the location measurements of contact points might be noisy. The constraints (7.10c) and (7.10d) together ensure that the contact force remains within positive $colspan(B)$ (i.e. friction cone). Constraint (7.10e) guarantees that the contact force f is realizable.

The weights $\alpha_1, \alpha_2, \alpha_3$ are used to balance different cost terms. They are tuned to penalize the magnitude of the contact force, the change of the contact force and the force tracking error, respectively. The tuning process considers the response speed of the real-world hand actuators and the force tracking performance.

Low-Level Tracking Controller Design

The low-level tracking controller is to realize the contact force command generated by manipulation controller. Feedback control is usually required since the dynamics model is not perfectly known for physical hands due to the backlash, friction or joint flexibility, etc. Depending on the sensor types and motor input, the tracking control may have different forms. In manipulation architecture, common sensor types for force control include force sensors (e.g. ATI F/T sensor nano17), tactile sensors (e.g. BioTac SP or PPS Tactile) on fingertips or the joint torque sensors (e.g. strain gauge or series elastic actuator) on finger joints. Excessive calibration (e.g. friction identification, backlash modeling) is required if no of above sensor types are equipped.

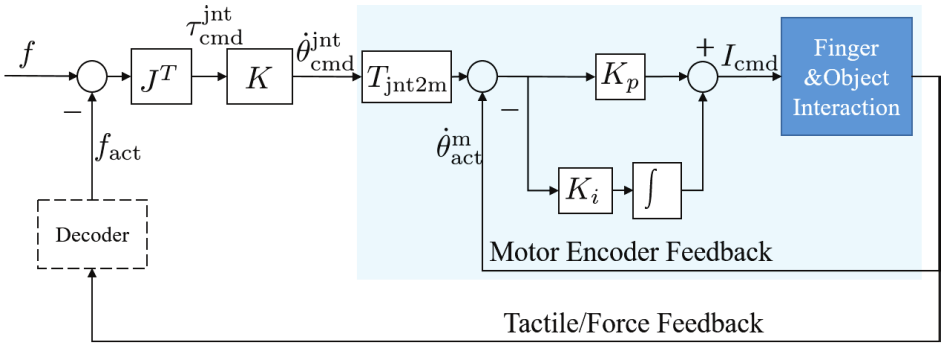


Figure 7.3: Illustration of the low-level force tracking control.

Option 1: Force Tracking Controller

In this section, we assume that the finger equips with force or tactile sensors on fingertips, and the motor takes current signal as input. To track the force command generated by manipulation controller, we design a simple force tracking controller shown in Fig. 7.3.

The force controller contains a force feedback layer with tactile/force sensors and a velocity feedback layer with motor encoders. The force controller is a Proportional control that converts the force error $f - f_{\text{act}}$ to joint velocity command $\dot{\theta}^{\text{jnt}}_{\text{cmd}}$. J^T denotes transpose of the finger Jacobian, and K is the proportional gain. For those finger systems with small damping (rigid fingertips and rigid joints), an additional Derivative channel is required to improve the stability, as shown in Section 8.6 of Chapter 8. The velocity controller is a Proportional-Integral control that converts the joint velocity command $\dot{\theta}^{\text{jnt}}_{\text{cmd}}$ to current command I_{cmd}. T_{jnt2m} denotes a transform from joint side velocity $\dot{\theta}^{\text{jnt}}$ to motor side velocity $\dot{\theta}^{m}$. K_p and K_i denote the proportional and integral gains.

Option 2: Torque Tracking Controller

In this section, we assume that the finger equips with joint torque sensors in each joint, and the motor takes current signal as input. To track the force command generated by manipulation controller, a similar torque tracking controller can be designed. Instead of feeding f_{act}, the torque feedback suppresses torque error $J^T f - \tau_{\text{act}}$ with a P or PD control. The inner layer is a velocity feedback same with Fig. 7.3. The block diagram is neglected for brevity.

7.3 Simulation Study

In this section, simulation results are presented to verify the effectiveness of the proposed manipulation architecture. The simulation video is available at [102].

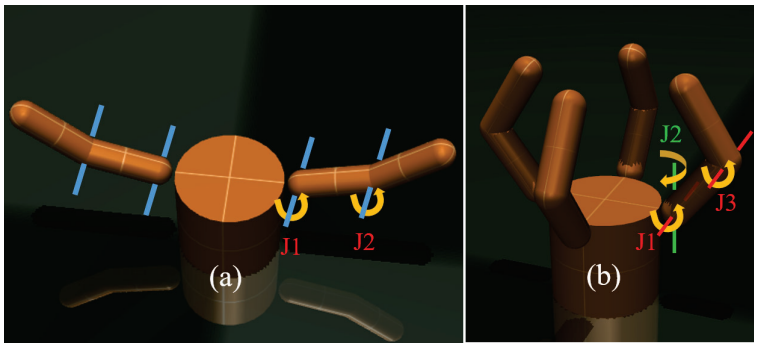


Figure 7.4: Two hands used in the simulation.

Simulation Setup

The controller was implemented in Mujoco physical engine [96]. The simulation time step was set to 2 ms. Our platform was a desktop with 4.0 GHz Intel Core Quad CPU, 32 GB RAM, running Windows 10 operating system.

Two types of hands were used in the simulation, including a two-fingered hand and a four-fingered hand for 2D and 3D manipulation, respectively (Fig. 7.4). For purposes of illustration, a planar hand with two identical fingers and 4 DOFs was set up, as shown in Fig. 7.4(a). The four-fingered hand has four identical fingers and each finger has three revolute joints, as shown in Fig. 7.4(b). Two hands are equipped with high-resolution position sensors for joint position/velocity measurements, motor torque sensors for motor torque feedback, one-dimensional distributive tactile sensors to measure normal force and infer surface normal. The manipulated object is approximately 0.5 kg. The dynamics parameters of the object are assumed to be unknown and estimated by vision sensors.

Comparison with Different Methods

For comparison, the proposed robust controller is compared with a disturbance observer (DOB) approach proposed in [58]. This approach is briefly reviewed in the following. This method assumes the dynamics of the system as $M\ddot{x}_o + C\dot{x}_o + N + d = F$. The dynamics can be rewritten as $\ddot{x}_o = u + w$, where $u = F - \bar{N}$ and $\bar{N}$ denotes nominal gravity value, and $w = \ddot{x}_o - (M\ddot{x}_o + C\dot{x}_o + N - \bar{N} + d)$ is the lumped disturbance. Similar to feedback linearization, the controller is set as $u = \ddot{r} - k_1\dot{e} - k_2 e - \hat{w}$, where $e = x_o - r$. In this manner, the error dynamics are:

$$\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ -k_2 & -k_1 \end{bmatrix}}_{A} \begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \underbrace{\begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}}_{B} e_w$$

where $e_w = w - \hat{w}$, $\hat{w} = \hat{w}_0 + K_o E$, and $\dot{\hat{w}}_0 = B^T P E$. P can be found from $A^T P + PA = -Q$, and K_o can be found from $K_o A + K_o B K_o + B^T P = 0$, with $Q \succ 0$ is a design parameter

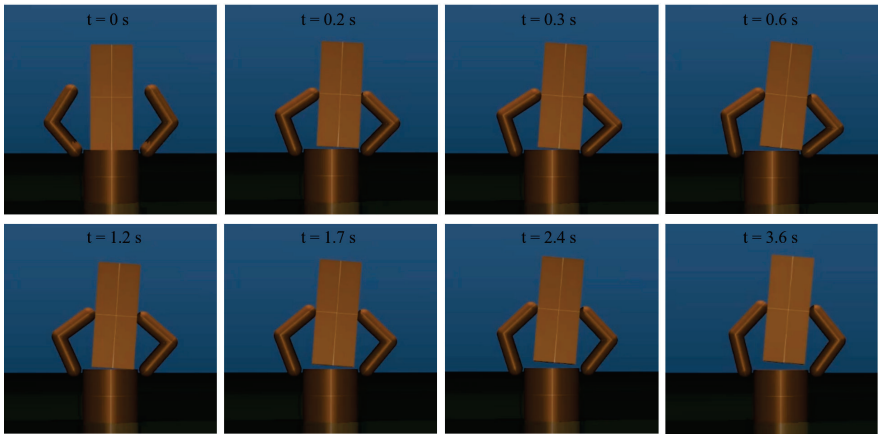


Figure 7.5: Snapshots of the 2D manipulation results with the proposed architecture.

and $E = [e^T, \dot{e}^T]^T$. The desired force on the object is $F = u + \bar{N}$.

Parameter Lists

For manipulation controller, $\alpha_1 = 0.01$, $\alpha_2 = 0.01$ and $\alpha_3 = 1000$. The desired parameter values for MIC in 2D manipulation were: $K^d = 50 \times \text{diag}([1, 1, 0.2])$, $B^d = 5 \times \text{diag}([1, 1, 0.02])$, $I^d = 50 \times \text{diag}([1, 1, 0.2])$. As for MIC in 3D manipulation: $K^d = 50 \times \text{diag}([1, 1, 1, 0.2, 0.2, 0.05])$, $B^d = 5 \times \text{diag}([1, 1, 1, 0.02, 0.02, 0.005])$, and $I^d = 50 \times \text{diag}([1, 1, 1, 0.02, 0.02, 0.005])$. The low-level tracking controller was bypassed due to the perfect modeling in simulation. The torque command $\tau_{cmd}^{jnt} = J^T f$ was feed to finger joints directly in an open-loop manner. The parameters for DOB based control are: $k_2 = 95 \times \text{diag}([1, 1, 0.01])$, $k_1 = 95 \times \text{diag}([1, 1, 0.008])$, and Q matrix is chosen as $Q = \text{diag}([51000, 51000, 6, 51000, 51000, 1])$.

Simulation Results

To simplify the validation process, a 2D manipulation task with the planar hand in Fig. 7.4(a) was first employed to verify the proposed architecture. Figure 7.5 shows the snapshots of the 2D manipulation task. The object was assumed to have 20% mass uncertainty. The simulation constrained the motion of the object into vertical plane, and the desired object motion was to move to (150 mm, −10 mm, 5°) from (139 mm, 0 mm, 0°).

Figure 7.6 compares the proposed modified impedance control with DOB controller on the same task under 20% mass uncertainty. The solid lines are average convergence profiles while the shaded batches are associated variations. In average, the proposed manipulation architecture with MIC converged to the desired pose with settling time 3.14 secs[1]. The convergence in gravitational direction exhibited larger variation for different uncertainties,

[1] 5% threshold was used for all settling time calculations.

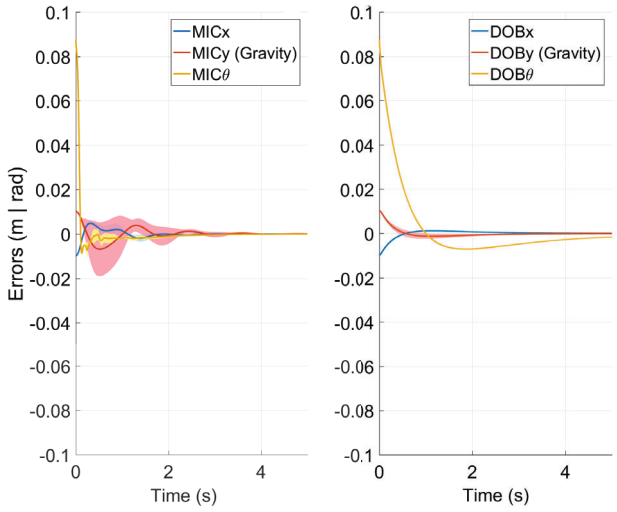


Figure 7.6: Comparison of the object pose tracking error under 20% mass uncertainty with (Left) MIC and (Right) disturbance observer in [58].

since it took longer time for the controller to accumulate force when the nominal gravity is lighter than the actual one. In comparison, the manipulation performance on the same task with a disturbance observer based controller is shown in Fig. 7.6(Right). The average settling time was 3.28 secs and the rotational direction was the critical direction that effects the convergence. The tuning of the parameters is described in [27].

The proposed architecture with MIC can be applied to general 3D manipulation with the four-fingered hand shown in Fig. 7.4(b). Furthermore, the architecture is able to supply as the low-level controller for a higher-level finger gaits planning to achieve the large-scale object motion beyond the hand workspace. The details of the finger gaits planning will be introduced in Chapter 9. The snapshots of the manipulation architecture in a finger gaiting task are shown in Fig. 7.7. The task was to rotate the ellipsoid around the gravitational axis for 180°. Despite the hybrid dynamics and disturbances during contact rebuilding, the MIC based architecture remained stability and fulfilled the manipulation task.

Figure 7.8 shows the performance of the manipulation controller under object dynamics uncertainty and external disturbances. The object is subject to 20% of mass uncertainty and 2N external disturbance. The manipulation controller is capable of driving the object to the desired pose.

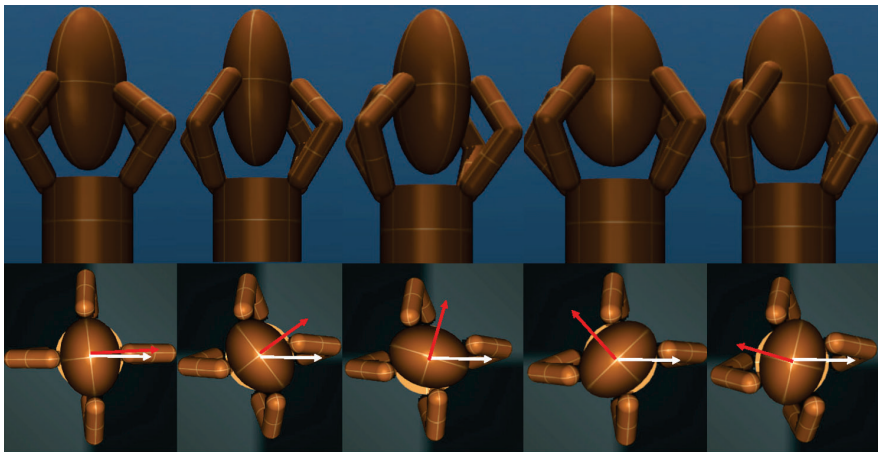


Figure 7.7: Snapshots of a finger gaiting task with the proposed architecture.

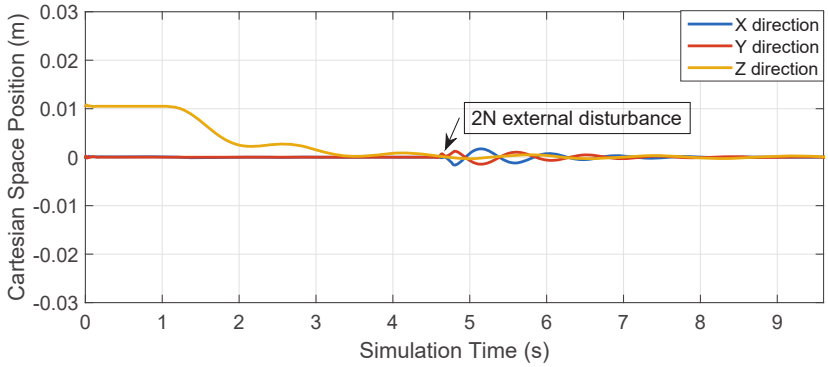


Figure 7.8: Illustration of architecture on dynamic uncertainty and external disturbances.

7.4 Experiment Study

Experiment Setup

This section presents the experimental validation of the proposed architecture. The algorithm was implemented to two physical hands shown in Fig. 7.9. Both hands have 3 fingers and 9 DOFs. Type A hand in Fig. 7.9(Left) equips with joint encoders in each joint and BioTac SP tactile sensors [31] on fingertips. The springs and excessive frictions in the drivetrain introduce extra unmodeled dynamics for precision control [70]. Type B hand in Fig. 7.9(Right) equips with joint encoders in each joint and ATI Nano17 force/torque sensor on each fingertip. This hand cannot provide precise contact location and contact normal information, and the dead zone in the drivetrain introduces extra uncertainties. Figure 7.10 shows the experimental setup. Kinect sensor was used to detect the object pose. The perceived signal was sent to host PC along with tactile/force signals. The host PC computes

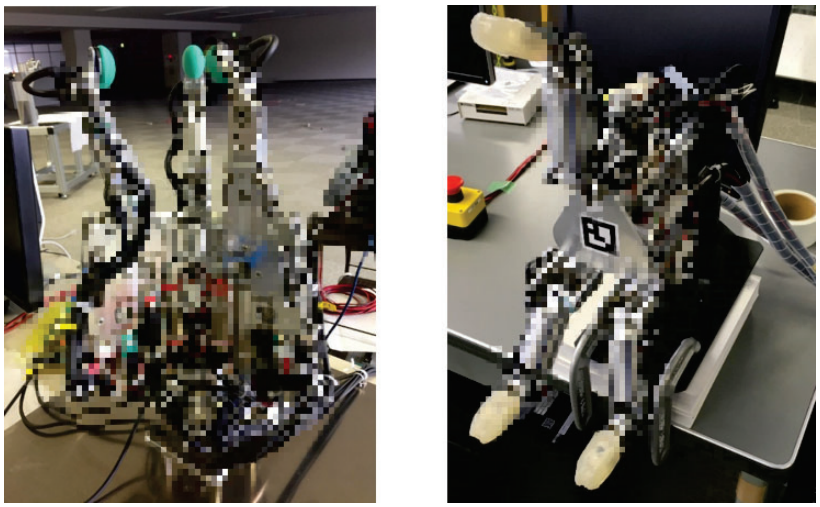


Figure 7.9: (Left) Type A and (Right) Type B hands for in-hand manipulation tasks.

Figure 7.10: Experimental setup for algorithm validation.

the actual object pose x^o_{act} and sends the pose error to the proposed algorithm for desired joint velocity $\dot{\theta}^{\text{m}}_{\text{cmd}}$. The target PC runs a low-level PID controller to track the desired joint velocity $\dot{\theta}^{\text{m}}_{\text{cmd}}$ and sends the torque command to hand motors.

Experimental Results

This section presents the experimental results with the proposed manipulation architecture. Figure 7.11 shows the experimental implementation of the architecture. First, the robust controller took object pose error as input, and produced desired Cartesian force on object (Fig. 7.11(a)). Then the Cartesian object was transformed to contact force on fingertips by the manipulation controller (Fig. 7.11(b)). Since the hands take motor current command as input, a force controller was designed to convert the contact force in manipulation controller into joint velocity (Fig. 7.11(c)). Finally a low-level velocity PID controller was designed to convert the joint velocity into the motor current to drive the hand (Fig. 7.11(d)). The following sections introduce the experimental results of the force controller and MIC.

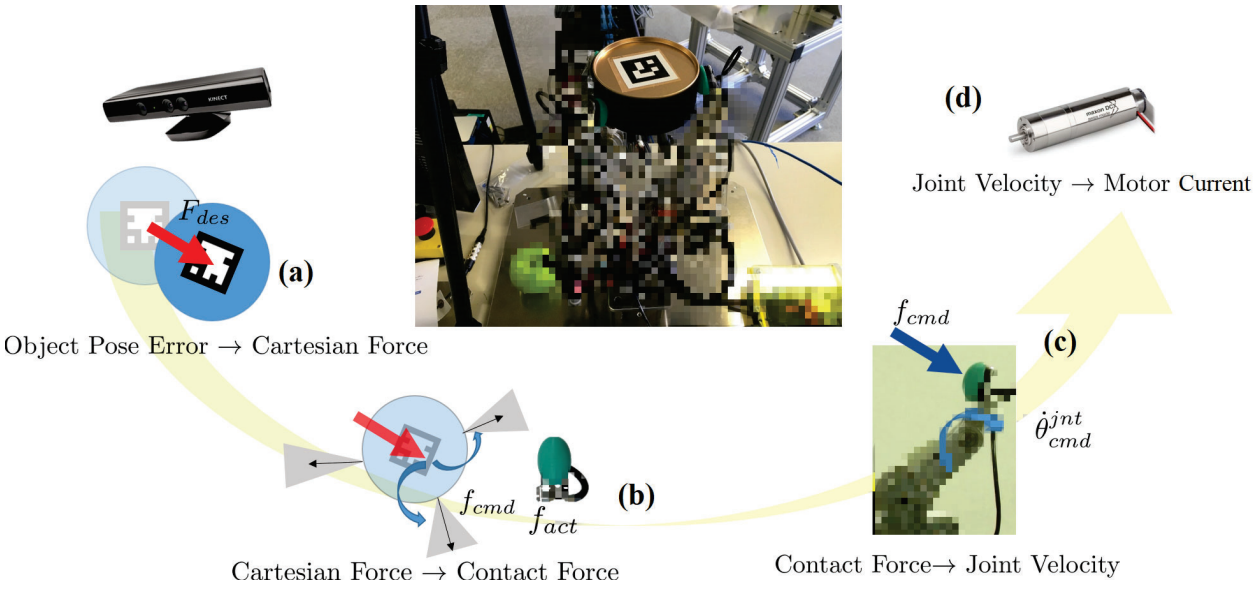


Figure 7.11: Control flow of the manipulation framework.

Force Control

To verify the effectiveness of the manipulation architecture, a low level contact force tracking controller was designed beyond the basic velocity control. To collect the force feedback for type A hand, we first calibrated the BioTac SP sensors. The calibration setup included a BioTac SP tactile sensor and a ATI mini45 force/torque sensor for ground truth force/torque values. The tactile sensor consists of a rigid inner core, deductive liquid and an elastic skin. The rigid core has an array of electrodes to detect 24 channels of impedances. The changes of impedance values are used to encode force/torque, contact locations and vibrations, etc. However, an analytical relationship between the impedance changes and the fingertip force/torque are absent. Therefore, we used a neural network adapted from [90] to model the impedance-force relationship. The contact position/normal was estimated by averaging the positions/normals of the top-4 electrodes with largest impedance changes.

The network we used was a 3-layer neural network with 393 parameters. The network had 24 impedance input channels and 3 force output channels. The torque measure was ignored since we used a point contact with friction model [69]. The network was trained with 1.2 million data collected in 20 mins.

Figure 7.12 shows the calibration results of the BioTac SP sensor. The error histogram indicates that the mean square error of the force is concentrated in $[-0.43 \sim 0.35]$ N, as shown in Fig. 7.12(a). Figure 7.12(b-d) show an example of the actual force and estimated force in all three axes. It is worth noting that the error became larger in sharp transition stage. These large errors caused erroneous force feedback and might affect the stability of the system. The mismeasured portions were regarded as external disturbances and were

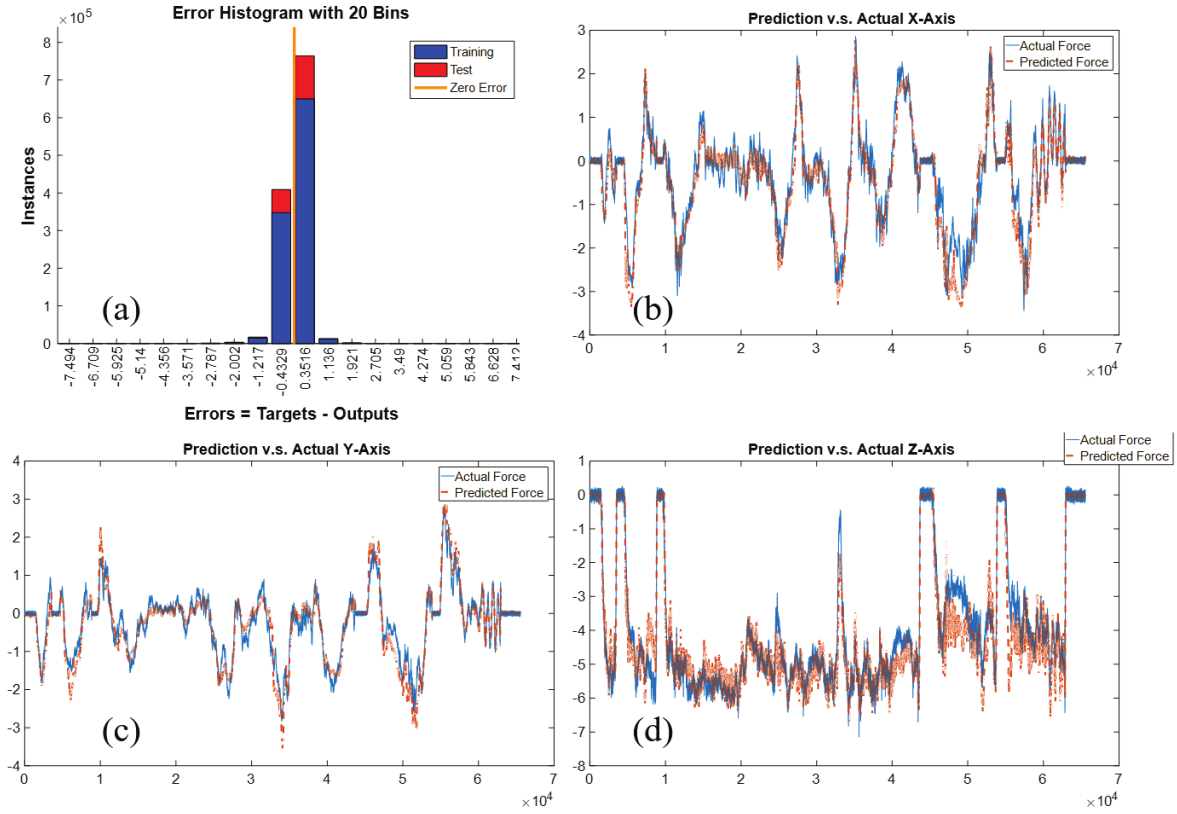


Figure 7.12: Calibration results of BioTac SP sensor.

suppressed by the proposed MIC.

Figure 7.13 shows the force tracking results with force controller in Fig. 7.3. Extra springs in the drivetrain of Type A hand are used to absorb the collision energy and improve the safety ratio during grasping. However, these springs introduce extra dynamics and pose a challenge to manipulation control. To be more specific, the generated velocity first stretches/compresses springs to increase/release potential energy before transmitting the energy to increase/reduce the contact force on fingertips. Consequently, the Type A hand has very small bandwidth, large phase lag and large magnitude loss. The largest bandwidth with acceptable force tracking performance is 0.4 Hz, as shown in Fig. 7.13(a). On the other hand, the Type B hand was able to track a 0.5 Hz sinusoidal force due to its rigid drivetrain and smaller friction, as shown in Fig. 7.13(b). The bandwidth for force control with Type B hand is 6 Hz.

Robust Grasping

In this section we demonstrate the proposed architecture on a robust grasping task. Robust grasping is a special manipulation application where the desired trajectory is degenerated

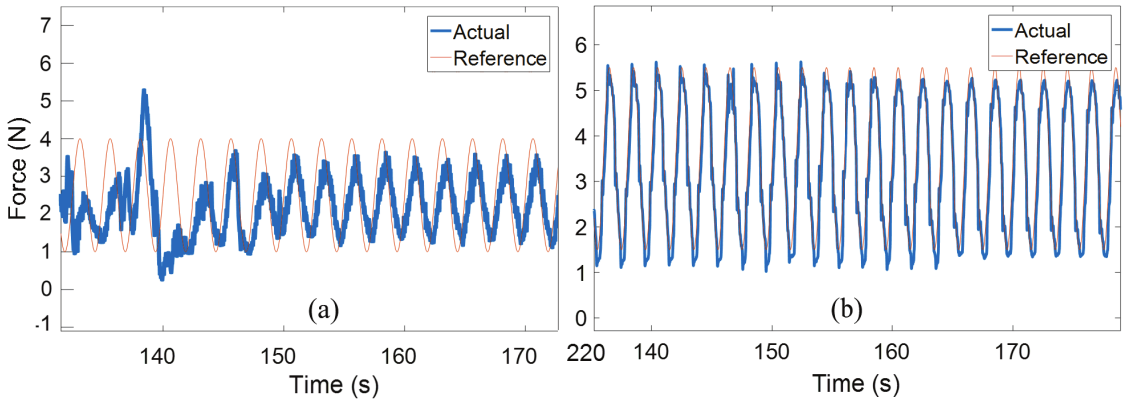


Figure 7.13: Force tracking with (a) Type A hand and (b) Type B hand.

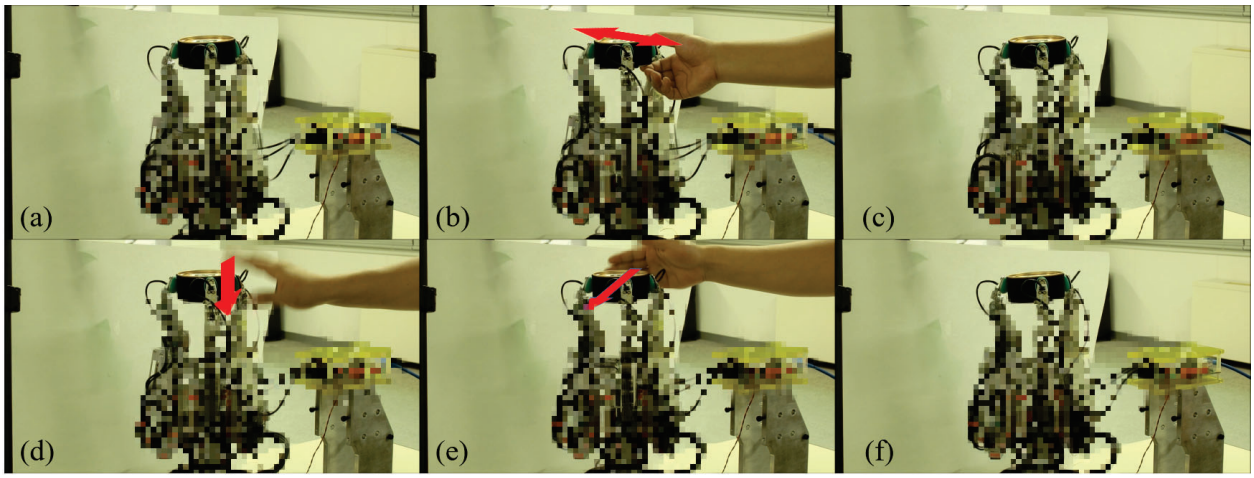


Figure 7.14: Illustration of the robust grasp with Type A hand.

into a point. Any external disturbance should be suppressed by observing the pose error and producing proper force command by MIC.

Figure 7.14 shows the robust grasping results with the Type A hand. The object is a cylinder box with an ArUco marker on the top surface. The object pose was observed by measuring the pose of the ArUco marker and transformed to the estimated center of the object. The external disturbance was exerted by human in several directions. The object was able to roughly maintain the original grasp pose when perturbing the object in different directions. The pose error in Fig. 7.14(f) is caused by 1) force estimation error, 2) finger singularity and 3) pose detection error. Despite the narrow distribution of the force estimation error in Fig. 7.12(a), a continuous force measurement indicates that the force estimation error may be excessively large ($\geq$1N) for several instances in each direction. The

finger singularity is a common problem shared by both types of hands. The singularity will be introduced later.

