OPTIMIZATION FOR CONTACT-RICH ROBOTIC MANIPULATION WITH HIGH-FIDELITY GEOMETRY

BY

MENGCHAO ZHANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Doctoral Committee:

Professor Geir Dullerud, Chair
Professor Kris Hauser, Director of Research
Associate Professor Joohyung Kim
Associate Professor Aaron Johnson

ABSTRACT

Contact interactions are intricate and pervasive phenomena encountered in various real-world scenarios. Humans and other organisms naturally navigate and interact with their surroundings through physical contact, in contrast to contemporary robots that avoid touching things as much as possible and shy away from complex manipulations such as pushing, pivoting, sliding, and rolling objects.

In bridging the gap between human-like contact interactions and current robotic capabilities, this thesis makes foundational contributions through introducing a novel computational model that employs a semi-infinite/infinite programming approach to effectively address the intricate nature of pervasive contact scenarios. The proposed model demonstrates applicability in various contexts, including object's 6D pose estimation within cluttered environments, contact-rich stable grasp pose optimization, and contact-rich manipulation trajectory optimization involving complex-shaped objects. In each of these scenarios, we address the problem that contact is an infinite phenomena involving continuous regions of interaction, which requires a discrete approximation to solve. Contrary to previous methods which need to discretize contacting surfaces a priori into a finite set of contact points, the proposed method operates directly on the continuous underlying geometry, and it dynamically identifies a finite set of constraints essential for problem resolution. As a result, the subsequent solving process is rendered feasible and tractable.

Additionally, to enable robust execution of planned trajectories, this thesis introduces two innovative techniques that combine the strengths of model-based planning and reinforcement learning. The first technique, Plan-Guided Reinforcement Learning, uses planned trajectories implicitly as demonstrations. It trains a control policy to mimic the demonstrated motion style while accomplishing the specified task. The policy synthesized in this way exhibits robustness in the face of uncertainties stemming from model parameter inaccuracies and the object's configuration. The second technique, Planned-Contact Informed Policy, utilizes the planned trajectory explicitly. The control policy takes features extracted from the planned trajectory as input, enabling it to anticipate changes in contact modes and adjust its actions accordingly. Consequently, control policies trained in this manner is capable of stabilizing the execution of similar trajectories planned for the same task type.

All the contributions presented in this thesis constitute fundamental building blocks towards automated robotic manipulation.

This dissertation is dedicated to my family for their love and support,
and to my younger self, for embarking on this challenging yet rewarding journey.

ACKNOWLEDGMENTS

I would first like to thank my advisor, Kris Hauser, for giving me the opportunity to join the Intelligent Motion Lab to pursue a Ph.D. in robotics. I still vividly remember the first time we met in his office when my research experience in robotics was minimal. Fortunately, it seems that my responses to his questions convinced him of my potential. From him, I learned to work smart rather than hard, though he works smart and hard. I learned how to be a better researcher, thinker, speaker, and writer from him in our weekly meetings, which he never failed to attend in the past five years no matter how busy he was. I cannot thank him enough for all the guidance and support he has provided throughout my Ph.D. journey.

I would also like to thank my committee members, Geir Dellerud, Joohyung Kim, and Aaron Johnson, for their guidance and support!

I am very grateful to all of my colleagues past and present in the Intelligent Motion Lab for their discussion, support, and encouragement. Thank you to Zherong Pan, Gao Tang, Fan Wang, Shihao Wang, Yifan Zhu, Joao Marques, William Edwards, Yeonju Kim, Patrick Naughton, Yu Zhou, Minkyung Kim, Shaoxiong Yao, Dohun Jeong, Pusong Li, Al Smith, Yixuan Wang, Jing-Chen Peng, Roger Qiu, James Nam, Simon Kato, David Null, Baoyu Li, Rachel Moan, Parsa Riazi Bakhshayesh, Chaoqi Liu, and Noah Franceschini.

I am fortunate to have had internship opportunities at Abbvie, MERL, and TRI, and I would like to extend my gratitude to the incredible individuals I had the pleasure of working with. Thank you to my internship supervisors Michelle Crouthamel, Sven Mensing, Devesh Jha, Arvind Raghunathan, Aykut Önol, Jose Barreiros, and Alex Alspach.

I would also like to extend my gratitude to those with whom I have collaborated or engaged in deep and joyful discussions. Thank you Yanran Ding, Chuanzheng Li, Jifei Xu, Hang Cui, Yu Chen, Yinan Pei, Chenzhang Xiao, Pranay Thangeda, Aditya Prakash, Tao Chen, Chenhao Li, Abhijat Biswas, Zhe Huang, and Binghao Huang.

Next, I want to thank my other close friends Bo Fu, Jiaxin Wu, Hongyu Shen, Yuhang Yang, Siyuan Chen, Shiyue Zhang, Xinchang Li, Haoran Qiu, Jiaqi Guan, Dawei Sun, Weichao Mao, and Sarang Bhagwat for the great time that we spent together! You have made this Ph.D. journey enjoyable!

Finally but most importantly, I want to thank my parents Zhongpu Zhang and Tong Meng, my mother-in-law Xiaoli Xue, my grandparents, and my lovely wife Hankun He. This dissertation would not be possible without their unwavering love, trust, and support.

TABLE OF CONTENTS

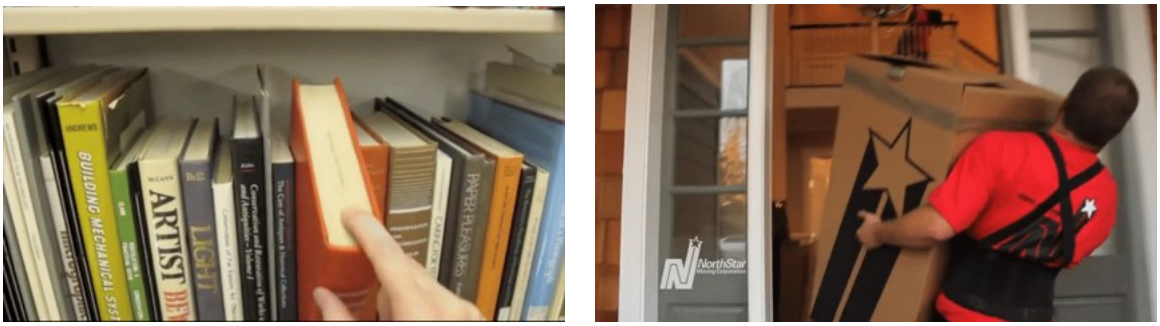
LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Humans leverage diverse strategies such as in-hand manipulation, whole-body manipulation, and environmental contact to manipulate a variety of objects in our daily lives as shown in Fig. 1.1(a). In contrast, even though the robotics field has long attempted to imitate these behaviors, the interactions of the majority of robots with their surroundings are primarily restricted to their end-effectors, and the motions are mostly pick and place, which are not as dexterous or energy-efficient as those performed by humans, as shown in Fig. 1.1(b).

The difference between the manipulation dexterity exhibited by humans and the limited manipulation capability demonstrated by robots is a result of lacking effective planning and control methods for robotic contact-rich manipulation, and the main difficulty of planning and control with contact is rooted in the non-smooth nature of contact dynamics.



(a) Manipulations performed by humans

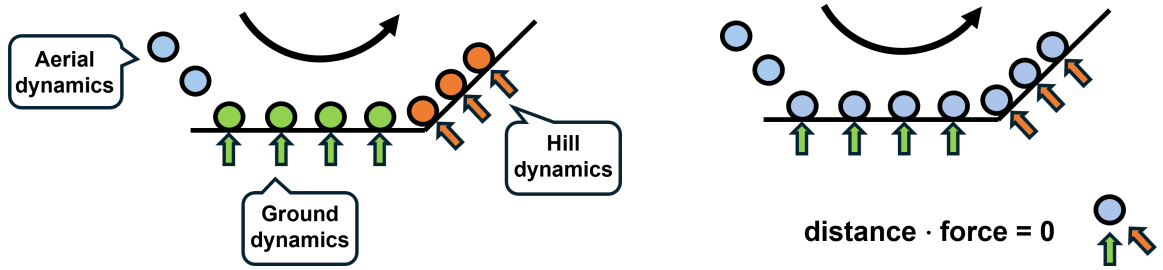
(b) Manipulations performed by robots

Figure 1.1: (a) Examples of various manipulations performed by humans in daily life: the left image shows a human retrieving a book from a shelf, while the right image depicts a human using his whole body to move a large, heavy box. (b) Examples of different manipulations performed by robots: the left figure illustrates the most common robotic manipulation—pick and place, while the right figure shows a humanoid robot moving a large box with its hands. Compared to human manipulations, robotic manipulations typically limit interactions to the end-effectors, and the movements are not as dexterous or energy-efficient as those performed by humans. [Best viewed in color.]

1.1 CONTACT-RICH MANIPULATION PLANNING

The relative motion of the contact points between two objects can be classified as separating, sticking, or sliding, which is usually called contact mode. When the contact modes for all contact points on an object is given, the dynamic equations of motion for the object will be determined accordingly. The dynamics of the object is smooth until any of the contact modes change.

Different planning methods have been proposed to address the difficulty caused by non-smooth contact dynamics caused by changing of contact modes, and they can be categorized into two branches, contact explicit and contact implicit, based on whether the method considers contact modes switches explicitly.



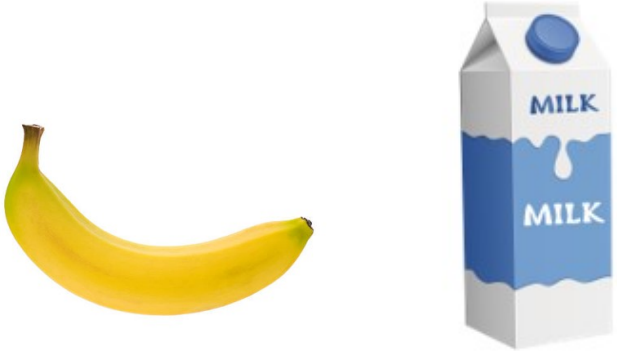
(a) Contact explicit (b) Contact implicit

Figure 1.2: An illustration of the differences between the contact explicit method and the contact implicit method in solving a simple trajectory optimization problem involving contact. A sphere moves from left to right, and the objective is to determine the sphere's position at every time instance along the trajectory, as well as the contact force between it and the environment. As shown in (a), the contact explicit method requires a pre-specified contact sequence (as the aerial, ground, and hill stages shown in the figure), with different segments stitched together using reset functions. In contrast, the contact implicit method, depicted in (b), does not require the contact sequence as an input and treats all states equally. Here, contact is modeled using complementarity constraints, which require that the product of the distance between the sphere and the environment and the force exerted by the environment on the sphere be zero. In practice, these complementarity constraints are typically relaxed and gradually tightened to facilitate solving. [Best viewed in color. Adapted from Prof. Aaron Johnson's presentation (https://www.youtube.com/watch?v=4MrqIoJ4kLI).]

As illustrated in Fig. 1.2(a), the main idea of methods in the contact explicit branch is to divide the contact dynamics into pieces, within each piece the contact is fixed and the dynamics is smooth consequently. Then, through specifying a set of resets that describe the discrete update to the state between each pair of connected pieces, we can stitch all the pieces together. Methods in this branch either require the contact mode sequence to be known apriori, or explored by an auxiliary discrete search. Hybrid trajectory optimization,

as a representative method in this branch, was initially applied to solving locomotion trajectories [1], where contact is limited to a few potential contact points (heel and toe) and humans can easily specify the contact mode sequence. However, manually specifying the contact sequence for a contact-rich manipulation task, in which there can be many more contacts and contact switches, can easily become non-tractable. Thus, in [2], the author through combining an automated contact mode enumeration algorithm [9] and a sampling-based planning algorithm extended the usage of contact explicit methods from locomotion to contact-rich manipulation planning.

As illustrated in Fig. 1.2(b), methods in the contact implicit branch do not require the contact mode sequence to be known apriori, and contacts are modeled by complementarity constraints there [3, 4]. Complementarity constraints, which require that the contact forces can be non-zero if and only if the contact is established, are challenging to handle from a numerical optimization perspective as they have no interior and violate constraint qualifications which are needed by gradient-based methods to converge. Contact-implicit trajectory optimization (CITO) [4], as a representative method in this branch, formats the planning problem with contact as a mathematical program with complementarity constraints (MPCC), and solves it through relaxing the complementarity constraints and gradually tightening the relaxation. Since the solving time of the MPCC rises sharply as the number of modeled contacts grows, the application of CITO was restricted to simple geometries.



(a) Banana (b) Milk carton

Figure 1.3: (a) An example of a daily object that cannot be accurately represented by a geometry primitive. (b) An example of a daily object whose geometry simplification requires expert and task-specific knowledge. [Best viewed in color.]

However, using simple geometries in robot manipulation planning has limitations in many aspects. Firstly, not all everyday objects can be accurately represented by a geometry primitive or a combination of geometry primitives, and the banana shown in Fig. 1.3(a) is an example. Also, how to simplify an object usually requires expert and task-specific knowledge.

For instance, if a robot wants to interact with the milk carton shown in Fig. 1.3(b), whether the lid and two side holes should be modeled will depend on whether the planning method can handle non-convex geometry, as well as whether the robot will need to interact with those specific regions of the object. Besides, no matter how much we simplify an object's geometry, there will always be a discrepancy between the real and simplified versions, and this difference can easily cause a failed execution of the planned trajectory. To enable robots to perform human-like manipulation with minimal human intervention, manipulation planning methods that directly use high-fidelity geometry representation such as point cloud or mesh are necessary.

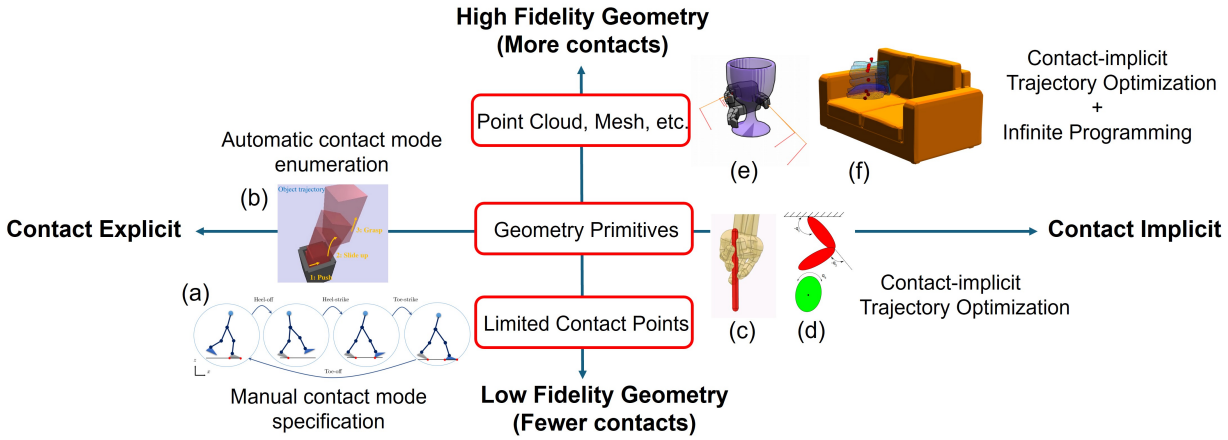


Figure 1.4: A comparison of representative contact explicit and contact implicit planning methods, highlighting their generalizability for planning with high-fidelity geometries. The contact explicit method was initially used to solve for locomotion trajectories as shown in (a) [Reproduced from [1]], where the contact sequence can be manually specified. Through combining an automatic contact mode enumeration algorithm and sampling-base planning algorithm, contact explicit method was applied to plan for manipulation trajectories as shown in (b) [Reproduced from [2]]. Contact implicit methods can also plan for manipulation trajectories as shown in (c) [Reproduced from [3]] and (d) [Reproduced from [4]]. However, both contact explicit and contact implicit methods were restricted to plan with geometry primitives due to their scalability towards working with more contact points. The methods proposed in this thesis are depicted in the upper right of the figure, marking the first instance in which planning is enabled for contact-rich stable grasp poses (e) and manipulation trajectories (f) with high-fidelity geometry separately. [Best viewed in color.]

The complexity of extending the contact explicit branch of methods to work with high-fidelity geometry is rooted in the complexity of contact mode enumeration. As shown in [9], the complexity of enumerating all 3D contact modes for one object is $\mathcal{O}(N^d)$, where N is the number of contacts and d is the effective degrees of freedom of the object. Considering planning a free movement for a high-fidelity geometry, N will be in the magnitude of hundreds

to thousands (or even larger), and d will be 6, which could easily make the enumeration non-tractable due to the exponential growth.