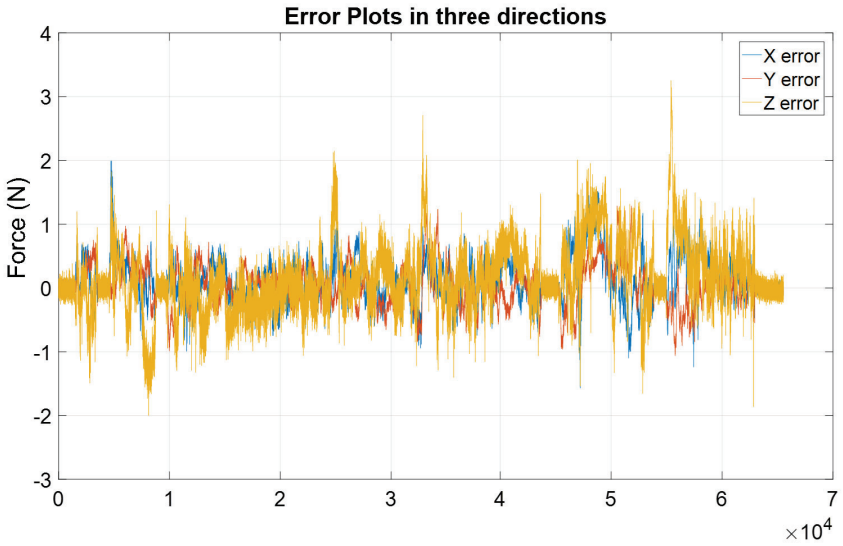


Figure 7.15: Force estimation error with the proposed network on a BioTac SP sensor.

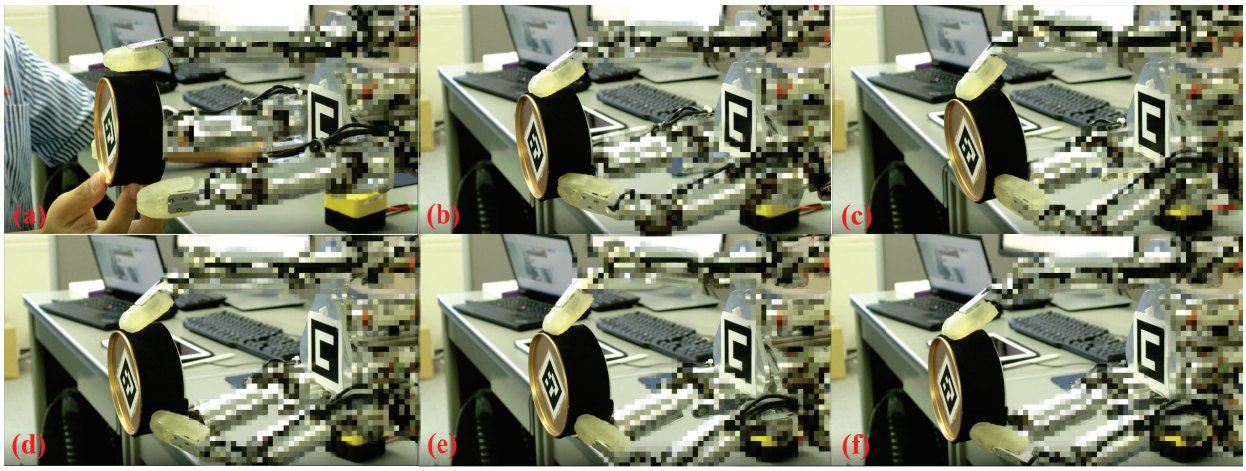


Figure 7.16: Snapshots of the manipulation result following a sinusoidal trajectory.

Dexterous Manipulation

Finally, we demonstrate the proposed architecture on a dexterous manipulation task with the Type B hand. The desired motion was to move sinusoidally around an axis perpendicular

to the palm. Figure 7.16 shows the snapshots of the tracking results. The initial object placement is shown in Fig. 7.16(a). The hand closed fingers and detected contacts based on the ATI nano17 F/T sensor reading. The snapshots in which the object was on its furtherest point are shown in Fig. 7.16(bdf) while the snapshots in which the object was on its closest point are shown in Fig. 7.16(ce). Background object (e.g. iPad) may be regarded as a reference for observing the object motion.

Similarly, there are several factors that affect the performance of the manipulation. These factors include: 1) contact uncertainty, 2) finger singularity, and 3) errors on object pose detection. Instead of estimating the contact position/normal by averaging the top-4 electrodes with the largest impedance change, the Type B hand with ATI nano17 F/T sensors on fingertips cannot provide exact contact positions/normals. Therefore, we roughly estimated the contact position and normal based on the kinematics of each finger.

As for the finger singularity, both types of hands have the same singularities, as shown in Fig. 7.17. The first kind of singularity appears when the contact point falls on the axis of joint 1 (Fig. 7.17(a) red line), as shown in Fig. 7.17(a). The finger with this singularity loses one DOF that moves perpendicular to the triangle formed by the links of the finger. The second kind of singularity appears when the finger reaches its joint limit. This singularity is trivial and can be avoided by [21].

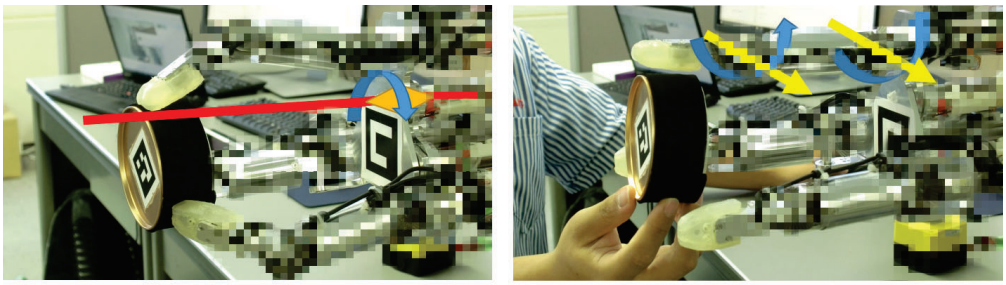


Figure 7.17: Two kinds of singularities of Type A/B hands.

7.5 Chapter Summary

This chapter presented an architecture for robust grasping and dexterous manipulation. The architecture contains a high-level modified impedance controller (MIC) to generate Cartesian force on an object from the pose feedback, a mid-level manipulation controller to produce contact forces on fingertips from the desired Cartesian force, and an optional low-level force tracking controller to execute the generated force command. The high-level MIC is robust to underlying uncertainties and external disturbances, the mid-level manipulation controller avoids slippage and reduces control efforts, and the low-level force controller bypasses the

unmodeled finger dynamics such as friction, joint flexibility and backlash. Both simulations and experiments including two types of hands with different sensors and drivetrains verified the proposed architecture. Preliminary qualitative results of the robust grasping with Type A hand and dexterous manipulation with Type B hand demonstrated the effectiveness of the proposed manipulation architecture. The experimental videos are available at [102].

The manipulation performance with both types of hands is greatly influenced by the finger singularity in Fig. 7.17(a). Under the current design, this singularity is unavoidable, frequently encountered and challenging to escape once being trapped by it. The traditional damped least-squares used in [21] does not work in force domain. Future work includes the investigation of a singularity solution compatible with the force control scheme and quantitative evaluation of the manipulation architecture.

Chapter 8

Robust Dexterous Manipulation under Various Uncertainties

8.1 Introduction

Chapter 7 introduced a comprehensive architecture for dexterous manipulation. The architecture exhibits certain robustness due to the high-level modified impedance controller. This chapter further improves the robustness of the architecture by employing the structure of the uncertainties. In a general manipulation task, the robotic hand usually has to manipulate objects with various types of uncertainties shown in Fig. 8.1. Type I uncertainty is caused by the inaccurate 3D model perception and density distribution of the object. It is separated into object dynamics uncertainty and center of mass (COM) uncertainty, as shown in Fig. 8.1(a) and Fig. 8.1(b). The uncertain terms in object dynamics uncertainty include both mass and moment of inertia, which are parameters of inertia matrix, Coriolis and gravity of the object. The uncertain terms in COM uncertainty include COM position and principal axes of the object, which influence the object dynamics and the force transformation from contact to object. Black frame in Fig. 8.1(b) shows the actual COM and principal axes, and red frames show possible variations. Type II uncertainty is caused by the inaccurate sensing and modeling of contacts under various surface properties. It is separated into contact dynamics uncertainty and tactile uncertainty, as shown in Fig. 8.1(c) and Fig. 8.1(d). The uncertain terms in contact dynamics uncertainty include stiffness, damping, coefficients of Coulomb friction, torsional friction and rolling friction, which influence the force transformed onto the object. The uncertain terms in tactile uncertainty include contact position and orientation, which affect contact force transformation, joint torque generation, and may cause the problems such as friction cone constraints violation and unexpected slippage. Black ones in Fig. 8.1(d) show actual position/normal, while red ones show the variations. Besides the disturbances from the contact dynamics, there might be unexpected contact and external perturbations from the environment. The controller should be designed with disturbance rejection ability.

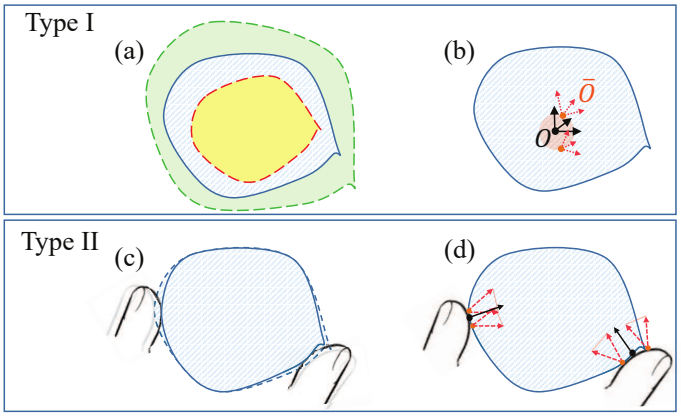


Figure 8.1: Different types of uncertainties in dexterous manipulation.

It is difficult to deal with such uncertainties in dexterous manipulation. First, the object is not directly controlled by actuators. Alternatively, energy is transferred from the fingertips to the object through unknown contact dynamics. Second, the robotic hand for dexterous manipulation can be a high degree-of-freedom (DOF) nonlinear system and can not be directly written into linear time-invariant (LTI) or linear parametric-varying (LPV) form.

A robust controller for contact uncertainties was proposed in [9]. The controller is designed for a LTI system linearized around an equilibrium point. A force-position controller using 6D tactile sensors was implemented to realize adaptive grasping [92]. Nonlinearities were ignored due to its constant-pose grasping property. In order to consider parameter variations caused by nonlinearities, a LPV control with smooth scheduling was applied in [44], assuming that the nonlinearities can be approximated through linear varying parameters. To deal with dynamics uncertainties, a disturbance observer (DOB) was proposed in [58] for tracking control. The nonlinearities and parameter uncertainties are lumped into a disturbance term. It assumes full state feedback, while in dexterous hand, the velocity feedback is difficult due to the size constraints and cost issue. Feedback linearization was applied to control an unmanned aerial vehicle [67]. A linear state observer and a DOB are combined to observe the state and the lumped disturbance. Similar to [58], the structures of the parameter uncertainties are ignored, and the linear state observer assumes a perfect model for state estimation.

Chapter 7 introduced a manipulation architecture with modified impedance controller (MIC) to resist system uncertainties and disturbances. This chapter adopts the architecture and further proposes a robust manipulation controller (RMC) for dexterous manipulation under various uncertainties and external disturbances. Distinctive features of this chapter are as follows. First, the nonlinearities are reduced by feedback linearization on a nominal model. Compared with LPV that assumes linear variations of parameters, the proposed method is more computationally efficient for broad-scale manipulations. Second, the robust controller is formulated as a μ-synthesis problem, and the structures of the uncertainties are considered

by descriptor form, instead of treating uncertainties as a lumped disturbance, which results in information loss and a larger disturbance to resist. Moreover, by the dual-stage formulation, the complicated contact modeling is bypassed, and the contact force is regulated and the slippage is prevented. Finally, RMC does not require velocity measurements of objects/joints.

The remaining of this chapter is organized as follows. Section 8.2 introduces the robust manipulation controller framework. Section 8.3 describes the system dynamics and its combination with the feedback linearization. The robust manipulation controller is presented in Section 8.4. The simulation and experiment results are shown in Section 8.5 and Section 8.6. Section 8.7 summarizes this chapter.

8.2 Robust Manipulation Controller Framework

Figure 8.2 shows the framework of the robust manipulation controller (RMC). In this figure, r, y, n and e denote the reference pose, the actual pose, the measurement noise and the pose error of the object, respectively. The signal u denotes the control input to the linearized plant. The signal u_{dis} is the external disturbance to the plant. F is the desired Cartesian space force on the object. The signal τ is the torque command to the hand in order to realize F. The objective is to: 1) track the desired pose r of the object, 2) be robust to object dynamics uncertainties (i.e. mass and inertia uncertainties) and external disturbances u_{dis}, and 3) realize firm contact without violating the friction cone constraints.

The robust manipulation controller consists of a robust controller and a manipulation controller from Chapter 7, as shown in Fig. 8.2. The robust controller takes e as input, and generate F to the object. It applies on a linearized nominal plant with nonlinear uncertainties. The linearized nominal plant is obtained by feedback linearization on nonlinear dynamics. The F obtained from feedback linearization and robust controller is converted into τ by the manipulation controller.

In robust controller design, the feedback linearization is directly connected to the nonlinear dynamics by the assumption that the actual force on the object after executing τ is close to F. The gap between these two forces is treated as part of u_{dis}.

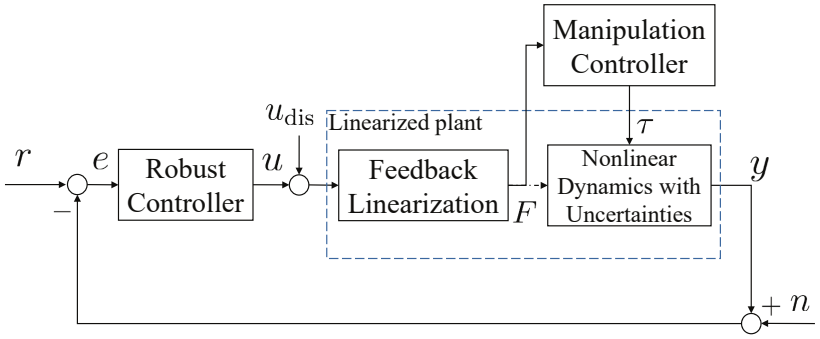


Figure 8.2: The general framework of the proposed robust manipulation controller.

8.3 Modeling of Uncertain Manipulation Dynamics

State-Space Dynamics

The dynamics of the hand and object are restated here for clarity:

$$\begin{aligned} &M_h(q)\ddot{q} + C_h(q,\dot{q})\dot{q} + N_h(q,\dot{q}) + J_h^T(q,x_o)f_c = \tau \\ &M_o(x_o)\ddot{x}_o + C_o(x_o,\dot{x}_o)\dot{x}_o + N_o = G(q,x_o)f_c \end{aligned} \tag{8.1}$$

where $M_{h/o}, C_{h/o}$ and $N_{h/o}$ are inertia matrices, Coriolis matrices and gravities for the hand/object. $q, \dot{q}$ and $\ddot{q} \in \mathbb{R}^{n_q}$ denote the joint angle, velocity and acceleration, with n_q as the total degree of freedoms (DOFs) of the hand. $x_o, \dot{x}_o$ and $\ddot{x}_o \in \mathbb{R}^{n_x}$ are a local parameterization of object position, velocity and acceleration, where n_x is the dimension of the pose of the object, with $n_x = 6$ for 3D manipulation ($n_x = 3$ for 2D manipulation). $f_c \in \mathbb{R}^{d_c n_c}$ and $\tau \in \mathbb{R}^{n_q}$ are contact force vector and joint torque vector, where d_c is the dimension of each contact, and n_c is the contact number. $J_h \in \mathbb{R}^{(d_c n_c)\times n_q}$ is the hand Jacobian and $G \in \mathbb{R}^{n_x \times (d_c n_c)}$ is the grasp map [69].

If the contacts are fixed w.r.t both object and fingertips, then

$$J_h(q,x_o)\dot{q} = G^T(q,x_o)\dot{x}_o \tag{8.2}$$

holds. Equation 8.2 assumes the contact forces remain in the friction cone.

The object and hand dynamics in (8.1) can be connected by (8.2):

$$M(q,x_o)\ddot{x}_o + C(q,\dot{q},x_o,\dot{x}_o)\dot{x}_o + N(q,x_o) = GJ_h^{-T}\tau \tag{8.3}$$

where:

$$\begin{aligned} M &= M_o + GJ_h^{-T}M_hJ_h^{-1}G^T \\ C &= C_o + GJ_h^{-T}C_hJ_h^{-1}G^T + GJ_h^{-T}M_h\frac{d(J_h^{-1}G^T)}{dt} \\ N &= N_o + GJ_h^{-T}N_h \end{aligned} \tag{8.4}$$

In some applications such as fruit harvesting, only the rough values of the mass m_o and the inertia $\mathcal{I}_o$ of the object can be estimated. Therefore, M_o, C_o, N_o cannot be exactly known and would exhibit some uncertainties. Suppose that the inertia, Coriolis and gravity can be represented as:

$$M = \bar{M} + \tilde{M}_o, \quad C = \bar{C} + \tilde{C}_o, \quad N = \bar{N} + \tilde{N}_o \tag{8.5}$$

with nominal values:

$$\begin{aligned} \bar{M} &= \bar{M}_o + GJ_h^{-T}M_hJ_h^{-1}G^T \\ \bar{C} &= \bar{C}_o + GJ_h^{-T}C_hJ_h^{-1}G^T + GJ_h^{-T}M_h\frac{d(J_h^{-1}G^T)}{dt} \\ \bar{N} &= \bar{N}_o + GJ_h^{-T}N_h \end{aligned}$$

where $\bar{M}_o, \bar{C}_o, \bar{N}_o$ are nominal object inertia, Coriolis, gravity, and $\tilde{M}_o, \tilde{C}_o, \tilde{N}_o$ are corresponding uncertainties. The torque command τ can be related to the object-centered force F:

$$\tau = J_h^T (G^\dagger F + N_G \lambda) \tag{8.6}$$

where N_G is the matrix composed by the basis of the null space of G, and λ is a free variable to control the magnitude and direction of the contact force.

The state space equation can be derived by plugging (8.5) and (8.6) into (8.3):

$$\left(\underbrace{\begin{bmatrix} \mathbb{I} & \mathbb{O} \\ \mathbb{O} & \bar{M} \end{bmatrix}}_{\bar{M}_{\text{aug}}} + \underbrace{\begin{bmatrix} \mathbb{O} & \mathbb{O} \\ \mathbb{O} & \tilde{M}_o \end{bmatrix}}_{\tilde{M}_{\text{aug}}} \right) \underbrace{\begin{bmatrix} \dot{x}_o \\ \ddot{x}_o \end{bmatrix}}_{\dot{x}} + \left(\underbrace{\begin{bmatrix} \mathbb{O} \\ \bar{N} \end{bmatrix}}_{\bar{N}_{\text{aug}}} + \underbrace{\begin{bmatrix} \mathbb{O} \\ \tilde{N}_o \end{bmatrix}}_{\tilde{N}_{\text{aug}}} \right) + \left(\underbrace{\begin{bmatrix} \mathbb{O} & -\mathbb{I} \\ \mathbb{O} & \bar{C} \end{bmatrix}}_{\bar{C}_{\text{aug}}} + \underbrace{\begin{bmatrix} \mathbb{O} & \mathbb{O} \\ \mathbb{O} & \tilde{C}_o \end{bmatrix}}_{\tilde{C}_{\text{aug}}} \right) \underbrace{\begin{bmatrix} x_o \\ \dot{x}_o \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} \mathbb{O} \\ \mathbb{I} \end{bmatrix}}_{B_F} F \tag{8.7}$$

where $\mathbb{I}, \mathbb{O} \in \mathbb{R}^{n_x \times n_x}$. Equation (8.7) can be rewritten as:

$$\dot{x} = -\bar{M}_{\text{aug}}^{-1} \bar{C}_{\text{aug}} x - \bar{M}_{\text{aug}}^{-1} \bar{N}_{\text{aug}} + \bar{M}_{\text{aug}}^{-1} B_F F - \bar{M}_{\text{aug}}^{-1} \tilde{M}_{\text{aug}} \dot{x} - \bar{M}_{\text{aug}}^{-1} \tilde{C}_{\text{aug}} x - \bar{M}_{\text{aug}}^{-1} \tilde{N}_{\text{aug}} \tag{8.8}$$

In 3D manipulation, the parameters of (8.8) can be decomposed as:

$$-\bar{M}_{\text{aug}}^{-1} \tilde{M}_{\text{aug}} = L_1 \mathbf{\Delta} R_1, \quad -\bar{M}_{\text{aug}}^{-1} \tilde{C}_{\text{aug}} = \sum_{j=1}^{2} L_{2j} \mathbf{\Delta} R_{2j} \tag{8.9}$$

when parameterizing the rotation matrix R of the object by Z-Y-X Euler angles E, with

$$\begin{aligned}
&L_1 = L_{21} = [0_{6\times 6}; \bar{M}^{-1}] \times \text{diag}(I_{3\times 3}, Q_E^T) \\
&\mathbf{\Delta} = \text{diag}(\delta_m I_{3\times 3}, \delta_{\mathcal{I}_1}, ...\delta_{\mathcal{I}_3}) \quad \text{with} \|\mathbf{\Delta}\|_\infty \leq 1 \\
&R_1 = -\text{diag}(\Delta m I_{3\times 3}, \Delta \mathcal{I}) \times [0_{6\times 6}, \text{diag}(I_{3\times 3}, Q_E)] \\
&R_{21} = -\text{diag}(\Delta m I_{3\times 3}, \Delta \mathcal{I}) \times [0_{6\times 6}, \text{diag}(0_{3\times 3}, \dot{Q}_E)] \\
&L_{22} = [0_{6\times 6}; \bar{M}^{-1}] \times \text{diag}(I_{3\times 3}, R(Q_E \dot{E})\hat{}) \\
&R_{22} = -\text{diag}(\Delta m I_{3\times 3}, \Delta \mathcal{I}) \times [0_{6\times 6}, \text{diag}(0_{3\times 3}, Q_E)]
\end{aligned}$$

where $\Delta m \in \mathbb{R}$ and $\Delta \mathcal{I} = \text{diag}(\Delta \mathcal{I}_1, ...\Delta \mathcal{I}_3)$ are the maximal mass and inertia uncertainties, $Q_E \in \mathbb{R}^{3\times 3}$ is a Jacobian matrix from Euler angle rate $\dot{E}$ to angular velocity of the object in body frame, and $(\bullet)\hat{}$ denotes the matrix representation of cross product.

With (8.9), the uncertainty term $-M_{\text{aug}}^{-1}(\tilde{M}_{\text{aug}} \dot{x} + \tilde{C}_{\text{aug}} x)$ in (8.8) can be represented by:

$$L_1 \mathbf{\Delta} \underbrace{(R_1 \dot{x} + R_{21} x)}_{z_1} + L_{22} \mathbf{\Delta} \underbrace{R_{22} x}_{z_2} = L_1 \underbrace{\mathbf{\Delta} z_1}_{w_1} + L_{22} \underbrace{\mathbf{\Delta} z_2}_{w_2} = L_1 w_1 + L_{22} w_2 \tag{8.10}$$

2D manipulation is used for illustration and comparison purpose. The Coriolis uncertainty can be eliminated by choosing the local parameterization as body frame translation

and rotation angle. Thus L_{21}, R_{21} and L_{22}, R_{22} are removed, and

$$\begin{aligned} L_1 &= [0_{3\times3}; \bar{M}^{-1}] \\ \mathbf{\Delta} &= \text{diag}(\delta_m I_{2\times2}, \delta_{\mathcal{I}_3}) \quad \text{with} \|\mathbf{\Delta}\|_\infty \leq 1 \\ R_1 &= [0_{3\times3}, -\text{diag}(\Delta m I_{2\times2}, \Delta\mathcal{I}_3)] \end{aligned} \tag{8.11}$$

In general 3D manipulation, the Coriolis term is typically ignored due to the low-speed operation condition, as shown in [51].

The control input u is F, and the augmented gravity $\bar{N}_{\text{aug}}$ can be compensated by an additional control input $u_0 = \bar{N}_{\text{aug}}$. The gravity uncertainty $\tilde{N}_{\text{aug}}$ is considered as part of the disturbance u_{dis}. Then the uncertain state space model is represented as:

$$\begin{aligned} \dot{x} &= \underbrace{-\bar{M}_{\text{aug}}^{-1}\bar{C}_{\text{aug}}}_{A} x + \underbrace{L_1}_{B_1} w_1 + \underbrace{\bar{M}_{\text{aug}}^{-1}B_F}_{B_2}(u - u_0 + u_{\text{dis}}) \\ z_1 &= C_1 x + \underbrace{R_1 L_1}_{D_{11}} w_1 + \underbrace{R_1\bar{M}_{\text{aug}}^{-1}B_F}_{D_{12}}(u - u_0 + u_{\text{dis}}) \\ y &= \underbrace{[I_{3\times3}, 0_{3\times3}]}_{C_2} x \qquad w_1 = \mathbf{\Delta} z_1 \end{aligned} \tag{8.12}$$

where $C_1 = -R_1\bar{M}_{\text{aug}}^{-1}\bar{C}_{\text{aug}}$. Equation (8.12) describes uncertainties by linear fractional transformation (LFT). Notice though the system is nonlinear, due to the state dependencies of the dynamics parameters.

Combining Feedback Linearization with Modeling

A challenge in robust control is the implementation on nonlinear systems. Although some extensions have been done for LPV systems, the application of robust control to a general nonlinear system is still challenging.

To reduce the influence of nonlinearities, feedback linearization is applied to linearize the model. More specifically, for (8.8), the command force may be:

$$F = \left(\bar{M}_{\text{aug}}^{-1}B_F\right)^\dagger \left[\bar{M}_{\text{aug}}^{-1}\bar{C}_{\text{aug}}x + \bar{M}_{\text{aug}}^{-1}\bar{N}_{\text{aug}} + B_F u\right] \tag{8.13}$$

Notice $\left(\bar{M}_{\text{aug}}^{-1}B_F\right)\left(\bar{M}_{\text{aug}}^{-1}B_F\right)^\dagger = [0_{3\times3}, 0_{3\times3}; 0_{3\times3}, I_{3\times3}]$, rather than identity. Therefore, (8.8) becomes:

$$\begin{aligned} \dot{x} &= Ax + B_F u - \bar{M}_{\text{aug}}^{-1}\tilde{M}_{\text{aug}}\dot{x} - \bar{M}_{\text{aug}}^{-1}\tilde{C}_{\text{aug}}x - \bar{M}_{\text{aug}}^{-1}\tilde{N}_{\text{aug}} \\ A &= \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \end{aligned} \tag{8.14}$$

Following the similar procedure as (8.9) and (8.11):

$$\begin{aligned} \dot{x} &= Ax + L_1 w + B_F(u + d) \\ z_1 &= R_1 Ax + R_1 L_1 w_1 + R_1 B_F(u + d) \\ y &= [I_{3\times3}, 0_{3\times3}]x \qquad w_1 = \mathbf{\Delta} z_1 \end{aligned} \tag{8.15}$$

The model would be a LTI system if there were no uncertainties. However, due to the parametric uncertainties, the feedback linearization based on nominal parameters will not be able to eliminate all the nonlinearities. Therefore, the remaining nonlinear uncertainties after feedback linearization are approximated by a LTI system evaluated around an equilibrium point. The resultant system has the same form as (8.15), except that the L_1 and R_1 are evaluated at the equilibrium point. The feasibility of this approximation is validated in Section 8.5.

The linearized plant described by (8.15) is controllable and observable. The robust controller will be designed based on this linearized plant.

8.4 Robust Manipulation Controller Design

Robust manipulation controller consists of a robust controller and a manipulation controller. This section presents the design process of the robust controller. The introduction of manipulation controller can be found in Chapter 7.

Design Scheme

Robust controller is to obtain desired Cartesian force of the object for motion tracking with guaranteed robust stability and performance. The generalized plant P_{general} that the robust controller will work on is shown in Fig. 8.3. G_{NL} and $\mathbf{\Delta}$ define an upper LFT w.r.t. $\mathbf{\Delta}$ (denoted as $F_u(G_{NL}, \mathbf{\Delta})$) to represent the nonlinear uncertain dynamics, as shown in red dash box. The feedback linearization described by $\alpha(x) + \beta(x)u$ is connected with the nonlinear uncertain plant to linearize the nominal model, as shown in the blue dash-dot box. Equation (8.15) is the combination of two boxes.

The inputs to the generalized plant P_{general} are $\{r, u_{\text{dis}}, n, u\}$, which denote the pose reference, the input disturbance, the noise and the control input to the plant. The outputs of the plant are $\{w_{\text{perf}}, w_u, e\}$, which denote the tracking performance, the action magnitude and the pose error. W_{perf} is to suppress tracking errors at different frequencies. W_u is to regulate the control input. W_{dis} is to shape the input disturbance. W_n is to shape the measurement noise. The structures of the weighting functions will be described in Section 8.4.

The connection between the generalized plant P_{general} and the controller K is described by Fig. 8.4. P_{general} and K define a lower LFT w.r.t. K as $F_l(P_{\text{general}}, K)$, to denote the closed-loop system. The closed-loop system concatenates the inputs $\{r, u_{\text{dis}}, n\}$ as $\mathbf{d}$ and the outputs $\{w_{\text{perf}}, w_u\}$ as $\mathbf{e}$. The objective of the robust controller design is to synthesize K to

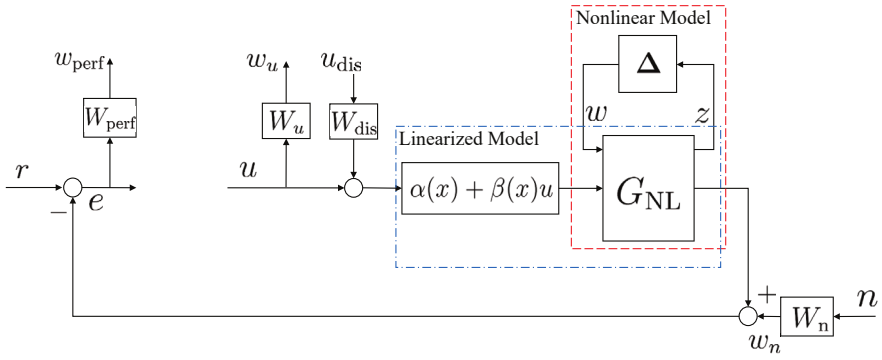


Figure 8.3: Generalized plant with weighting functions.

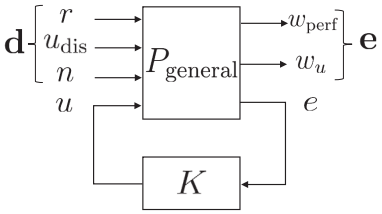


Figure 8.4: Illustration of the closed-loop system.

keep $\mathbf{e}$ small for all reasonable $\mathbf{d}$. The small is in the sense of infinity norm, i.e.

$$\begin{aligned} &K = \underset{K}{\operatorname{argmin}} \|F_L(P_{\text{general}}, K)\|_\infty \\ &\text{with:} \\ &\mathbf{e} = F_L(P_{\text{general}}, K)\mathbf{d} \\ &\|F_L\|_\infty := \max_{\omega \in \mathbb{R}} \bar{\sigma}(F_L(j\omega)) \end{aligned} \tag{8.16}$$

The D-K iteration is applied to solve (8.16):

$$\min_K \inf_D \|DF_L(P_{\text{general}}, K)D^{-1}\|_\infty < 1 \tag{8.17}$$

Readers can refer [4] for implementation details.

The designed controller K will be used to calculate u based on the pose error e. Then the output of the controller is combined with feedback linearization (8.13) to obtain the desired Cartesian space force F for the object.

Design of Weighting Functions

The general form of a weighting function $W(s)$ in P_{general} can be written as:

$$W(s) = \text{diag}([a_1 W_{1,1}(s), a_2 W_{2,2}(s), a_3 W_{3,3}(s)])$$

where a_i is the weight to the i-th channel. $W_{i,i}$ is a SISO filter determined by high-frequency gain G_h, low-frequency gain G_l, cross-over frequency ω_c, and order n. In this section, the principle for choosing parameters is introduced. The concrete values for these parameters will be shown in Section 8.5.

Design of Performance Weighting Function $W_{\mathbf{perf}}$

W_{perf} penalizes the tracking error caused by the general disturbance $\mathbf{d}$. High cross-over frequency w_c penalizes the disturbance with large bandwidth. With larger w_c, the system takes shorter time to settle down, and the desired force tends to change at higher frequencies. Consequently, the error oscillates at higher frequencies. The low-frequency gain G_l penalizes the magnitude of low-frequency disturbance. When G_l is very small, the low-frequency error is large, but the high-frequency error is small, which means that the system takes shorter time to converge. On the other hand, the high-frequency gain G_h penalizes the magnitude of high-frequency disturbances. Increasing G_h will speed-up the convergence. However, the oscillation will be enlarged, and the low-frequency performance will be compromised. As for n, large order n makes the system have more freedom to choose the best controller, while an excessive large n increases the order of the controller. The motivation for tuning a_i is the fact that the behavior in translation directions and rotation direction are usually different because of different parameter scales.

Design of Action Weighting Function W_u

The actions at different frequencies are penalized equally. This is a special case when $G_l = G_h$, which means the weighting function is a constant. Similar as before, large $G_{l/h}$ penalizes the magnitude of action. A larger $G_{l/h}$ results in more penalty to control effort, thus the force generated by the controller is smaller. The smaller force can result in slower convergence speed and poor disturbance rejection. On the contrary, a small $G_{l/h}$ can make the controller generate excessive large force and damage the object. The influences of a_i and n can be reflected into changing $G_{l/h}$.

Design of Disturbance Weighting Function $W_{\mathbf{dis}}$

The disturbance weighting function is used to shape the exogenous disturbance in the generalized plant P_{general}. The cross-over frequency ω_c indicates the shaping bandwidth. Generally, it enlarges the magnitude of low-frequency disturbances and shrinks the magnitude of high-frequency disturbances. A large G_l will create a virtual disturbance with large low-frequency gain. Therefore, the controller would concentrate on reducing the low-frequency disturbance. In our implementation, the gravity is treated as static disturbance. Therefore, increasing G_l makes the actual system response faster by using the larger control effort. The high-frequency gain G_h specifies the shaping factor to high-frequency disturbances. A large value makes the system consider the disturbance rejection in full scale, and the low-frequency dis-

turbance response will be compromised. Similar with W_{perf}, a_i specifies the scales of shaping for different channels, and n specifies the freedom of designing W_{dis}.

Design of Noise Weighting Function W_n

The W_n is designed to be a high-pass filter to shape the noise to the generalized plant P_{general}. The reasons are twofolds. First, the vision sensor used for object pose detection has high-frequency noises. Second, the manipulation controller used for desired force approximation is a low-pass filter, which may result in additional high-frequency approximation error. The tuning of noise weighting is similar with disturbance weighting tuning.

8.5 Simulation Study

Simulation Setup

The controller was implemented in the Mujoco physics engine [96]. The simulation time step was set to be 2 ms. Our platform was a desktop with 4.0 GHz Intel Core Quad CPU, 32 GB RAM, and running Windows 10 operating system.

The hand models used in the simulation are shown in Fig. 8.5. The general hand model used in 3D manipulation is shown in Fig. 8.5(a). It has four identical fingers and 12 DOFs. Each finger has three revolute joints. For illustration purpose, a planar hand with two identical fingers and 4 DOFs was set up, as shown in Fig. 8.5(b). Two hands are equipped with joint encoders, motor torque sensors, and one-dimensional distributive tactile sensors. The manipulated object is approximately 0.5 kg. To mimic an actual real-world finger, the density of each finger link is set to 10000 Kg/m^3. The manipulated objects for four-finger hand and three-finger hand are approximately 0.5 Kg and 0.3 Kg. The 3D mesh models of objects are unknown to the planner. Rather, a vision system can be employed to obtain the pose of the object by tracking the features on it. Currently, the object pose is obtained from Mujoco. In future real world experiments, approach in [59] can be employed to estimate the pose of objects.

Parameter Lists

The parameters of the weighting functions in Section 8.4 are shown in Table 8.1:

Table 8.1: Parameters of Weighting Functions

Weightings	ω_c	G_l	G_h	(a_1, a_2, a_3)	n
W_{perf}	2π	1100	0.9	(1,1,2)	2
W_u	N/A	0.0001	0.0001	(1,1,0.5)	1
W_{dis}	200π	80	0.1	(1,1,10)	2
W_n	20π	0.1	10	(1,1,1)	1

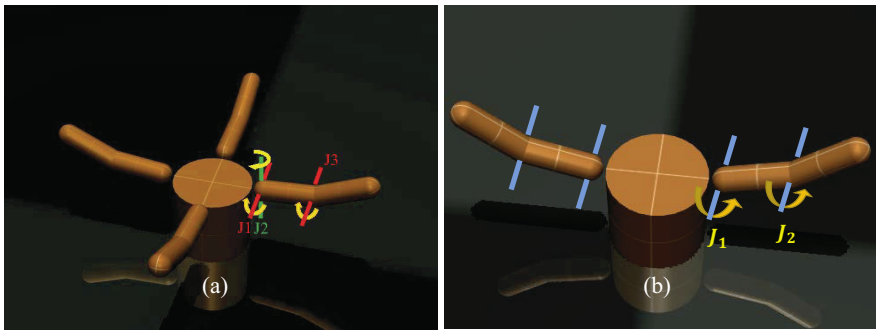


Figure 8.5: Two hand models used in the simulation.

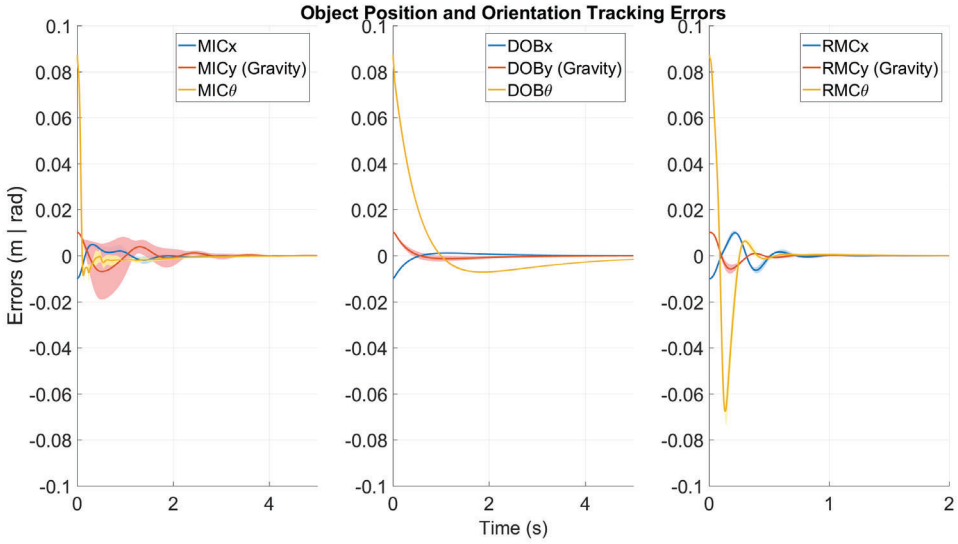


Figure 8.6: Comparison of the proposed RMC with the DOB and MIC.

The parameters of manipulation controller were set as follows: the joint torques were constrained by $\tau_{\min} = -0.5$ Nm and $\tau_{\max} = 0.5$ Nm. 0.5 Nm is twice the joint torque in static case. The weights for different cost terms in (7.10) were $\alpha_1 = 0.1, \alpha_2 = 0.1, \alpha_3 = 1000$. The dimensionality of the contact space in the simulation was set as 6, with sliding, torsional and rolling friction coefficients 1, 0.005 and 0.0001, respectively. In manipulation controller design, we used the point contact with friction model [69] and assumed a conservative sliding friction coefficient 0.5774. The values associated with various uncertainties mentioned in Fig. 8.1 would be introduced below. $f_{\min} = 2$ N and $f_{\max} = 20$ N.

Simulation Results

Comparison with Different Methods

To simplify comparison, a 2D manipulation task is first employed to compare the proposed RMC with other methods. The desired object motion is to move to (150 mm, −10 mm, 5°) from (139 mm, 0 mm, 0°). The equilibrium point is chosen as the configuration upon contact, which can be planned by grasp planning. In this chapter, the equilibrium point is prerecorded for simplicity, and the nominal parameters required for modeling can be calculated accordingly.

The controller was designed to be robust to 40% mass and 50% moment of inertia uncertainties. The resultant robust stability margin was 1.73, which means that the system could withstand about 73% more uncertainties than were specified in the uncertain elements without going unstable. The proposed method is compared with the modified impedance control (MIC) from Chapter 7 and the disturbance observer (DOB) based tracking in [58], as shown in Fig. 8.6. All these three methods assume the variations of mass and moment of inertia are within 20% and 50%, respectively.

The object motion tracking result using MIC is shown in Fig. 8.6(a). The solid lines are average convergence profiles while the shaded batches are associated variations[1]. The average settling time[2] for different uncertainties is 3.14 secs. The convergence in gravitational direction exhibits larger variation for different uncertainties, since it takes longer time accumulate force when the belief gravity is lighter than actual gravity.

The convergence behavior using DOB is shown in Fig. 8.6(b). The average settling time is 3.28 secs and the rotational direction is the critical direction that effects the convergence. The tuning of the parameters are described in [27]. The object pose tracking using the proposed RMC is shown in Fig. 8.6(c). The average settling time is 0.72 sec. Consequently, the oscillation is large compared with other methods. The error profiles have small variations under different mass and moment of inertia uncertainties.

The feedback linearization is applied in Section 8.3 to eliminate the nonlinearities of the nominal system. However, the nonlinearities still exist in the uncertain terms, as shown in (8.14). The remaining uncertainties are approximated as linear by evaluating parameters at an equilibrium point. The error introduced by this approximation could be treated as a disturbance and is called the disturbance from LTI approximation d_{LTI}:

$$d_{\text{LTI}} = (I - \bar{M}_{\text{eq}}\bar{M}^{-1})(\tilde{M}_o\ddot{x}_o + \tilde{N}_o) - \bar{M}_{\text{eq}}\bar{M}^{-1}\tilde{C}_o\dot{x}_o \tag{8.18}$$

where $\bar{M}_{\text{eq}}$ is the nominal inertia matrix at the equilibrium point. The magnitudes of d_{LTI} in both time and frequency domain are shown in Fig. 8.7. The magnitude and spectrum of d_{LTI} are shown in Fig. 8.7. The disturbance caused by LTI approximation has small magnitude compared with disturbance rejection introduced later, and mainly lies in low-frequency region(<12 Hz), thus can be suppressed by the proposed robust controller.

[1] ± standard derivation is used as the boundary of variation

[2] 5% threshold is used for all settling time calculations.

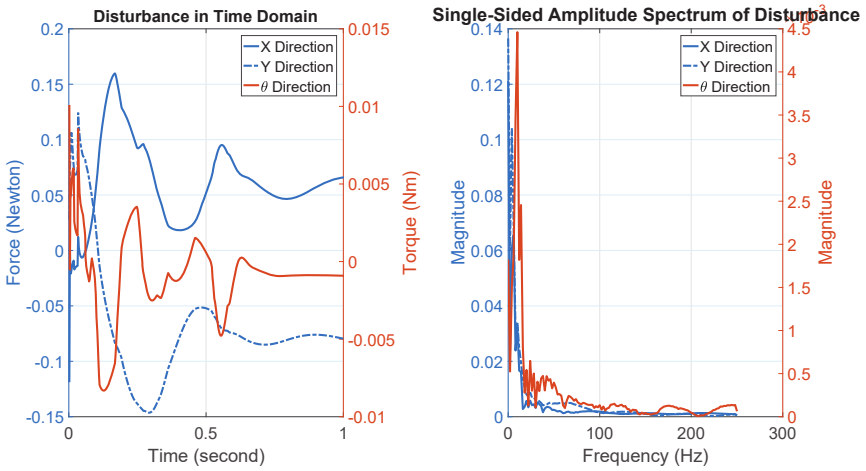


Figure 8.7: Disturbance caused by LTI approximation.

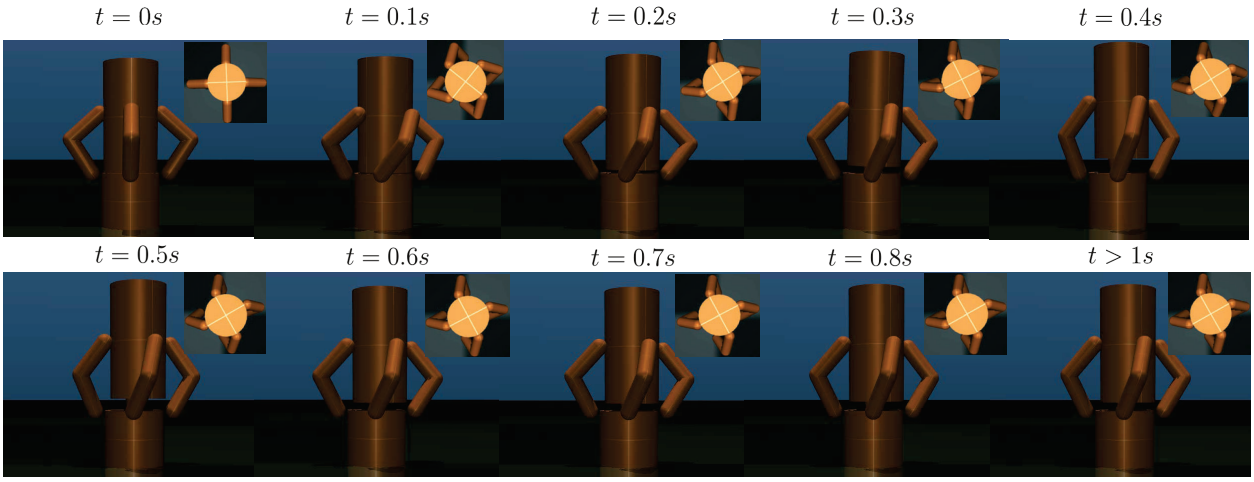


Figure 8.8: Performance of 3D manipulation using the proposed algorithm.

Robustness to Uncertainties

A general 3D manipulation task using four-finger hand is presented to demonstrate the robustness of the proposed RMC to various uncertainties. The desired object displacements are $(4, 4, 11)$ mm, $(0, 0, 0.5)$ rad. The velocity measurement is not required since the Coriolis force in (8.3) is neglected. The snapshot of this type of manipulation is presented in Fig. 8.8. The object is a cylinder with mass 0.535 Kg and radius 35 mm. The proposed RMC is robust to object dynamics uncertainties. The robust controller was designed to be robust to 40% mass and 50% moment of inertia uncertainties, while the mass and moment of inertia uncertainties in the controller vary from $\{-40\%, -80\%\}$ to $\{+50\%, +80\%\}$ of their true

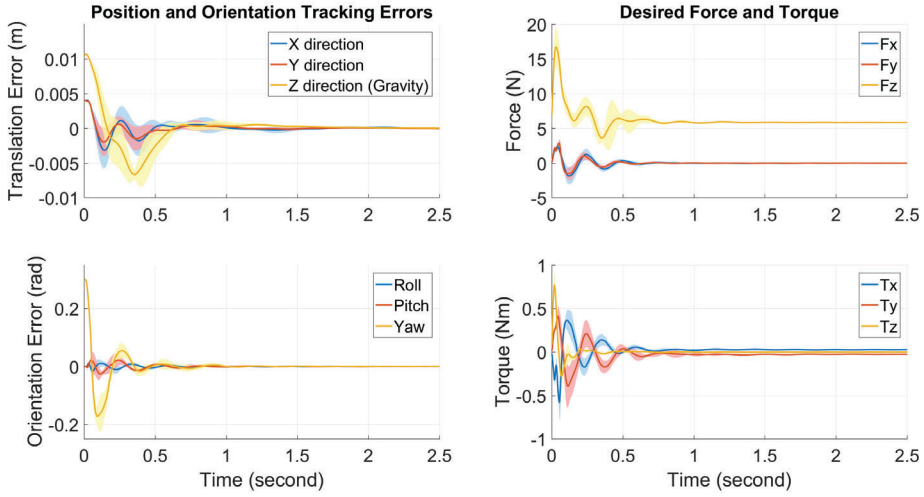


Figure 8.9: The pose tracking errors under different mass and moment of inertia uncertainties.

values. The influence of these parameter variations is described in Appendix A.1. The tracking errors and the corresponding desired force on object of RMC with the sampled uncertainties are shown in Fig. 8.9(Left) and Fig. 8.9(Right). Similar as before, the solid lines represent the average values and the shaded patches represent the variations for different uncertainties. The average settling time for all the samples are 1.31 secs, while the largest settling time 2.25 secs appears in x-direction when the uncertainties are $\{-40\%, -80\%\}$.

In addition, the proposed RMC is also robust to COM uncertainty. The maximum position deviation from the actual COM of the object is $\pm\{10, 10, 15\}$ mm, while the maximum orientation derivation from the actual principal axes of the object is $\pm\{0.3, 0.3, 0.3\}$ rad in ZYX Euler angle. The influence of these parameter variations is shown in Appendix A.1. The object has 20% mass and 50% moment of inertia uncertainties at the same time. The tracking errors and the associated desired forces on the object are shown in Fig. 8.10(Left) and Fig. 8.10(Right). The average settling time for all the samples are 2.13 secs, while the largest settling time 2.46 secs appears in x-direction when the uncertainties are $\{5, 5, -15\}$ mm and $\{0.3, 0.3, 0.3\}$ rad. The Cartesian force converges to effective gravitational force to compensate the gravity for the composite system.

Thirdly, we demonstrate the robustness of the RMC to contact dynamics uncertainties. In Mujoco, the reference acceleration a_{ref} after contact is modeled by a virtual spring with stiffness and damping $\{k_c, b_c\}$, with $a_{\text{ref}} = -k_c r_c - b_c v_c$, where r_c, v_c are residual and velocity, and the implemented acceleration a_{imp} is interpolated by $a_{\text{imp}} = d a_{\text{ref}} + (1-d)a_0$, where d is an interpolation factor and a_0 is the acceleration in absence of constraint. In the simulation, $\{k_c, b_c\}$ vary from $\{4440.9, 133.3\}$ to $\{63131.5, 505.2\}$, and the object has 20% mass and 50% moment of inertia uncertainties in the meantime. The tracking errors and the corresponding

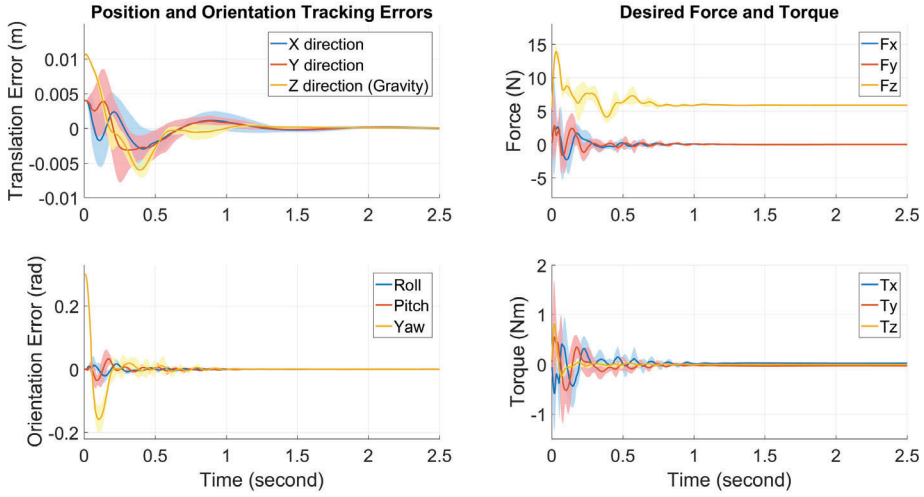


Figure 8.10: Tracking errors and desired force under COM uncertainty.

desired forces under the sampling of these parameters are shown in Fig. 8.11. The average settling time for all samples are 1.29 secs, and the largest settling time 2.28 secs appears in x direction when $k_c = 4440.9$ and $b_c = 133.3$. The Cartesian force converges to effective gravitational force to compensate the gravity for the composite system. The robustness to friction uncertainties will be described in next Chapter.

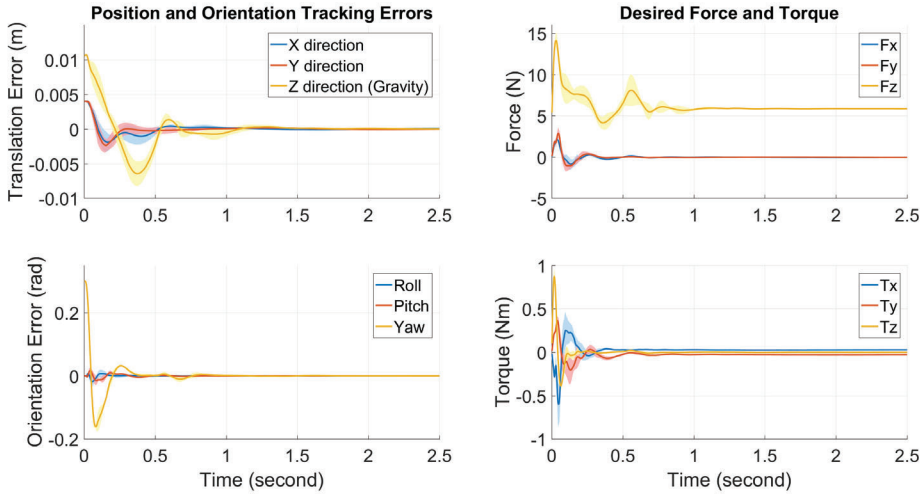


Figure 8.11: Tracking errors and desired force under contact dynamics uncertainties.

Finally, the proposed RMC is robust to tactile uncertainty. The nominal contact position $\bar{c}_i$ and rotation $\bar{R}_{c_i}$ for the i-th contact are computed by $\bar{c}_i = c_i + \delta c, \bar{R}_{c_i} = R_{c_i} \cdot \delta R_c$,

where c_i and R_{c_i} are actual contact position and rotation, and δc and δR_c are uncertainties added to the contact, with $\delta c \sim \mathcal{N}(m_{\delta c}, \sigma^2_{\delta c})$ and δR_c generated from Euler angle uncertainty $\delta E_c \sim \mathcal{N}(m_{\delta E}, \sigma^2_{\delta E})$. The nominal contact positions for finger 1 and 3 $\bar{c}_{1/3}$ were set to $c_{1/3} - \delta c$ to avoid the influence of symmetry. In the simulation, $m_{\delta c}$ varies from $[-5, -5, -5]$ to $[5, 5, 5]$ mm, $\sigma_{\delta c} = 3\text{diag}([1, 1, 1])$ mm, and $m_{\delta E}$ varies from $[-0.2, -0.2, -0.2]$ to $[0.3, 0.3, 0.3]$ rad and $\sigma_{\delta E} = 0.1\text{diag}([1, 1, 1])$ rad. The influence of these parameter variations is shown in Appendix A.1. In the meantime, the object has 20% mass and 50% moment of inertia uncertainties. The tracking errors and the associated force on object under above uncertainties are shown in Fig. 8.12. The average settling time for all samples within the range is 1.39 secs, and the largest settling time 2.58 secs appears in x direction when $m_{\delta c} = [5, 5, 5]$ mm and $m_{\delta E} = [0.2, 0.2, 0.2]$ rad.

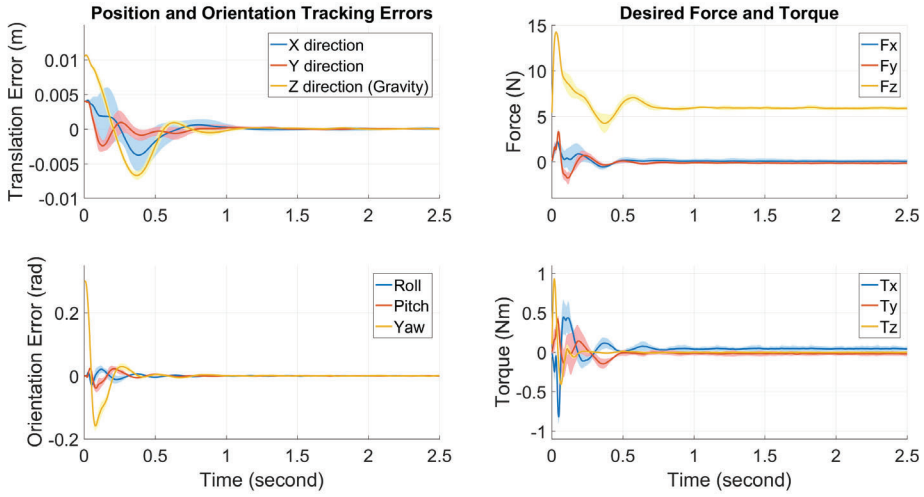


Figure 8.12: Tracking errors and desired force under tactile uncertainty.

8.6 Experiment Study

Experimental Setup

This section presents experimental validation of the proposed RMC algorithm. The algorithm was implemented to BarrettHand BH8-282. The BarrettHand has 3 fingers, 4 DOFs and 8 joints. Each finger has 2 joints driven dependently by 1 motor through gears and wires. An additional motor controls the spread joints of both finger 1 and 2, causing dependent motion of two spread joints. It equips a joint encoder in each joint, pressure profile systems (PPS) tactile sensors on fingertips and palms, and a strain gauge joint torque sensor at the distal link of each finger.

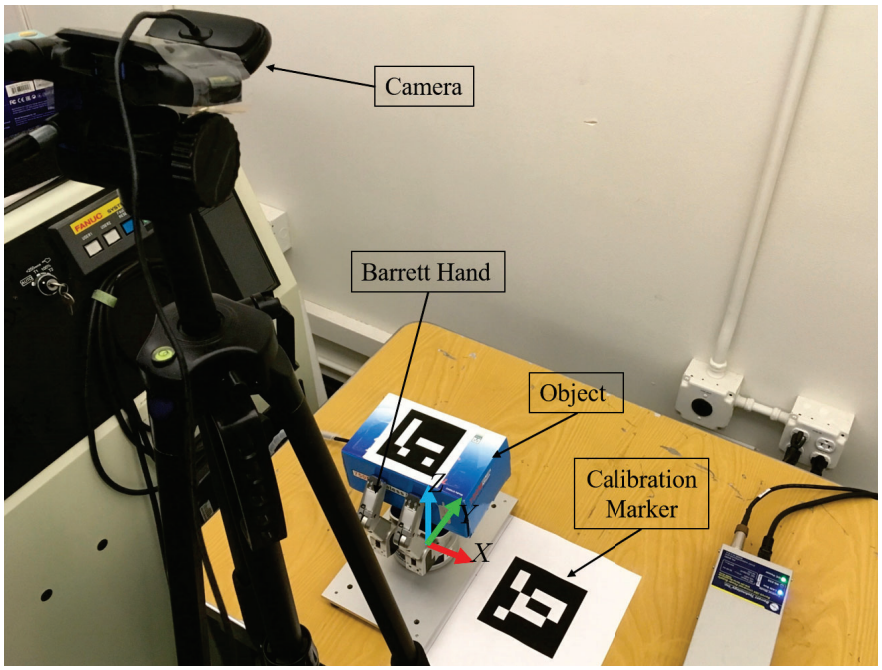


Figure 8.13: Experimental setup for RMC validation.

Figure 8.13 shows the experimental setup for RMC validation. A Logitech C270 webcam was used to perceive the scene and capture the motion of ArUco markers. Marker detection and pose estimation algorithms from OpenCV were applied to estimate the object 3D position and orientation. The pose error was then sent to proposed RMC algorithm to produce desired contact force for hand-object interaction. Since the wire-driven introduces excessive frictions for direct open-loop force implementation, tactile and torque signals were fused to estimate actual contact force and provide feedback for a low-level contact force tracking controller.

Figure 8.14 shows the control structure of the proposed implementation. It consists of two control levels including a high-level robust manipulation controller (RMC) shown in red and a low-level contact force tracking controller shown in yellow. RMC takes object pose as input, and generates the desired contact force f_{cmd}. The contact force tracking controller tracks the contact force on fingertips with the force feedback. For BarrettHand BH8-282 with fingertip tactile sensors and distal joint torque sensors, the actual contact force f_{act} on each fingertip is obtained from pressure readings of PPS sensors multiplying effective contact areas as well as the strain gauge torque readings of distal links. The force tracking controller is a proportional-derivative (PD) controller. The derivative channel increases the damping of interaction and improves the stability of closed-loop system. The details of the sensor fusion is introduced below.