On the other side, for the CITO method, the number of optimization variables and constraints in the resulting MPCC increases linearly with respect to the number of modeled contacts. However, the solving time of the resulting MPCC does not grow linearly regarding the number of modeled contacts due to the matrix inversions usually needed in an optimization-solving process, which makes directly putting all the contacts in a high-fidelity geometry into CITO non-tractable.

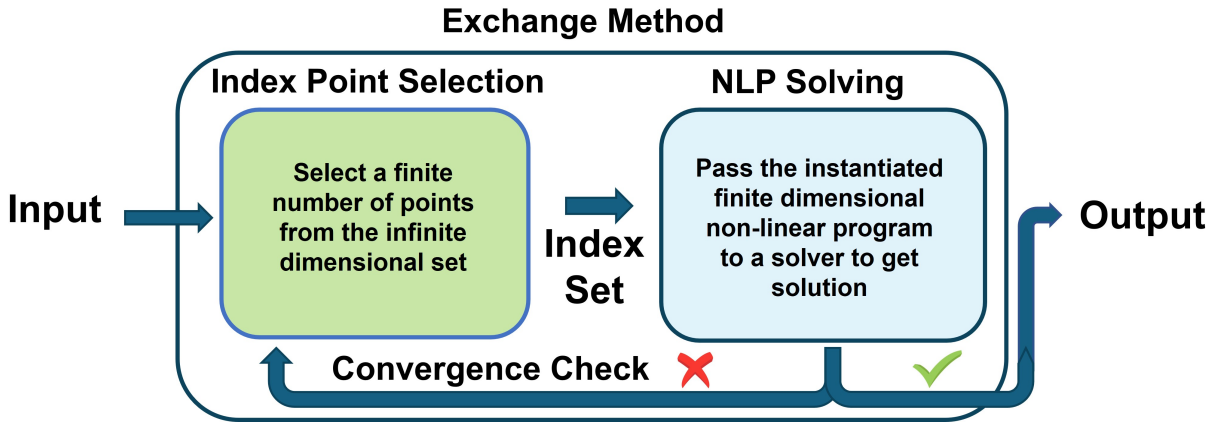


Figure 1.5: An illustration of the exchange method used to address instantiated infinite programming problems discussed in this thesis. The index point selection step, commonly referred to as an oracle, strategically selects a finite number of points from an infinite-dimensional set where the constraint is defined. Based on these selected points, defined as an index set, a finite-dimensional optimization problem is instantiated and forwarded to a solver for solution. The solution must pass a convergence check, which evaluates the satisfaction of the original infinite-dimensional optimization problem, before being output. This thesis makes foundational contributions to contact-rich manipulation planning through using this method. [Best viewed in color.]

In this thesis, we extend the application of the CITO method, enabling it to work with high-fidelity geometry by innovatively embedding it into an infinite programming framework. In optimization theory, infinite programming (IP) is an optimization problem with an infinite number of variables and constraints. Although it is not possible to optimize an infinite number of variables and constraints directly, a finite subset of variables and constraints, suitably chosen, is sufficient to define an optimum [10]. The exchange method (shown in Fig. 1.5), as an effective approach to solving IP problems, instantiates a series of finite dimensional optimization problems, each of which progressively adds some number of variables and constraints. The key to the success of an exchange method is the designing of a judicious constraint selection procedure, called an oracle, to make the series of instantiated problems

converge toward one that contains a true optimum. We introduce different oracle designs for solving contact-rich stable grasp pose planning and manipulation trajectory planning, demonstrating for the first time that contact-rich manipulation planning with high-fidelity geometry is tractable.

1.2 CONTACT-RICH MANIPULATION CONTROL

Except for the difficulty of planning for contact-rich manipulation, controlling a robot to do contact-rich manipulation tasks is also challenging because the constraints introduced by the contacts can cause the task to fail in different ways. Due to the discrete nature of contacts, a small disturbance to the system could result in a drastic change in contact modes and the system's dynamics. Then the original planned robot control will likely be invalid under the new contact modes. Executing the planned robot motion under a different contact mode can easily fail the task, and could possibly incur large forces to damage the robot or the object. Thus, to let robots perform contact-rich manipulation robustly and safely, closed-loop feedback control is essential.

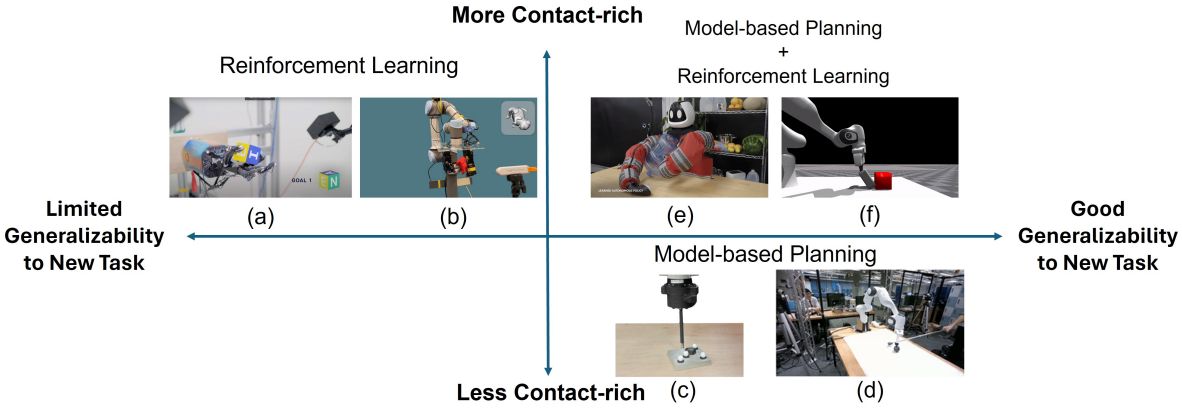


Figure 1.6: A comparison of representative closed-loop feedback control methods for manipulation, focusing on their generalizability to new tasks and the contact-richness of the tasks they can address. Reinforcement learning can solve contact-rich in-hand manipulation tasks very well as shown in (a) [Reproduced from [5]] and (b) [Reproduced from [6]]. However, its generalizability to new tasks is limited due to the heavy computation needed and task-specific reward function and training scheme design. Model-based planning is good at generalizing to new tasks. However, due to the slow solve time of contact-rich motion planners, online replanning is only possible for tasks involving a limited number of contacts as shown in (c) [Reproduced from [7]] and (d) [Reproduced from [8]]. The methods proposed in this thesis, as depicted in (e) and (f), enable closed-loop feedback control for contact-rich manipulation tasks and can also be easily generalized to new tasks. [Best viewed in color.]

The recent advancements in reinforcement learning (RL) have enabled closed-loop control for contact-rich dexterous in-hand manipulation and have shown good robustness [5, 6]. However, a successful RL policy usually requires a large amount of offline computation and task-specific reward function and training procedure (such as privileged learning) design, which makes it hard to generalize to a new task. On the other hand, model-based approaches are known for their good generalizability to new tasks through task parameterization. However, due to the complexity of planning with contact, the solving time of model-based contact-rich motion planners is not fast enough to be incorporated into closed-loop feedback controllers such as Model Predictive Controller to do replanning online except for some simple tasks with a limited number of contacts [7, 8].

In this thesis, we introduce two distinct approaches to integrating model-based planning with RL, demonstrating that a method strategically combining these two techniques has the potential to capitalize on the strengths of both.

1.3 CONTRIBUTIONS AND THESIS STRUCTURE

This thesis makes fundamental contributions to the advancement of automatic contact-rich robotic manipulation, offering specific contributions such as a 6D pose estimation algorithm for cluttered scenes, grasp/manipulation trajectory planning methods that can plan with high-fidelity geometric representations of objects, and the development of controllers designed to execute contact-rich manipulation trajectories with increased robustness.

Chapter 2 presents a novel 6D pose estimation algorithm, the Non-Penetration Iterative Closest Points (NPICP), an enhancement of the Iterative Closest Points (ICP) method. This chapter elaborates on the integration of non-penetration constraints between close objects, and between objects and the free space to ICP through using a semi-infinite programming approach. The NPICP method, applied as a post-processing step to the outcomes generated by deep neural network-based object detection and pose estimation systems, demonstrates superior performance over the standard ICP method in single-view, multi-object 6D pose estimation tasks within the IC-BIN dataset.

Chapter 3 presents a novel formulation, infinite programming with complementarity constraints (IPCC), that optimizes a force distribution over points of contact, and encodes constraints and/or objectives on the force distribution, e.g., force balance and Coulomb friction. Additionally, this chapter delineates a local optimization algorithm for IPCC, employing the exchange method. Through iteratively choosing a finite subset of contact points and forces to consider, and solving a series of corresponding finite-dimensional nonlinear constrained optimization problems for step directions, the series of problems converges toward one that

contains a true optimum of the original problem. The application of IPCC to stable grasp pose optimization demonstrates rapid convergence to local optima, even when dealing with highly complex geometries, achieving solutions within seconds to tens of seconds.

Chapter 4 introduces a novel contact-implicit trajectory optimizer, Simultaneous Trajectory Optimization and Contact Selection (STOCS), that optimizes contact-rich manipulation trajectories for non-convex objects and improves computational efficiency by dynamically instantiating non-penetration, force balance, and complementarity constraints at contact points selected within the optimization loop. STOCS extends the application of IPCC from pose optimization to manipulation trajectory optimization, and it is the first method capable of planning contact-rich manipulation trajectories with high-fidelity geometric representations in 3D. Experimental results demonstrate its efficacy in planning for pushing, pivoting, sliding, and rolling trajectories using dense point clouds of daily objects.

STOCS is contact-implicit for object-environment contact, but it still requires a pre-selected robot-object contact state. In Chapter 5, we introduce a sampling-based planner, Multi-Modal Manipulation Planner (MMMP), that uses STOCS as a local optimizer to incrementally construct a manipulation tree, which helps it solve for longer-horizon trajectories that change manipulation mode between pushing, pivoting, and grasping. MMMP's capability of generating plans involving changes of manipulator contact state is validated on several manipulation tasks in simulation and on a real robot.

Chapter 6 introduces a plan-guided reinforcement learning framework that uses manipulation trajectories planned by a model-based planner as a demonstration to train a closed-loop control policy using reinforcement learning. The proposed approach is tested to address a complex whole-body large object manipulation task. The experimental results show that this approach can enable generating control policies efficiently even from a single, infeasible example motion plan with a simple reward function. The trained policy is evaluated in both simulation environments and on real hardware employing Toyota Research Institute's Punyo robot.

Chapter 7 introduces another method, Planned-Contact Informed Policy (PCIP), that integrates model-based manipulation planning and reinforcement learning and uses planned trajectories more explicitly during the training of a control policy. Different features, especially the novel-designed planned contact feature, are extracted from the planned manipulation trajectories and given as input to the control policy. The inclusion of planned contact information in the policy's observation allows it to anticipate future contact modes without requiring privileged information or adding sensing modalities to the robot. Experimental results of evaluating PCIP on two task types: peg insertion and non-prehensile pivoting show that PCIP achieves superior performance to standard RL and residual RL, and that

planned contact information plays a vital role in PCIP's performance.

Chapter 8 provides the concluding remarks and introduces possible future research directions from this thesis.

Additionally, during my Ph.D., I completed some extra work that, while not directly related to the topic of this dissertation, is listed here for reference. I proposed a hybrid planning framework for generating complex dynamic motion plans that enable legged robots to traverse challenging terrains [11]. I also developed an experience-based technique for the sample-efficient adaptive control of nonlinear systems, addressing dynamical modeling errors (under review for IROS 2024).

CHAPTER 2: NON-PENETRATION ITERATIVE CLOSEST POINTS FOR SINGLE-VIEW MULTI-OBJECT 6D POSE ESTIMATION

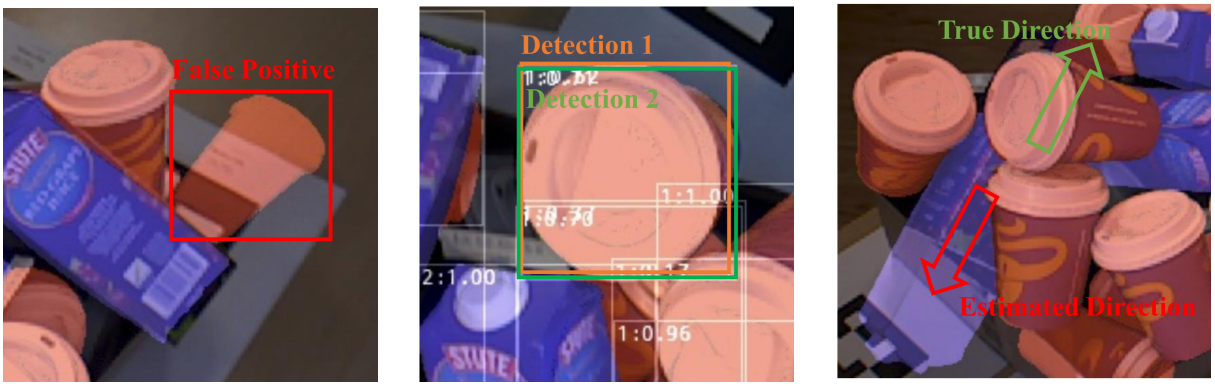
In this chapter, we introduce a novel 6D pose estimation algorithm, the Non-Penetration Iterative Closest Points (NPICP) [12], which enhances the traditional Iterative Closest Points (ICP) method [13, 14] by incorporating non-penetration constraints between close objects and between objects and the free space. We detail the processing steps required to implement NPICP as a post-processing technique for improving results obtained from deep neural network-based object detection and pose estimation systems. Furthermore, we discuss how non-penetration constraints are imposed between non-convex, complex-shaped objects using a semi-infinite programming approach [15]. Experiments conducted on single-view, multi-object 6D pose estimation tasks with the IC-BIN dataset [16] show that NPICP outperforms the conventional ICP method and achieved state-of-the-art performance at the time of publication. [1]

2.1 INTRODUCTION

Accurate detection and estimation of the six degree-of-freedom (6-DOF) pose (position and orientation) of objects in a scene is needed for augmented reality and robot manipulation tasks, so pose estimation has been an active topic of research for many years. Nevertheless, state-of-the-art object detection [17] and pose estimation [18] systems are still noisy, often including many false positives, duplicate detections, and inaccurate pose estimates as shown in Fig. 2.1. These challenges are compounded when scenes involve object clutter and multiple instances of the same object. In such scenarios, local information is often insufficient to disambiguate poses, whereas global information about the scene such as geometric non-penetration and support relationships can effectively narrow down the feasible range of poses.

This study delves into the application of non-penetration constraints to enhance the precision of local optimization for single-object and multi-object pose estimation. Specifically, we address the problem that complex, non-convex 3D geometries impose a large number of constraints, which can number in the tens or hundreds of thousands. Rather than resorting to conventional nonlinear programming (NLP) techniques, we draw upon the concept of the exchange method in semi-infinite programming (SIP) [10]. The optimizer progressively gen-

[1] This chapter is reproduced from Mengchao Zhang, and Kris Hauser, "Non-Penetration iterative closest points for single-view multi-object 6D pose estimation". In the International Conference on Robotics and Automation (ICRA) 2022.