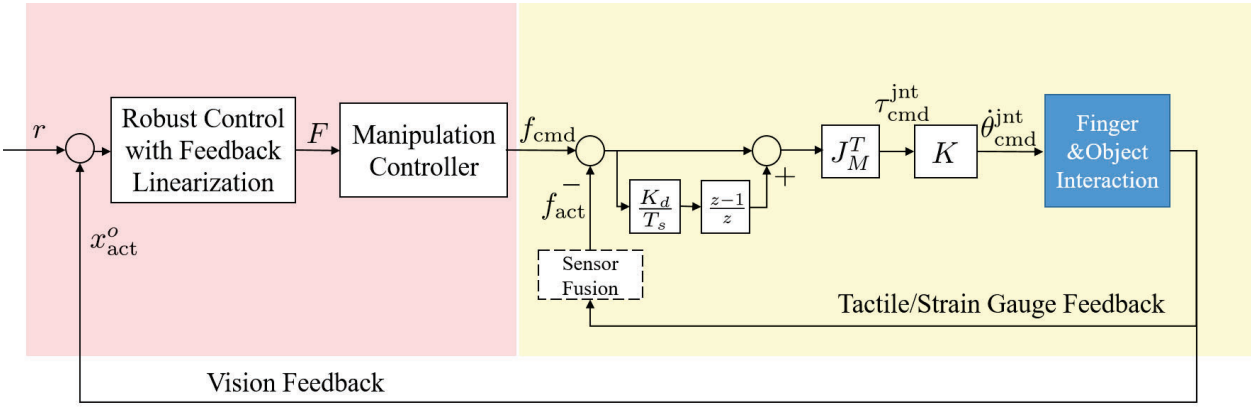


Figure 8.14: Manipulation structure for experimental validation.

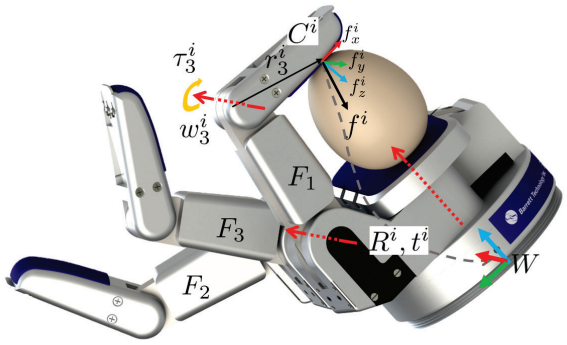


Figure 8.15: Illustration of the contact force calculation.

Experimental Results

Sensor Fusion and Force Control

The force control is to track the contact force command f_{cmd} generated by manipulation controller. With point contact with friction (PCWF) model [69], the contact force on the i-th fingertip $f^i = [f^i_x, f^i_y, f^i_z]^T \in \mathbb{R}^3$, where f^i_z, f^i_x, and f^i_y represent projection of f along the normal, finger, and joint axis directions, respectively, as shown in Fig. 8.15. The PPS tactile sensor equipped on the fingertip is a one-dimensional pressure sensor and only f^i_z is measurable by multiplying the perceived pressure readings with effective contact areas. To further obtain the remaining force elements, we employ the strain gauge torque sensor in each distal joint. More specifically, the relationships between tactile/torque observations and

contact force are represented as:

$$\begin{aligned}
\tau_3^i &= (w_3^i \times r_3^i)^T R^i f^i \quad i = 1, ..., 3 \\
f_z^i &= [0, 0, 1]^T \cdot f^i \quad i = 1, ..., 3 \\
\tau_1^i &= (w_1^i \times r_1^i)^T R^i f^i \quad i = 1, 2 \\
\tau_{sp} &= \tau_1^1 + \tau_1^2
\end{aligned} \tag{8.19}$$

where τ_3^i, w_3^i, r_3^i represent the joint torque, joint axis, and rotation radius of the distal joint of i-th finger, R^i represent the rotation from contact frame C^i to world frame, f_x^i, f_y^i, f_z^i denote the decomposition of contact force f^i towards the axes of C^i, and τ_1^i denotes the torque of joint 1 of i-th finger and τ_{sp} is the torque measurement of the spread joint.

Equation (8.19) defines 7 independent functions with 9 unknown variables, so the contact force is not observable. Further kinematic analysis shows that the contact forces along the normal and finger directions $f_x^i, f_z^i,\ i = 1, 2, 3$ are solvable by first two sets of equations. The components f_y^1, f_y^2 are perpendicular to the finger planes and are not controllable or observable, given the fact that finger 1 and 2 are moved dependently controlled with one spread motor, and the torque τ_{sp} on the load side is not directly measurable. f_y^3 is structural force and not observable. The contact force tracking control uses the force estimation of $(f_x^1, f_z^1, f_x^2, f_z^2, f_x^3, f_z^3)$.

Figure 8.16 shows the tracking results of the contact force in the normal direction (f_z) with finger 2. The force tracking controller contains proportional-derivative (PD) feedback channels (P gain is 0.5 and D gain is 0.2) and ran on 30 Hz to convert the force command to motor velocity, as shown in Fig. 8.14. The red curves are desired force trajectories with frequencies span from 0.2 Hz to 2 Hz, and the blue curves are the actual force from PPS tactile sensor multiplying effective contact areas. The tracking controller suffers from large nonlinearities, as shown in the downside of the sinusoidal curves. The nonlinearities may come from the large Coulomb friction or the backlash in the drivetrain. The phase error becomes excessive large when tracking a 2 Hz force curve.

Compared with Type A hand in Chapter 7, the BarrettHand has more rigid drivetrains, thus a derivative channel is required to increase the damping and improve stability. The rigidity of the drivetrain increases the bandwidth of force tracking from 0.4 Hz to 1 Hz. Compared with Type B hand in Chapter 7, the BarrettHand has larger friction, backlash and lower control rate, which introduce more nonlinearities and latencies. Therefore, the bandwidth for force control with Type B hand (6 Hz) is higher than BarrettHand (1 Hz).

Robust Manipulation Results

This section shows the experimental results using the proposed robust manipulation controller (RMC) with a low-level contact force tracking controller.

The object to be manipulated is 0.29 Kg, while the estimated mass is 0.2 Kg (31% mass uncertainty). RMC was designed to resist 40% mass and 50% moment of inertia uncertainties. Due to the large sampling time of the force tracking control, the continuous plant was

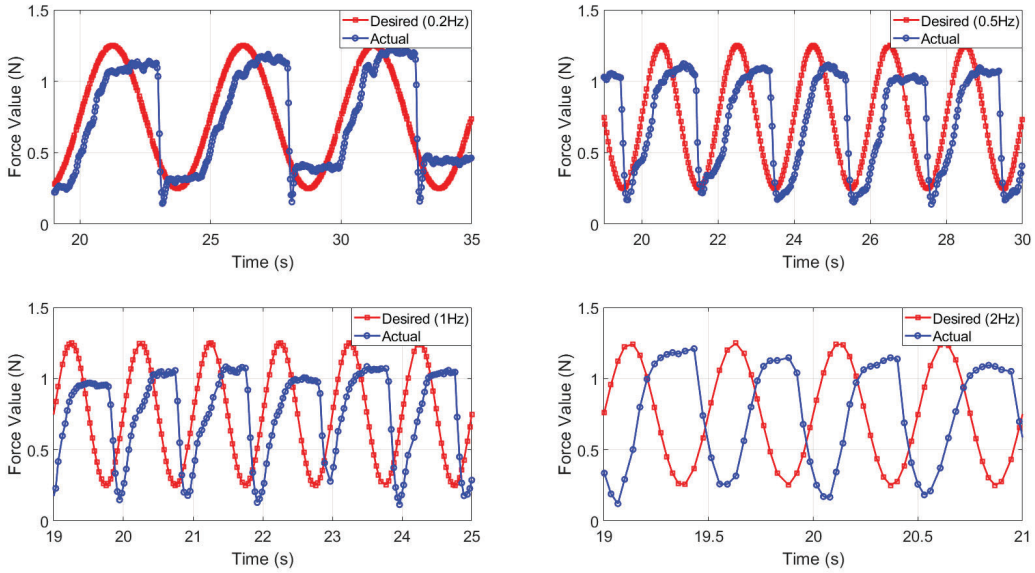


Figure 8.16: Illustration of the normal contact force tracking results.

first discretized with $T_s = 0.033$ s before RMC design. With the plant discretization, the parameters of weighting functions in Section 8.4 are redesigned and shown in Table 8.2:

Table 8.2: Weighting Functions of Barrett Experiment

Weightings	ω_c	G_l	G_h	(a_1, a_2, a_3)	n
W_{perf}	8π	20	0.9	(1,1,1,1,1,1)	2
W_u	N/A	0.0003	0.0003	(1,1,1,0.1,0.1,0.1)	1
W_{dis}	8π	32	0.1	(1,1,1,1,1,1)	2
W_n	8π	0.1	10	(1,1,1,1,1,1)	1

The closed-loop system with the designed RMC has robust stability margin is around 1.13, which means that the system can withstand about 13% more uncertainty than is specified in the uncertain elements without going unstable.

Fixed Target Pose Tracking The achievable object motion is affected by both kinematic constraints caused by limited sensors for force tracking and low-DOFs of the hand. On one hand, the contact force components $f_y^i,\ i = 1, ..., 3$ are not measurable by the equipped PPS tactile sensors or strain gauge torque sensors. On the other hand, the fundamental grasp constraint [69] describes the relationship between finger velocity and object velocity under non-slippage condition:

$$J_h(\theta, x_o) M \dot{\theta}_m = G^T(\theta, x_o) \dot{x}_o$$

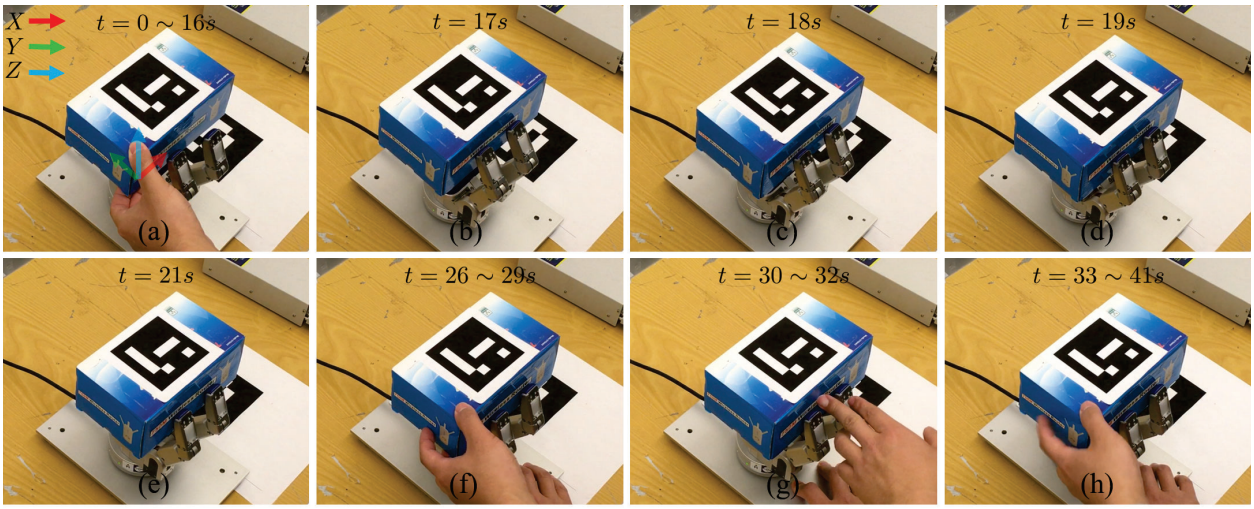


Figure 8.17: Snapshots of fixed pose tracking with RMC.

where $x_o \in \mathbb{R}^6, \theta \in \mathbb{R}^8, \theta_m \in \mathbb{R}^4$ denote object pose in local coordinates, joint angle and motor angle, respectively, $J_h M \in \mathbb{R}^{9\times4}$ is the hand Jacobian in motor side and $M \in \mathbb{R}^{8\times4}$ is the transformation from joint angle to motor angle, and $G^T \in \mathbb{R}^{9\times6}$ denotes transpose matrix of the grasp map. Grasp manipulability [69] is used to characterize the existence of hand motion $\dot{\theta}_m$ to realize arbitrary $\dot{x}_o$. The grasp is manipulable on a configuration (x_o, θ) if and only if $colspan(G^T) \subset colspan(J_h(\theta, x_o)M)$. In general three-point contact condition of BarrettHand, $rank(G^T) = 6 > 4 = rank(J_h M)$, thus the grasp is not manipulable and cannot achieve arbitrary object motion. The realizable object motion $\dot{x}_{o,real} = \{\dot{x}_o | G^T \dot{x}_o \in colspan(J_h M)\}$.

Further analysis on force-torque relationship reveals the deficiency of BarrettHand on dexterous manipulation. The relationship between object Cartesian force and motion torque of BarrettHand is:

$$F = \underbrace{G(J_h M)^{-T}\tau_m}_{controllable} + \underbrace{G\mathcal{N}\left((J_h M)^T\right)\lambda}_{structural}, \tag{8.20}$$

where $F \in \mathbb{R}^6, \tau_m \in \mathbb{R}^4$ denote object Cartesian force and motor torque, and $\mathcal{N}(\bullet)$ denotes the null space of $\bullet$. $\lambda \in \mathbb{R}^{N_n}$ denotes a free variable to affect magnitude and direction of the structural force, where $N_n \geq 5$ is the number of independent bases of the null space. The dimension of the controllable force space is $dim(colspan(G(J_h M)^{-T})) \leq 4$. The remaining Cartesian force is not controllable though it may be compensated by structural force.

A general analytical formulation of the feasible space of object motion is challenging under the unknown structural force and the non-neglectable rolling effect. Therefore, this section only demonstrates the manipulation performance on trivial directions by fixing the spread motor (Fig. 8.17), since the spread motor provides limited contribution for object manipulation. Besides the feasible space of motion, RMC has to address undesired prop-

erties of the hand-object system. These undesired properties include: 1) force estimation error, 2) high friction in drivetrain, 3) various uncertainties, and 4) noise of sensors. Firstly, the normal contact force (f_z^i) estimation from tactile sensors generally smaller than ground truth, causing f_x^i calculated from (8.19) significantly larger than real value, as shown in Fig. 8.19(Bottom). Secondly, the contact force is assumed to be tracked instantly, while the high friction in the drivetrain compromises the performance of force tracking control (Fig. 8.16) and introduces unmodeled dynamics to RMC. Thirdly, there are various uncertainties including 31% mass uncertainty, center of mass uncertainty (~ 0.01 m), and tactile uncertainty (position ~ 1.5 mm and orientation ~ 0.2 rad). These uncertainties may affect the stability and performance of the closed loop system. Lastly, the controller may be sensitive to the noises from vision, tactile sensors and strain gauge torque sensors.

Figure 8.17 illustrates the tracking performance to a fixed position $(0, -0.02, 0.175)$ m and rotation $(0, 0, 0)$ rad[3]. The hand started from fully open state and gradually closed three fingers until contacting with the object (Fig. 8.17(a)), after which RMC generated contact force command for low-level force tracking controller. The force drove the object to move towards the target pose, and the pose error of this time step was fed to RMC for force generation of the next time step, as shown in Fig. 8.17(b-e). The object converged to non-zero pose error within 5 secs under the aforementioned undesired properties. The robustness of the grasp to external disturbances is further demonstrated in Fig. 8.17(f-h).

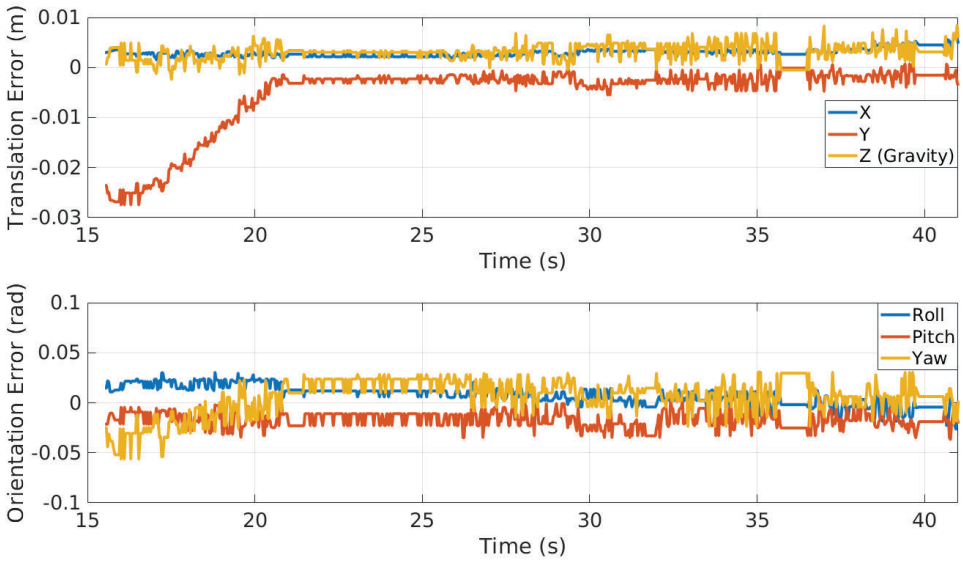


Figure 8.18: Pose error profile of fixed pose tracking with RMC.

Figure 8.18 and Fig. 8.19 show the pose error and the corresponding contact force profile of the fixed pose tracking. RMC starts at 15.5 s and the force tracking controller interacts

[3]ZYX Euler angle is used to represent rotation.

with object to track the desired object pose. The actual force exhibits high frequency oscillation during the tracking due to the effect of excessive nonlinear friction in low-speed motion. Consequently, the pose error exhibits certain level of oscillation. Nevertheless, the tracking errors converge to their minimum values after 5 secs. The robustness test starts at 26 s by manually exerting force disturbance ($\sim$ 1.5 N) to rotate object around Z axis and push it along Y axis, as shown in snapshots Fig. 8.17(fg) and in force profile Fig. 8.19. The RMC algorithm is able to comply with excessive disturbance to avoid damage of fingers while maintaining the stability of the object. There are small oscillations in Y, Z and yaw directions caused by the compliance of the closed-loop system, as shown in 26 $\sim$ 41 s of Fig. 8.18. Notice that the pose error cannot converge to zero since the grasp is not manipulable and the target grasp is not in the feasible space of object motion with the current initial configuration.

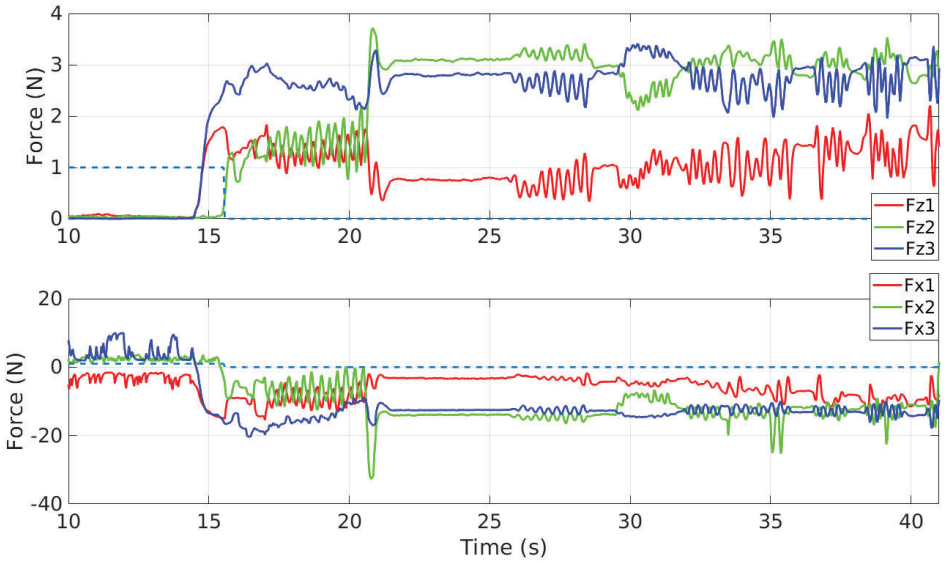


Figure 8.19: Force profile of fixed pose tracking with RMC.

Comparison of Different Methods The performance of RMC was compared with modified impedance control (MIC) in Chapter 7 using a fixed pose tracking with target position $0, 0.03, 0.16$ m and rotation $0, 0, 0$ rad. For MIC implementation, we used parameters $K_d = diag([50, 50, 50, 2.5, 2.5, 2.5])$; $I_d = diag([50, 50, 50, 1.0, 1.0, 5.0])$. Both RMC and MIC ran for 4 times. The snapshots of both methods are shown in Fig. 8.20. The pose error profiles for MIC and RMC are shown in Fig. 8.21 and Fig. 8.22. The shaded areas represent error variations and quantified by standard deviation, and the bold curves represent average error values. With the aforementioned undesired properties of hand-object system, the grasps with MIC cannot maintain firm contacts with object and exhibit large variations in both Y and pitch directions, as shown in Fig. 8.20(Top). On the other hand, the grasps with

RMC are able to maintain firm contacts and perform consistently under all undesired properties including force estimation error, nonlinear friction, dynamics uncertainties and sensor noises, as shown in Fig. 8.20(Bottom). Both MIC and RMC suffer from the reachability issue caused by the hand manipulability.

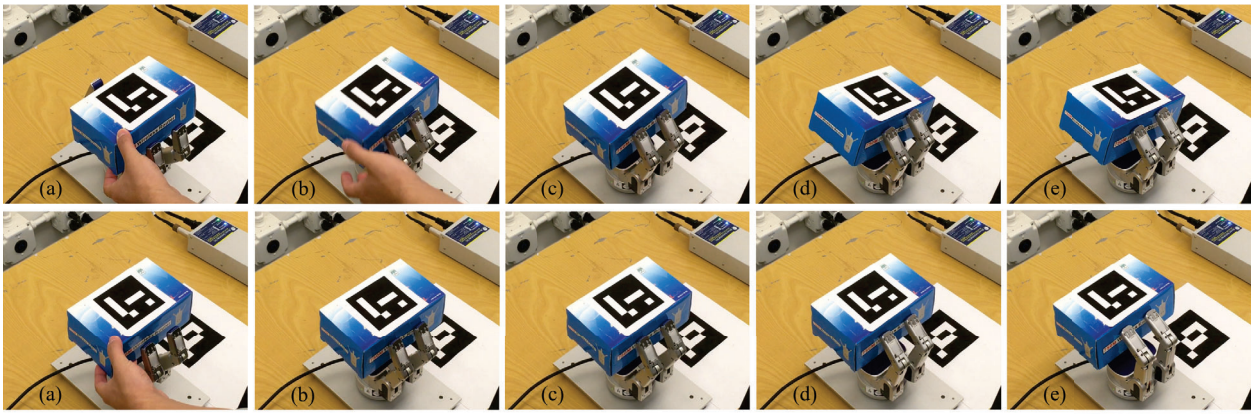


Figure 8.20: Snapshots of fixed pose tracking with (Above) MIC and (Bottom) RMC.

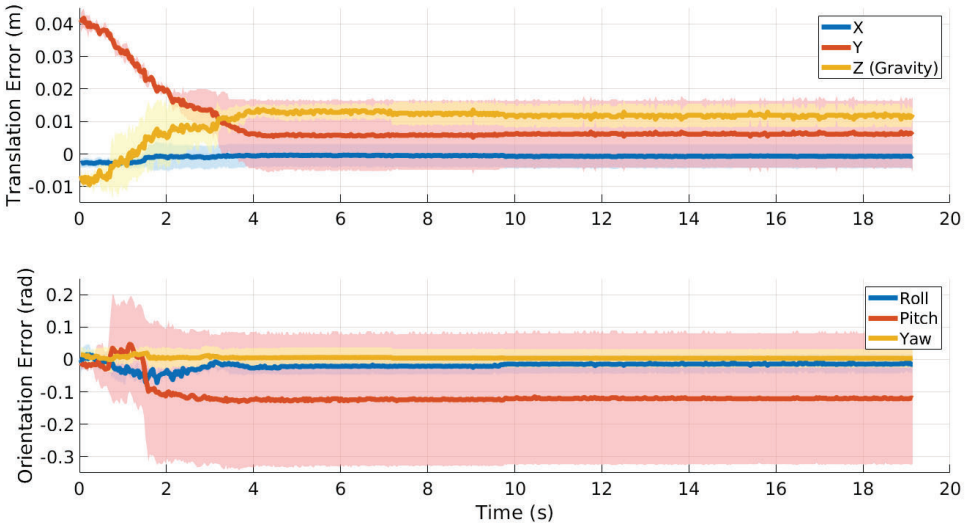


Figure 8.21: Fixed pose tracking error with MIC.

Desired Trajectory Tracking Finally, we demonstrate the performance of RMC on tracking a desired trajectory that is within the feasibility space of object motion. Without considering the spread motion, the desired trajectory was pre-recorded during tracking

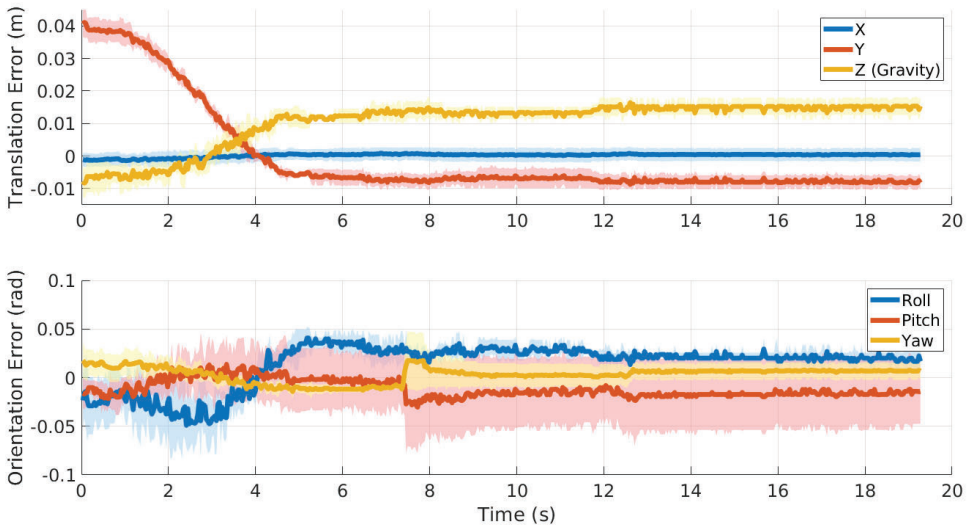


Figure 8.22: Fixed pose tracking error with RMC.

to fixed position $(0, 0.03, 0.16)$ m and rotation $(0, 0, 0)$ rad (Fig. 8.22), though it could be computed analytically given the initial contacts. Figure 8.23 and Fig. 8.24 illustrate the execution snapshots and the error profile for the trajectory tracking. RMC tracked the target motion in Y and Z directions and maintained a firm grasp to keep the stability of other directions. It can be seen that the desired motion on Y axis can be accurately tracked, though the Z direction exhibit 0.005 m error. Nevertheless, the error is much smaller than fixed-pose tracking in Fig. 8.22 (0.015 m). The robustness of the proposed algorithm to external disturbances is demonstrated at last from $31 \sim 35$ s. Similar with fixed pose tracking in Fig. 8.18, RMC is able to comply with the external disturbance to avoid excessive large contact force while maintaining the firm grasp without losing the stability, as shown in $31 \sim 35$ s of Fig. 8.24.

8.7 Chapter Summary

This chapter proposed a robust manipulation controller (RMC), which includes a robust controller and a manipulation controller, to achieve dexterous manipulation under various uncertainties and external disturbances. Feedback linearization was applied to reduce the nonlinearities of the composite hand-object system. By utilizing the structures of the uncertainties, the proposed robust controller can achieve faster convergence and tolerate more uncertainties compared with other methods based on disturbance observer and modified impedance controller (MIC) from Chapter 7. RMC skipped complicated contact modeling, and was able to regulate contact force and prevent slippage. Simulation verified that the

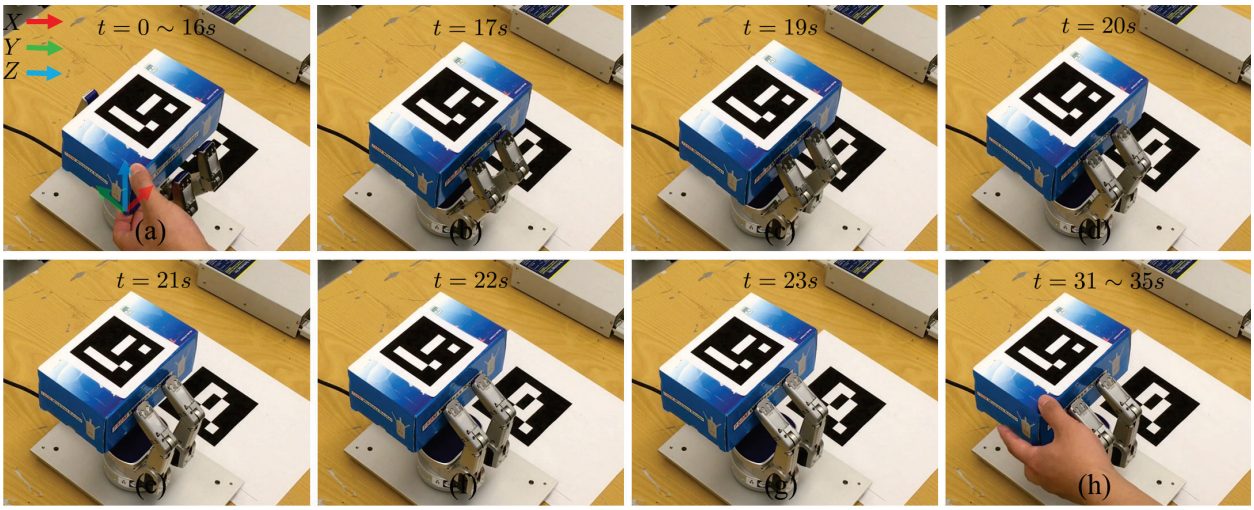


Figure 8.23: Snapshot of the feasible reference tracking with RMC.

proposed RMC is robust to various uncertainties including dynamics, center of mass, tactile uncertainties. Moreover, it did not require joint/object velocity measurement and was able to achieve fast tracking performance. Further experiments were performed on a Barrett-Hand BH8-282 with minor adaptations of the algorithm. The experiment verified that the proposed RMC is able to manipulate objects to different target poses robustly and stably under various undesired properties of the hand-object system.

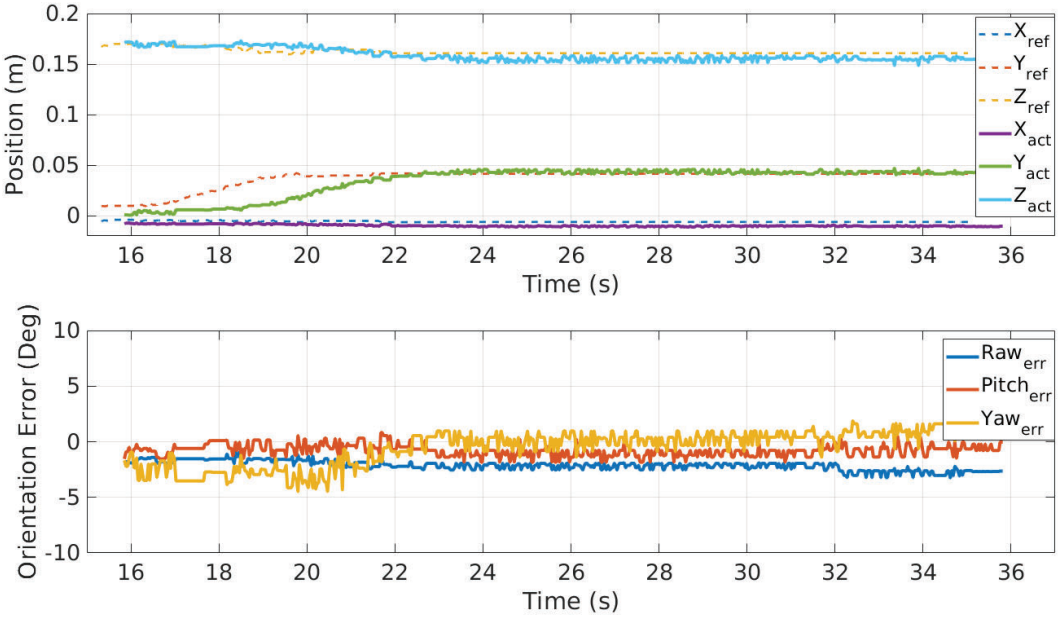


Figure 8.24: Error profile of the feasible reference tracking with RMC.

Chapter 9

Finger Gaits Planning for Robust Dexterous Manipulation

9.1 Introduction

Chapter 8 introduced a robust manipulation controller (RMC) to manipulate objects and track the desired trajectories of the objects within the workspace of the hand. To perform long-range complex manipulations, the robotic hand may have to change its grasping status by relocating fingers during the manipulation, which gives the hand more dexterity and robustness. Such strategy is called finger gaits planning. However, the optimization of the finger gaiting under complicated grasp quality metrics is computationally expensive [6]. The optimization searches optimal contact points on a nonlinear object surface by maximizing object stability and hand manipulability metrics [69]. These two metrics are represented in different spaces, and associated by a high degree of freedoms (DOFs) nonlinear forward kinematics. The optimization becomes more challenging when the object 3D surface model is not available.

As a result, problems related to dexterous manipulation and finger gaits planning have received significant attention. The challenges of dexterous manipulation and finger gaiting were summarized in [6], namely, the analysis and control of hybrid systems during gaits planning, and the optimization of the plans. A task-specific finger gaiting policy was trained in [2] by the covariance matrix adaptation method, given the goal states of objects. However, the learned policies cannot be adapted to other objects and tasks. A high-speed hand and a high-speed vision system were applied in [33] to perform dynamic re-grasping. However, the object model should be precisely known, and the presented success rate (35%) is not suitable for many applications. Impedance parameters were learned from human demonstration for robust grasping and dexterous manipulation in [51]. A tangle topology was used in [101] to reproduce object pose from learned human demo. However, the object gravity is not considered during their gaits changing process. A set of controllers is used in [75] to realize unknown object grasping by sliding on the surface to maximize grasp stability. The unknown

surface of the object was explored in [76, 68] by designing a global re-grasping planner and searching local optimal contact points. However, predefined finger gaits are used in these approaches, and the exploration of local optima does not incorporate necessary constraints, such as joint velocity and acceleration limitations. As a result, these approaches tend to be slow in re-grasping and manipulation, and the predefined finger gaits might be inapplicable to other objects and robotic hands. A sampling-based method was proposed in [106] to plan finger gaits. In [68], a contact-invariant optimization method was used to compute the states of the hand and the object, given the high-level goals. These approaches are not computationally efficient for real-time finger gaits planning.

In this chapter, a dual-stage optimization based planner is developed for real-time robust finger gaits planning under the object and contact uncertainties. The dual-stage planner consists of a finger gaits planner and a robust manipulation controller (RMC) in Chapter 8. To achieve real-time computation, the finger gaits planner is formulated in the velocity level, instead of the position level; formulation in the position level results in a complicated nonlinear constrained optimization problem. At each time step, the optimal joint velocities are computed to improve the hand manipulability as well as the object grasp quality, and the computed joint velocities are fed into motors by a velocity-force controller. The proposed RMC is formulated as a robust control and a contact force optimization. Feedback linearization and μ-synthesis are combined for the robust controller design to deal with nonlinearities, dynamics uncertainties and external disturbances, as introduced in Chapter 8.

The contributions of this chapter include: 1) The velocity-level finger gaits planner is cast into a linear programming (LP), which is computationally efficient and can be solved in real-time (< 1 ms). Furthermore, the velocity-level gaits planning incorporates joint kinematic constraints, which makes the generated motions feasible. 2) The two-level optimization based planner is robust to different types of uncertainties as described in Chapter 8. The velocity-level finger gaits planner utilizes velocity-force control to detect the object surface and searches motions in tangent space. Moreover, RMC can handle 50% mass and 80% moment of inertia uncertainties, $(10, 10, 15)$ mm COM deviations and $(0.3, 0.3, 0.3)$ rad principal axes variations, robust to 10 times differences on stiffness, and withstand $(5, 5, 5)$ mm contact position and $(0.3, 0.3, 0.3)$ orientation variations. 3) The proposed dual-stage optimization based planner reduces the cost of the dexterous hand. To be more specific, it does not require precise 3D reconstruction for exact object surface model, and high resolution encoder or accelerometer for velocity measurement. The efficacy of the proposed controller is verified by simulations. The video demo is available at [102].

The remainder of this chapter is organized as follows. Section 9.2 shows the overall dual-stage optimization based planner framework. Section 9.3 introduces the finger gaits planner. The design of robust manipulation controller is presented in Chapter 8. Section 9.4 shows simulation results on two robotic hands with different fingers and DOFs. Section 9.5 summarizes the chapter.

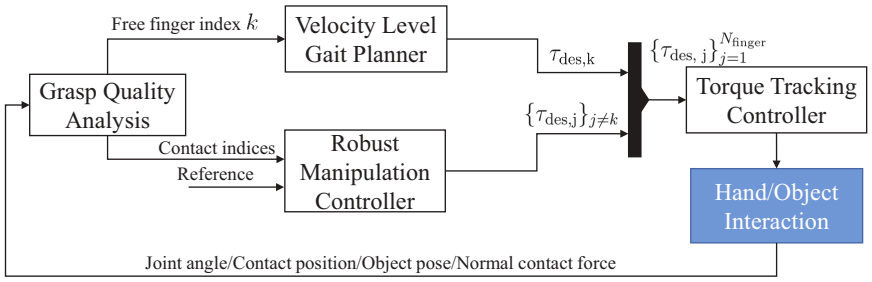


Figure 9.1: The general framework of the proposed optimization based planner.

9.2 Dual-Stage Manipulation and Gaiting Framework

Figure 9.1 shows the proposed dual-stage optimization based planner framework. First, grasp quality analysis is conducted by combining hand manipulability and object grasp quality, and the free finger index k is chosen to change gait once the overall quality drops below a predefined threshold. A velocity-level finger gaits planner is evoked by this event, and the planner generates the torque command $\tau_{\text{des,k}}$ to drive the selected finger towards the better quality region. The remaining fingers are controlled by RMC in Chapter 8 to generate desired torque $\{\tau_{\text{des,j}}\}_{j\neq k}$, to manipulate the object stably and track the reference motion of the object, as shown in Fig. 9.1. If the overall quality is above the threshold, all fingers will be controlled by RMC. The detailed introduction of RMC is in Chapter 8.

A joint level torque tracking controller is used to track the desired torque command $\{\tau_{\text{des, j}}\}_{j=1}^{N_{\text{finger}}}$, where N_{finger} denotes the number of fingers. The torque tracking controller uses a PID scheme and runs at a higher frequency, in comparison with the finger gaits planner and the robust manipulation controller (500 Hz).

9.3 Real-Time Finger Gaits Planning

Grasp Quality Analysis

Grasp quality has been well explored in [69, 79]. It is desired that both hand manipulability and object grasp quality are considered during the finger gaits planning. The hand manipulability describes the ability for a hand to manipulate the object to realize arbitrary object motions. The object grasp quality describes the capacity to resist external disturbances given a group of contact points on the object. This chapter adopts a quality metric in [54] to represent the hand manipulability Q_h: $Q_h = -0.5\sum_{j=1}^{N_{\text{finger}}}\sum_{i=1}^{N_{\text{joint}}}\left((q_j^i - \bar{q}_j^i)/(q_{\text{max,j}}^i - q_{\text{min,j}}^i)\right)^2$, where q_j^i is the i-th joint angle of the j-th finger, $q_{\text{min,j}}^i$ and $q_{\text{max,j}}^i$ are the limits of q_j^i, $\bar{q}_j^i = (q_{\text{max,j}}^i + q_{\text{min,j}}^i)/2$ is the middle position of the corresponding joint, N_{joint} is the number of joints per finger. The object grasp quality Q_o is represented as: $Q_o = 2\text{Area}\left(\{p_j\}_{j\in I_c}, \text{proj}(p_k)\right)$ [91], where I_c is the set of indices of all fingertips that are in contact with the object. p_j is the

contact position in Cartesian space for the j-th fingertip. $\text{proj}(p_k)$ denotes the projection operation of p_k onto the plane specified by $\{p_j\}_{j\in I_c}$.

The overall quality Q can be obtained by combining Q_o and Q_h:

$$Q = w_1 Q_o + w_2 Q_h \tag{9.1}$$

where $w_i > 0$ is the weight for the corresponding term.

Once the overall grasp quality Q drops below a threshold, the finger gaits should be replanned to adjust contact points on the object. It is observed that humans tend to relocate their fingers one by one during the finger gaiting. This idea is adopted and the finger gaits are sequentially planned. Thus, the proposed algorithm will compare all the fingers and choose one of them to initialize finger gaits planning, if all fingertips are in static contacts and $Q < \delta_Q$, where δ_Q is a threshold. The free finger is selected based on the finger manipulability of itself and the grasp quality of the remaining fingers to the object. To be more specific, the finger manipulability for the k-th finger is $-0.5\sum_{i=1}^{N_{\text{joint}}}\left((q_k^i - \bar{q}_k^i)/(q_{\text{max,k}}^i - q_{\text{min,k}}^i)\right)^2$. The grasp quality of remaining fingers to the object is the area of convex hull spanned by the remaining fingertips. The candidate free finger for gaiting is the one with small finger manipulability and large remaining grasp quality. If there is already one free finger that are not in contact with the object, that finger will continue its gaiting.

In this section, the related position-level finger gaits planner is first presented; then the velocity-level finger gaits planner is proposed to resolve the problems in the position-based planner.

Position-Level Finger Gaits Planning

Position-level finger gaits planner consists of a contact optimization and a trajectory planning. The contact optimization searches an optimal contact point to maximize the overall grasp quality (9.1), and the trajectory planning generates trajectories to relocate the finger to the optimal contact point. The contact optimization can be formulated in the following form:

$$\begin{aligned}
\max_{p_k, q_k} \quad & Q && \text{(9.2a)}\\
s.t. \quad & p_k \in \partial O && \text{(9.2b)}\\
& \|p_k - p_o\| \le \epsilon && \text{(9.2c)}\\
& p_k = \text{FK}(q_k) && \text{(9.2d)}\\
& q_{l,k} \le q_k \le q_{u,k} && \text{(9.2e)}
\end{aligned}$$

Constraint (9.2b) indicates that the fingertip position p_k of the free finger k should be on the surface of object ∂O. Constraint (9.2c) means that the searching region should be constrained in certain region ϵ from original position p_o to keep the stability of the object. Constraint (9.2d) is the forward kinematics of the finger. Constraint (9.2e) means that the

joint space searching should be in the feasible region. After finding the optimal contact point, the trajectory planning algorithm is required to generate a feasible trajectory.

The position-level finger gaits planning has the following drawbacks: First, the problem has nonlinear equality constraints. Therefore, it is difficult for real-time computation. In addition, this method requires a trajectory planning to avoid collision with the object and reach the planed optimal point. Moreover, the trajectory planning should consider the collision avoidance. Furthermore, the equality constraint (9.2b) corresponding to object surface is usually unknown in advance. Lastly, the contact optimization (9.2) uses current contacts $\{p_j\}_{j\in I_c}$ to find the optima, while $\{p_j\}_{j\in I_c}$ actually keep moving during the contact optimization, trajectory planning and execution. With all aforementioned issues, an efficient velocity-level gaits planner is proposed below.

Velocity-Level Finger Gaits Planning

The task of the finger gaits planner is to generate commands to change the contact location of the free finger, to achieve better object grasp quality and finger manipulability in real time. However, searching contact position by maximizing the quality (9.1) is challenging. First, the search of p_k should be conducted on the surface of the object, and the formulation of the object surface requires 3D reconstruction and surface modeling. Second, the searching of q_k should be constrained within the joint limits, and q_k is coupled with p_k by forward kinematics. Third, after finding the optimal contact point, a trajectory planning algorithm is required to generate a feasible trajectory. In our previous work [22], a velocity level finger gaits planner is proposed to overcome the aforementioned challenges.

In this planner, the contact optimization is modified into a short-term optimization. To be more specific, rather than finding an optimal contact point, an optimal moving velocity of the fingertip of the free finger is calculated at each time step, and the finger is actuated by a velocity-force controller to achieve that velocity. Formally, instead of optimizing Q in (9.1), we optimize $\dot{Q}$ with joint velocity of the k-th finger $\dot{q}_k$ as the decision variable. The solution $\dot{q}_{\text{des,k}}$ is used to control the robotic hand in each time step.

The intuition behind it is the Taylor series expansion. Q is a function of states $\{q_k, p_k\}$, and the states are the functions of time t. Therefore, Q is a function of t. By this interpretation, Q in time instant $t + T_s$ can be written as:

$$Q(t + T_s) \approx Q(t) + \dot{Q}(t)T_s \tag{9.3}$$

where T_s is the time step. In this equation, higher order terms have been omitted, because T_s is usually a small period. Therefore, designing control policy to maximize $Q(t + T_s)$ is equivalent to maximizing $\dot{Q}(t)$.

With the short term approximation, $\dot{Q}$ becomes:

$$\begin{aligned}\dot{Q} &= w_1\dot{Q}_o + w_2\dot{Q}_h \\ \dot{Q}_o &= \|p_{j_2} - p_{j_3}\|_2 n_{j_1}^T v_{p_k} \\ \dot{Q}_h &= \sum_{i=1}^{N_{\text{joint}}} \left(\frac{\bar{q}_k^i - q_k^i}{(q_{\text{max,k}}^i - q_{\text{min,k}}^i)^2} \dot{q}_k^i \right)\end{aligned} \tag{9.4}$$

We assume that $\{p_j\}_{j\in I_c} = \{p_{j_1}, p_{j_2}, p_{j_3}\}$, n_{j_1} is a normal vector of line segment $\overline{p_{j_2}p_{j_3}}$ in the plane specified by $\{p_j\}_{j\in I_c}$, and v_{p_k} is the velocity of contact point p_k. $\dot{q}_k^i$ is joint velocity of the i-th joint for the k-th finger. In this optimization, the states v_{p_k} and $\dot{q}_k$ in $\dot{Q}_o$ and $\dot{Q}_h$ are coupled linearly by $v_{p_k} = J(q_k)\dot{q}_k$, where $J(q_k)$ is the Jacobian matrix of the k-th finger. By plugging in the coupled term, $\dot{Q}$ becomes:

$$\dot{Q} = w_1\|p_{j_2} - p_{j_3}\|_2 n_{j_1}^T J\dot{q}_k + w_2 \sum_{i=1}^{N_{\text{joint}}} c_k^i \frac{\bar{q}_k^i - q_k^i}{(q_{\text{max,k}}^i - q_{\text{min,k}}^i)^2} \dot{q}_k^i \tag{9.5}$$

c_k^i is a weighting function added to (9.5) to address influence of joint limits:

$$c_k^i = \begin{cases} \ln(\frac{\bar{q}_k^i - q_{\text{min},k}^i - q_{\text{thres}}^i}{q_k^i - q_{\text{min},k}^i}) + 1, & q_k^i - \bar{q}_k^i < -q_{\text{thres}}^i \\ 1, & |q_k^i - \bar{q}_k^i| \le q_{\text{thres}}^i \\ \ln(\frac{q_{\text{max},k}^i - \bar{q}_k^i - q_{\text{thres}}^i}{q_{\text{max},k}^i - q_k^i}) + 1, & q_k^i - \bar{q}_k^i > q_{\text{thres}}^i \end{cases}$$

where q_{thres}^i is a threshold where the weighting should start to increase. In the meantime, constraints (9.2b) and (9.2d) become $n_{p_k}^T J(q_k)\dot{q}_k = 0$, where n_{p_k} is the surface normal of the object at p_k, and $\dot{q}_k = [\dot{q}_k^1, ..., \dot{q}_k^{N_{\text{joint}}}]^T$. The surface normal can be inferred by the tactile sensor on the fingertip. The constraint (9.2c) is eliminated because we are working on short-term optimization, and the optimization would be solved in each time step.

With above analysis, a new optimization can be formulated to approximate the original contact optimization:

$$\dot{q}_{\text{des,k}} = \arg\max_{\dot{q}_k} \ \dot{Q} \tag{9.6a}$$

$$s.t. \quad \dot{q}_{\text{min,k}} \le \dot{q}_k \le \dot{q}_{\text{max,k}} \tag{9.6b}$$

$$n_{p_k}^T J(q_k)\dot{q}_k = 0 \tag{9.6c}$$

$$\|\dot{q}_k - \dot{q}_{\text{des,prev}}\|_\infty \le \sigma \tag{9.6d}$$

where constraint (9.6b) means that the desired joint velocity $\dot{q}_k$ should be bounded in $[\dot{q}_{\text{min,k}}, \dot{q}_{\text{max,k}}]$. Constraint (9.6c) indicates that p_k must move perpendicular to current surface normal n_{p_k}. Constraint (9.6d) limits the joint acceleration by σ/T_s, where T_s denotes

the sampling time of the system. $\dot{q}_{\text{des,prev}}$ is the desired joint velocity in previous time step. The optimization (9.6) is a linear programming, which can be solved in real-time.

After obtaining the desired joint velocity $\dot{q}_{\text{des,k}}$ by solving (9.6), a velocity-force controller is implemented to calculate the desired torque for the k-th finger:

$$\tau_{\text{des,k}} = K_v \dot{q}_{\text{des,k}} + K_f J(q_k)^T (f^n_{\text{des}} - f^n_{\text{act,k}}) \tag{9.7}$$

where $\tau_{\text{des,k}}$ and f^n_{des} are the desired torque and desired contact force in the normal direction. f^n_{des} is set to be a constant small force during finger gaiting. $f^n_{\text{act,k}}$ is the actual contact force in normal direction and can be measured by 1D tactile sensor. The force component $K_f J(q_k)^T (f^n_{\text{des}} - f^n_{\text{act,k}})$ in (9.7) attempts to maintain the contact between the fingertip and the surface, which makes the normal vector n_{p_k} measured from the tactile sensor updated.

The velocity-level gaits planner that composed of (9.6) and (9.7) has several advantages. First, the proposed planner is computationally efficient. The optimization (9.6) is an LP that can be solved in each time step. Second, the 3D object model is not required. Instead, 1D tactile sensors are employed to detect contact points on the hand and infer the surface normals by the known fingertip structures, and the sensor update can be accomplished by the force component in the velocity-force controller.

The grasp quality is expected to be improved at the beginning of gaits planning. The velocity-level gaits planner can be terminated when there is little grasp quality improvement (i.e. $\dot{Q} < \delta$, where δ is a small positive number), or when the grasp quality is above the threshold (i.e. $Q > \delta_Q$).

Similarities Between Position-Level and Velocity-Level Planners

This section shows the similarity of the performance between the velocity-level planner (9.6) and (9.7) and one step of (9.2) if solved by gradient projection method ([81]).

An abbreviated form of (9.2) is formulated for notational convenience:

$$\max_x Q \quad \text{s.t. } h(x) = 0, g(x) \leq 0 \tag{9.8}$$

where $x = \left[c_k^T, q_k^T\right]^T$, $h(x) = 0$ represents the equality constraints (9.2b) and (9.2d), and $g(x) \leq 0$ represents inequality constraints (9.2c) and (9.2e).

In each step of the gradient projection method, the search direction d is found by projecting ∇Q onto the tangent space of equality constraints $T = \{y | \nabla h^T y = 0\}$, as shown in Fig. 9.2(a). Then, an iterative technique is employed to project the points along d onto $h(x) = 0$, until the projected point x_{k+1} lies in the set $\{x | h(x) = 0, g(x) \leq 0\}$.

Instead of projecting the gradient into the tangent space, the proposed planner searches optimal direction $v_{c_k,\text{des}}$ in tangent space by LP (9.6) with consideration of the feasibility of the motion (9.6d) and (9.6b), as shown in Fig. 9.2(b). Moreover, the velocity-force controller (9.7) is a physical actualization of projecting $\hat{y}$ onto $h(x) = 0$ by maintaining the contact force between the fingertip and the surface. As the LP (9.6) is solved in each time step T_s, the search step $\Delta x_k = x_{k+1} - x_k$ is quite small. Thus, the planned $\dot{q}_{\text{des,k}}$ in (9.6) is usually smooth.

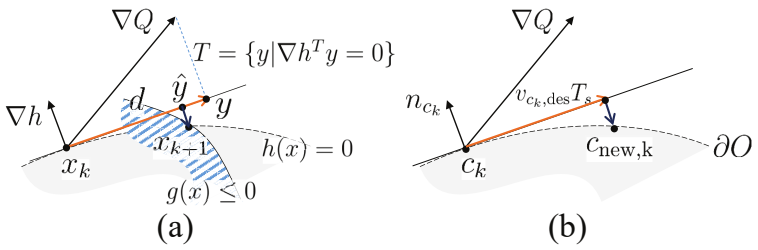


Figure 9.2: Comparison of the gradient projection method and the proposed planner.

Finger Gaiting with Jump Control

The above finger gaits planner achieves short-range sliding along the convex or concave object surfaces. However, the finger gaits planner would not able to slide through the sharp edges or hollows where the curvatures are excessive large, since the normal direction estimation from tactile sensors might be noisy due to the possible multiple-point contact conditions.

To avoid the failure of the finger gaits planner around the sharp edge or hollows, we observe the human behavior and propose a strategy called short-range jump control. First, the rough pose of the edge is estimated either by prior-knowledge, vision or tactile sensor. The jump is triggered by check whether 1) the distance between the fingertip and edge is within certain threshold, and 2) the desired velocity of the fingertip generated by the finger gaiting is towards the edge. Once the jump is triggered, the jump controller would choose a reference point and jump around the point, as shown in Fig. 9.3. In this figure, A and S denote the reference jump point and the fingertip. $\mathbf{n}$ and $\mathbf{t}$ are normal and tangent vectors computed from the current configuration. The red dash circle with radius r_{thres} is the desired path for jumping. A LP is applied to generate velocity command for the fingertip:

$$\begin{aligned}
\max_{v_t, \dot{q}_k} \quad & v_t && \text{(9.9a)}\\
s.t. \quad & J_{v,k}\dot{q}_k = \mathbf{t}v_t + \mathbf{n}k_n(r_{\text{thres}} - \|AS\|) && \text{(9.9b)}\\
& \dot{q}_{\text{min,k}} \leq \dot{q}_k \leq \dot{q}_{\text{max,k}} && \text{(9.9c)}\\
& v_t \geq 0 && \text{(9.9d)}
\end{aligned}$$

where v_t is the fingertip velocity in tangent direction $\mathbf{t}$, and $\|AS\|$ is the current distance between the fingertip and the reference point. The (9.9) aims to maximize the moving speed along the tangent direction. Constraint (9.9b) regulates the fingertip along the desired circular path to avoid unexpected collision with the edge. The object velocity is not included due to the measurement noise, and the assumption that the object moves slow. Constraints (9.9c)(9.9d) ensure that the jump is feasible and progressive.

The desired joint velocity $\dot{q}_{\text{des,k}}$ generated from (9.9) is converted into the joint torque by $\tau_{\text{des,k}} = K_v \dot{q}_{\text{des,k}}$. During the jump, the finger gaiting is bypassed. The jump control is reset

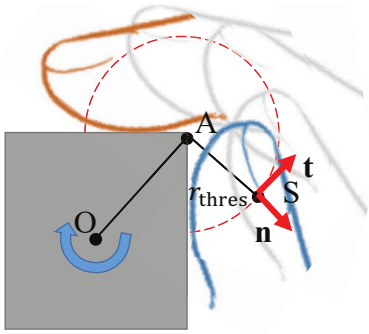


Figure 9.3: Illustration of jump control strategy on sharp edges.

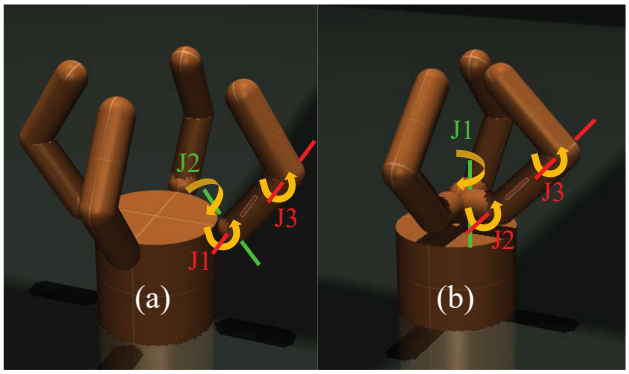


Figure 9.4: Two hand models used in the simulation.

once the fingertip recontact with the object. r_{thres} is small value, thus the grasp quality (9.1) is assumed not effected by the contact change due to the short-range jump control.

9.4 Simulation Study

In this section, simulation results are presented to verify the effectiveness of the proposed dual-stage optimization based planner. The simulation video is available at [102].

Simulation Setup

Same as Chapter 8, the algorithm was implemented in Mujoco physical engine [96]. The simulation and planning time steps were set to 0.5 ms and 2 ms, respectively. Our platform was a desktop with Intel Core i7-6700K CPU and 32 GB RAM, running Windows 10 operating system. The hand models used in the simulation are shown in Fig. 9.4. Both the three-finger hand and the four-finger hand are presented to demonstrate the effectiveness of the algorithm on different types of hands. The four-finger hand is set up with twelve DOFs

and each finger has three revolute joints J1, J2, and J3, as shown in Fig. 9.4(a). The joint angles of J1, J2 and J3 are constrained in $[-10°, 135°]$, $[-45°, 45°]$ and $[-10°, 170°]$, respectively. The three-finger hand is set up with nine DOFs and each finger has three revolute joints J1, J2, and J3, as shown in Fig. 9.4(b), and each joint is constrained in $[-90°, 90°]$. Two hands are equipped with joint encoders for joint angle feedback, motor torque sensors for joint torque feedback, and one-dimensional distributive tactile sensors for normal force feedback and contact pose estimation. To mimic an actual real-world finger, the density of each finger link is set to 10000 Kg/m^3. The manipulated objects for four-finger hand and three-finger hand are approximately 0.5 Kg and 0.3 Kg. The 3D mesh models of objects are unknown to the planner. Rather, a vision system can be employed to obtain the pose of the object by tracking the features on it. Currently, the object pose is obtained from Mujoco. In future real world experiments, approach in [21] can be employed to estimate the pose of objects.

Parameter Lists

The parameter values for the LP (9.6) are: $w_1 = 0.99, w_2 = 0.01$. $q^i_{\text{thres}} = 0.5(q^i_{\text{max},k} - q^i_{\text{min},k})/2$. $\dot{q}_{\text{min,k}} = -1$ rad/s, $\dot{q}_{\text{max,k}} = 1$ rad/s, $\sigma = 0.002$ rad/s, $\delta = 10^{-5}$. The parameter values for velocity-force controller are: $K_v = 0.1 \sim 0.25 \times \text{diag}([1, 1, 1]), K_f = 1.5 \sim 3.4 \times \text{diag}([1, 1, 1])$. The planner time step $T_s = 2$ ms, and the simulation time step $t_s = 0.5$ ms.

The parameters of RMC are the same as Chapter 8.

Simulation Results

Finger gaiting on smooth surface under uncertainties

The proposed finger gaits planning and the RMC algorithm are validated by a lifting and rotating task. The desired object motion is to move along Z-axis by 11 mm, rotate continuously around Z-axis with 0.2 rad/s, and rotate sinusoidally around Y-axis with 0.4 rad/s. The manipulation surface of the object is smooth. The following uncertainties are included in **every** simulation of this section: 1) object dynamics uncertainty: The object has 20% mass and 50% moment of inertia uncertainties; 2) COM uncertainty: the COM position has $(3, 3, -10)$ mm offset and principal axes has $(0.1, 0.1, 0.1)$ rad offset represented by Euler angle; 3) tactile uncertainty: the contacts measured by tactile sensors have $(2, 2, 2)$ mm position offset and $(0.05, 0.05, 0.05)$ rad orientation offset, plus additional noises; 4) contact dynamics uncertainty: the planner uses Coulomb friction and point contact with friction (PCWF) model, while the contacts in the simulator also contain torsional friction and rolling friction. The stiffness $k_c = 15783$ and damping $b_c = 253$.

The performances of manipulating a cylindrical object without/with the proposed finger gaits planner are presented in Fig. 9.5. The initial and the current pose of the object are shown by white and red arrows. Without the proposed finger gaits planner, the object can not be rotated over 90° due to the decreasing finger manipulability, as shown in Fig. 9.5 (Top).

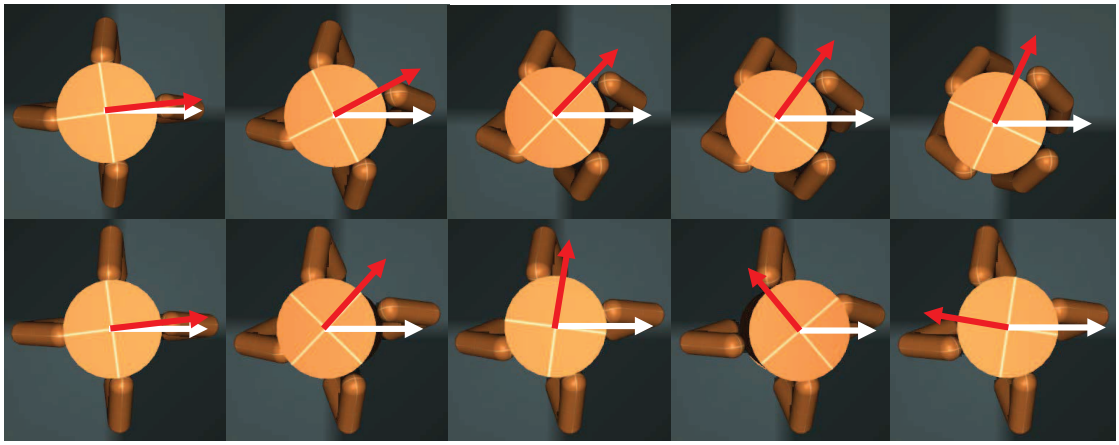


Figure 9.5: A lift and rotation task without (Top) and with (Bottom) the finger gaits planner.