(a) False positive (b) Duplicated detection (c) Flipped direction

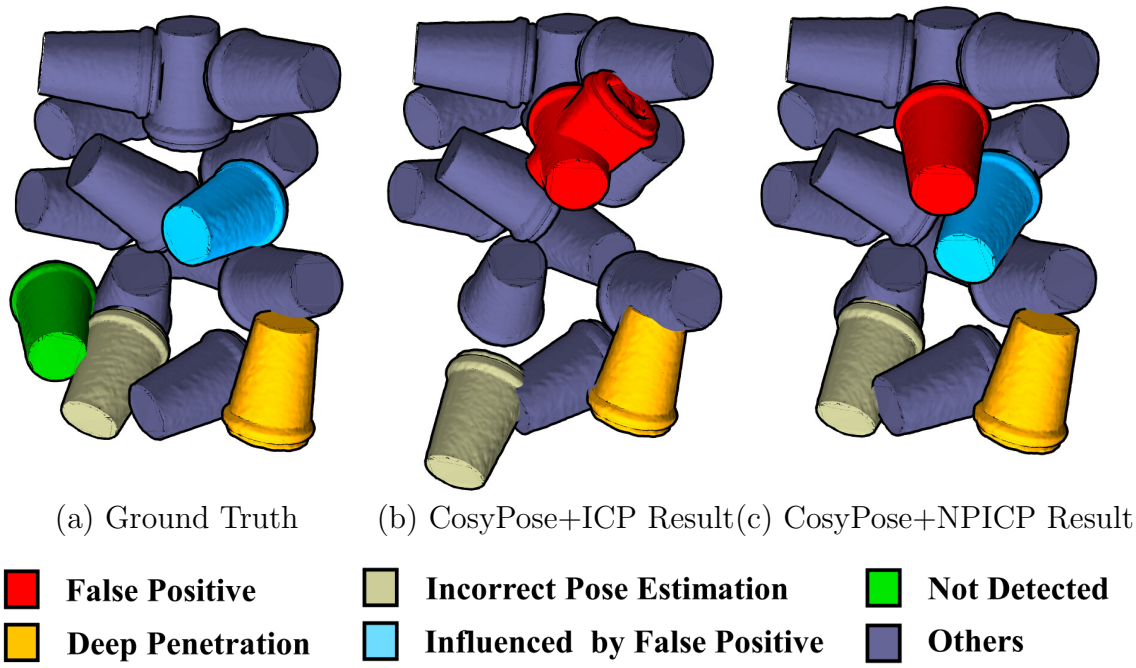
Figure 2.1: Common problems in DNN object detection and pose estimation results. The blue and red shapes are rendered objects at the estimated poses. (a) A coffee cup is incorrectly detected in free space. (b) The object is detected twice. (c) The estimated pose of the juice box is in a flipped direction compared with the ground truth. [Best viewed in color.]

erates some constraints to be included in a smaller NLP, and the series of problems converges toward a true optimum of the original problem.

Our proposed NPICP algorithm integrates the SIP non-penetration approach in an Iterative Closest Points (ICP) algorithm that minimizes a point cloud registration energy while respecting non-penetration constraints between objects and the free-space of an RGB-D image. It is evaluated on the single-view multi-object 6D pose estimation problem of the IC-BIN dataset [16]. As a post-processor for a deep neural network (DNN) object detector/-pose estimator, NPICP performs geometry-based "cleanup" of physically implausible poses. NPICP's efficacy is most pronounced within complex scenes, such as the illustrative instance from the IC-BIN dataset [16] shown in Fig. 2.2. Notably, NPICP excels particularly when operating with accurate detections, capitalizing on its local nature, though its effectiveness diminishes with inadequate initialization. Nevertheless, it outperforms ICP across conditions of both ideal and noisy detections and outperforms the top-performing pose estimator for the IC-BIN dataset in the Benchmark for 6D Object Pose Estimation (BOP Challenge) [19].

2.2 RELATED WORK

Accurately determining the six degrees-of-freedom (6-DOF) pose, encompassing both position and orientation, of objects within a scene holds critical significance for robotic manipulation tasks. Consequently, pose estimation has garnered considerable research attention over the years. Despite advancements in object detection [17] and pose estimation [18],



(a) Ground Truth (b) CosyPose+ICP Result (c) CosyPose+NPICP Result

Figure 2.2: Views from below of the reconstruction results of Scene 1 in the IC-BIN dataset, showing (a) ground truth, (b) results of the best performer on the IC-BIN dataset in the BOP Challenge which uses ICP to refine the result of the CosyPose detector, (c) NPICP used in place of vanilla ICP. Compared to the results in (b), NPICP resolves the deep penetration of the yellow cup with the cup next to it, and corrects the pose of the beige cup. It is also partially able to recover the pose of the cyan cup, but is influenced by a false positive from CosyPose (red). [Best viewed in color.]

contemporary systems continue to exhibit noise, encompassing false positives, redundant identifications, and imprecise pose estimations. These challenges are exacerbated in scenarios involving cluttered scenes and the presence of multiple instances of the same object.

The BOP challenge [20] serves as an open competition aimed at showcasing the cutting-edge developments in the realm of 6-DOF object pose estimation from RGB-D images. Point pair features (PPF) methods [21] are frequently employed, involving the matching of pairs of oriented 3D points between the model of a 3D object and the point cloud of the test scene. Subsequently, a voting scheme is utilized to deduce the object's pose. However, PPF methods exhibit vulnerability to background clutter and sensor noise, coupled with the drawback of quadratic computational complexity [22].

Deep neural network (DNN) methods, propelled by their successes in object detection [17, 23], have significantly advanced pose estimation [18, 24]. While PPF methods dominated early editions of the BOP challenge, DNN methods gained ground in 2020 [25]. Yet, DNN-

based pose estimation results may be prone to noise. Consequently, post-DNN refinements, such as local registration through ICP [14], are common to improve point cloud-object geometry alignment. However, the efficacy of local registration diminishes when significant object occlusion is involved.

Under highly cluttered circumstances, local information is often insufficient to disambiguate poses, whereas global information about the scene such as geometric non-penetration and support relationships can help narrow down the space of valid poses. Leveraging interactions between objects has been explored to enhance pose estimation accuracy. Mitash et al. (2019) [26] introduced a method generating multiple candidate poses per object, followed by a search across the Cartesian product of these pose candidates to identify optimal scene hypotheses. However, the exponential increase in runtime with the number of objects restricts its applicability, as demonstrated in experiments with up to three objects.

A method capable of improving pose estimation accuracy by incorporating non-penetration constraints between nearby objects and between objects and unoccupied space, with a computational runtime that is roughly linear in relation to the number of objects in the scene, is highly desirable for addressing the pose estimation problem in cluttered environments.

2.3 METHOD

2.3.1 Problem Setup

We address the single-view multi-object 6D pose estimation problem, that is estimating the poses of N objects in an RGB-D image I, assuming known 3D object geometries. Since our method is most useful as a local registration technique for a DNN-based initializer, we assume an object detector generates a set of estimated objects $e_{1:n} = [e_1, \cdots, e_n]$, where $e_i = (e_i^c, e_i^q, e_i^s, e_i^M)$ contains the object class $e_i^c \in \{1, \cdots, C\}$, the 6D object pose $e_i^q \in SE(3)$ with respect to the camera, a confidence score $e_i^s \in [0, 1]$ and a binary mask image e_i^M. Our approach generates a refined set of object estimates $o_{1:m} = [o_1, \cdots, o_m]$ where $o_i = (o_i^c, o_i^q, o_i^s, o_i^M)$. Note that the first stage of our algorithm filters out some proposed object estimates, so m is often smaller than n.

To represent the object's geometry, we use a dual representation, consisting of both the signed distance function (SDF) and point cloud sampled from the surfaces of the object. We denote the SDF of o_i in its local frame as $g_i(\cdot) : \mathbb{R}^3 \to \mathbb{R}$. The surface of o_i is also denoted as a point cloud in its local reference frame as PC_i^l.

2.3.2 Non-Penetration Iterative Closest Points

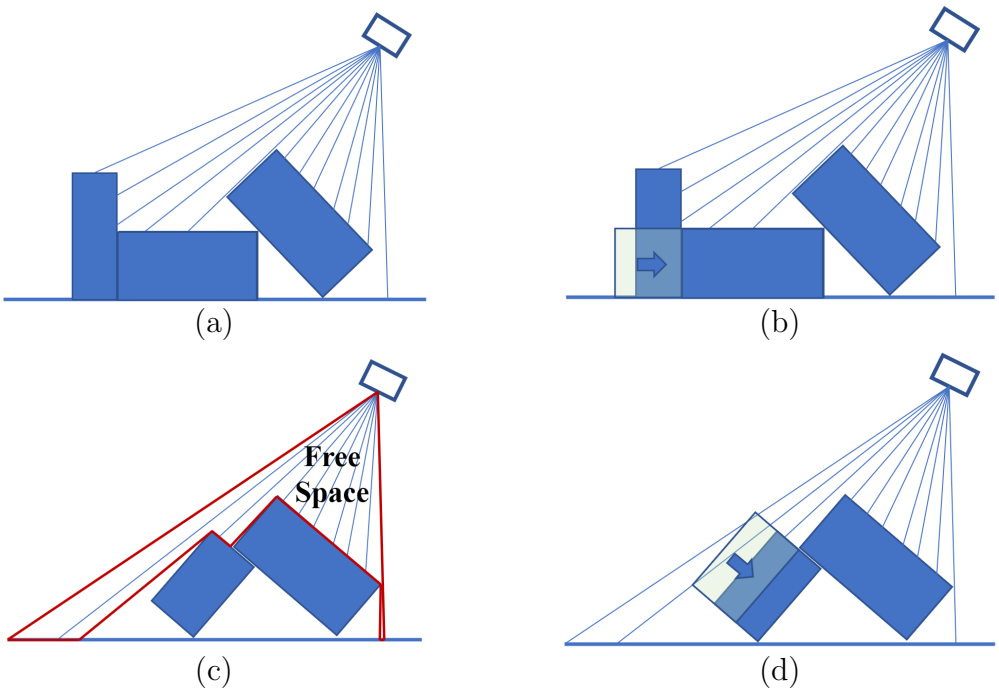


Figure 2.3: Illustration of how non-penetration information is helpful to improve pose estimation accuracy. Pure ICP cannot distinguish between poses in (b) or poses in (d), whereas non-penetration constraints are able to disambiguate the correct pose. (a) and (b) illustrate how object non-penetration constraints disambiguate the correct pose. (c) and (d) illustrate how object-free space non-penetration constraints disambiguate the correct pose. [Best viewed in color.]

Our approach uses two forms of non-penetration constraints to improve pose estimation accuracy: object-object penetration and object-free space penetration (Fig. 2.3). ICP only works to minimize the error of matches between points and object geometry, but for occluded objects or those with little texture, it has little information to leverage to distinguish the true pose.

As illustrated in Fig. 2.3 (b), object non-penetration constraints can find shift poses closer to the ground truth. Free-space constraints are also very informative, because each point in the depth image is the closest point between the camera and the objects in the scene along that direction, so all objects should be behind the observed surface. So, from the RGB-D image I and the intrinsic parameters K of the camera we build a free space object o_{free} consisting of these line segments in Fig. 2.3 (c). Fig. 2.3 (d) illustrates how object-free space non-penetration can improve pose estimation by pushing the left object downward under the observed surface.

However, directly adding all the estimated objects to NPICP and optimizing their poses jointly can lead to low accuracy and low efficiency. The false positives may take the space of correctly detected objects, and very deep penetration can cause trouble for NPICP. Besides, adding more object to the optimization will result in longer solve time.

Thus, we design a two-step method. In Step 1, Estimation Association (EA), we filter out bad pose estimations using geometry information, and we run NPICP individually on each object to improve the estimated poses, which only considers the non-penetration between the object and the free space. In step 2, Joint NPICP, we run NPICP over all the kept objects to optimize their poses jointly, using the non-penetration both between objects and between objects and the free space. The pipeline is illustrated in Fig. 2.4.

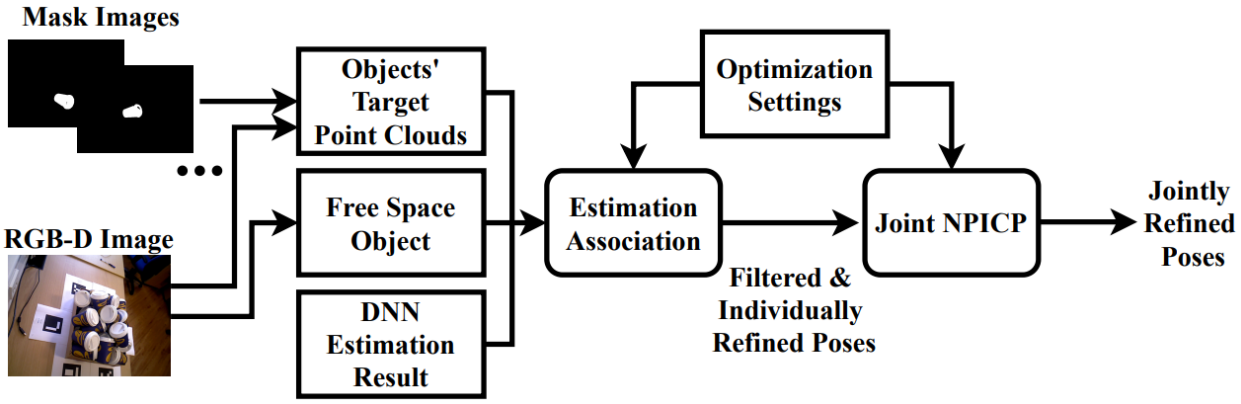


Figure 2.4: The workflow of NPICP.

Next, we discuss each of the steps in detail.

Step 1: Estimation association

DNN object detection/pose estimation results can include many false positives, and these are critical to remove because spurious objects can cause NPICP joint pose optimization to lose accuracy. Otherwise, NPICP will try to prevent penetration between phantom objects, which can cause poor estimates to cascade. Our first step (Alg. 2.1) filters out likely false positives, and it also generates the target (scene) point clouds which are used for the registration energy in NPICP. It proceeds by examining each estimated object, in order of most to least confident, by the following steps:

Low Quality Mask Removal Line 3 removes estimations whose masks have fewer than n_{min} valid pixels. DNN object detection methods sometimes give detections that consist of

very few pixels, which cannot generate good pose estimation through ICP or its variants, so these are filtered out.

Translation Adjustment DNN pose estimation methods often produce large errors in the estimated distance from the object to the camera, so Lines 4–6 use the point cloud to adjust the initial translation estimation. To do so, we render the object into a depth image where the object is placed at the estimated configuration e_i^q and then extract points PC_{ren} belonging the mask e_i^M. We shift the estimated pose to match the centroid of the rendered point cloud PC_{ren} to the centroid of the portion of the scene point cloud containing e_i

This stage also extracts a target scene point cloud PC_{scn} to be used in the NPICP registration energy. The extraction is performed by $GetObjectPointcloud(I, K, e_i^M, n_{obj}, n_n, r_{std})$ in Alg. 2.1. First, we use the object mask and camera intrinsics to determine a dense point cloud PC_{reg}^s, then we down-sample to n_{obj} points to accelerate the optimization. To make sure the down-sampled points are representative, the iterative farthest point method is used. Then, we run the statistical outlier removal method in Open3D [27] on the down-sampled point cloud, with parameters n_n and r_{std} specifying how many neighbors are taken into account in order to calculate the average distance for a given point, and r_{std} is an outlier rejection threshold level based on the standard deviation of distances.

Individual NPICP To make sure that the object is roughly at the ground truth pose, such that the following estimation quality check step will not filter out good results, we run NPICP on each estimated object before we pass it to the quality check. This individual NPICP only considers the penetration between the object and the free space object.

False Positive Removal Next, Line 8 tests the penetration with prior objects $o_{1:m}$ and o_{free}. If all the absolute penetration distances are smaller than a threshold d_1, the sum of the absolute penetration distances is smaller than a threshold d_2, and the absolute penetration distances with the free space is smaller than another threshold d_3, then we will go to the next step. Otherwise, we discard the estimation. All of d_1, d_2 and d_3 are determined by the size of the geometry to be examined.

Duplicated Detection Removal DNN object detection methods sometimes detect an object instance multiple times. So, for the next estimated object e_i to be examined, Line 9 checks the distance $\delta q_{ij} = \|e_i^q.t - o_j^q.t\| \ \forall j \in 1, ..., m$, where $e_i^q.t$ is the translation of the estimated pose. If any δq_{ij} is smaller than some threshold δq_{min}, then we consider the estimation as a duplication.

Algorithm 2.1 Estimation Association

Require:

- RGB-D image I, camera intrinsic parameters K.
- Estimations $e_{1:n}$.
- Free space object o_{free}.
- Maximum iterations N_{max}^{single}, step size tolerance ϵ, index addition and deletion threshold g_{max}.
- Minimum # of pixels occupied in object mask n_{min}.
- Maximum # of points retained in object point clouds n_{obj}.
- Outlier removal parameters n_n, r_{std}, inlier percentage p.

Number of retained objects $m = 0$
for e_i in $Sort(e_{1:N})$ **do**
  if $ValidPixel(e_i^M) > n_{min}$ **then**
    $PC_{scn} \leftarrow GetObjectPointcloud(I, K, e_i^M, n_{obj}, n_n, r_{std})$
    $PC_{ren} \leftarrow RenderPointcloud(e_i^c, e_i^q, e_i^M, K)$
    $e_i^q.t \; +\!\leftarrow Centroid(PC_{scn}) - Centroid(PC_{ren})$
    $e_i^q \leftarrow NPICP([e_i], [PC_{scn}], o_{free}, N_{max}^{single}, g_{max}, \epsilon, p)$
    if not $FPRemoval(e_i, o_{1:m})$ **then**
      if not $DuplicateRemoval(e_i, o_{1:m})$ **then**
        $m \leftarrow m + 1$
        $o_m \leftarrow e_i$
        $PC_m^s \leftarrow PC_{scn}$
      end if
    end if
  end if
end for
$o_{1:m}^s \leftarrow ModifyConfidenceScore(o_{1:m}^s, o_{1:m}^M)$
return $o_{1:m}, PC_{1:m}^s$

Score Modification For all retained objects, we adjust their confidence scores from the DNN object detection method according to a heuristic so that objects with fewer occlusions have higher confidence scores (these are used in the scoring function in BOP challenge; this step is not strictly necessary). We modify the score according to the number of pixels in the mask image o_i^M, setting $o_i^s = o_i^s + \frac{ValidPixel(o_i^M)}{Pixel_{max}} \; \forall i = 1, \dots m$, where $Pixel_{max} = \max_{i=1}^{m} ValidPixel(o_i^M)$.

At the end of step 1, all objects are considered to be inliers and NPICP only modifies their poses hereafter.

Step 2: Non-penetration Iterative Closest Point

The next stage is the main contribution of this study, which is an ICP implementation with non-penetration constraints. This includes the non-penetration constraints between each object and the free space, and the non-penetration constraints between each pair of objects.

We use the local SDF representations of each object, $g_i(\cdot)$, to measure penetration. Denoting the volume of the free space object o_{free} as $\mathcal{V}_{free}$, the non-penetration constraint between o_{free} and o_i is expressed as $g_i((o_i^q)^{-1} \cdot y) \geq 0 \ \forall y \in \mathcal{V}_{free}$. The non-penetration constraint between object i and object j is expressed as $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \geq 0 \ \forall y \in PC_j^l$. Although there are an intractable number of points in these constraints, we follow the SIP formulation of [28] which uses fast deepest-penetrating-point techniques to construct a manageable number of constraints [28].

The point cloud registration energy of object o_i is defined by the sum of squared distances of the geometry to the scene point cloud PC_i^s. A standard technique used in ICP is to reject outliers by only registering the closest p percent of points in PC_i^s to o_i [14]. This improves the the accuracy and stability of pose estimation by rejecting invalid matches. Denote $d_i(o_i^q) = [g_i((o_i^q)^{-1} y_{i,1}^s)^2, \cdots, g_i((o_i^q)^{-1} y_{i,N_i^s}^s)^2]^T$ as the vector concatenating the squared distances of points in PC_i^s to o_i, with $N_i^s = |PC_i^s|$. Also, let $\mathbb{1}_p(d_i)$ be an inlier selection function, which selects the smallest p percent points from d_i. The returned vector has the same dimension as d_i, and $d_{ij} = 1$ if the j'th distance is an inlier and $d_{ij} = 0$ if it is an outlier. Then the NPICP optimization problem is given by the following form:

$$\begin{aligned}
\arg\min_{o_{1:m}^q} \quad & \frac{1}{2} \sum_{i=1}^{m} \mathbb{1}_p(d_i(o_i^q))^T d_i(o_i^q) && \text{(2.1a)} \\
\text{s.t.} \quad & g_i((o_i^q)^{-1} y) \geq 0 \ \forall i \in 1, \cdots, m \ \ \forall y \in \mathcal{V}_{free} && \text{(2.1b)} \\
& g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \geq 0 \ \ \forall y \in PC_j^l && \text{(2.1c)} \\
& \forall i \in 1, \cdots, m, \ \forall j \in 1, \cdots, m, \ i \neq j. && \text{(2.1d)}
\end{aligned}$$

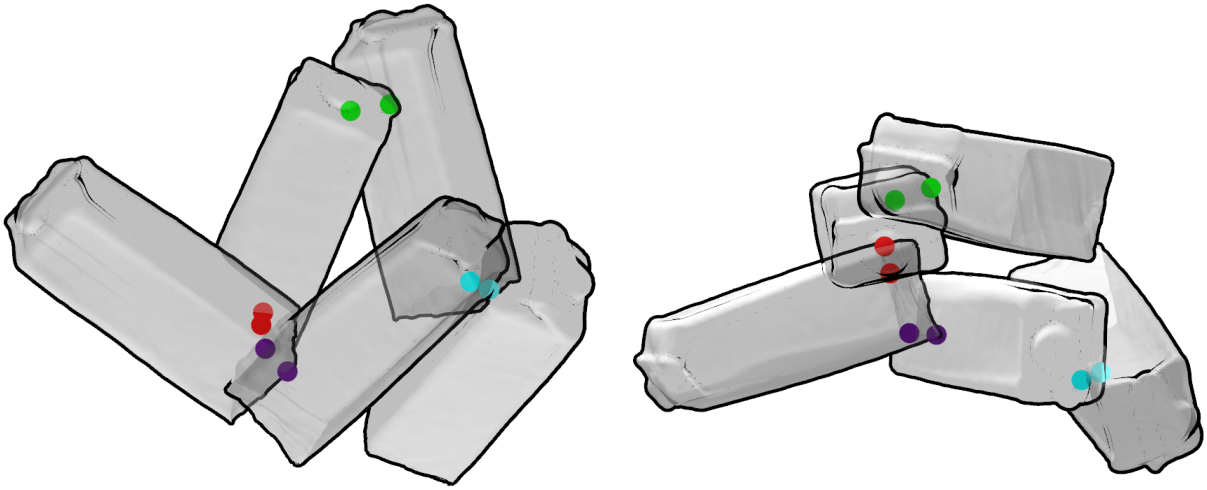
The primary challenge is that $\mathcal{V}_{free}$ and each of the PC_i^l may contain tens or hundreds of thousands of points, making direct optimization of (2.1) using an NLP solver intractable. We leverage semi-infinite programming (SIP) techniques to improve the speed of optimization. SIP allows constraints to depend on free variables, known as index points, which may range over an infinite set. Certainly, an infinite number of constraints cannot be solved directly in a standard NLP solver. Thus, an exchange method [29] is used to instantiate a series of finite dimensional optimization problems, each of which progressively adds some number

of constraints. Also, through using a judicious index point selection procedure, called an oracle, the series of problems converges toward one that contains a true optimum of the original infinite dimensional problem. The same approach can be applied to greatly accelerate optimization in problems with a very large number of constraints, like ours.

We use a maximum-violation oracle [10], which identifies a parameter value that has a large effect on the next iterated solution, to instantiate index points. Specifically, on iteration k of the optimization, we calculate the most violating parameter of each non-penetration constraint:

$$y_{i,j}^{min} \leftarrow \arg \min_{y \in PC_j^l} g_i((o_i^q)^{-1} \cdot o_j^q \cdot y) \tag{2.2}$$

and similarly for the free-space violation constraint. An example of the index points between objects instantiated by the most-violation oracle is shown in Fig. 2.5.



(a) Front View (b) Top View

Figure 2.5: Index points between objects generated by the maximum-violation oracle when the objects are at the configurations shown in the figure. The points are the closest points (or deepest penetration points) between each pair of objects. The index points between different pair of objects are represented by different colors. [Best viewed in color.]

We define the set of instantiated index parameters as Y, and choose an index addition and deletion threshold g_{max}. Then if $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y_{i,j}^{min}) \leq g_{max}$, we will add $(y_{i,j}^{min}, i, j)$ to Y. To simplify notation, we regard the free space object as object 0 ($o_{free} \equiv o_0$) and set its pose to the identity transform. We also delete constraints from the constraint set when they are not deemed necessary. To be specific, if any index parameter $(y_{i,j}^{min}, i, j)$ in Y that satisfies $g_i((o_i^q)^{-1} \cdot o_j^q \cdot y_{i,j}^{min}) > g_{max}$, then we will delete it from Y to accelerate the following

NLP solving.

Letting Y^k be the index set generated at iteration k, and freezing the inlier selection vector as $\mathbb{1}^k \leftarrow \mathbb{1}_p(d_i(o_i^{q,k-1}))$ for the poses at the start of the iteration, we solve the following the nonlinear program to determine a desired set of poses at iteration k:

$$\arg\min_{o^q_{1:m}} \quad \frac{1}{2}\sum_{i=1}^{m}(\mathbb{1}^k)^T d_i(o_i^q) \tag{2.3a}$$

$$\text{s.t.} \quad g_i((o_i^q)^{-1}\cdot o_j^q \cdot y_j^l) \geq 0 \;\; \forall (y_j^l, i, j) \in Y^k. \tag{2.3b}$$

Simply taking the NLP solution as a step could cause infeasibility for a non-instantiated constraint, so instead we use a line search. The line search ensures a sufficient decrease in a merit function that penalizes the worst-case violation of a constraint. Let $g^*_{i,j}(o_i^q) \equiv \min_{y \in PC_i^l} g_i((o_i^q)^{-1}\cdot o_j^q \cdot y) \geq 0$ be the worst-case violation of the collision constraint between objects i and j. This is then used in the merit function $\phi(o^q_{1:m}, Y, w_1, w_2) = f(o^q_{1:m}) + w_1 \times |D(o^q_{1:m}, Y)^-| + w_2 \times |g^*(o^q_{1:m})^-|$ (see Alg. 2.2 Line 7). In this function, $D(o^q_{1:m}, Y)$ returns a stack of the distances of the instantiated index points, and $g^*(o^q_{1:m})$ returns a stack of the closest distances between each pair of objects and the closest distance between each object and the free space object, where $(\cdot)^- \equiv \min(\cdot, 0)$. We also add the penetration points detected during line search to the index set in the next iteration.

NPICP is summarized in Alg. 2.2.

2.4 EXPERIMENTS AND DISCUSSION

We tested our method on the IC-BIN dataset, which contains two object types that appear in multiple locations with heavy occlusion in a bin-picking scenario (Fig. 2.6). These scenes also have severe foreground occlusions and background distractors.

We followed the evaluation methodology provided by the BOP Challenge to evaluate the pose estimation results. The performance of a method on a dataset is measured by the Average Recall: $AR = (AR_{VSD} + AR_{MSSD} + AR_{MSPD})/3$. VSD is the Visible Surface Discrepancy [20, 30], which measures the distance difference in the depth image using only the visible part of the object in the image. MSSD is the Maximum Symmetry-Aware Surface Distance [31], which measures the maximum surface deviation and is relevant to robotic manipulation. MSPD [30] is the Maximum Symmetry-Aware Projection Distance, which changes the average distance in 2D projection [32] by the maximum distance and is relevant for augmented reality applications. An object is only considered in the evaluation if at least

Algorithm 2.2 Non-Penetration Iterative Closest Point

Require:

- Estimations $o_{1:m}$.
- Target point clouds $PC^s_{1:m}$.
- Free space object o_{free}.
- Maximum iteration number N_{max}, step size tolerance ϵ.
- Index addition and deletion threshold g_{max}.
- Inlier selection parameter p.

1: $k \leftarrow 0$
2: $o^k_{1:m} \leftarrow o_{1:m}$, $Y^k \leftarrow \{\}$, $Y^k_{LS} \leftarrow \{\}$
3: **for** $k = 1, \ldots, N_{max}$ **do**
4: $\quad Y^k \leftarrow DeleteIndex(Y^{k-1}, o^{k-1}_{1:m}, o_{free}, g_{max})$
5: $\quad Y^k \leftarrow Oracle(Y^k, Y^{k-1}_{LS}, o^{k-1}_{1:m}, o_{free}, g_{max})$
6: $\quad$ Solve 2.3 to get $o^{q,k}_{1:m}$
7: $\quad$ Run line search on ϕ to get step size α and new penetration points detected during line search Y^k_{LS}
8: $\quad \Delta o^q_{1:m} \leftarrow \alpha(o^{q,k}_{1:m} - o^{q,k-1}_{1:m})$
9: $\quad o^{q,k}_{1:m} \leftarrow o^{q,k-1}_{1:m} + \Delta o^q_{1:m}$
10: $\quad \triangleright$ Test for convergence
11: $\quad$ **if** $\|\Delta o^q_{1:m}\|/m \leq \epsilon$ **then return** $o^k_{1:m}$
12: $\quad$ **end if**
13: **end for**

10% of its surface is visible. All experiments were run on a single core of a 3.6 GHz AMD Ryzen 7 processor.

We use the following parameters in all the experiments:

- $N^{single}_{max} = 10$, $N^{joint}_{max} = 10$, $\epsilon = 5e-4$, $g_{max} = 5\ mm$.
- $n_{min} = 100$, $n_{obj} = 200$, $n_n = 50$, $r_{std} = 2$, $p = 95$.
- Denoting the smallest dimension of the bounding box of the object to be examined as d^*, we use $d_1 = \frac{d^*}{3}$, $d_2 = \frac{d^*}{2}$ and $d_3 = \frac{d^*}{3}$.

2.4.1 Perfect detection, disturbed ground truth pose

The first set of experiments compares ICP and NPICP when we have perfect detection but imperfect pose estimation. In this case, we know the true number of objects $m = n$ and their classes, and we skip the Estimation Association filters (Lines 3, 8, and 9 in Alg. 2.1) and directly set $o_{1:m} := e_{1:n}$. The initial estimated poses are the ground truth poses plus random disturbances, with each disturbance sampled as a random rotation in the range

(a) Scene 1 (b) Scene 2 (c) Scene 3

Figure 2.6: Example images of the IC-BIN dataset. [Best viewed in color.]

Table 2.1: The results of ICP and NPICP given perfect detection on the three scenes of IC-BIN seperately.

Scene	Method	AR	Pen/obj (mm)
1	Init	0.376	10.84
	ICP	0.695	5.18
	NPICP	0.808	1.53
2	Init	0.425	9.04
	ICP	0.788	2.07
	NPICP	0.814	0.90
3	Init	0.400	11.50
	ICP	0.686	4.37
	NPICP	0.823	1.42

± 0.25 *rad* and translation in the range ± 3 *cm*. We run both ICP and NPICP on three scenes of IC-BIN. The ICP baseline we used is the point-to-point ICP in Open3D [27]. For a fair comparison, we run the translation adjustment for both algorithms.

The results of this experiment are shown in Table 2.1. Init is the initial guess. AR is the average recall and Pen/obj is the average penetration of an object with all its surrounding objects. Compared with ICP, NPICP greatly increases the AR and decreases the penetration between objects. Also, the improvement of NPICP over ICP is larger when the scene is highly cluttered, as in scenes 1 and scene 3.

To validate the design choice of our algorithm, we test the following NPICP variants:

- NPICP Ind: Run NPICP individually on each object (skipping Joint NPICP in Fig. 2.4).
- NPICP Joint: Run NPICP jointly over all the objects (skipping line 7 in Alg. 2.1).
- NPICP: Run NPICP both individually and jointly.

Results on the IC-BIN dataset are shown in Table 2.2. Time/img is the average time used

for the pose optimization of each image. NPICP provides the best results of all the three variants while taking the most time. However, NPICP is only slightly slower than NPICP Joint, which we believe is because the individual optimization makes the initial condition of the joint optimization more ideal, which enables the joint optimization to converge more quickly.

The Pen/obj of NPICP Ind is larger than that of ICP because NPICP Ind only constrains penetration with the free space object, such that there is no guarantee that it will decrease the penetration between objects.

Table 2.2: Results of ICP and three variants of NPICP given perfect detection on the IC-BIN dataset. Pen/obj is the average penetration of the objects selected by the BOP AR calculation procedure. T_{err} is the average pose error compared with the ground truth pose, which sums the translational error (in mm) and rotational error (in degrees).

Method	AR	Pen/obj (mm)	T_{err}/obj		Time/img (s)
			μ	σ	
Init	0.393	10.88	31.3	7.6	-
ICP	0.704	4.40	19.2	14.4	17.6
NPICP Ind	0.743	4.85	14.9	17.5	26.8
NPICP Joint	0.812	1.59	12.0	18.0	106.5
NPICP	0.815	1.40	11.8	17.6	108.6

2.4.2 DNN-based detection, DNN-based pose estimation

The second experiment compares ICP and NPICP given DNN object detection and pose estimations. We use the results of CosyPose [33] as the initial guess.

The variant of CosyPose which has the best performance additionally applies an ICP refinement, and is currently the best result on the IC-BIN dataset in the BOP Challenge. We run all the three variants of NPICP as a post-processing method for CosyPose (without ICP), and compare their results with the ICP variant of CosyPose.

The results of this experiment are shown in Table 2.3. CP is CosyPose for short. Given the results of CosyPose, all the NPICP variants increase the average recall and decrease the penetration between objects, even though the margin of performance increase is not as large as with perfect detection. We believe that this is caused by the following reasons: 1) Some objects are not detected, thus the interactions between that object and the objects surrounding it could not be used in the joint optimization. 2) The confidence score of the object detection module are not reliable. Some bad detections could take the place of good

detections if they have higher scores, and could also influence its surrounding objects. Given imperfect detection, NPICP still performs the best among all the three NPICP variants.