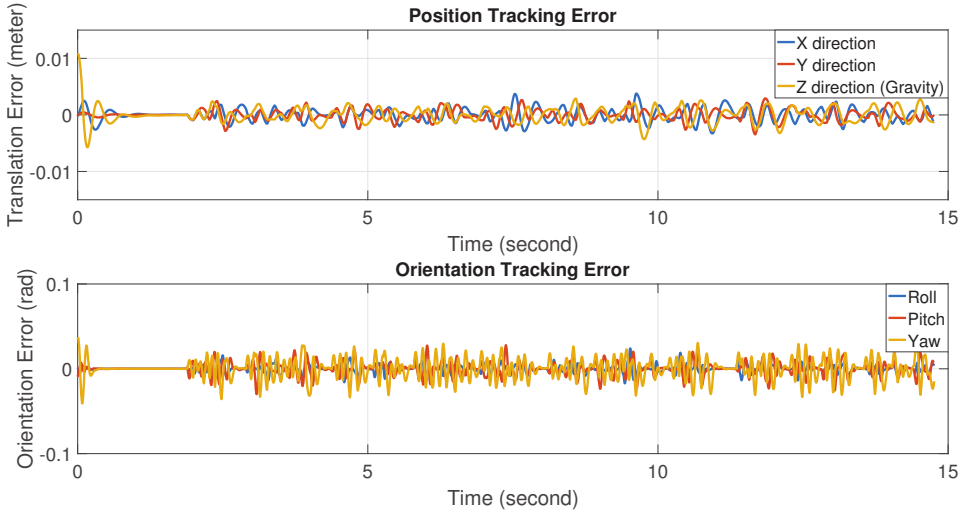


Figure 9.6: Tracking errors for the dual-stage optimization based planner.

With the proposed finger gaits planner, the object can be rotated continuously with the desired velocity, as shown in Fig. 9.5 (Bottom). The finger gaits planner ensures the grasp quality (9.6) is kept above a threshold, and the RMC guarantees robust stability and robust performance. The computation time for solving the (9.6) and (7.10) is less than 1 ms for each planning step.

Figure 9.6 shows the pose tracking errors during the lifting and rotation. RMC is able to drive the object to track the desired motion in $0 \sim 2$ secs. The finger gaiting is triggered at 2 secs since the grasp quality drops below a threshold. The pose errors do not converge to zero due to the disturbance introduced by contact allocating. The maximum position error

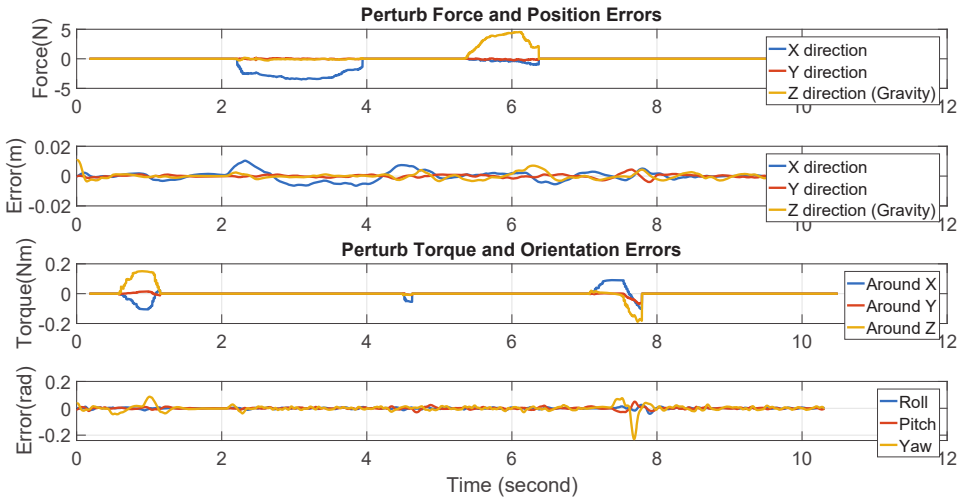


Figure 9.7: The response of two-level planner under external disturbances.

is 1.6 mm in x direction, and the maximum orientation error is 0.019 rad (1.09°) around z axis.

The disturbance rejection of the proposed two-level planner is shown in Fig. 9.7. Besides the disturbances from various uncertainties, external perturbation force/torque are exerted to the object. The force perturbation and the associated position errors are shown in Fig. 9.7(a)(b), and the torque perturbation and the associated orientation errors are shown in Fig. 9.7(c)(d). The system can resist at least 5 N force and 0.2 Nm torque in different directions without going unstable.

The robustness of the proposed two-level planner to different shapes is demonstrated by lifting and rotating an ellipsoid, as shown in Fig. 9.8. Figure. 9.8(top) is the top view, and Fig. 9.8(bottom) is the lateral view. The mass of the ellipsoid is 0.34 Kg. An identical two-level planner (finger gaits planner + RMC) is used for the ellipsoid manipulation, though the ellipsoid has different geometries and dynamics with cylinder.

The quality rate $\dot{Q}$ in (9.6) during a typical contact relocation period are shown in Fig. 9.9. The rate is above the δ, which means that the proposed finger gaits planner is able to continuously improve the grasp quality.

The robustness of RMC to different contact conditions has been presented in Section 8.5. However, RMC requires a conservative pyramid approximation of the friction cone specified by the Coulomb friction coefficient μ_c. However, this approximation might be too conservative if the contact dynamics are uncertain. One potential solution is to detect slippage and adaptively adjust μ_c used in RMC, and slippage detection can be difficult by 1D tactile sensor [29]. In this chapter, the finger gaits planner is employed to relocate the slipping finger if the quality drops below a threshold, instead of developing complex algorithm to prevent slippage. Besides other uncertainties, the nominal Coulomb friction coefficient $\bar{\mu}_c$ in

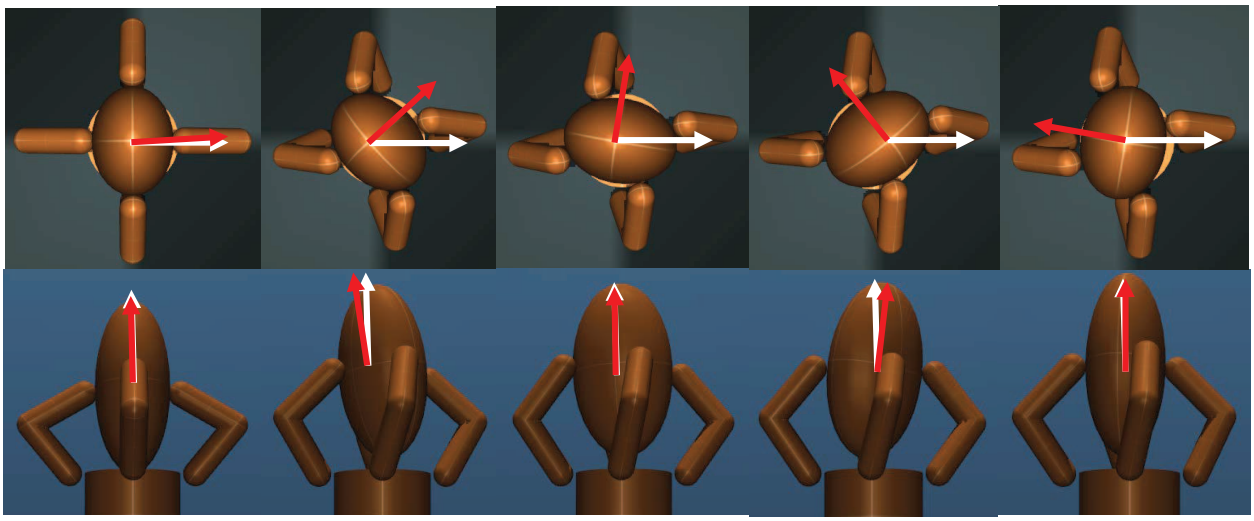


Figure 9.8: Lift and rotation task for ellipsoid using an identical planner as cylinder.

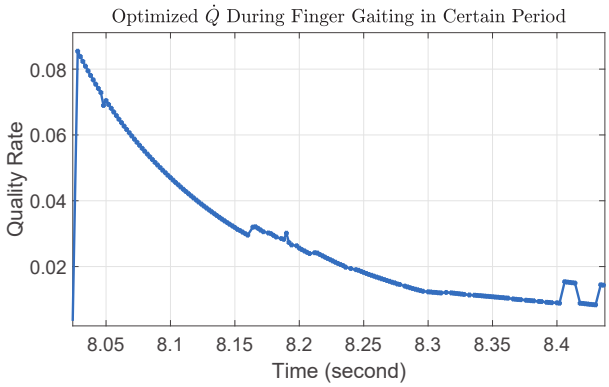


Figure 9.9: Optimal quality rate from (9.6) in a typical contact relocation period.

the planner is set as 0.5236, while the actual $\mu_c = 0.4$. The unexpected slippage of the fingers is compensated by the finger gaits planner. The tracking errors of the two-level planner under friction overestimation are shown in Fig. 9.10.

Finger gaiting of three-finger hand

Finally, we demonstrate the proposed two-level planner on different hands and tasks. A box flipping task using a three-finger hand shown in Fig. 9.4(b) is illustrated. The desired flipping speed is 0.5 rad/s. The box to be manipulated has 20% mass and 50% MoI uncertainties. Compared with above simulations, the box has sharp edge, and the task cannot be finished without changing manipulation surfaces. Therefore, a jump controller is desired to drive

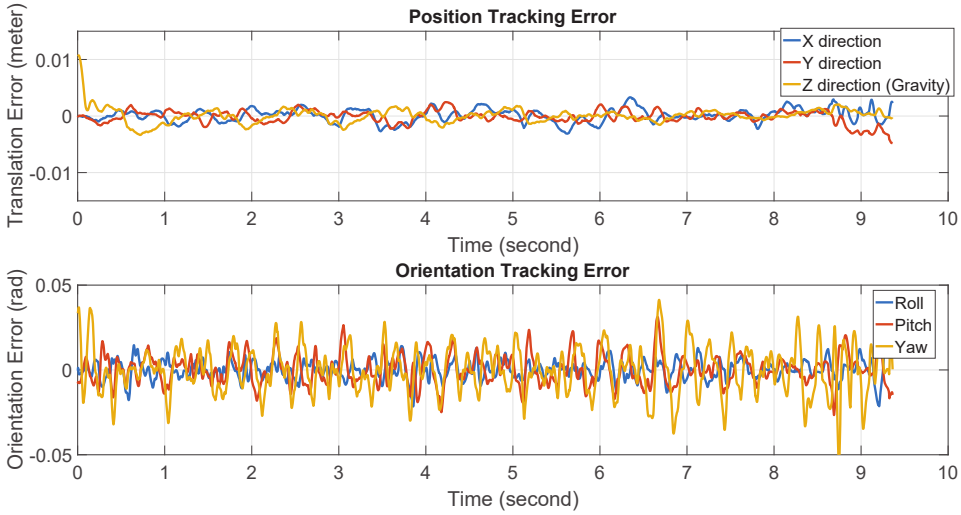


Figure 9.10: Tracking errors under friction overestimation.

the middle finger through the edge, as shown in Fig. 9.11 and 9.12. Figure 9.11 shows the snapshots of the box flipping process. The finger gaiting starts whenever losing contact or the quality drops below the threshold, and the jumping starts from 1.78 secs and rotates along the edge until contacting with the other surface. Compared with sliding, the re-contacting after jumping results in larger disturbance, thus the pose tracking errors oscillate to some extent, as shown in Fig. 9.12. The fingers on two sides manipulate the box by RMC. Notice though RMC is not able to maintain the robust performance during jumping, since the remaining 6 DOFs cannot form force closure grasps and the box cannot track the desired pitch motion, as shown in Fig. 9.12(b). The manipulation employs point contact with friction (PCWF) model, while the contact in Mujoco is set up with both torsional friction and rolling friction. The additional frictions act as a disturbance and the value is shown in Fig. 9.12(c).

9.5 Chapter Summary

This chapter proposed a two-level optimization based planner, which includes a velocity-based finger gaits planner and a robust manipulation controller, to achieve real-time finger gaiting under different types of uncertainties. The finger gaits planner searches optimal velocities to improve the object grasp quality and the hand manipulability, rather than directly finding optimal contact points by nonlinear programming methods. The proposed planner is computationally efficient and can be solved in real-time. Besides, the planner does not rely on precise 3D reconstruction for surface modeling and high resolution encoders for velocity measurements. The presented two-level optimization based planner can handle 50% mass and 80% moment of inertia uncertainties of the object, robust to the friction

Figure 9.11: Snapshots of box flipping using a three-finger hand.

overestimation and unexpected slippage, and address the complex dynamically critical tasks such as box-flipping with low-DOF hands. Simulations showed that the proposed method can achieve real-time finger gaiting, and realize long-range object motions that are infeasible without the proposed finger gaits planner.

In the future, the authors would like to test the proposed method on more complex objects such as non-convex ones, and perform experiments on a real world robotic hand.

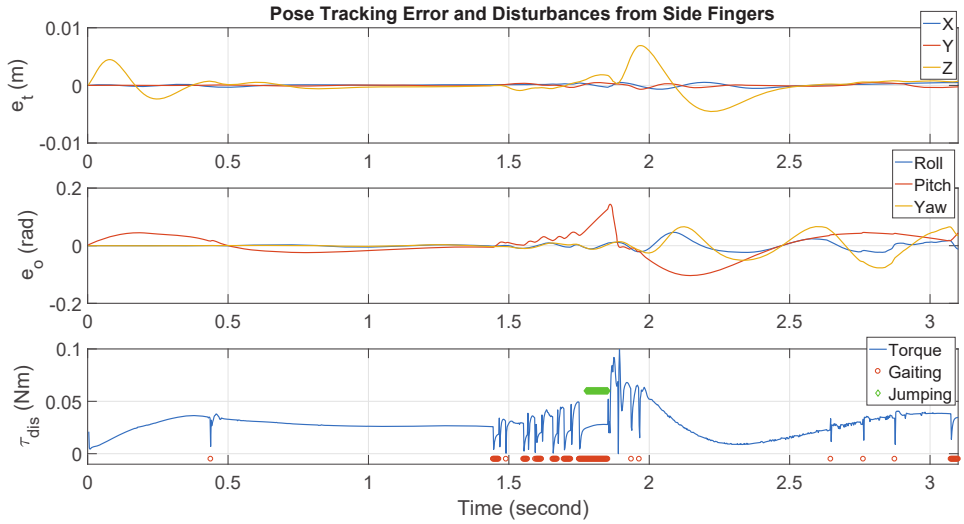


Figure 9.12: Tracking errors and the disturbance torque from sides fingers in box flipping.

Part III

Assembly

Chapter 10

Learning Industrial Assembly by Guided-DDPG

10.1 Introduction

Chapter 2 $\sim$ 6 described the development of grasp planning algorithms to extend the dexterity of industrial manipulators in kinematic level. Chapter 7 $\sim$ 9 described the development of manipulation control algorithms to extend the dexterity in dynamic level. In this chapter, the dexterity of the industrial manipulators will be further extended to skill level by developing a learning framework for industrial assembly. Automatic precision assembly is important for industrial manipulators to improve the efficiency and reduce the cost. Most of the current assembly tasks rely on dedicated manual tuning to provide trajectories for specific tasks, which requires intensive labors and is not robust to uncertainties. To reduce the human involvement and increase the robustness to uncertainties, more researches are focusing on learning the assembly skills.

There are three types of learning in Psychology [41]: classical conditioning, observational learning and operant conditioning. The second and third types correspond to supervised learning and reinforcement learning, respectively. The supervised learning is ideal when the training data is sufficient. Practically, collecting data is inefficient under various uncertainties of the environment. A Gaussian mixture model (GMM) is trained in [93] from human demonstration to learn a peg hole insertion skill. The peg hole insertion task is simplified by constraining the policy into planar motion and the trained policy is not adaptable to different environments.

The reinforcement learning (RL) learns a sequence of optimal actions by exploring the environment to maximize the expected reward. Different types of RL methods include the direct policy gradient such as REINFORCE [103], Q-learning based methods such as deep Q-learning (DQN) [66], as well as the actor-critic framework such as deep deterministic policy gradient (DDPG) [55] or proximal policy optimization (PPO) [85]. These methods are called model-free RL since the dynamics model is not used during exploration. Despite

lack of dynamics, the model-free RL has been successfully applied to assembly tasks [100, 39]. The model-free RL requires considerable data to explore the state/action space and reconstruct the transitions of the environment. Consequently, it is less data-efficient and time-efficient.

Model-based RL is proposed to increase the data efficiency [47, 48]. It fits dynamics models and applies optimal control such as iterative linear quadratic regulator (iLQR) or iterative linear quadratic Gaussian (iLQG) [94] to compute the optimal trajectories. The exploration is conducted by adding random noise to the actions during the optimization. Then the optimized trajectories are used to train a neural network policy in a supervised manner. Compared with model-free RL, the model-based RL has larger exploit-exploration ratio, thus explores narrower space and converges faster than the model-free RL. The performance of the model-based RL depends on the behavior of the optimal controller (i.e. supervisor), which in turn is effected by the accuracy of the local dynamics model. For the rigid robot dynamics with force/torque as states, the dynamics model is less smooth[1], which makes the dynamics fitting not effective. Consequently, the model-based RL cannot converge consistently. In practice, people usually use soft robotics model (Baxter, PR2) [48] with position/velocity states by ignoring the force/torque feedback.

This chapter proposes a learning framework to train a more natural assembly policy by incorporating both the force/torque and the positional feedback signals. The proposed framework combines the model-based RL with the model-free actor-critic to learn the manipulation skills for precision assembly tasks. The model-based RL computes for the optimal trajectories with both positional and force/torque feedback. The performance of the controller might be affected by the smoothness of the local fitted dynamics model. To avoid the problem of inconsistency or tedious parameter tuning of the optimal controller, a critic network is introduced to learn the correct critic value (Q-value). Instead of training the policy network by pure supervision, we train an actor network by combining the supervised learning with the policy gradient. To accelerate the training efficiency of the critic network, the Q-value from the optimal control is employed to train the critic network.

The contributions of this work are as follows. First, the optimal controller is able to constrain the exploration space in safe region compared with the random exploration at the first iterations of actor-critic methods. Secondly, the optimal controller is more data-efficient when exploring in a narrower space and solving for the optimal trajectory mathematically. Thirdly, the combined critic network is able to address the potential inconsistency and instability of the optimal controller caused by the rigid robotics system and force/torque feedback, and build up a ground truth critic for the policy network.

The remainder of this chapter is described as follows. The related work is stated in Section 10.2, followed by a detailed explanation of the proposed learning framework in Section 10.3. Simulation and experiment results are presented in Section 10.4. Section 10.5 concludes the chapter and proposes future works.

[1]The dynamics change dramatically as the trajectory slightly changes.

10.2 From Model-Free RL to Model-Based RL

The objective of an assembly task is to learn an optimal policy $\pi_\theta(a_t|o_t)$ to choose an action a_t based on the current observation o_t in order to maximize an expected reward:

$$\min_{\pi_\theta} E_{\tau\sim\pi_\theta}(l(\tau)), \tag{10.1}$$

where θ is the parameterization of the policy, $\tau = \{s_0, a_0, s_1, a_1, ..., s_T, a_T\}$ is the trajectory, $\pi_\theta(\tau) = p(s_0)\prod_1^T p(s_t|s_{t-1}, a_{t-1})\pi_\theta(a_t|s_t)$, and l is the loss of the trajectory τ.

Equation (10.1) can be solved by optimization once a global dynamics $p(x_t|x_{t-1}, u_{t-1})$ is explicitly modeled. For a contact-rich complex manipulation task, the global dynamics model is extremely difficult to obtain. Therefore, the assembly task either avoids using dynamics [39] or fits the a linear dynamics model [93, 47, 48].

On one hand, the RL without dynamics requires excessively data to explore the space and locate to the optimal policy due to the potential high-dimensionality of the action space. On the other hand, the performance of the [47, 48] can be downgraded once the robotic system is rigid or the force/torque feedback is included in the optimal controller.

We propose a learning framework that combines the actor-critic framework and optimal control for efficient high-accuracy assembly. The optimal controller is adapted from the model-based RL [47], while the actor-critic framework is modified from the DDPG algorithm. These two algorithms will be briefly introduced below.

Deep Deterministic Policy Gradient (DDPG)

The DDPG algorithm collects sample data (s_j, a_j, s_{j+1}, r_j) from the replay buffer R and trains a critic network Q_ϕ and actor network u_θ parameterized by ϕ and θ. More specifically, the critic network is updated by:

$$\begin{aligned} \phi &\leftarrow \underset{\phi}{\operatorname{argmin}} \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} \left(y_j - Q_\phi(s_j, a_j)\right)^2, \\ y_j &= r_j + \gamma Q_{\hat{\phi}}(s_{j+1}, u_{\hat{\theta}}(s_{j+1})), \end{aligned} \tag{10.2}$$

where N_{dd} is the batch size for DDPG, $\hat{\phi}, \hat{\theta}$ are parameters of the target critic network and target actor network, and γ is the discount for future reward.

The policy network is updated by:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmax}} \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} Q_{\hat{\phi}}(s_j, u_\theta(s_j)), \tag{10.3}$$

where θ is the parameters for the policy network to be optimized. Policy gradient is applied to update the parameters of the actor network:

$$\theta \leftarrow \theta + \alpha \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} \nabla_a \hat{Q}(s, a)|_{s=s_j, a=a_j} \nabla_\theta u_\theta(s)|_{s=s_j}, \tag{10.4}$$

where the α is the learning rate of the actor network.

The target networks are updated by

$$\begin{aligned}\hat{\phi} &\leftarrow \delta\phi + (1-\delta)\hat{\phi},\\ \hat{\theta} &\leftarrow \delta\theta + (1-\delta)\hat{\theta},\end{aligned} \tag{10.5}$$

where δ is the target update rate and is set to be small value ($\delta \approx 0.01$).

Guided Policy Search (GPS)

With the involvement of guiding distribution $p(\tau)$,Problem (10.1) can be rewritten as

$$\min_{\pi_\theta, p} E_p(l(\tau)), \quad s.t. \;\; p(\tau) = \pi_\theta(\tau). \tag{10.6}$$

GPS solves (10.6) by alternatively minimizing the augmented Lagrangian with respect to primal variables p, π_θ and updating the Lagrangian multipliers λ. The augmented Lagrangian for θ and p optimization are:

$$\begin{aligned}L_p(p,\theta) &= E_p(l(\tau)) + \lambda\left(\pi_\theta(\tau) - p(\tau)\right) + \nu D_{KL}\left(p(\tau)\|\pi_\theta(\tau)\right),\\ L_\theta(p,\theta) &= E_p(l(\tau)) + \lambda\left(\pi_\theta(\tau) - p(\tau)\right) + \nu D_{KL}\left(\pi_\theta(\tau)\|p(\tau)\right),\end{aligned} \tag{10.7}$$

where λ is the Lagrangian multiplier, ν is the penalty parameter for the violation of the equality constraint, and D_{KL} represents the KL-divergence. The optimization of primal variable p is called trajectory optimization. It optimizes the guiding distribution p with learned local dynamics. To assure the accuracy of dynamics fitting, the optimization is constrained within the trust region ϵ:

$$\min_p L_p(p,\theta), \quad s.t. \;\; D_{KL}(p(\tau)\|\hat{p}(\tau)) \le \epsilon, \tag{10.8}$$

where $\hat{p}$ is the guiding distribution of the previous iteration. The Lagrangian of (10.8) is:

$$\mathcal{L}(p) = L_p(p,\theta) + \eta(D_{KL}(p(\tau)\|\hat{p}(\tau)) - \epsilon), \tag{10.9}$$

where η is the Lagrangian multiplier for the constraint optimization. With the Gaussian assumption of the dynamics, (10.9) is solved by iLQG [97]. To avoid large derivation from the fitted dynamics, η is adapted by comparing the predicted KL-divergence with the actual one.

The optimization of the policy parameters θ can be written as a supervised learning problem. With the Gaussian policy $\pi_\theta(a_t|o_t) = \mathcal{N}(u_\theta(o_t), \Sigma_t^\pi)$, we can rewrite $L_\theta(p,\theta)$ in (10.7) as:

$$\begin{aligned}L_\theta(\theta,p) = \frac{1}{2N_b}\sum_{i,t=1}^{N_b,T} E_{p_i(s_t,o_t)}[\mathrm{tr}\left(C_{ti}^{-1}\Sigma_t^\pi\right) - \log|\Sigma_t^\pi| + \\ \left(u_\theta(o_t) - u_{ti}^p(s_t)\right)^T C_{ti}^{-1}\left(u_\theta(o_t) - u_{ti}^p(s_t)\right) + 2\lambda_t^T u_\theta(o_t)],\end{aligned} \tag{10.10}$$

where $p_i(u_t|s_t) \sim \mathcal{N}(u_{ti}^p(s_t), C_{ti})$ is the guiding distribution. Equation (10.10) contains the decoupled form of the variance optimization and policy optimization. Refer [48] for more details.

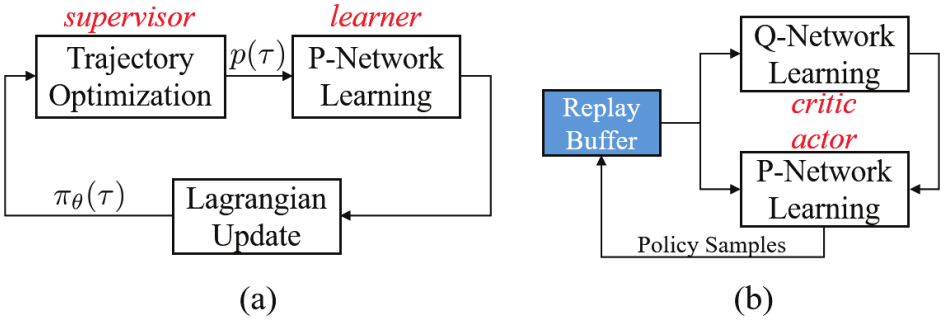


Figure 10.1: (a) Guided Policy Search (GPS). (b) deep deterministic policy gradient (DDPG).

Comparison of GPS and DDPG

GPS decouples RL into a trajectory optimization (*supervisor*) and a supervised policy network learning (*learner*), as shown in Fig. 10.1(a). The performance of the learner relies on the quality of the supervisor. By fitting the dynamics from sampling data and computing the supervisor with the optimal control, GPS is more efficient than the DDPG and many other model-free RL algorithms. However, the performance of the learner would be compromised if the system has high stiffness and has force/torque feedback as states due to the less smooth dynamics and smaller trust region.

In comparison, DDPG uses rollout samples to jointly train the Q-network (*critic*) and policy network (*actor*), as shown in Fig. 10.1(b). The critic gradually builds up the Q-value from physical rollouts, and the Q-value is applied to train the actor network based on policy gradient. The actor-critic framework provides more stable policy in the tasks with non-smooth dynamics. These tasks are common in high precision industrial assembly where the system has higher stiffness and contains force/torque feedback in the states. Despite the reliable performance, the actor-critic framework is less data efficient due to the intensive exploration, which is usually unnecessary since assembly tasks only requires exploration in narrow trajectory space.

10.3 Guided-Deep Deterministic Policy Gradient (Guided-DDPG)

Precision industrial assembly usually has large system stiffness in order to achieve precise tracking performance and reduce the vibration. With large stiffness, small clearance and force/torque feedback, both the model-free RL and model-based method cannot accomplish the task efficiently and stably. In this chapter, we propose a learning framework that combines the actor-critic with the model-based RL for high precision industrial assembly. The

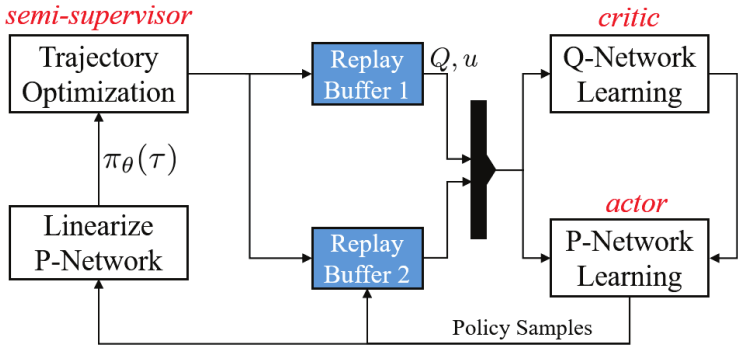


Figure 10.2: Illustration of the proposed guided-DDPG.

framework is named as guided-deep deterministic policy gradient (guided-DDPG). Guided-DDPG behaves more efficient than the actor-critic and more stable/reliable than the model-based RL.

Figure 10.2 illustrates the proposed guided-DDPG algorithm. Due to the discontinuity of the fitted dynamics in rigid precise systems, the trajectory optimization can have inconsistent behavior or requires dedicated parameter tuning. Therefore, a pure supervised learning from trajectory optimization cannot fulfill the task consistently. The actor-critic is incorporated to the framework to address this issue. The trajectory optimization serves as a *semi-supervisor* to train the actor-critic to establish the initial critic and constrain the network in narrow task space. The involvement of the supervision will be reduced as the training progresses and the critic network becomes more accurate, since the actor-critic exhibits superior performance than the semi-supervisor.

To be more specific, the trajectory optimization (semi-supervisor) has the following form:

$$\min_{p} E_p(l(\tau)), \quad s.t. \;\; D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) \leq \epsilon, \tag{10.11}$$

where $\hat{p}_\theta$ is set as the trajectory distribution generated by actor policy at the first sub-iteration, and is set as the previous trajectory distribution $\hat{p}$ for the successive $N_{trajopt} - 1$ sub-iterations. Equation (10.11) is optimized by the dual:

$$\max_{\eta}\{\min_{p} E_p(l(\tau)) + \eta(D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) - \epsilon)\}. \tag{10.12}$$

The optimization of p is solved by LQG with fixed η and dynamics, and the optimization of η is done heuristically: decrease η if $D_{KL}(p(\tau)\|\hat{p}_\theta(\tau)) < \epsilon$, otherwise increase η. The trust region ϵ varies based on the expected improvement and actual one. ϵ would be reduced once the actual improvement is far smaller from the expected one, thus the network focuses on penalizing the KL divergence from $\hat{p}_\theta(\tau)$.