Table 2.3: The results of the ICP variant of CosyPose and using three variants of NPICP as post-processing methods on CosyPose. *The average run time of CP+ICP is 11.358 s as listed on BOP Challenge's website. It was run on a machine with 20-core Intel Xeon 6164 @ 3.2 GHz CPU and Nvidia V100 GPU. As a reference, the average run time of CP without ICP is 0.678 s on the same machine.

Method	AR	Pen/obj (mm)	Time/img (s)
CP+ICP	0.647	4.24	11.4*
CP+NPICP Ind	0.667	2.83	40.5
CP+NPICP Joint	0.672	1.33	73.9
CP+NPICP	0.674	1.28	99.2

2.4.3 Analysis

The advantages of NPICP are the best with perfect detection, and given imperfect detection, although the improvements beyond ICP degrade, it still outperforms the best result on the IC-BIN dataset in the BOP Challenge.

The run time of NPICP are shown in Table 2.2 and Table 2.3, which only counts the optimization time and does not include the time used for processing the geometries. The run time depends roughly linearly on the following factors: 1) The number of initial estimation N. 2) The number of kept objects m. 3) The iteration number N_{max}^{joint}, N_{max}^{single}. 4) The number of point in $PC_{1:m}^{s}$.

Although the run time of the problem is tens of seconds, the computational performance of the current implementation could be significantly improved. We coded in Python for the purpose of rapid prototyping, and the run time could be reduced if we rewrite the code in a compiled language such as C++. Also, although the run time is not ideal for real time application, as far as we know, methods which could deal with highly occluded scenes that have a similar number of objects, and could provide similar accuracy are all far from real time.

2.4.4 Parameter variations

To compare the influence of false positive removal, we tested different choices for d_1-d_3 values in Alg. 2.1. When thresholds are large ($d_1 = 1/2\ d^*, d_2 = 3/4\ d^*, d_3 = 1/2\ d^*$), some deeply penetrated objects are not discarded by Alg. 2.1, which decrease the AR to 0.657

and increases the penetration to 4.96 mm. When thresholds are small ($d_1 = 1/6\ d^*, d_2 = 1/4\ d^*, d_3 = 1/6\ d^*$), some good detections could be discarded, which decrease the AR to 0.667 but slightly decrease the penetration to 1.09 mm.

CHAPTER 3: INFINITE PROGRAMMING WITH COMPLEMENTARITY CONSTRAINTS FOR POSE OPTIMIZATION

In this chapter, we extend the semi-infinite programming approach introduced in Chapter 2 by incorporating force variables into the optimization to impose balance constraints on the object. We introduce a novel formulation, Infinite Programming with Complementarity Constraints (IPCC) [34], designed to optimize force distribution across contact points while encoding constraints and objectives on the force distribution, such as force balance and Coulomb friction. We also detail a local optimization algorithm for solving IPCC using the exchange method [10]. This involves the strategic design of oracles, which iteratively select a finite subset of contact points and forces from an infinitely large set of potential contact points on the object's surface, and solving a sequence of finite-dimensional nonlinear constrained optimization problems instantiated on the chosen contact points. Applying IPCC to stable grasp pose optimization problems involving non-convex, complex-shaped objects, we demonstrate its efficacy in achieving rapid convergence to local optima. This marks the pioneering demonstration that contact-rich manipulation planning with high-fidelity geometry is indeed feasible. [1]

3.1 INTRODUCTION

Optimization is a central tool in robot planning, but a major limitation with existing tools is the representation of geometric contact in optimization, which is not easily captured as a numeric-valued constraint except for simple geometries like polyhedra and geometric primitives [35, 36]. In this study, we introduce an innovative approach designed to address the intricacies of handling complex and irregular geometries to optimize the poses of objects in contact while upholding conditions of force and torque equilibrium.

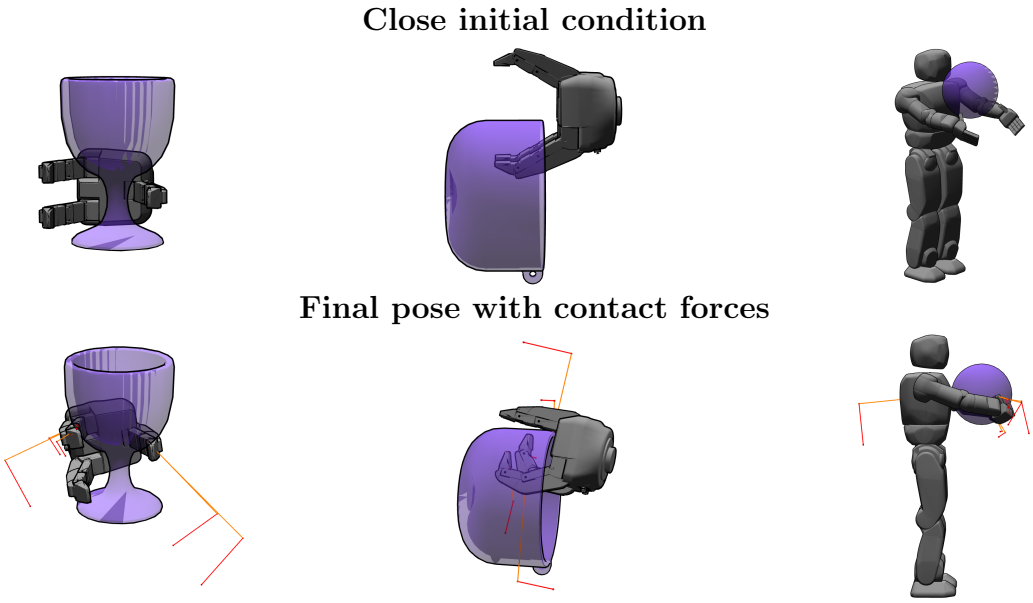
Our approach is based on the infinite programming (IP) paradigm, which represents a collision constraint as an infinite set of simpler constraints and dynamically instantiates a finite subset of these constraints during optimization. We present a novel formulation *infinite programming with complementarity constraints* (IPCC) that optimizes a force distribution over points of contact and encodes constraints and/or objectives on the force distribution, e.g., force balance and Coulomb friction. The complementarity condition, dictating that the force at the point may only be nonzero if contact is made at that point, bestows tractability

[1] This chapter is reproduced from Mengchao Zhang, and Kris Hauser, "Semi-infinite programming with complementarity constraints for pose optimization with pervasive contact". In the International Conference on Robotics and Automation (ICRA) 2021.

upon this method since the solver only needs to reason about the force distribution at points of active contact.

We present a local optimization algorithm for IPCC that uses the exchange method [29], which iteratively chooses a finite subset of contact points and forces to consider, and then formulates a finite-dimensional nonlinear constrained optimization problem to solve for a step direction. Using a judicious constraint selection procedure, called an oracle, the series of problems converges toward one that contains a true optimum of the original problem. Our discourse delves into the influence of various design choices pertaining to implementation, encompassing factors like the oracle, merit function, inner-loop optimization step size, and outer-loop line-search step-shrinking coefficient.

The proposed model is instantiated on both a gripper and a humanoid robot, aimed at optimizing their static poses to securely grasp objects while upholding force and torque equilibrium. Remarkably, the algorithm demonstrates rapid convergence to local optima even when confronted with intricate geometries, and the algorithm efficiently yields feasible poses within a matter of tens of seconds.



(a) Gripper, Wine glass (b) Gripper, Bowl (c) Humanoid, Sphere

Figure 3.1: Our algorithm optimizes for stability considering multiple contacts anywhere on the robot. The first row shows the initial configurations for three examples, and the bottom row shows final poses and contact forces. Normal forces are drawn in orange, and friction forces are drawn in red, translated to the end of the normal forces. [Best viewed in color.]

3.2 RELATED WORK

Grasping pose optimization is concerned with determining a robot's optimal pose for securely gripping an object. The pose quality assessment encompasses factors such as robot-self collision avoidance, robot-object collision avoidance, and ensuring static equilibrium of the object in the grasp.

Model-based grasp planners systematically explore poses using the object and robot geometry information. This exploration is accomplished through various techniques, including sampling-based approaches such as GraspIt! [37] and continuous optimization methods like those proposed by Chen et al. [38]. Sampling-based planners offer flexibility in defining grasp quality metrics and inherently address collision concerns. Nonetheless, they tend to converge slowly toward an optimum, especially when the robot's pose dimensionality is high. Conversely, continuous optimization techniques converge more swiftly, demanding differentiable geometry representations for encoding collision constraints and pose-function relationships. Recent research has introduced a two-stage grasp planning strategy [39] involving global grasp point optimization and subsequent gripper pose computation through inverse kinematics and collision constraints. However, this approach assumes convex geometries where each robot link contacts the object at a single point.

Learning-based grasp planners forecast grasp points and/or gripper poses by leveraging observed environmental data. Nonetheless, these methods necessitate substantial data generation efforts. For instance, in a study by Lu et al. [40], over 1500 grasp attempts were conducted in simulation, yielding only 159 successful grasps due to the gripper's consistent initialization and finger-closing protocol.

A compelling need exists for an efficient optimization-based pose planning method that can adeptly handle complex geometries while also guaranteeing the stability of the grasped object.

3.3 METHOD

3.3.1 Infinite Programming with Complementarity Constraints

Adding complementarity into semi-infinite programs

A semi-infinite programming problem is an optimization problem in finitely many variables $x \in \mathbb{R}^n$ on a feasible set described by infinitely many constraints:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.1a}$$

$$s.t. \quad g(x, y) \geq 0 \quad \forall y \in Y, \tag{3.1b}$$

where $g(x, y) \in \mathbb{R}^m$ is the constraint function, y denotes the index parameter, and $Y \in \mathbb{R}^p$ is its domain. In the case of pose optimization with collision constraints, x describes the pose of objects in the scene, y is a contact point on the surface of one object, Y denotes that surface, and g is the distance from the point to the other object. Typically, optimal solutions will be supported by contact at some points, i.e., $g(x^\star, y) = 0$ will be met for the optimal solution $x^\star$ at some set of points y.

An additional challenge is posed when forces need to be considered as part of the solution to ensure force and torque balance, since force variables need to be introduced to the optimization problem at each point of contact. We do this by defining a continuous field $z : Y \to \mathbb{R}^r$ in which is an optimization variable. Given an infinite number of contact points, infinitely many optimization variables will be instantiated, which makes the optimization become an infinite programming (IP) problem in which the number of variables and the number of constraints are both possibly infinite [41]. To ensure that forces are only felt at points where objects are in contact, the field is required to satisfy a complementarity condition $z(y)g(x, y) = 0 \quad \forall y \in Y$, which ensures that the force is nonzero only if the distance between the geometries is zero. Meanwhile, there may be some other inequality constraints $h(x, y, z(y)) \leq 0$ that need to be satisfied pointwise, such as friction constraints. Finally, we may require some constraints on the integral of the field over the domain, such as force and torque balance. In this way, we define an IPCC as a problem in the form:

$$\min_{x \in \mathbb{R}^n, z \in Y \to \mathbb{R}^r} f(x, z) = f_x(x) + \int_{y \in Y} f_z(x, y, z(y)) dy \tag{3.2a}$$

$$s.t. \quad g(x, y) \geq 0 \quad \forall y \in Y \tag{3.2b}$$

$$z(y)g(x, y) = 0 \quad \forall y \in Y \tag{3.2c}$$

$$z(y) \geq 0 \quad \forall y \in Y \tag{3.2d}$$

$$h(x, y, z(y)) \geq 0 \quad \forall y \in Y \tag{3.2e}$$

$$s(x, z) = s_x(x) + \int_{y \in Y} s_z(x, y, z(y)) dy = 0, \tag{3.2f}$$

To solve the IPCC problems using numerical methods, we hope that z only is non-zero at

a finite number of points. Indeed, if an optimal solution $x^\star$ is supported by a finite subset of index points $\tilde{Y} = (y^1, y^2, \ldots, y^N) \in Y$, then it suffices to solve for the values of z at these supporting points, since z should elsewhere be zero. We borrow this concept, which is used in the exchange method used to solve SIP problems [42], to solve the IPCC problem.

Exchange method and oracle

A discretization of an IPCC problem creates constraints and variables corresponding to a finite number N of instantiated index points $\tilde{Y} = (y^1, \ldots, y^N)$. Force variables $z = (z^1, \ldots, z^N)$ are instantiated for each index point (slightly abusing notation). To replace integrals with sums, the true distribution $z(y)$ is represented by a set of Dirac impulses: $z(y) = \sum_i \delta(y - y^i) z^i$. In this way, the IPCC problem is converted into a standard mathematical program with complementarity constraints (MPCC) over $(n + Nr)$ variables:

$$\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^{Nr}} f_x(x) + \sum_{i=1}^{N} f_z(x, y^i, z^i) \tag{3.3a}$$

$$s.t. \quad g(x, y^i) \geq 0 \quad \forall i = 1, ..., N \tag{3.3b}$$

$$z^i g(x, y^i) = 0 \quad \forall i = 1, ..., N \tag{3.3c}$$

$$z^i \geq 0 \quad \forall i = 1, ..., N \tag{3.3d}$$

$$h(x, y^i, z^i) \geq 0 \quad \forall i = 1, ..., N \tag{3.3e}$$

$$s_x(x) + \sum_{i=1}^{N} s_z(x, y^i, z^i) = 0, \tag{3.3f}$$

where r is the number of force variables for each index point.

In a slight abuse of notation, when discussing discretized IPCC problems we will write the discretized objective as $(x, \tilde{Y}, z)$, and stacked vectors of inequality constraints as $g(x, \tilde{Y})$ and $h(x, \tilde{Y}, z)$, and the force and torque balance constraint as $s(x, \tilde{Y}, z)$.

For a solution of the MPCC to correspond to a feasible impulse solution of the original IPCC, we note that the constraint function h must be conic in z, i.e., $h(x, y, z(y)) \geq 0 \implies h(x, y, c \cdot z(y)) \geq 0$ for any scaling $c > 0$.

To apply the exchange method to IPCC, we progressively instantiate index sets $\tilde{Y}_1, \tilde{Y}_2, \ldots$ and their finite-dimensional MPCCs whose solutions converge toward the true optimum [10]. Specifically, define P as the original IPCC, Q_k as the MPCC instantiation corresponding to $\tilde{Y}_k$, and let x_k^* be the solution to Q_k. If the index sets are chosen wisely, we expect that the iterates $x_1^\star, x_2^\star, \ldots$ will eventually approach an optimum of P. A naive approach would sample points incrementally from the domain Y (e.g., randomly or on a grid), and hopefully,

with a sufficiently dense set of points, the MPCC solutions will approach an optimal solution.

But this approach is inefficient as most samples will not affect the iterated solutions, and also whether the iterated solutions approach an optimal solution of the IPCC using this strategy is an open question.

The key question here is which new constraint should be selected, and oracle is a subroutine that performs this selection process. In [28], a maximum-violation oracle that selected closest / deepest penetrating points was used to avoid collisions between the robot and its environment. We argue that adding such points is necessary to solve our problem, but maximum-violation alone may not be sufficient because closest points may not encourage the instantiated MPCC toward finding a solution to the force balance constraints. We show that an oracle that balances the residual of the collision constraint against that of the force balance constraint, as described in Section 3.3.2, yields improved solve rates at the expense of additional computation time.

Also, it is possible to delete constraints from the constraint set when they are not deemed necessary (the "exchange"), which saves time in later MPCC solve steps. In our implementation, we delete the index points whose distances to the robot are bigger than a threshold and the forces assigned on that index points are smaller than some other threshold.

Optimization of instantiated MPCCs

MPCCs are generally difficult to solve since they are highly degenerate problems and they do not satisfy the majority of Constraint Qualifications established for standard nonlinear optimization [43]. But recently, they have attracted significant attention of operation researchers. The standard form of an MPCC problem is given in (3.4), where $0 \leq g(x, y) \perp z \geq 0$ means the two vectors are positive and orthogonal, $c_{\mathcal{I}}$ is the inequality constraint and $c_{\mathcal{E}}$ is the equality constraint.

$$\min_{x \in \mathbb{R}^n, z \in \mathbb{R}^p} f(x, z) \tag{3.4a}$$

$$\text{s.t. } c_{\mathcal{I}}(x, z, g(x, y)) \leq 0 \tag{3.4b}$$

$$c_{\mathcal{E}}(x, z, g(x, y)) = 0 \tag{3.4c}$$

$$0 \leq g(x, y) \perp z \geq 0. \tag{3.4d}$$

We convert (3.3) into an MPCC in the form of (3.4) to get the search direction $p = (d_x, d_z)$.