We collect the trajectory after $N_{trajopt}$ sub-iterations to replay buffer R_1 for supervised training of actor-critic nets, and feed all the sample data during $N_{trajopt}$ executions to replay

buffer R_2. With the supervision from R_1, the critic is trained by:

$$\phi \leftarrow \underset{\phi}{\operatorname{argmin}} \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} (y_j - Q_\phi(s_j, a_j))^2 + w_{to} \frac{1}{N_{to}} \sum_{i=1}^{N_{to}} \|Q_\phi(s_i, a_i) - Q_i^{to}\|^2 \tag{10.13}$$

where w_{to}, N_{to} are the weight and batch size of the semi-supervisor, y_j has the same form as (10.2). (s_i, a_i, Q_i^{to}) is the supervision data from R_1, and (s_j, a_j, r_j, s_{j+1}) is the sample data from R_2.

The actor is trained by:

$$\theta \leftarrow \underset{\theta}{\operatorname{argmax}} \frac{1}{N_{dd}} \sum_{j=1}^{N_{dd}} Q_{\hat{\phi}}(s_j, u_\theta(s_j)) + w_{to} \frac{1}{N_{to}} \sum_{i=1}^{N_{to}} \|u_\theta(s_i) - a_i\|^2 \tag{10.14}$$

The supervision weight w_{to} decays as the number of training rollouts N_{roll} increases. We use $w_{to} = \frac{c}{N_{roll}+c}$, where c is a constant to control the decay speed.

The guided-DDPG algorithm is summarized in Alg. 10. The critic and actor are initialized in Line 2. Guided-DDPG runs for EP epochs in total. In each epoch, semi-supervisor is first executed to update the trajectories for supervision. With the high stiffness, small clearance and the force/torque feedback, the fitted dynamics (Line 7) is discontinuous and has small trust region. Therefore, the trajectories generated from the semi-supervisor might be sub-optimal. Nevertheless, they are sufficient to guide the initial training of the actor-critic. The actor-critic is trained in Line (12 - 22) following the standard procedure of DDPG with the modified objective function (Line (19)). The supervision weight w_{to} is decreased as the training progresses due to the superior performance of the actor-critic than the semi-supervisor.

10.4 Simulations and Experiments

This section presents both the simulation and experimental results of the guided-DDPG to verify the effectiveness of the proposed learning framework. The videos are available at [102].

To compare the performance of the guided-DDPG with other state-of-the-art RL algorithms, we built up a simulation model using the Mujoco physics engine [96]. The host computer we used was a desktop with 32 GB RAM, 4.0 GHz CPU and GTX 1070 GPU. A 6-axis UR5 robot model from universal robotics was used to perform the tasks. Two different assembly tasks were simulated, the first one was the Lego brick insertion, and the second one was the U-shape joint assembly, as shown in Fig. 10.3.

Parameter Lists

The number of the maximum epoch is set to $EP = 100$, initial number of rollouts for DDPG and trajectory optimization were $N_{ddpg} = 21$ and $N_{trajopt} = 3$, respectively. To ensure

Algorithm 10 Guided-DDPG

input: $EP, N_{ddpg}, N_{inc}, N_{trajopt}, N_{roll} = 0, R_{1/2} \leftarrow \Phi$
init: $Q_\phi(s,a), u_\theta(s)$, set target nets $\hat{\phi} \leftarrow \phi, \hat{\theta} \leftarrow \theta$
for $epoch = 0 : EP$ **do**
  $p_{prev} \leftarrow u_\theta$
  for $it = 0 : N_{trajopt}$ **do**
    $\mathcal{S} \leftarrow sample\_data(p_{prev})$, $R_2 \leftarrow R_2 \cup \mathcal{S}$
    $f_{dy} \leftarrow fit\_dynamics(\mathcal{S})$
    $\hat{p}_\theta \leftarrow linearize\_policy(p_{prev}, \mathcal{S})$
    $p \leftarrow update\_trajectory(f_{dy}, \hat{p}_\theta), p_{prev} \leftarrow p$
  end for
  $S \leftarrow sample\_data(p)$, $R_1 \leftarrow R_1 \cup \mathcal{S}, R_2 \leftarrow R_2 \cup \mathcal{S}$
  for $it = 0 : N_{ddpg}$ **do**
    $\mathcal{N}_{ex} \leftarrow exploration\_noise()$
    $s_0 \leftarrow observe\_state(), w_{to} = \frac{c}{c+N_{roll}{}^{++}}$
    for $t = 0 : T$ **do**
      $a_t = u_\theta(s_t) + \mathcal{N}_{ex}(t)$, observe s_{t+1}, r_t
      $R_2 \leftarrow R_2 \cup (s_t, a_t, s_{t+1}, r_t)$
      sample N_{to}, N_{dd} transitions from R_1, R_2
      update critic and actor nets by (10.13) and (10.14)
      update target nets by (10.5)
    end for
  end for
  $N_{ddpg} \leftarrow N_{ddpg} + N_{inc}$
end for

less visits of trajectory optimization as the training progresses, we increased the number of rollouts by $N_{inc} = 15$ for each DDPG iteration. The sizes of the replay buffer R_1, R_2 were 2000 and 1E6, respectively. The soft update rate $\gamma = 0.001$ in (10.5). The batch size for trajectory optimization N_{to} and DDPG N_{dd} were both 64. The algorithm used a cost function $l(s,a) = 0.0001\|a\|_2 + \|FK(s) - p_{tgt}(s)\|_2$, where FK represents the forward kinematics and p_{tgt} is the target end-effector points.

Simulation Results

The simulation results on U-shape joint assembly and Lego brick insertion are shown by Fig. 10.4. Both simulations were trained with assembly clearance as 0.1 mm. Guided-DDPG takes poses and force/torque measurements of the end-effector as the states, and generates joint torques as action to drive the robot. The U-shape joint has more complicated surface than the Lego brick, and a successful assembly requires matching the shapes twice, as shown in Fig. 10.4 (Top). Despite the difficulties, the proposed algorithm was able to train the

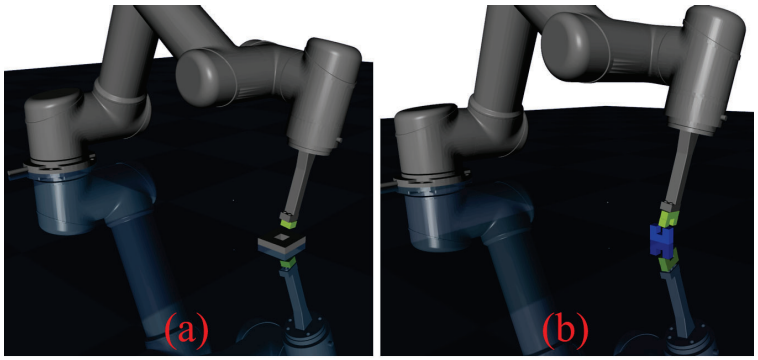


Figure 10.3: Two simulation tasks for algorithm evaluation.

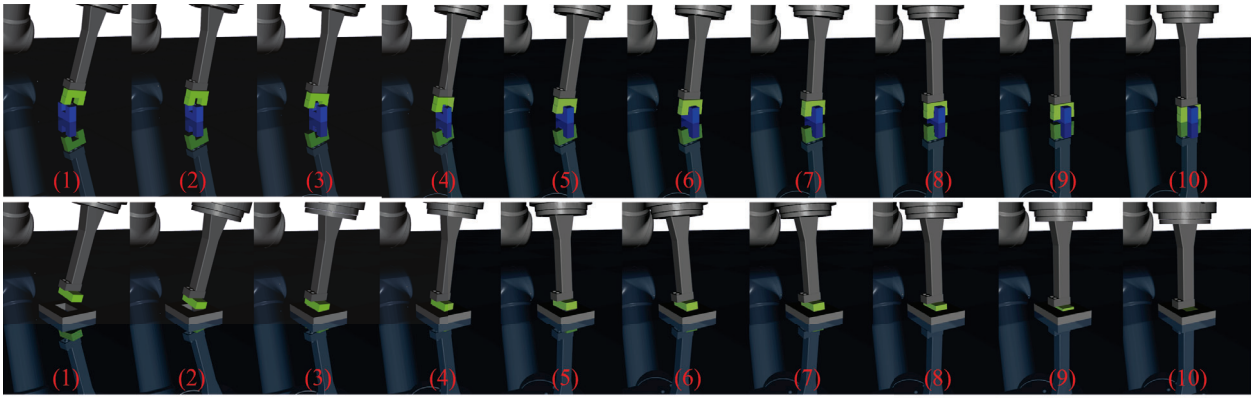


Figure 10.4: Simulation animations on (Top) U-shape joint assembly and (Bottom) Lego brick insertion.

policy within 1000 rollouts. We also visualized the adaptability of the trained policy on the Lego brick insertion task, as shown in Fig. 10.4 (Bottom). The policy was trained with a brick of size 2×2 and clearance 0.1 mm and tested with a brick of size 4×2 and clearance 1 μm. Moreover, the brick position had an unknown offset (1.4 mm) to the network. The proposed network was able to address these uncertainties and successfully inserted the brick to a tighter hole with uncertain position.

Comparison of Different Supervision Methods

The proposed learning framework guides both the critic and actor. To illustrate the necessity of the proposed guidance, we compared the results of guided-DDPG with several other supervision methods, including the guided-DDPG with partial guidance, pure-DDPG with supervision data to replay buffer (no supervision on objective function) and the pure-DDPG. The result was shown in Fig. 10.5. The proposed guided-DDPG achieved the best perfor-

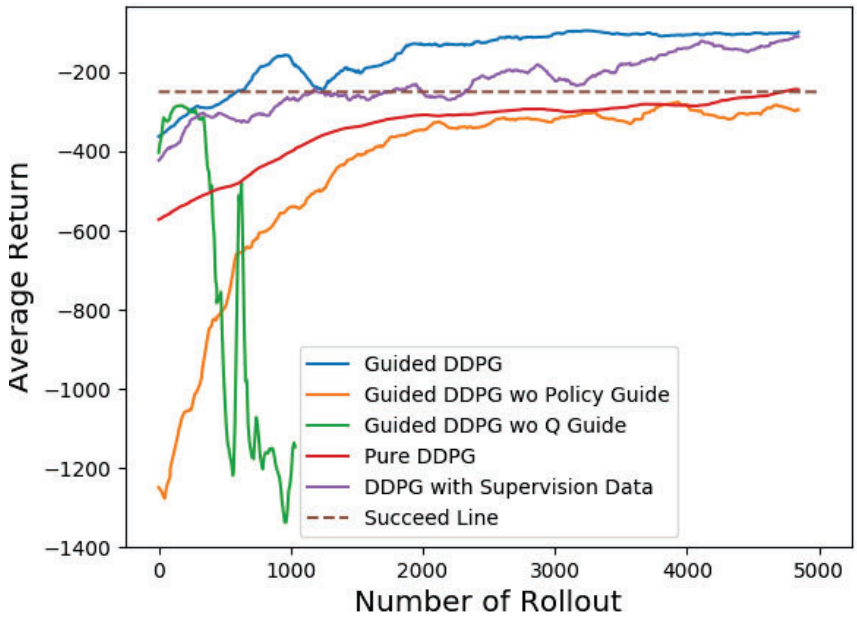


Figure 10.5: Comparison of different supervision methods with Lego brick insertion.

mance. The partial guidance without critic (Fig. 10.5 Green) was able to guide the actor and realized safe exploration at the beginning. However, the actor network behaved worse as the involvement of the semi-supervisor reduced and the weight of the critic increased, since the critic is trained purely by the contaminated target actor (10.2). In contrast, the partial guidance without actor (Fig. 10.5 Orange) had poorly behaved actor since the actor was trained purely by the policy gradient using the contaminated critic (10.3). The pure-DDPG with supervision data (Fig. 10.5 Purple) achieved better performance than pure-DDPG, since the trajectories obtained from semi-supervisor were better behaved than the initial rollouts of DDPG. This kind of supervision is similar with the human demonstration in [100].

Effects of the Supervision Weight w_{to}

The supervision weight w_{to} balances the model-based supervision and model-free policy gradient in actor/critic updates, as shown in (10.14) and (10.13). The results of different weights on Lego brick insertion are shown in Fig. 10.6. With $c = 1$, the supervision weight is $w_{to} = \frac{1}{1+N_{roll}}$. The weights starts with 1 and decays to 0.001 as $N_{roll} = 1000$, while $c = 100$ makes w_{to} decay to 0.1 as $N_{roll} = 1000$. Slower decay provides excessive guidance by the semi-supervisor and contaminates the original policy gradient and makes the DDPG unstable. Empirically, $c = 1 \sim 10$ achieves comparable results.

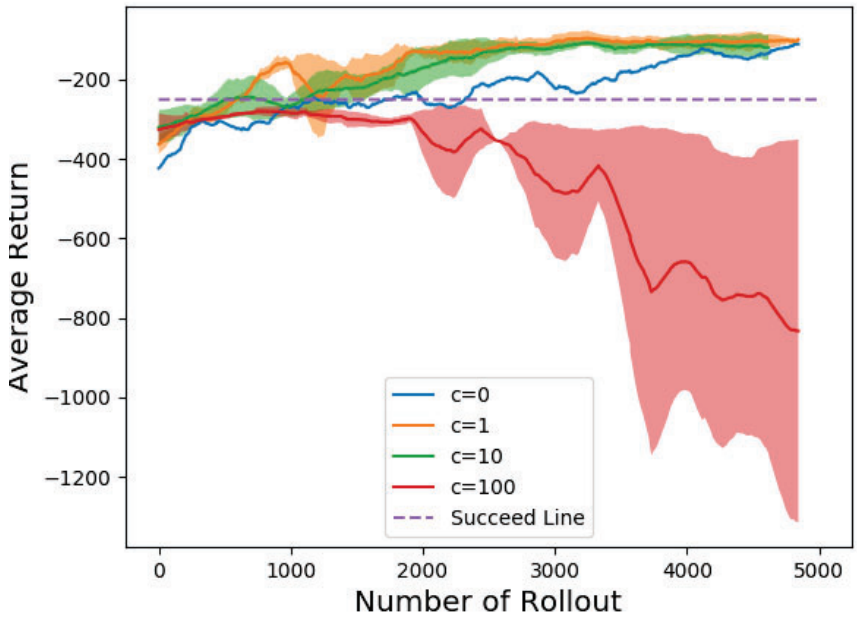


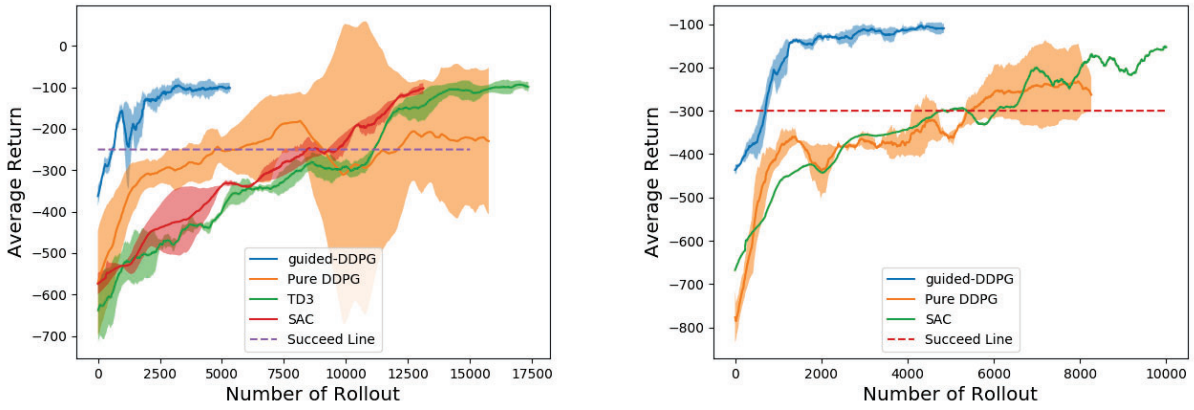
Figure 10.6: Illustration of the supervision weights on Lego brick insertion.

Comparison of Different Algorithms

The proposed learning framework was compared with other state-of-the-art algorithms, including the pure-DDPG, twin delayed deep deterministic policy gradients (TD3) [32] and the soft actor-critic (SAC) [36]. Default parameters were used for TD3, as shown in [77]. As for SAC, we used the default parameters in [77] with tuned reward scale as 10. The comparison result on Lego brick insertion task is shown in Fig. 10.7(a). The proposed guided-DDPG passed the success threshold (shaded purple line) at the 800 rollouts and consistently succeeded the task after 2000 rollouts. In comparison, the pure DDPG passed the success threshold at the 5000 rollouts and collapsed around 10000 rollouts. The performance of pure DDPG was inconsistent in seven different trials. TD3 and SAC had the similar efficiency with pure DDPG. The comparison of the algorithms on U-shape joint assembly is shown in Fig. 10.7(b). Similar with Lego brick insertion, the guided-DDPG achieved more stable and efficient learning. The time efficiency and data-efficiency of the DDPG and guided-DDPG are compared in Table 10.1.

Adaptability of the Learned Policy

The adaptability of the learned policy is discussed in this section. Three different types of uncertainties were considered. The first type was the unknown hole position. The learned

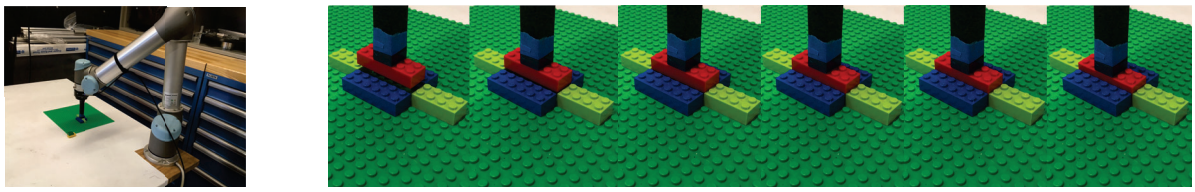


(a) (b)

Figure 10.7: Comparison of different algorithms for (a) Lego insertion and (b) joint assembly.

Table 10.1: Comparison of DDPG and guided-DDPG

items	DDPG	Guided-DDPG
time (min)	83	**37.3**
data (rollouts)	7000	**1500**



(a) (b)

Figure 10.8: (a) Experimental setup, and (b) experimental results for Lego brick insertion.

policy was able to successfully insert the brick when moving the hole to an uncalibrated position (maximum offset is 5 mm, hole has width of 16 mm). The second type of uncertainty was the shapes of peg/hole. We found that the learned policy is robust to different shapes shown in Fig. 10.9. Figure 10.9(a) shows the 2×2 brick used in training, Fig. 10.9(b) shows the 4×2 brick scenario, Fig. 10.9(c) shows the 4×2 brick with incomplete hole, and Fig. 10.9(d) shows a cylinder brick. The third type was the different clearance. The policy was trained with clearance 0.1 mm and tested successfully on insertion tasks with clearance 10 μm, 1 μ and 0. The simulation videos are available at [102].

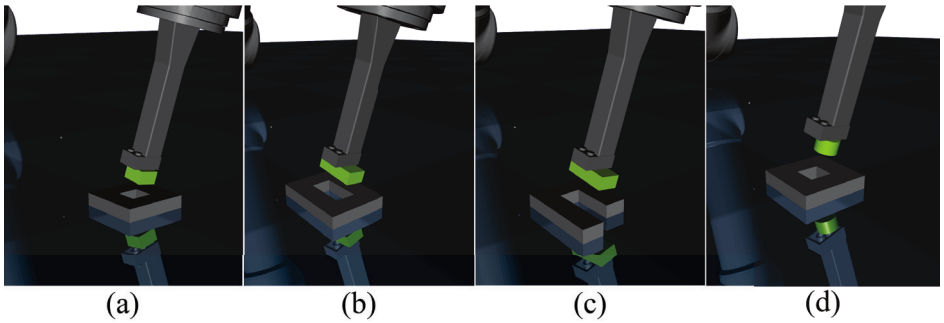


Figure 10.9: Adaptability validation of the proposed guided-DDPG.

Experimental Results

Experimental results are presented in this section. The Lego brick was attached to a 3D printed stick at the end-effector of the Universal robot (UR5). A Robotiq FT 300 force torque sensor was used to collect the force/torque signal at the wrist. The experimental setup is shown in Fig. 10.8(a). The policy took the estimated hole position and the force/torque reading as inputs, and generated transitional velocities for the end-effector. The velocity was tracked by a low-level tracking controller. The clearance of the Lego brick is less than 0.2 mm. The target position of the hole had 0.5 mm uncertainty, yet the policy was able to successfully locate the hole and insert the brick, as shown in Fig. 10.8(b). It took 2 hours for pure-DDPG to find a policy in the exploration space bounded within 1 mm around the hole, and took 1.5 hours for guided-DDPG to find a policy in a larger exploration space bounded within 3 mm around the hole. The experimental videos are shown in [102].

10.5 Chapter Summary

This chapter proposed a learning framework for automatic assembly task. The framework contains a trajectory optimization and an actor-critic structure. The trajectory optimization was served as a semi-supervisor to provide initial guidance to actor-critic, and the critic network established the ground-truth quality of the policy by learning from both the semi-supervisor and exploring with policy gradient. The actor network learned from both the supervision of the semi-supervisor and the policy gradient of the critic. The involvement of critic network successfully addressed the stability issue of the trajectory optimization caused by the high-stiffness and the force/torque feedback. The proposed learning framework constrained the exploration in a safe narrow space, improved the consistency and reliability of the model-based RL, and reduced the data requirements to train a policy. Simulation and experimental results verified the effectiveness of the proposed learning framework.

In the future, the authors would evaluate the algorithm on more realistic industrial applications such as connector insertion, furniture assembly and tight peg-in-hole tasks.

Chapter 11

Conclusions and Future Works

11.1 Conclusions

This dissertation developed algorithms to improve the dexterity of industrial manipulators. Three major levels of dexterity, including kinematic dexterity, dynamic dexterity and skill dexterity, were discussed and explored.

Overview

To achieve the kinematic dexterity and reduce the redesigning and reprogramming efforts to operate workpieces, a unified grasping framework with both customized end-effectors and general hands was developed and verified. First, a surface fitting algorithm was proposed for grasp planning with customized industrial grippers to extend the kinematic dexterity and minimize hardware modification (Chapter 2). The planning algorithm was incorporated with a learning-based explorer in Chapter 3 to improve the efficiency and keep the interpretability and reliability of the planning algorithm. Grasp planning algorithms for multi-fingered hands were further developed to extend the kinematic dexterity on more general grasping tasks. A finger splitting algorithm was developed in Chapter 4 to transfer grasps from parallel grippers to multi-fingered hands. A stronger optimization model was proposed in Chapter 5 to plan grasps with multi-fingered hands directly without the initialization of parallel grasps. A general grasping framework was finalized in Chapter 6 by incorporating the optimization model with a multi-dimensional surface fitting, to produce versatile grasps robustly under various uncertainties.

To form a stable grasp and operate workpieces without placing and re-grasping, the kinematic dexterity was incorporated with dynamic dexterity. First, a comprehensive in-hand manipulation architecture was proposed and validated in Chapter 7. Chapter 8 further proposed a robust manipulation controller to enhance the robustness to various uncertainties and undesired hand properties. A finger gaits planner was introduced in Chapter 9 to realize long-range object motion and reduce the cycle time of re-grasping.

Besides kinematic and dynamic dexterity, industrial manipulators are desired to develop skill dexterity to assemble workpieces into personalized products with minimal system reconfiguration. An intelligent assembly algorithm called guided-deep deterministic policy gradient was designed in Chapter 10 to learn assembly skills from uncertain environments. The summaries of different chapters are as follows.

Chapter Summary

Chapter 2 proposed an iterative surface fitting (ISF) algorithm to plan grasps for customized grippers. ISF searches for optimal grasps by iteratively solving for optimal palm pose and finger displacement with closed-form solutions. Guided sampling was introduced to initialize ISF searching and avoid local optima. The proposed grasp planning algorithm was applied to a series of simulations and experiments. ISF achieved 64.4 ms average searching time to find a collision-free grasp on the objects in simulations. A grasp planning experiment was further implemented in clutter environments to grasp objects from unsegmented point clouds.

Chapter 3 proposed a learning framework to plan robust grasps for customized grippers. The learning framework includes a low-level ISF planner from Chapter 2 and a high-level learning-based explorer. The learning-based explorer was introduced with a region-based convolutional neural network (R-CNN) to search for desired low-regret regions to initialize ISF search. A series of experiments on robotic bin picking were performed to evaluate the proposed method. Experiments indicated that the proposed learning framework with RCNN-ISF achieved a more efficient planning on heavy clutter environments, by significantly decreasing the average searching time from 17.23 secs to 1.52 secs.

Chapter 4 proposed a finger splitting strategy for grasp planning with multi-fingered hands by transferring the grasps of parallel grippers. The splitting was initialized by the planning result of the parallel gripper, and was optimized continuously by a novel iterative contact point optimization - palm pose optimization (CPO-PPO) algorithm. The CPO and PPO were both solved analytically. The iterative CPO-PPO algorithm was able to find a locally optimal collision-free grasp within one second on average for the objects studied in simulations.

Chapter 5 proposed an efficient optimization model for precision grasp planning. To optimize the quality and avoid collision, the planning problem was formulated into an optimization with penalties and solved by iterating between palm pose optimization and joint position optimization (PPO-JPO). Iterative PPO-JPO was able to locate a collision-free grasp within 0.50 sec based on 120 grasps on 12 objects in different categories. Experiments with a BarrettHand BH8-282 further demonstrated the effectiveness of the algorithm.

Chapter 6 proposed an efficient framework for grasp generation and execution by combining the surface fitting from Chapter 2 and optimization model from Chapter 5. The framework includes a multi-dimensional iterative surface fitting (MDISF) and a grasp trajectory optimization (GTO). The MDISF algorithm searches for optimal grasps by minimizing the hand-object fitting error and penalizing the collision, and the GTO algorithm

plans finger trajectories for grasp execution with the point cloud representation of the object. The MDISF-GTO exhibits certain robustness to the incomplete/noisy point cloud and various underlying uncertainties. On average, it took 0.40 sec for MDISF to find a collision-free grasp, and took 0.61 sec for GTO to optimize the trajectory to reach the grasp. The effectiveness of the framework was verified by simulations and experiments.

Chapter 7 presented an architecture for robust grasping and dexterous manipulation. The architecture contains a high-level modified impedance controller (MIC) to generate Cartesian force on object from the pose feedback, a mid-level manipulation controller to produce contact forces on fingertips from the desired Cartesian force, and a low-level force tracking controller to execute the generated force command. The high-level MIC is robust to underlying uncertainties and external disturbances, the mid-level manipulation controller avoids slippages and reduces the control effort, and the low-level force controller bypasses the unmodeled finger dynamics such as friction, joint flexibility and backlash. Both simulations and experiments with two different hands verified the proposed architecture. Preliminary qualitative results of the robust grasping with Type A hand and dexterous manipulation with Type B hand demonstrated the effectiveness of the proposed manipulation architecture.

Chapter 8 proposed a robust manipulation controller (RMC) within the manipulation architecture from Chapter 7. RMC includes a robust controller and a manipulation controller to achieve dexterous manipulation under various uncertainties and external disturbances. Feedback linearization was applied to reduce the nonlinearities of the composite hand-object system. By utilizing the structures of the uncertainties, the proposed robust controller can achieve faster convergence and tolerate more uncertainties compared with other methods based on disturbance observer and MIC from Chapter 7. The dual-stage formulation skipped complicated contact modeling, and was able to regulate contact force and prevent slippage. Simulation verified that the proposed RMC is robust to various uncertainties. Moreover, it did not require joint/object velocity measurement and was able to achieve fast tracking performance. Experiments performed on a BarrettHand BH8-282 verified the effectiveness of RMC.

Chapter 9 incorporated a novel velocity-based finger gaits planner with the robust manipulation controller from Chapter 8, to achieve real-time finger gaiting under different types of uncertainties. The finger gaits planner searches optimal velocities to improve the object grasp quality and the hand manipulability, rather than directly finding optimal contact points by nonlinear programming methods. The proposed planner is computationally efficient and can be solved in real-time. Besides, the planner does not rely on 3D reconstruction for surface modeling and high resolution encoders for velocity measurements. The presented planner is robust to the friction overestimation and unexpected slippage, and is able to address the complex dynamically critical tasks such as box-flipping with low-DOF hands. Simulations showed that the proposed method can achieve real-time finger gaiting, and realize long-range object motion that is infeasible without the proposed finger gaits planner.

Chapter 10 proposed a learning framework called guided-deep deterministic policy gradient (guided-DDPG) for high precision assembly task. The framework contains a trajectory optimization and an actor-critic structure. The trajectory optimization was served as a

semi-supervisor to provide initial guidance to actor-critic, and the critic network established the ground-truth quality of the policy by learning from both the semi-supervisor and exploring with policy gradient. The actor network learned from both the supervision of the semi-supervisor and the policy gradient of the critic. The involvement of critic network successfully addressed the stability issue of the trajectory optimization caused by the high-stiffness and the force/torque feedback. The proposed learning framework constrained the exploration in a safe narrow space, improved the consistency and reliability of the model-based RL, and reduced the data requirements to train a policy. Simulation and experimental results verified the effectiveness of the proposed learning framework.

11.2 Discussion and Future Works

Several open questions are revealed during the dexterity study of industrial manipulators. This section discussed these questions and directions of future works.

Grasping

Current implementations from Chapter 2 to Chapter 6 avoided manual heuristics by optimizing more abstract quality indices for desired grasps. While optimization-based approaches are more general than manual heuristics [99, 88] and more interpretable and predictable than deep learning [49], they rely on heavy online computation. To increase the optimization efficiency, the accuracy of quality indices was sacrificed by twisting rigorous wrench-based qualities to geometrical ones. For example, the Ferrari-Canny metric [30] in Chapter 5 was replaced by a geometric measure that combined 1) the distance between contact polygon and object center and 2) normal alignment errors between fingertips and the object. With the geometric quality measures, gradient-based approaches were applied to achieve timely search without exhaustive sampling.

Deep learning based methods learn a mapping function from the images to qualities [62, 49, 63]. The quality networks were trained by either mathematically rigorous grasps [62, 49] or empirical trials [63] to reflect reality. With domain randomization [95, 63], the learning methods are robust to sensing noise, positioning errors and background variations. Moreover, the learning based methods are able to handle the situation where the actual grasp positions are missing or not visible in the depth or RGB images. Therefore, they are suitable for grasping in clutter environments.