Anitescu shows that SQP with elastic mode converges globally for MPCCs [44]. Moreover, Fletcher et al. [45] presented a large collection of MPCC test problems and compared the performance of standard NLP solvers (SNOPT [46], Knitro [47] and loqo [48]) on those

problems. SQP methods are proved to be the most robust at solving MPCCs, and SNOPT had the best performance among all the tested solvers.

We use SNOPT in our implementation, and also take the most common smoothing approach for the complimentarity gap, which replaces $g(x,y)\cdot z = 0$ by $g(x,y)\cdot z \leq \tau$ and gradually driving τ to 0.

IPCC Outer Iteration

Once we get a step toward a desired point x_d, z_d, the IPCC outer loop moves from the current iterate (x_k, z_k) toward x_d, z_d. However, due to nonlinearity, the full step may lead to worse constraint violation. To avoid this problem, we perform a line search over the following merit function that balances reducing the objective and reducing the constraint error:

$$\phi(x, \tilde{Y}, z; \mu) = f(x,z) + \mu \|v(x, \tilde{Y}, z)\|_1, \tag{3.5}$$

where v denotes the vector of constraint violations of (3.3), which includes the negative components of each $g^-(x,y^i)$ and $h^-(x,y^i,z^i)$ term, each complementarity term $z^i g(x,y^i)$, and the force/torque balance term $s_x(x) + \sum_{i=1}^N s_z(x,y^i,z^i)$. We denote the negative component of a term as $\cdot^- \equiv \min(\cdot, 0)$. Also, in SIP for collision geometries, a serious problem is that using existing instantiated index parameters, a step may go too far into areas where the minimum of the inequality $g^*(x) \equiv \min_{y\in Y} g(x,y)$ violates the inequality, and the optimization loses reliability. So we add the max-violation $g_k^{-*}(x)$ to v. Moreover, since the scaling of the complementarity constraint is in different units from either g, h, or s, we allow a user-defined weight on each entry of v. Note that, with a fixed index set, our problem becomes a standard MPCC, which is solved using SQP. Therefore, the search direction of the inner problem must be a descendent direction of the above l_1-merit function, as analyzed in [49].

In [50], Eq. (18.33) gives a reasonable method for choosing μ to ensure that the chosen descent direction p_k computed by SNOPT descends the merit function. It is sufficient to choose any μ_k satisfying

$$\mu_k > \frac{\nabla_{xz} f(x_k, \tilde{Y}_k, z_k)^T p_k}{\|v(x_k, \tilde{Y}_k, z_k)\|}, \tag{3.6}$$

and hence we simply double the right-hand side.

One notable challenge is the complementarity constraint. Because (x_d, z_d) satisfies the constraint and (x_k, z_k) often is quite close to satisfying it, the midpoint between these points is likely to have a large constraint error. Hence, we perform a line search that backtracks

only by a small amount on each iteration (we use a 20% reduction).

For an accepted point of the line search that decreases the merit function, we also enforce the requirement that the penetration depth between two geometries is not too deep, because the quality of penetration depth and normal estimation degrades with depth. When the penetration depth exceeds a threshold, that is $g^\star(x) \leq g_{min}$, we add the detected penetration point as an index point to $\tilde{Y}_{k+2}$ in the next iteration.

Warm starting

Coherence between problems Q_{k-1} and Q_k may suggest the use of warm-starting to speed up solve. We also keep the forces assigned for each kept index point from Q_{k-1} and use it as the initial value for Q_k. By default, we set the guessed forces for new index points to 0 as listed in Alg. 3.1 Line 7.

The SNOPT iteration count should be set lower than the default value to avoid spending too much time on problems whose constraint sets are not sufficiently populated. Also, the step size of SNOPT should be set lower than the default value to avoid a deep penetration step.

3.3.2 Stable Grasping Formulation

We apply IPCC to solve for a gripper/humanoid robot to hold an object, while making sure that the object is in both force and torque balance. We assume that

1. the object geometry is known;
2. the mass and center of mass (CoM) of the object are known;
3. the friction coefficient between the robot and object is known;
4. the base joint of the gripper can provide arbitrary large force and torque.

Geometry modeling

To handle collision avoidance, we establish a infinite constraint $g(x, y)$ between the robot and the object. We perform some pre-computation to accelerate IPCC solve. The object is represented as a point cloud with resolution 1 mm, and the robot links are represented as a signed distance field (SDF) with resolution 0.8 mm, which supports $O(1)$ depth lookup and $O(1)$ gradient estimation at a point.

Algorithm 3.1 IPCC Solver

Require: N_{outer}^{max}, N_{inner}^{max}, step size tolerance ϵ_x, complementarity gap tolearance ϵ_{gap}, balance tolerance ϵ_s, penetration tolerance ϵ_p, index deletion thresholds z_{min} and g_{max}, initial guess $x_{init} \in \mathbb{R}^n$

$x_0 \leftarrow x_{init}$
$\tilde{Y}_0 = [\,]$ ▷ Initialize empty constraint set
$z_0 \leftarrow \emptyset$ ▷ Initialize empty force vector
for $k = 0, \ldots, N_{outer}^{max} - 1$ **do**
    ▷ Update constraint set and guessed forces $\tilde{z}_k$
    For any $y^i \in \tilde{Y}_k$ with corresponding z^i where $z^i \geq z_{min}$ and $g(x_k, y^i) \leq g_{max}$, add y^i to $\tilde{Y}_{k+1}$ and z^i to $\tilde{z}_k$
    Run the oracle to add one or more new points to $\tilde{Y}_{k+1}$. Initialize their force(s) z^i to 0
    ▷ Solve for step direction
    $\tau_0 \leftarrow \tilde{z}_k^T g(x_k, \tilde{Y}_{k+1})$
    Run SNOPT to yield the desired endpoint x_d, z_d
    ▷ Line Search
    $\alpha \leftarrow 1$
    $\Delta x \leftarrow x_d - x_k$, $\Delta z \leftarrow z_d - \tilde{z}_k$
    Calculate μ_k from (3.6)
    $score_0 = \psi(x_k, \tilde{Y}_{k+1}, \tilde{z}_k; \mu_k)$
    converged $\leftarrow$ false
    while ¬converged **and** $n_{inner} < N_{inner}^{max}$ **do**
        $x \leftarrow x_k + \alpha \Delta x$, $z \leftarrow \tilde{z}_k + \alpha \Delta z$
        $score = \psi(x, \tilde{Y}_{k+1}, z; \mu_k)$
        if $score \leq score_0$ **then**
            converged $\leftarrow$ true
            if $\min_{y \in Y} g(x, y) \leq g_{min}$ **then**
                Add $y^* = \arg\min_{y \in Y} g(x, y)$ to $\tilde{Y}_{k+2}$
            end if
        else
            $\alpha \leftarrow \alpha \cdot 0.8$
            $n_{inner} \leftarrow n_{inner} + 1$
        end if
    end while
    ▷ Update state and test for convergence
    $x_{k+1} \leftarrow x_k + \alpha \Delta x$
    $z_{k+1} \leftarrow \tilde{z}_k + \alpha \Delta z$
    if $\alpha \|\Delta x\| \leq \epsilon_x$ and $z_{k+1}^T g(x_{k+1}, \tilde{Y}_{k+1}) \leq \epsilon_{gap}$ and $\sum g_{k+1}^{-*}(x) < \epsilon_p$ and $\|s_0\| < \epsilon_s$ **then return** x_{k+1},z_{k+1}
    end if
end for

Friction force modeling

We model the contacts between the robot and each index point on the object as point contacts with dry friction. The i'th force variable $z^i = (f_i^N, f_{i,1}^F, \ldots, f_{i.j}^F)$ represents the force applied at the i'th contact y^i, which is divided into the normal component f_i^N and j frictional components $f_{i,j}^F$ along the edges of a polyhedral approximation of the friction cone [51]. Given the normal vector $\mathbf{n}_i$ along the outward surface normal, and d tangential directions $\mathbf{t}_{i,j}$, the overall force applied at at y^i is $\mathbf{f}_i = f_i^N \mathbf{n}_i + \sum_j f_{i,j}^F \mathbf{t}_{i,j}$. Each of the components of z^i is required to be non-negative ($z^i \geq 0$), and given the friction coefficient μ_i, the friction cone constraints are given by $\sum_j f_{i,j}^F \leq \mu_i f_i^N$.

Force and torque balance

We establish an integral equality constraint on the object force and torque balance. Force balance requires that the force exerted by the object on the robot matches the gravity force of the object, $\sum_i \mathbf{f}_i = m\mathbf{g}$. Torque balance requires that the torque applied to the object to be zero, $\sum_i \mathbf{r}_i \times (-\mathbf{f}_i) = 0$, where $\mathbf{r}_i = y^i - \text{CoM}$ is the vector from the CoM of the object to the index point y^i.

We adopt a heuristic to reduce the number of constraints in the complementarity condition and accelerate solve times. By only applying complementarity to the normal component, $g(x, y^i) \cdot f_i^N = 0$, we reduce the number of complementarity constraints from Nr to N, and the problem is unchanged because $f_i^N = 0 \Rightarrow z^i = 0$ due to the friction cone constraint.

Custom oracle

As discussed in Section III.B, an oracle should be designed to instantiate a sequence of contact points y that help the IPCC find a feasible solution. The ***Maximum violation oracle*** (MVO) chooses the closest point y for each link $\arg\min_{y \in Y} g(x, y)$, but fails to consider the balance and complementarity constraints.

In Alg. 3.1, any (deepest) penetrating point is automatically added to the constraint set, so we turn our attention to non-penetrating index points. The complementarity constraint alone is not sufficient to select y, since for any y for which $g(x, y) > 0$, we can let $z^i = 0$ to satisfy complementarity. Instead, we must consider the interaction of force-torque balance with the other constraints. The balance residual at a new point y, if we were to find a force z, is $s_0(x) + s_z(x, y, z)$, where $s_0(x) = s_x(x) + \sum_{i=1}^N s_z(x, y^i, z^i)$ is the residual at the current index points. Then, an ideal new index point y, would enable solving for a new pose x' and force z to simultaneously satisfy $g(x', y) \geq 0$, $h(x', y, z) \geq 0$, and $s_0(x') + s_z(x', y, z) = 0$.

Solving these constraints directly would require finding a constraint-minimizing (x', z) for every $y \in Y$, which would be expensive to solve. So we propose an approximate score that is expected to take large values for high quality y. We define $-s_0(x) \cdot s_z(x, y, 1)$ as a score for the balance constraint, and $\exp(-g(x, y))$ as a score for the contact constraint, since a point with a small $g(x, y)$ has a good chance to satisfy the complementarity constraint if a force would be assigned at that point.

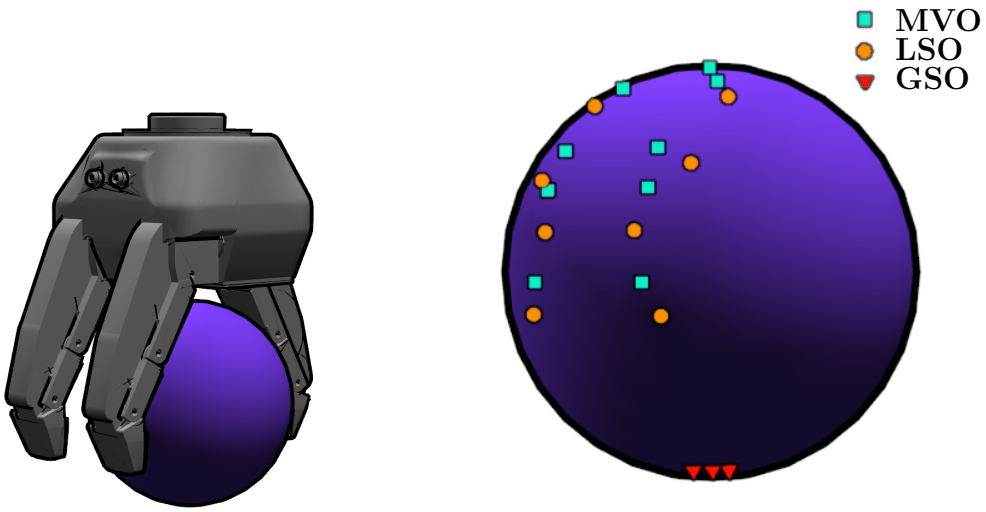


Figure 3.2: Comparison of different oracles at the given pose. Squares, circles and triangles indicate the index points instantiated by MVO, LSO and GSO, respectively. [Best viewed in color.]

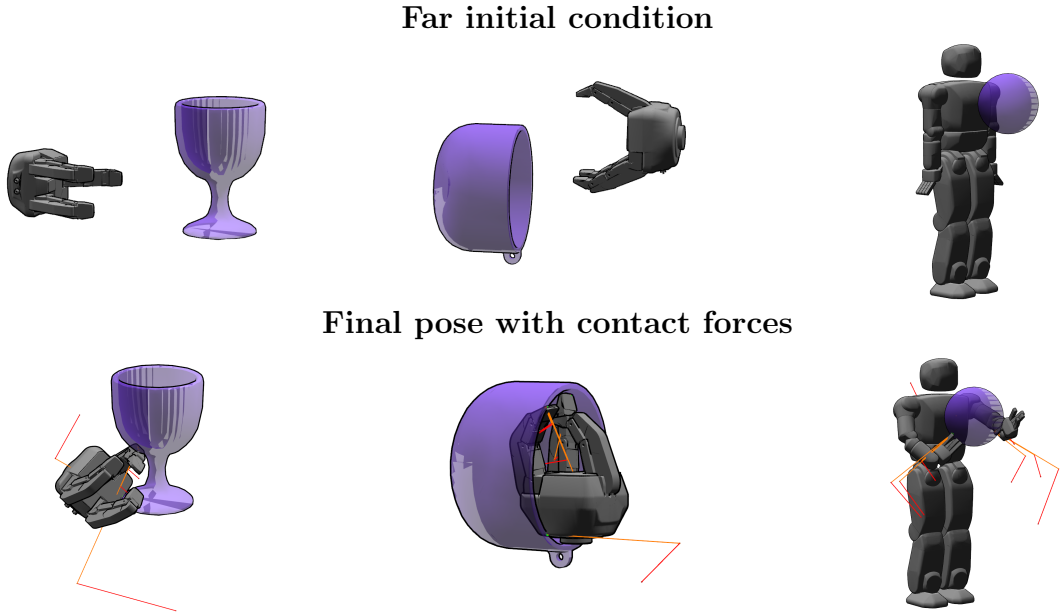
We define an overall score $w_1 \cdot (s_0 \cdot s_z(x, y, 1)) + w_2 \cdot exp(-g(x, y))$ that should be maximized at a high-quality point, where w_1 and w_2 are custom weights. This is too expensive to minimize over the entire domain Y, so we propose an alternate approach, called the ***Local score oracle*** (LSO), that only minimizes over a neighborhood of the MVO point. Overall, we choose $y^* = \arg\max_{y \in B_r(y_M^*)} [w_1 \cdot (s_0 \cdot s_z(x, y, 1)) + w_2 \cdot exp(-g(x, y))]$, where $B_r(y_M^*)$ is an neighborhood of the MVO point y_M^* with radius r. Here, r is a user-defined parameter. When $r = 0$, LSO is equivalent to MVO, and when $r = \infty$, the optimization occurs over the entire object, which we call the ***Global Score Oracle*** (GSO). We perform this computation for each robot link.

The index points instantiated by the three oracles at the same configuration are shown in Fig. 4.2. For LSO, r is chosen to include $2,000$ points around the MVO point. From the results we can see that the GSO finds points that would immediately cause balance constraints to be met, but does not respect the initial pose. LSO strikes a balance between proximity and points that have better normals to apply upward forces.

3.4 EXPERIMENTS AND DISCUSSION

The IPCC algorithm is implemented with front end in the Python programming language, with the SNOPT solver [46], and custom C++ collision detection software. The communication between Python and SNOPT is through the pyOptSparse [52]. The Klamp't library is used for robot kinematics, collision queries, and visualization [53]. The objects are taken from Princeton Shape Benchmark [54] and Ycb benchmark object and model set [55]. All experiments were run on a single core of a 3.6 GHz AMD Ryzen 7 processor.