A common drawback of the learning-based approaches is the learning dimension. Due to the limit of data, the learned grasps are usually constrained in a top-down manner with parallel grippers. Some recent researches have broken the limitation of learning dimension [105] and learned 6D versatile grasps with multi-fingered hands, yet the performance on a broad scale physical experiment is unknown. Another common shortcoming is the absence of collision avoidance and motion planning modules, making the end-to-end system fragile when

picking heavy metal workpieces. In comparison, our grasp trajectory generation in Chapter 6 is able to generate collision-free optimal trajectories reliably.

A promising direction of future research is to incorporate learning with the proposed grasping framework. The goal is to enable the robot to 1) learn grasps from partially observable environments with single depth image, and 2) learn push-through grasps in clutter environments, since avoiding each obstacle in heavy clutter environments makes the grasp excessively conservative. The grasp planning at algorithmic level with BarrettHand provides valuable feedback for hardware design. Therefore, another research direction is to design an adaptive, safe and affordable multi-fingered hands for industrial bin picking.

Manipulation

Ideally 3D force/tactile sensors or joint torque sensors would not be required if the finger model was perfectly known. The force/torque command could be simply sent to motor in an open-loop manner. However, due to the unmeasurable dynamics (inertial and Coriolis force), unmodeled dynamics (friction, backlash and joint flexibility), action noise and time delay, the open-loop method is usually not enough to track the desired force. One method is to model the finger exhaustively by identifying the parameters of unmodeled dynamics and then apply feed-forward control. This method, however, has several drawbacks. First, it requires a rough model of the unmodeled dynamics. For example, the friction may need to be modeled into LuGre model with 6 parameters [71, 10]. Second, the identification usually requires high resolution encoders and accelerometers to capture the high-order motion [10, 107]. Moreover, the calibrated model may be affected by trajectories, payloads and pretension of the drivetrains, as shown in Chapter 7.

Another method is to use force sensors (Type B hand), tactile sensors (Type A hand) or combination of PPS sensor and strain gauge (BarrettHand) and perform a feedback control. However, the force/tactile sensors are costly and easy to abrade, and the sensor modeling error is destructive to the stability of closed-loop system.

Deep learning based methods [3] provide an alternative solution for dexterous manipulation with minimum number of sensors. The robustness of the policy is achieved by training with domain randomization [95]. However, the current implementation of deep learning methods usually exhibit several drawbacks. First, the training with domain randomization is exhaustive and brute-forcing, which means the training scenarios will be exponentially increased when randomizing any dimensions with uncertainties. Secondly, the network designing and hyperparameter tuning are time-consuming and require expertise. Moreover, the learned policy is not interpretable and lacks of stability guarantee compared with the proposed model-based control methods in Chapter 8.

Essentially, the manipulation policy trained with deep learning is embedded into the hand-object-task system. The embedded nature enables the learned policy to 1) use raw signals as input and 2) be robust to the sensor calibration/modeling errors. However, the learned policy is not easily adaptable to different hands, objects or different tasks. The controller designed by the proposed method, however, is parameterized by key parameters of the

object and independent to the hand and tasks. Therefore, the designed controller is able to 1) handle objects with different sizes and weights, 2) achieve continuous trajectory tracking, gaiting and pivoting tasks for dangled objects with stability guarantee, and 3) be invariant to hand and sensor types, as shown in Chapter 7 and Chapter 8. Moreover, the system with the designed controller guarantees safety by complying with excessive disturbances.

Nevertheless, the learned policy is able to 1) allow multiple contacts for each finger and 2) achieve complex object motions. The designed controller is difficult to model the hybrid dynamics with multiple contacts in each finger, and the realizable trajectories are constrained by feasible space of object motion (Chapter 8).

A promising direction of future research is to combine control with learning-based methods on manipulation. More specifically, a deep network with parameterization and abstraction of the environment will be designed to enable the deep learning to play side-by-side instead of being embedded to the hardware and task. A model-based domain randomization may also be investigated to avoid exhaustive sampling in uncertain dimensions.

Assembly

The performance of the automatic assembly has been improved dramatically by deep learning [48, 39, 64, 28]. On one hand, supervised learning is reliable when the training data is sufficient. Practically, collecting data under various uncertainties is inefficient. On the other hand, reinforcement learning (RL) learns a sequence of optimal actions by exploring the environment to maximize the expected reward [28]. With the abstract reward, RL also suffers from data shortage and the low efficiency issue. Meanwhile, both types of learning methods and the policies are embedded into the trained environment, causing overfitting and challenges to adapt to different environments. Model-based reinforcement learning [47, 48] has been proposed to address the data shortage by fitting models from data. The model fitting would be erroneous and the performance would be compromised if the contact surface is stiff or jerky, as shown in Chapter 10. Moreover, defining a physically safe training environment is very challenging for different tasks at the end-user side. Domain randomization [95] improves robustness by randomizing uncertain states and training exhaustively in simulation. The force responses in the physical world, however, are extremely difficult to be captured accurately, especially for those tasks with stiff and jerky contacts. The contact dynamics become more different when constraining the workspace in precise assembly.

A recent successful peg-hole-insertion experiment with a simple Proportional controller[1] reveals the strength of compliance control. It can be seen that the human or soft robotic systems usually have better performance because of the system compliance. Therefore, a group of assembly learning with compliance control and impedance control [93, 74] have been successfully applied. The compliance is provided by the controller to the closed-loop system in order to compensate the jerky and high stiffness of the contact surfaces.

[1]Experiment with FANUC LRMate 200iD/7L manipulators and H7/h7 Aluminum peg and hole. P controller is tuned trivially. Experiment credit: Shiyu Jin in Mechanical Systems Control lab.

Chapter 10 combined the model-based reinforcement learning with off-policy actor-critic to address the high stiffness and improve the efficiency and robustness. The method, however, still requires excessive time to train. A promising direction of future research is to integrate the impedance/compliance control into the guided-DDPG to compensate the effect of the discontinuous dynamical model. Another direction is to evaluate the algorithm on more realistic industrial applications such as connector insertion, furniture assembly and tight peg-in-hole tasks.

Appendix A

Robust Manipulation Control

A.1 Uncertainties Modeling

Type I/II uncertainties influence the effective object dynamics (8.3) in a complicated manner. This appendix first describes these uncertainties mathematically, then provides an approach to model all the uncertainties.

Mathematical Description of Uncertainties

To evaluate their impacts, we first present the relation between the motion in body frame and the motion in local coordinate.

$$\begin{aligned} v_o &= R_o^T \dot{p} \\ \omega_o &= Q_E \dot{E} \end{aligned} \tag{A.1}$$

where v_o, ω_o are translational and angular velocities of the object in body frame, and $\dot{p}, \dot{E}$ are same velocities represented in a local frame, where the translation is in inertial frame, while the rotation is a special frame constructed by ZYX Euler angle E. R_o is the rotation matrix of the object in inertial frame, and Q_E is the associated transform matrix with the form:

$$Q_E(R_o) = \begin{bmatrix} -\sin(\theta) & 0 & 1 \\ \cos(\theta)\sin(\phi) & \cos(\phi) & 0 \\ \cos(\theta)\cos(\phi) & -\sin(\phi) & 0 \end{bmatrix}$$

where ϕ, θ are elements of ZYX Euler angle $E = [\psi, \theta, \phi]$. The derivation of Q_E are based on the fact that R_o is a function of E and $\hat{\omega}_o = R_o^T \dot{R}_o$.

Object Dynamics Uncertainty

Without considering other uncertainties, the m_o, I_o uncertainties appear in inertia matrix:

$$M_o = \begin{bmatrix} m_o \mathbb{I}_3 & \mathbb{O}_3 \\ \mathbb{O}_3 & Q_E^T I_o Q_E \end{bmatrix} \quad N_o = \begin{bmatrix} m_o \mathbf{g} \\ 0_3 \end{bmatrix} \tag{A.2}$$

where $\mathbb{I}_3, \mathbb{O}_3 \in \mathbb{R}^{3\times3}$ are identity and zero matrix, and $\mathbf{g}, 0_3 \in \mathbb{R}^3$ are gravitational acceleration vector and zero vector. The Coriolis force is neglected. The $Q_E^T I_o Q_E$ in (A.2) is derived based on $\tau_o^T \omega_o = \tau_E^T \dot{E}$ and (A.1). Therefore,

$$\tau_E = Q_E^T \tau_o = Q_E^T I_o \dot{\omega}_o = Q_E^T I_o Q_E \dot{E}$$

τ_E, τ_o are the moment of object in local frame and object frame.

COM Uncertainty

In COM uncertainty, our belief to object COM and principal axes are erroneous. Suppose the nominal body center is $\bar{O}$ and actual body center is O ($\bar{O} \neq O$), then based on [89]:

$$M_{\bar{o}} = \begin{bmatrix} m_o \mathbb{I}_3 & m_o R_{\bar{o}} \hat{p}_{\bar{o}o}^T \bar{Q}_E \\ m_o \bar{Q}_E^T \hat{p}_{\bar{o}o} R_{\bar{o}}^T & \bar{Q}_E^T R_{o\bar{o}}^T I_o R_{o\bar{o}} \bar{Q}_E \end{bmatrix} \quad N_{\bar{o}} = \begin{bmatrix} m_o \mathbf{g} \\ 0_3 \end{bmatrix} \tag{A.3}$$

where $R_{\bar{o}}$ is the object rotation in the frame specified by $\bar{O}$ and the nominal principal axes. $\bar{Q}_E$ is the associated transfer matrix. $p_{o,\bar{o}}, R_{o,\bar{o}}$ are relative position and rotation of nominal object frame w.r.t. actual object frame.

Meanwhile, the grasp map w.r.t. nominal frame $G_{\bar{o}}$ becomes:

$$G_{\bar{o}} = \begin{bmatrix} R_{c_1} & ... & R_{c_{n_c}} \\ \bar{Q}_E^T R_{\bar{o}}^T \hat{p}_{c_1} R_{c_1} & ... & \bar{Q}_E^T R_{\bar{o}}^T \hat{p}_{c_{n_c}} R_{c_{n_c}} \end{bmatrix} \tag{A.4}$$

where p_{c_i}, R_{c_i} are position and orientation of the contact frame measured by tactile sensor. The point contact with friction (PCWF) model [69] is used to evaluate (A.4).

With the coupling of $R_{\bar{o}}$ and $p_{\bar{o}}$ in $M_{\bar{o}}$ and $G_{\bar{o}}$, separating the effective object inertia matrix into the form in (8.3) is extremely difficult.

Tactile Uncertainty

In tactile uncertainty, the measurement of contact position and orientation p_{c_i}, R_{c_i} are uncertain. The uncertainty appears in grasp map evaluation (A.4). Besides, by assuming each finger has at most one contact point, we can evaluate the hand Jacobian by:

$$J_h = \text{diag}\left(R_{c_1}^T J_{v,1}(p_{c_1}), ..., R_{c_{n_c}}^T J_{v,n_c}(p_{c_{n_c}})\right) \tag{A.5}$$

where $J_{v,i}(p_{c_i})$ is the translational part of the finger Jacobian, and it maps the joint speed of the i-th finger to inertial velocity of p_{c_i}.

The formulations of (A.4) and (A.5) introduce challenges to both robust controller design in Section 8.4 of Chapter 8 and manipulation controller in Section 7.2 of Chapter 7.

An Approach to Model Uncertainties

Compared with object dynamics uncertainty, the COM uncertainty and tactile uncertainty are difficult to model explicitly. Firstly, these two uncertainties influence both the effective object dynamics and the kinematics (i.e. grasp map and hand jacobian). Secondly, the position and orientation variations are coupled together for later two uncertainties. Moreover, the variations of $R_\bullet \in SO(3)$ in later two uncertainties generate a multiplicative uncertainty on $R_\bullet = \bar{R}_\bullet \tilde{R}_\bullet$, and separating effective object inertia, grasp map and hand Jacobian in this case becomes extremely challenging. The approximation $R_\bullet = \bar{R}_\bullet + \Delta R_\bullet$ only works for small variations, which still results in excessively high order controller[1]. The influence of the contact dynamics uncertainty shall be directly treated as disturbance, since the parameters of the effective object dynamics are not directly effected by the variations on contact stiffness, damping and friction coefficients.

Based on the analysis above, we choose to model the object dynamics uncertainties explicitly, and treat the influences of other uncertainties as disturbances. Therefore, the object inertia matrix (A.3), grasp map (A.4) and hand Jacobian (A.5) become:

$$\begin{aligned} M_{\bar{o}} &= \begin{bmatrix} m_o \mathbb{I}_3 & \mathbb{O}_3 \\ \mathbb{O}_3 & \bar{Q}_E^T I_{\bar{o}} \bar{Q}_E \end{bmatrix} \quad N_{\bar{o}} = \begin{bmatrix} m_o \mathbf{g} \\ 0_3 \end{bmatrix} \\ G_{\bar{o}} &= \begin{bmatrix} \bar{R}_{c_1} & ... & \bar{R}_{c_{nc}} \\ \bar{Q}_E^T R_{\bar{o}}^T \hat{\bar{p}}_{c_1} \bar{R}_{c_1} & ... & \bar{Q}_E^T R_{\bar{o}}^T \hat{\bar{p}}_{c_{nc}} \bar{R}_{c_{nc}} \end{bmatrix} \\ J_h &= \operatorname{diag}\left(\bar{R}_{c_1}^T J_{v,1}(\bar{p}_{c_1}), ..., \bar{R}_{c_{nc}}^T J_{v,n_c}(\bar{p}_{c_{nc}}) \right) \end{aligned} \tag{A.6}$$

where $\bar{R}_{c_i}$ is the average of the last ten R_{c_i} collected in 1000 Hz (sampling rate of the tactile sensor).

Starting from (8.5), the object inertia matrix $\bar{M}_o$, grasp map G and hand Jacobian J_h are replaced by $M_{\bar{o}}$, $G_{\bar{o}}$ and J_h in (A.6).

A.2 Equivalence of Different Disturbance Placements

The objective of this appendix is to prove the equivalence of placing the lumped disturbance u_{dis} before nonlinear dynamics, as shown in Fig. 8.2, and placing u_{dis} before feedback linearization, as shown in Fig. 8.3. The u_{dis} includes external perturbation and other disturbances converted from uncertainties.

u_{dis} as Input of Nonlinear Dynamics

When treating the u_{dis} as the input of the dynamics equation, the state space model of nonlinear system can be represented as:

$$\dot{x} = \bar{M}_{\text{aug}}^{-1} B_F F - \bar{M}_{\text{aug}}^{-1} \bar{C}_{\text{aug}} x - \bar{M}_{\text{aug}}^{-1} \bar{N}_{\text{aug}} + B_F u_{\text{dis}} + Lw \tag{A.7}$$

[1]Based on the trials of authors, approximation methods only works when the variations $r < 0.1$ rad in $\Delta R_\bullet = e^{\hat{r}}$, and the controller order can reach 264.

Plugging (8.13) into (A.7), we have:

$$\dot{x} = \begin{bmatrix} \mathbb{O} & \mathbb{I} \\ \mathbb{O} & \mathbb{O} \end{bmatrix} x + B_F(u + u_{\text{dis}}) + Lw \tag{A.8}$$

u_{dis} as Input of Feedback Linearization

When treating the u_{dis} as the input of feedback linearization, the state space model of nonlinear system can be represented as:

$$\dot{x} = \bar{M}_{\text{aug}}^{-1} B_F F - \bar{M}_{\text{aug}}^{-1} \bar{C}_{\text{aug}} x - \bar{M}_{\text{aug}}^{-1} \bar{N}_{\text{aug}} + Lw \tag{A.9}$$

The feedback linearization is:

$$F = \left(\bar{M}_{\text{aug}}^{-1} B_F\right)^{\dagger} \left[\bar{M}_{\text{aug}}^{-1} \bar{C}_{\text{aug}} x + \bar{M}_{\text{aug}}^{-1} \bar{N}_{\text{aug}} + B_F(u + u_{\text{dis}})\right] \tag{A.10}$$

By plugging (A.10) into (A.9), we have the exact same formulation as (A.8).

Bibliography

[1] Jacopo Aleotti and Stefano Caselli. "Part-based robot grasp planning from human demonstration". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 4554–4560.

[2] Sheldon Andrews and Paul G Kry. "Goal directed multi-finger manipulation: Control policies and analysis". In: *Computers & Graphics* 37.7 (2013), pp. 830–839.

[3] Marcin Andrychowicz et al. "Learning dexterous in-hand manipulation". In: *arXiv preprint arXiv:1808.00177* (2018).

[4] Gary J Balas et al. "μ-analysis and synthesis toolbox". In: *MUSYN Inc. and The MathWorks, Natick MA* (1993).

[5] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor Fusion IV: Control Paradigms and Data Structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–607.

[6] Antonio Bicchi. "Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity". In: *IEEE Transactions on robotics and automation* 16.6 (2000), pp. 652–662.

[7] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". In: *Acta numerica* 4 (1995), pp. 1–51.

[8] Torgny Brogårdh. "Present and future robot control development-An industrial perspective". In: *Annual Reviews in Control* 31.1 (2007), pp. 69–79.

[9] A Caldas et al. "Object-level impedance control for dexterous manipulation with contact uncertainties using an LMI-based approach". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3668–3674.

[10] Wenjie Chen. "Intelligent Control of Robots with Mismatched Dynamics and Mismatched Sensing". PhD thesis. UC Berkeley, 2012.

[11] Yang Chen and Gérard Medioni. "Object modelling by registration of multiple range images". In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[12] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. "Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem". In: *Robotics: Science and Systems Manipulation Workshop-Sensing and Adapting to the Real World*. Citeseer. 2007.

[13] Mark R Cutkosky. "On grasp choice, grasp models, and the design of hands for manufacturing tasks". In: *IEEE Transactions on robotics and automation* 5.3 (1989), pp. 269–279.

[14] Mark Cutkosky and Paul Wright. "Modeling manufacturing grips and correlations with the design of robotic hands". In: *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*. Vol. 3. IEEE. 1986, pp. 1533–1539.

[15] Mathieu Desbrun et al. "Implicit fairing of irregular meshes using diffusion and curvature flow". In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1999, pp. 317–324.

[16] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[17] Yongxiang Fan and Masayoshi Tomizuka. "Efficient Grasp Planning and Execution with Multi-Fingered Hands by Surface Fitting". In: *arXiv preprint arXiv:1902.10841* (2019).

[18] Yongxiang Fan, Xinghao Zhu, and Masayoshi Tomizuka. "Optimization Model for Planning Precision Grasps with Multi-Fingered Hands". In: *arXiv preprint arXiv:1904.07332* (2019).

[19] Yongxiang Fan et al. "A Learning Framework for Robust Bin Picking by Customized Grippers". In: *arXiv preprint arXiv:1809.08546* (2018).

[20] Yongxiang Fan et al. "Grasp planning for customized grippers by iterative surface fitting". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2018, pp. 28–34.

[21] Yongxiang Fan et al. "Object position and orientation tracking for manipulators considering nonnegligible sensor physics". In: *Flexible Automation (ISFA), International Symposium on*. IEEE. 2016, pp. 450–457.

[22] Yongxiang Fan et al. "Real-time finger gaits planning for dexterous manipulation". In: *The 20th World Congress of the International Federation of Automatic Control (IFAC)* (2017, to be presented).

[23] Yongxiang Fan et al. "Real-time finger gaits planning for dexterous manipulation". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 12765–12772.

[24] Yongxiang Fan et al. "Real-time grasp planning for multi-fingered hands by finger splitting". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4045–4052.

[25] Yongxiang Fan et al. *Real-Time Robust Finger Gaits Planning under Object Shape and Dynamics Uncertainties*. Youtube. 2017. URL: `https://youtu.be/fvsFtnAyODs`.

[26] Yongxiang Fan et al. "Real-time robust finger gaits planning under object shape and dynamics uncertainties". In: *Intelligent Robots and Systems (IROS) 2017 IEEE/RSJ International Conference on*. IEEE. 2017, pp. 1267–1273.

[27] Yongxiang Fan et al. "Robust dexterous manipulation under object dynamics uncertainties". In: *Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on* (2017 to appear).

[28] Yonxiang Fan, Jieliang Luo, and Masayoshi Tomizuka. "A Learning Framework for High Precision Industrial Assembly". In: *arXiv preprint arXiv:1809.08548* (2018).

[29] Raul Fernandez et al. "Micro-vibration-based slip detection in tactile force sensors". In: *Sensors* 14.1 (2014), pp. 709–730.

[30] Carlo Ferrari and John Canny. "Planning optimal grasps". In: *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE. 1992, pp. 2290–2295.

[31] Jeremy Fishel, Gary Lin, and Gerald Loeb. "Biotac® product manual". In: *SynTouch LLC, February* (2013).

[32] Scott Fujimoto, Herke van Hoof, and Dave Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *arXiv preprint arXiv:1802.09477* (2018).

[33] Noriatsu Furukawa et al. "Dynamic regrasping using a high-speed multifingered hand and a high-speed vision system". In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE. 2006, pp. 181–187.

[34] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[35] Corey Goldfeder et al. "The columbia grasp database". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 1710–1716.

[36] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *arXiv preprint arXiv:1801.01290* (2018).

[37] Kaiyu Hang, Johannes A Stork, and Danica Kragic. "Hierarchical fingertip space for multi-fingered precision grasping". In: *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE. 2014, pp. 1641–1648.

[38] Kaiyu Hang et al. "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation". In: *IEEE Transactions on robotics* 32.4 (2016), pp. 960–972.

[39] Tadanobu Inoue et al. "Deep reinforcement learning for high precision assembly tasks". In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE. 2017, pp. 819–825.

[40] Timothée Jost and Heinz Hugli. "A multi-resolution ICP with heuristic closest point search for fast and robust 3D registration of range images". In: *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*. IEEE. 2003, pp. 427–433.

[41] James W Kalat. *Introduction to psychology*. Nelson Education, 2016.

[42] Byoung-Ho Kim et al. "Optimal grasping based on non-dimensionalized performance indices". In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Vol. 2. IEEE. 2001, pp. 949–956.

[43] Ellen Klingbeil et al. "Grasping with application to an autonomous checkout robot". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2837–2844.

[44] Hakan Koc et al. "Modeling and robust control of winding systems for elastic webs". In: *IEEE Transactions on control systems technology* 10.2 (2002), pp. 197–208.

[45] Suresh Kotha. "Mass customization: implementing the emerging paradigm for competitive advantage". In: *Strategic Management Journal* 16.S1 (1995), pp. 21–42.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". In: *International Conference on Neural Information Processing Systems*. 2012, pp. 1097–1105.

[47] Sergey Levine and Vladlen Koltun. "Guided policy search". In: *International Conference on Machine Learning*. 2013, pp. 1–9.

[48] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[49] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with large-scale data collection". In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 173–184.

[50] Miao Li et al. "Dexterous grasping under shape uncertainty". In: *Robotics and Autonomous Systems* 75 (2016), pp. 352–364.

[51] Miao Li et al. "Learning object-level impedance control for robust grasping and dexterous manipulation". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 6784–6791.

[52] Ying Li, Jiaxin L Fu, and Nancy S Pollard. "Data-driven grasp synthesis using shape matching and task-based pruning". In: *IEEE Transactions on Visualization and Computer Graphics* 13.4 (2007), pp. 732–747.

[53] Zexiang Li and S Shankar Sastry. "Task-oriented optimal grasping by multifingered robot hands". In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 32–44.

[54] Alain Liegeois. "Automatic supervisory control of the configuration and behavior of multibody mechanisms". In: *IEEE transactions on systems, man, and cybernetics* 7.12 (1977), pp. 868–871.

[55] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[56] Hsien-Chung Lin et al. "Real-time collision avoidance algorithm on industrial manipulators". In: *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE. 2017, pp. 1294–1299.

[57] C Karen Liu. "Dextrous manipulation from a grasping pose". In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 59.

[58] Chia-Shang Liu and Huei Peng. "Disturbance observer based tracking control". In: *Journal of Dynamic Systems, Measurement, and Control* 122.2 (2000), pp. 332–335.

[59] Kok-Lim Low. "Linear least-squares optimization for point-to-plane icp surface registration". In: *Chapel Hill, University of North Carolina* 4 (2004).

[60] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*. Vol. 2. Springer, 1984.

[61] Raymond R Ma and Aaron M Dollar. "On dexterity and dexterous manipulation". In: *2011 15th International Conference on Advanced Robotics (ICAR)*. IEEE. 2011, pp. 1–7.

[62] Jeffrey Mahler et al. "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1957–1964.

[63] Jeffrey Mahler et al. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics". In: *arXiv preprint arXiv:1703.09312* (2017).

[64] David Martínez, Guillem Alenya, and Carme Torras. "Relational reinforcement learning with guided demonstrations". In: *Artificial Intelligence* 247 (2017), pp. 295–312.

[65] Andrew T Miller and Peter K Allen. "Graspit! a versatile simulator for robotic grasping". In: *IEEE Robotics & Automation Magazine* 11.4 (2004), pp. 110–122.

[66] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), p. 529.

[67] Abdellah Mokhtari et al. "Feedback linearization and linear observer for a quadrotor unmanned aerial vehicle". In: *Advanced Robotics* 20.1 (2006), pp. 71–91.

[68] Igor Mordatch, Zoran Popović, and Emanuel Todorov. "Contact-invariant optimization for hand manipulation". In: *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association. 2012, pp. 137–144.

[69] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[70] Ehtesham Nazma and Suhaib Mohd. "Tendon driven robotic hands: A review". In: *Int. J. Mech. Eng. Robot. Res.* 1 (2012), pp. 1520–1532.

[71] Henrik Olsson et al. "Friction models and friction compensation". In: *Eur. J. Control* 4.3 (1998), pp. 176–195.

[72] Jeremie Papon et al. "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. Portland, Oregon, June 2013.

[73] Andreas ten Pas and Robert Platt. "Using geometry to detect grasp poses in 3d point clouds". In: *Robotics Research*. Springer, 2018, pp. 307–324.

[74] Luka Peternel, Tadej Petrič, and Jan Babič. "Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation". In: *Autonomous Robots* 42.1 (2018), pp. 1–17.

[75] Robert J Platt Jr. "Learning and generalizing control-based grasping and manipulation skills". PhD thesis. Citeseer, 2006.

[76] R Platt, Andrew H Fagg, and Roderic A Grupen. "Manipulation gaits: Sequences of grasp control tasks". In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 1. IEEE. 2004, pp. 801–806.

[77] Vitchyr Pong. *rlKit: reinforcement learning framework and algorithms implemented in PyTorch*. `https://github.com/vitchyr/rlkit.git`. 2018.

[78] Markus Przybylski, Tamim Asfour, and Rüdiger Dillmann. "Planning grasps for robotic hands using a novel object representation based on the medial axis transform". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE. 2011, pp. 1781–1788.

[79] Máximo A Roa and Raúl Suárez. "Grasp quality measures: review and performance". In: *Autonomous robots* 38.1 (2015), pp. 65–88.

[80] Eric Rohmer, Surya PN Singh, and Marc Freese. "V-REP: A versatile and scalable robot simulation framework". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 1321–1326.

[81] JB Rosen. "The gradient projection method for nonlinear programming. Part II. Nonlinear constraints". In: *Journal of the Society for Industrial and Applied Mathematics* 9.4 (1961), pp. 514–532. URL: `http://epubs.siam.org/doi/pdf/10.1137/0109044`.

[82] Szymon Rusinkiewicz and Marc Levoy. "Efficient variants of the ICP algorithm". In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE. 2001, pp. 145–152.

[83] Jean-Philippe Saut and Daniel Sidobre. "Efficient models for grasp planning with a multi-fingered hand". In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 347–357.

[84] John Schulman et al. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." In: *Robotics: science and systems*. Vol. 9. 1. Citeseer. 2013, pp. 1–10.

[85] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[86] Jane Shi and Gurdayal S. Koonjul. "Real-Time Grasping Planning for Robotic Bin-Picking and Kitting Applications". In: *IEEE Transactions on Automation Science & Engineering* 14.2 (2017), pp. 809–819.

[87] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[88] Peng Song, Zhongqi Fu, and Ligang Liu. "Grasp planning via hand-object geometric fitting". In: *The Visual Computer* 34.2 (2018), pp. 257–270.

[89] Jarosław Strzałko et al. "General Motion of a Rigid Body". In: *Dynamics of Gambling: Origins of Randomness in Mechanical Systems* (2009), pp. 23–39.

[90] Zhe Su et al. "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 297–303.

[91] Tamara Supuk, Timotej Kodek, and Tadej Bajd. "Estimation of hand preshaping during human grasping". In: *Medical engineering & physics* 27.9 (2005), pp. 790–797.

[92] Taro Takahashi et al. "Adaptive grasping by multi fingered hand with tactile sensor based on robust force and position control". In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 264–271.

[93] Te Tang et al. "Teach industrial robots peg-hole-insertion by human demonstration". In: *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. IEEE. 2016, pp. 488–494.

[94] Yuval Tassa, Tom Erez, and Emanuel Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 4906–4913.

[95] Josh Tobin et al. "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)*. IEEE. 2017, pp. 23–30.

[96] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[97] Emanuel Todorov and Weiwei Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 300–306.

[98] Nikolaus Vahrenkamp, Tamim Asfour, and Rudiger Dillmann. "Simultaneous grasp and motion planning: Humanoid robot ARMAR-III". In: *IEEE Robotics & Automation Magazine* 19.2 (2012), pp. 43–57.

[99] Nikolaus Vahrenkamp et al. "Planning high-quality grasps using mean curvature object skeletons". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 911–918.

[100] Matej Vecerík et al. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". In: *CoRR, abs/1707.08817* (2017).

[101] Phongtharin Vinayavekhin, Shunsuke Kudoh, and Katsushi Ikeuchi. "Towards an automatic robot regrasping movement based on human demonstration using tangle topology". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 3332–3339.

[102] Website for Towards Robotic Dexterity: A Study on Grasping, Manipulation and Assembly. https://www.decf.berkeley.edu/%7Eyongxiangfan/thesis/thesis.html.

[103] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[104] Thomas Wimböck et al. "Comparison of object-level grasp controllers for dynamic dexterous manipulation". In: *The International Journal of Robotics Research* 31.1 (2012), pp. 3–23.

[105] Bohan Wu, Iretiayo Akinola, and Peter K Allen. "Pixel-Attentive Policy Gradient for Multi-Fingered Grasping in Cluttered Scenes". In: *arXiv preprint arXiv:1903.03227* (2019).

[106] Jijie Xu, Tak-Kuen John Koo, and Zexiang Li. "Sampling-based finger gaits planning for multifingered robotic hand". In: *Autonomous Robots* 28.4 (2010), pp. 385–402.

[107] YU ZHAO. "Intelligent Control and Planning for Industrial Robots". PhD thesis. UC Berkeley, 2018.

[108] Timo Zinßer, Jochen Schmidt, and Heinrich Niemann. "A refined ICP algorithm for robust 3-D correspondence estimation". In: *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*. Vol. 2. IEEE. 2003, pp. II–695.