To show the generality of our algorithm, we first solve for a Robotiq Adaptive 3-finger gripper holding a wine glass and a bowl, and a HUBO2 humanoid robot (which can be viewed as a giant gripper) holding a large sphere. In this and subsequent experiments, we set the minimum signed distance to $g_{min} = -10^{-3}$ m. The solver terminates with success if the L_1 norm of penetrations of all links is smaller than 1 mm, the complementary gap is smaller than 10^{-2} N·m, and the balance residual is smaller than 0.01.



(a) Gripper, Wine glass (b) Gripper, Bowl (c) Humanoid, Sphere

Figure 3.3: The same test cases as in Fig. 3.1 but with farther initial conditions. IPCC is still able to find a pose to hold the object, and sometimes determines unusual strategies. [Best viewed in color.]

We start the robot first from a close initial position (Fig. 3.1) and then a faraway initial position (Fig. 3.3). The results show that our method can generate contact with "unusual" parts of the geometry. The gripper supports the object using the side of the finger, the end of the finger, the palm, and even the back of the finger. The humanoid cradles the sphere

between its torso and hands. Moreover, multiple contacts can happen on the same robot link, which distinguishes our method from prior works on grasp planning, e.g. [39], where precision grasps are planned with designated contact points. The test results are summarized in Table 3.1.

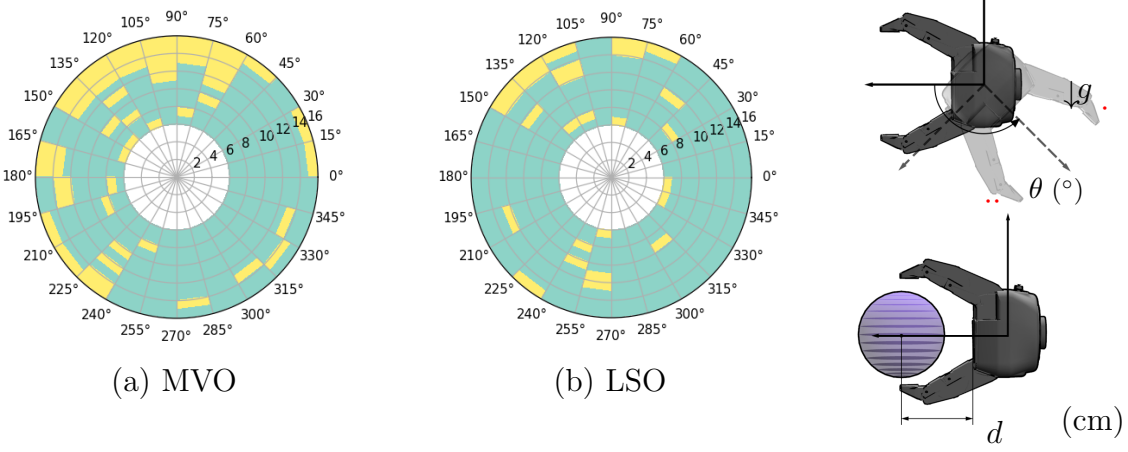
Table 3.1: Test results, listing # points in the point cloud (# in PC), robot degrees of freedom (DoF), outer iterations (Iter), computation time (Time), complementarity gap at final pose (Comp Gap), balance residual at final pose (Bal Res), sum of penetration depth of all the robot links (Pen), # contact points at final pose (# Contact), average # of index points instantiated in each iteration (Ave Index) and average # of index points kept after "exchange" (Ave Active Index).

Test Case	# in PC	DoF	Initial	Iter	Time (s)	Comp Gap (N · m)	Bal Res	Pen (m)	# Contact	Ave Index	Ave Active Index
Gripper, Wine Glass	436,804	18	Close	13	33.1	7.1e-3	3.5e-13	3.7e-4	10	27.0	4.8
			Far	10	30.6	4.8e-3	3.7e-13	8.5e-4	6	23.8	1.4
Gripper, Bowl	674,594	18	Close	11	36.2	1.2e-3	2.1e-14	2.6e-4	21	27.8	7.8
			Far	10	27.3	9.9e-6	1.2e-14	8.6e-4	10	24.7	3.2
Humanoid, Sphere	28,362	63	Close	13	103.9	2.0e-3	2.8e-13	5.4e-4	8	61.7	3.8
			Far	23	185.3	5.0e-3	1.7e-6	6.0e-5	5	59.7	1.4

Next, we examine the convergence of IPCC under different initial conditions and oracle choices (Fig. 3.4). The gripper is required to grasp a sphere. We initialize the gripper from different positions (0.06–0.12 m, 0.02 m/step) and orientations ($[0,2\pi)$, $\frac{\pi}{12}$/step) around a circle. MVO and LSO oracles are compared, and for LSO r is chosen so that $B_r(y_M^*)$ contains 2,000 points. We observe that MVO's performance decreases rapidly when the initial distance between the gripper and the object increases, while LSO is much more robust. Moreover, the success rate of the left half circle is lower than the right half circle, which is caused by the fact that the single-finger side of the gripper is underneath the object when $\theta \in (90°, 270°)$. Besides, the success rate of the upper half circle is lower than the lower half circle, and this is because when the initial index points allow the inner optimization to find a feasible solution, the algorithm is likely to find a solution. Otherwise, more burden is put on the oracle to choose index points to guide the gripper to a feasible configuration. Detailed test results are summarized in Table 3.2.

Table 3.2: Convergence test results include mean, standard deviation (Std) and median of the computation time (Time), sum of penetration depth of all the robot links (Pen), complimentarity gap at the final pose (Comp Gap), balance residual at the final pose (Bal Res), and success rate.

Oracle		Time (s)	Pen (m)	Comp Gap (N · m)	Bal Res	Success (%)
LSO	Mean	27.71	5.2e-4	1.2e-3	2.3e-3	89.17
	Std	15.52	2.8e-4	1.7e-3	1.2e-3	
	Median	23.03	5.3e-4	5.1e-4	9.3e-16	
MVO	Mean	13.99	4.9e-4	1.6e-3	1.4e-3	78.75
	Std	12.53	2.8e-4	2.1e-3	8.1e-4	
	Median	11.28	5.1e-4	8.1e-4	7.1e-16	



(a) MVO (b) LSO

Figure 3.4: Convergence results of MVO and LSO oracles, for sphere grasping. Green is success and yellow is fail. The angular axis represents the CCW rotation angle of the gripper in the vertical plane, with gravity pointing downwards. The radial axis represents the distance in cm between the CoM of the sphere and the palm plane of the gripper. Red dots indicate the number of fingers on each side. [Best viewed in color.]

CHAPTER 4: SIMULTANEOUS TRAJECTORY OPTIMIZATION AND CONTACT SELECTION

In Chapter 3, we introduced the innovative formulation of Infinite Programming with Complementarity Constraints (IPCC) and its application to stable grasp pose planning problems. This chapter expands IPCC's application from pose optimization to contact-rich manipulation trajectory optimization, introducing the novel contact-implicit trajectory optimizer, Simultaneous Trajectory Optimization and Contact Selection (STOCS) [56]. STOCS, as an extension of IPCC, inherits IPCC's advantage of planning with non-convex, complex-shaped objects. It represents the first method capable of planning contact-rich manipulation trajectories using high-fidelity geometric representations in 3D. Experimental results demonstrate its efficacy in planning for pushing, pivoting, sliding, and rolling trajectories using dense point clouds of daily objects. [1]

4.1 INTRODUCTION

Humans and other organisms treat contact as a fact of life and utilize contact to perform dexterous manipulation of objects and agile locomotion. In contrast, the majority of current robots avoid making contact with objects as much as possible and tend to avoid contact-rich manipulations like pushing, pivoting, sliding, and rolling objects [57, 58].

Trajectory optimization [59] is a tool used throughout robotics to generate robot motion, but the representation of making and breaking contact remains a major research challenge. A hybrid trajectory optimization approach divides a trajectory into segments in which the set of contacts remains constant, but it requires the contact mode sequence to be known in advance [60] or explored by an auxiliary discrete search.

Contact-implicit trajectory optimization (CITO) is a more flexible approach that allows the optimizer to choose the sequence of contact by formulating contact as a complementarity constraint, which ensures that the contact forces can be non-zero if and only if a point is in contact [4]. Although the resulting mathematical programming with complementary constraint (MPCC) formulation is less restrictive than hybrid trajectory optimization, it still requires a set of predefined allowable contact points on the object. Moreover, it is well known that MPCC becomes very challenging to solve as the number of complementarity constraints increases, so past CITO methods were strictly limited to a small handful of

[1] This chapter is adapted from Mengchao Zhang, Devesh K. Jha, Arvind Raghunathan, and Kris Hauser, "Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning." In Robotics: Science and Systems (RSS), 2023, and its journal extension that is in preparation.

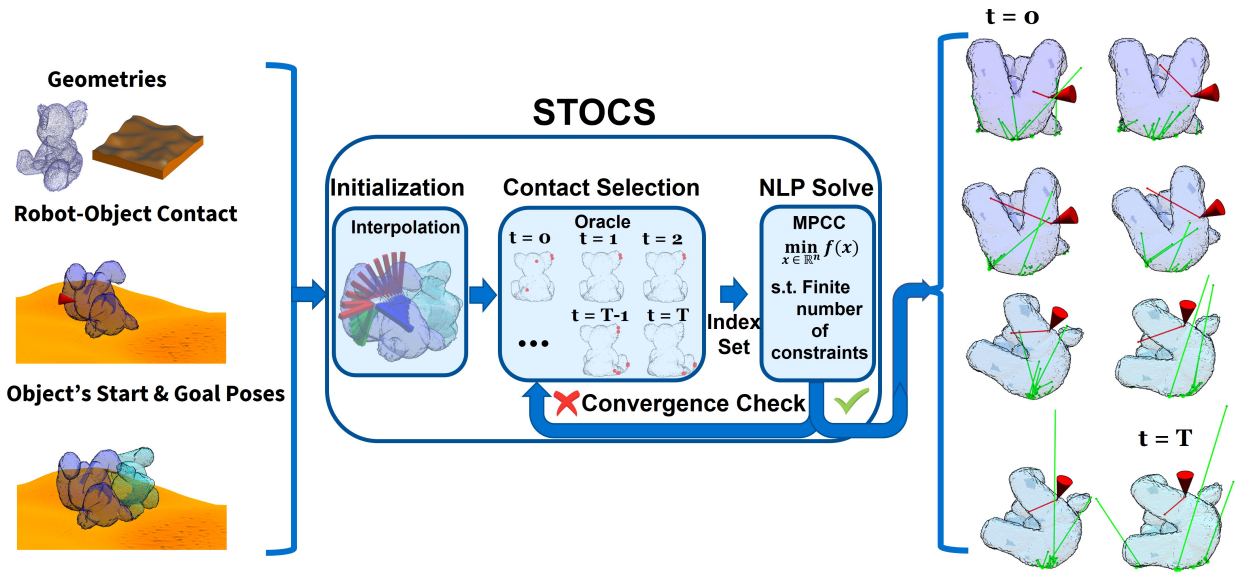


Figure 4.1: Leveraging the high-fidelity geometric representations of the object (represented by a dense point cloud) and the environment (represented by a signed distance field) as shown in the upper left figure, alongside the robot's contact as depicted in the middle left figure and the specified start and goal poses of the object as shown in the bottom left figure. The STOCS algorithm efficiently plans for a trajectory that includes the interaction between the object and the environment throughout its course, as illustrated in the upper right figure. For enhanced clarity, the object is depicted from a different angle. Additionally, the configuration trajectory of the object is detailed in the middle right figure, while the desired joint manipulation force is represented by red lines in the bottom right figure. [Best viewed in color.]

potential contact points, which restricted their applicability to objects with non-convex and complex shapes.

This study introduces the simultaneous trajectory optimization and contact selection (STOCS) algorithm to address the scaling problem in contact-implicit trajectory optimization. It applies an infinite programming (IP) approach to dynamically instantiate possible contact points between the object and environment inside the optimization loop, and hence the resulting MPCCs become far more tractable to solve. Experimental results demonstrate that STOCS can solve for manipulation trajectories involving changes of contact between complex-shaped objects and environments.

4.2 RELATED WORK

Trajectory optimization [61], which solves for an optimal control solution that is valid from a specific initial condition for high-dimensional systems, stands as a crucial tool in robotics

for generating robot motion. However, the representation of making and breaking contact remains a major research challenge, which impedes the planning of trajectories involving complex contact interactions.

One approach, hybrid trajectory optimization [62, 63], segments a trajectory into intervals in which the set of contacts remains constant, and it allows to cast trajectory optimization as one large non-linear optimization which can be solved to optimize the timings and variables associated with each of the individual modes [64]. It requires the contact mode sequence to be known in advance [60] or explored by an auxiliary discrete search, which becomes intractable in all but the simplest problems.

An alternative, contact-implicit trajectory optimization (CITO), which has been studied extensively in dexterous manipulation and legged locomotion literature [3, 4, 65, 66], is a more flexible approach that allows the optimizer to choose the sequence of contact by formulating a complementarity constraint. The complementarity constraint ensures that the contact forces can be non-zero if and only if a point is in contact [4]. Although this mathematical programming with complementary constraint (MPCC) formulation is less restrictive than hybrid trajectory optimization, it still requires a set of predefined allowable contact points on the object. Moreover, running times rise sharply as the number of allowable points grows, which restricts its use to simple geometries.

The low efficiency in handling complex geometries of CITO significantly limits its applicability. Adhering to the complex object model leads to prolonged solution times, while simplifying the object model may cause execution failures on real hardware due to the resulting model mismatch. A planner capable of directly operating on the continuous geometry of complex objects holds the potential to generate more physically realistic motions, facilitating smoother transfer to hardware.

4.3 METHOD

Given a start pose and a goal pose, a trajectory that includes the object's motion and the control inputs of the manipulator needs to be planned. In this section, we describe the inputs and outputs of our method in detail, and explain how we use STOCS to solve contact-rich manipulation trajectory optimization problems.

4.3.1 Problem Description

Our method requires the following information as inputs:

1. Object initial pose region: $Q_{init} \subset SE(2)$ or $SE(3)$.
2. Object goal pose region: $Q_{goal} \subset SE(2)$ or $SE(3)$.
3. Object properties: a rigid body $\mathcal{O}$ whose geometry, mass distribution, and friction coefficients with both the environment μ_{env} and the manipulator μ_{mnp} are known.
4. Environment properties: rigid environment $\mathcal{E}$ whose geometry is known.
5. Robot's contact with the object: c_{mnp}.

Our method will output a trajectory τ which includes the following information at time t:

1. Object's configuration: q_t.
2. Manipulator's configuration: q_t^{mnp}.
3. Manipulation force: u_t.
4. Object's contact with the environment: y_t.
5. Object-environment contact force: z_t.

In this study, we treat all objects and environments as rigid bodies and we assume the contact between the manipulator and the object is sticking contact. Furthermore, users are provided with the flexibility to opt for either quasistatic or quasidynamic assumptions based on their specific requirements. Under the quasistatic paradigm, inertial forces are considered negligible, necessitating that the object remains in a state of force and torque equilibrium at all times. Quasidynamic manipulation accounts for scenarios where tasks may involve occasional dynamic periods, during which accelerations do not integrate into substantial velocities, and both momentum and the effects of impact restitution are negligible [67].

4.3.2 STOCS Trajectory Optimizer

STOCS is a novel CITO algorithm for manipulation, and it enables the application of CITO on complex object and environment geometries by embedding the detection of salient contact points and contact times inside the trajectory optimization outer loop. Each instantiated MPCC iteration has relatively few constraints and is optimized for a handful of inner iterations before new contact points are identified.

Semi-infinite programming (SIP), infinite programming (IP) for contact-rich trajectory optimization. To illustrate how STOCS works, we start from the SIP/IP problem

that STOCS is designed to solve, and explain where the infinitely many optimization variables and constraints come from. An SIP problem is an optimization problem in finitely many variables $q \in \mathbb{R}^n$ on a feasible set described by infinitely many constraints:

$$\min_{q \in \mathbb{R}^n} f(q) \tag{4.1a}$$

$$\text{s.t. } g(q, y) \geq 0 \quad \forall y \in Y \tag{4.1b}$$

where $g(q, y) \in \mathbb{R}^m$ is the constraint function, y denotes the index parameter, and $Y \in \mathbb{R}^p$ is its domain.

In the case of pose optimization with collision constraints, q describes the pose of the object, y is a point on the surface of the object $\mathcal{O}$, where $Y \equiv \partial\mathcal{O}$ denotes that surface of $\mathcal{O}$ that has infinitely many points, and $g(\cdot, \cdot)$ is the signed distance from a point to the environment. Typically, optimal solutions will be supported by some points, i.e., $g(q^\star, y) = 0$ will be met for the optimal solution $q^\star$ at some set of points y.

In the case of trajectory optimization, constraints need to be instantiated in space-time, which means $y = (y_t, y_p)$ includes both active time y_t and active contact point y_p on the object's surface. Thus $Y \equiv \{(y_t, y_p) \mid y_t \in [0, t_{end}] \text{ and } y_p \in \mathcal{O}\}$ where t_{end} is the end time of the trajectory. In Section 4.3.3, we introduce two different oracles, the Maximum Violation Oracle (MVO) and the Time-active Maximum Violation Oracle (TAMVO). MVO assumes that a contact point, once established, remains active throughout the entire trajectory, which is straightforward to implement and makes the optimization stable. However, it introduces superfluous constraints into the optimization problem, given that the active contact points at the start of a trajectory may differ substantially from those at its conclusion. TAMVO allows different contact points to be selected for different time steps along the trajectory, resulting in a smaller optimization problem compared with MVO. We define $Y_t \equiv \{(y_t, y_p) \mid y_t = t \text{ and } y_p \in \mathcal{O}\}$ as the domain of index points for a specific time t.

To ensure the force and moment balance of the object, force variables need to be introduced to the optimization problem for each index point. We do this by defining $z : Y \to \mathbb{R}^r$ as an optimization variable. In 2D scenario, for a given contact point y, $z = [z^n, z^+, z^-]$ is expressed in a reference frame with z^n normal to the contact surface, and z^+, z^- tangent to the contact surface. In 3D scenario, following [4], to preserve the MPCC structure of the resulting optimization problem, we use a polyhedral approximation of the friction cone [51]. For a given index point y, $z = [z^n, z^{t1}, \cdots, z^{td}]$ is expressed in a reference frame with z^n normal to the contact surface, and $z^{t1}, \cdots, z^{td}$ tangent to the contact surface. The convex hull of the unit vectors along the directions of $z^{t1}, \cdots, z^{td}$ in $\mathbb{R}^2$ forms the polyhedral

approximation. Given an infinite number of contact points, infinitely many optimization variables will be instantiated, which makes the optimization become an **IP** problem in which the number of variables and the number of constraints are both possibly infinite [41].

In our formulation, at time t along a trajectory, the following constraints need to be satisfied:

1. **Bound Constraint**:

$$q_t \in \mathcal{Q},\ u_t \in \mathcal{U},\ z(y) \in \mathcal{Z} \quad \forall y \in Y_t \tag{4.2}$$

2. **Distance Complementarity Constraint**:

$$0 \leq z(y) \perp g(q_t, y) \geq 0 \quad \forall y \in Y_t \tag{4.3}$$

3. **Force Inequalities**:

$$h(q_t, y, z(y)) \geq 0 \quad \forall y \in Y_t \tag{4.4}$$

4. **Control Inequalities**:

$$c(q_t, u_t) \geq 0 \tag{4.5}$$

5. **Integral Constraint**:

$$\underbrace{s_{q,u}(q_t, u_t) + \int_{y \in Y_t} s_z(q_t, y, z_t(y)) dy}_{=: s(q_t, u_t, z; Y_t)} = 0 \text{ or } M\dot{v} \tag{4.6}$$

Eq. (4.3) ensures that nonzero forces are only exerted at points where objects are in contact, i.e. $z(y) \geq 0$ only if contact is made at y_p at time y_t, that is $g(q, y) = 0$. Friction cone constraints are included in the inequalities $h(q, y, z(y)) \geq 0$ in (4.4). We constrain the control input in (4.5) to make sure the manipulator can only push the object rather than pull the object. Additionally, this constraint guarantees that the manipulation contact remains a sticking contact, with the contact force maintained within the bounds of the corresponding friction cone. Finally, force and torque balance is expressed in (4.6) as an integral of the force field over the domain Y_t. Here, $s_{q,u}(q, u)$ represents the force and torque applied by gravity and the manipulator, and $s_z(q, y, z(y))$ represents the force and torque applied on an index point y by the contact force $z(y)$. The integral gives the net force and torque experienced by the object, which should be 0 if we assume quasistatic and be $M\dot{v}$ if we

assume quasidynamic, where M is the inertia matrix of the object $\mathcal{O}$ and v is the velocity of the object.

In practice, we solve the trajectory optimization problem in discrete time, where the time duration t_{end} of a trajectory is discretized into T time steps with step duration Δt, and then $\dot{v}$ can be written as $v/\Delta t$. The above constraints are imposed at each of the discretized time steps along the trajectory, along with additional constraints that make sure the relative tangential velocity at a contact is zero when the corresponding friction force is inside the friction cone ((4.7e) below). Hence, we formulate the following infinite programming with complementarity constraints trajectory optimization (IPCC-TO) problem denoted as $P(Y)$:

$$\min_{q,v,u,z} \; f(q,v,u,z) \tag{4.7a}$$

$$\text{s.t. } q_0 \in \mathcal{Q}_{init}, q_T \in \mathcal{Q}_{goal} \tag{4.7b}$$

$$(2),(3),(5),(6), v \in \mathcal{V} \quad \forall t \in \mathcal{T} \tag{4.7c}$$

$$q_t - q_{t+1} + v_{t+1}\Delta t = 0 \quad \forall t \in \mathcal{T} \setminus \{T\} \tag{4.7d}$$

$$0 \le w(q_t, v_t, y) \perp h(q_t, y, z_t(y)) \ge 0 \quad \forall y \in Y_t, \; \forall t \in \mathcal{T} \tag{4.7e}$$

where $f(q,v,u,z) \coloneqq \sum_{t\in\mathcal{T}}[f_{q,v,u}(q_t,v_t,u_t) + \int_{y\in Y_t} f_z(q_t,y,z(y))dy]$. For the sake of brevity, we use the notation $q = [q_0, \cdots, q_T]$, $v = [v_0, \cdots, v_T]$, $u = [u_0, \cdots, u_T]$, and $z = [z_0, \cdots, z_T]$, in which $z = [z(y)\ \forall y \in Y_t]$ is a concatenation of all the instantiated variable for all $y \in Y_t$.

To solve the IPCC-TO problem, STOCS (Alg. 4.1) includes the following subroutines.

Exchange method. The IPCC-TO problem $P(Y)$ not only has infinitely many constraints, but also introduces a continuous infinity of variables in z. To solve it using numerical methods, we hope that z only is non-zero at a finite number of points. Indeed, if an optimal solution $q^\star$ is supported by a finite subset of index points $\tilde{Y} \in Y$, then it suffices to solve for the values of z at these supporting points, since z should elsewhere be zero. This concept is used in the exchange method to solve SIP problems [42], and we extend it to solve IPCC-TO.

Since we discretize the duration of a trajectory into T time steps, we have $\tilde{Y} \equiv [\tilde{Y}_0, \cdots, \tilde{Y}_T]$. The exchange method progressively instantiates finite index sets $\tilde{Y}$ and their corresponding finite-dimensional MPCCs whose solutions converge toward the true optimum [10]. The solving process can be viewed as a bi-level optimization. In the outer loop, index points are selected by an oracle to be added to the index set $\tilde{Y}$, and then in the inner loop, the optimization $P(\tilde{Y})$ is solved. The outer loop will then decide how much should move toward

the solution of $P(\tilde{Y})$. Specifically, if we let $(\tilde{x}^* = [q^*, v^*, u^*], \tilde{z}^*)$ be the optimal solution to $P(\tilde{Y})$, then as $\tilde{Y}$ grows denser, the iterates of $(\tilde{x}^*, \tilde{z}^*)$ will eventually approach an optimum of $P(Y)$.

Given a finite number of instantiated contact points $\tilde{Y} \subset Y$, we can solve a discretized version of the problem which only creates constraints and variables corresponding to $\tilde{Y}$. Also, through replacing the distribution of $z(y)$ with Dirac impulses, integrals are replaced with sums and we formulate the finite MPCC problem $P(\tilde{Y})$ in the following form:

$$\min_{q,v,u,z} \ \tilde{f}(q, v, u, z) \tag{4.8a}$$

$$\text{s.t. } (2), (3), (5), v_t \in \mathcal{V} \quad \forall t \in \mathcal{T} \tag{4.8b}$$

$$(4.7b), (4.7d), (4.7e) \tag{4.8c}$$

$$\tilde{s}(q_t, u_t, z_t; \tilde{Y}_t) = 0 \quad \forall t \in \mathcal{T} \tag{4.8d}$$

where $\tilde{f}(q, v, u, z) \coloneqq \sum_{t=0}^{T} [f_{q,v,u}(q_t, v_t, u_t) + \sum_{y \in \tilde{Y}_t} f_z(q_t, y, z(y))]$, and $\tilde{s}(q_t, u_t, z; \tilde{Y}_t) = s_{q,u}(q_t, u_t) + \sum_{y \in \tilde{Y}_t} s_z(q_t, y, z(y)) = 0$.

Oracle. The optimal solution to an IP problem typically relies on a finite subset of index points. Identifying an effective selection strategy for these points from the domain Y, which comprises an infinite number of index points, constitutes the key to solving the IP problem. A naive approach would sample index points incrementally from Y_t (e.g., randomly or on a grid) at each of the discretized time steps along the trajectory, and hopefully, with a sufficiently dense set of points the iterates of solutions will eventually approach an optimum. However, this approach is inefficient, as most new index points will not yield active contact forces during the iteration.

In this study, we introduce two different oracles, the Maximum Violation Oracle (MVO) and the Time-active Maximum Violation Oracle (TAMVO), which are more effective than the naive sampling approach. The details of these two oracles will be discussed in 4.3.3.

Merit function for the outer iteration. After solving $P(\tilde{Y})$ in an outer iteration, we get a step direction from the current iterate $(\tilde{x}, \tilde{z})$ toward $(\tilde{x}^*, \tilde{z}^*)$, where $x = [q, v, u]$ is a concatenation of all the optimization variables except for z. However, due to nonlinearity, the full step may lead to a worse constraint violation for the original problem $P(Y)$. To avoid this, we perform a line search over the following merit function that balances reducing the objective and reducing the constraint error on the infinite dimensional problem $P(Y)$:

$$\phi(x, z; \mu) = f(x, z) + \mu \|b(x, z)\|_1, \tag{4.9}$$

Algorithm 4.1 STOCS

Require: q_{start}, q_{goal}, c_{mnp}

$Y^0 = [\,]$ ▷ Initialize empty constraint set

$z_0 \leftarrow \emptyset$ ▷ Initialize empty force vector

$x_0 \leftarrow$ initialize trajectory(q_{start}, q_{goal}, c_{mnp})

for $k = 1, \ldots, N^{max}$ **do**

    ▷ Update constraint set and guessed forces z_k

    Add all points in Y^{k-1} to Y^k, and initialize their forces in z_k with the corresponding values in z_{k-1}

    Call the oracle to add new points to Y^k, and initialize their corresponding forces in z_k

    $x_k \leftarrow x_{k-1}$

    ▷ Solve for step direction

    Set up inner optimization $P_k = P(Y^k)$

    Run S steps of an NLP solver on P^k, starting from x_k, z_k

    if P_k is infeasible **then return** INFEASIBLE

    else

        Set x^*, z^* to its solution, and $\Delta x \leftarrow x^* - x_k$, $\Delta z \leftarrow z^* - z_k$

        Do backtracking line search with at most N_{LS}^{max} steps to find optimal step size α such that $\phi(x_k + \alpha\Delta x, z_k + \alpha\Delta z; \mu) \leq \phi(x_k, z_k; \mu)$

    end if

    ▷ Update state and test for convergence

    $x_k \leftarrow x_k + \alpha\Delta x$, $z_k \leftarrow z_k + \alpha\Delta z$

    if Convergence condition is met **then**

    end ifreturn x_k,z_k

end forreturn NOT CONVERGED

where b denotes the vector of constraint violations of Problem (8). Also, in SIP for collision geometries, a serious problem is that using existing instantiated index parameters, a step may go too far into areas where the minimum of the distance function $g^*(q_t) \equiv \min_{y \in Y_t} g(q_t, y)$ greatly violates the inequality, and the optimization loses reliability. So we add the max-violation $g^{*-}(q_t)$ to b, in which we denote the negative component of a term as $\cdot^- \equiv \min(\cdot, 0)$.

Convergence criteria. We denote the index set $\tilde{Y}$ instantiated at the k^{th} outer iteration as Y^k, the corresponding MPCC as $P_k = P(Y^k)$, and the solved solution as $(x_k,\, z_k)$.

The convergence condition is defined as $\alpha \|[\Delta x, \Delta z]\| \leq \epsilon_x \cdot n_{xz}$ and $|z_k|^T |g(q_k, Y^k)| + |v(q_k, \dot{q}_k, Y^k)|^T |h(q_k, Y^k, z_k)| \leq \epsilon_{gap} \cdot n_{cc}$ and $|s(x_k, z_k, Y^k)| \leq \epsilon_s \cdot T$ and $\sum_t g_k^{-*}(x_{k,t}) < \epsilon_p \cdot T$, where n_{xz} is the dimension of the optimization variable and n_{cc} is the number of complementarity constraints, ϵ_x is the step size tolerance, ϵ_{gap} is the complementarity gap tolerance, ϵ_s is the balance tolerance, and ϵ_p is the penetration tolerance. With a little abuse of notation, $g(x_k, Y^k)$ is the concatenation of the function value of all the points in Y^k, and similar for $v(q_k, \dot{q}_k, Y^k)$ and $h(q_k, Y^k, z_k)$.

4.3.3 Maximum Violation Oracle and Time-Active Maximum Violation Oracle

Algorithm 4.2 Maximum-Violation Oracle

Input $q_{0:T}$, Y^{k-1}, adding threshold d^*_{min} and d^*_{max}
Output Y_k

$Y^k \leftarrow Y^{k-1}$
for $t = 0, \ldots, T$ **do**
  $y^* = arg\,min_{y \in Y_t} g(q_t, y)$
  $d^* = g(q_t, y^*)$
  if y^* is not in Y_t^k and $d^* < d^*_{max}$ **then**
    if $\|y - y^*\| > d^*_{min}\ \forall y \in Y_t^k$ **then**
      for $t = 0, \ldots, T$ **do**
        add $y*$ to Y_t^k
      end for
    end if
  end if
end for

The Maximum-Violation Oracle (Alg. 4.2) adds the closest or deepest penetrating points between the object and environment at each time step along the trajectory to all the Y_t^ks.

This method is more efficient than incrementally sampling index points in its domain. However, it may still include index points that do not generate active contact forces during the iteration. For instance, as illustrated in Fig. 4.2(a), the closest or deepest penetrating points at time step t are typically active only around that specific period. Thus, we introduce another oracle, the Time-Active Maximum-Violation Oracle (Alg. 4.3), which allows different contact points to be selected for different time steps along the trajectory.

(a) Closest point on the object to the environment at each time step

(b) Index points selected by MVO at each time step

(c) Index points selected by TAMVO without SD and TS at each time step

(d) Index points selected by TAMVO with TS ($n_s = 1$) at each time step

(e) Spatial Disturbance

Figure 4.2: This figure illustrates various Oracles for selecting index points, with the object transitioning from left to right and undergoing clockwise rotation. (a) depicts the closest point on the object relative to the environment at each time step. The Maximum Violation Oracle, shown in (b), imposes constraints on all identified closest points at every time step. In contrast, the Time-Active Maximum Violation Oracle without Spatial Disturbance and Spatial Disturbances, illustrated in (c), restricts its focus to imposing constraints solely on the closest point at the current time step. The Time Smoothing technique with $n_s = 1$, demonstrated in (d), extends constraint imposition to the closest points identified at adjacent time steps. (e) presents the Spatial Disturbance technique applied at a specific time step, with only disturbed rotation illustrated. [Best viewed in color.]

This TAMVO with $n_t = 0$ and $N_s = [\,]$ adds only the closest or most deeply penetrating points at the current time step t to Y_t^k. Consequently, the index set is no longer the same for all time steps, and each time step has its distinct index set.

Upon iteration k, the oracle selects index points based on the solution of q from the previous iteration, which is different from the optimal solution of $P(Y)$. Therefore, relying

Algorithm 4.3 Time-Active Maximum-Violation Oracle

Input $q_{0:T}$, Y^{k-1}, adding threshold d^*_{min} and d^*_{max}, time smoothing step n_t, spatial disturbances N_s
Output Y^k

$Y^k \leftarrow Y^{k-1}$
$Y' \leftarrow [[\,]_0, [\,]_1, \ldots, [\,]_T]$
for $t = 0, \ldots, T$ **do**
    $y^* = arg\,min_{y \in Y_t} g(q_t, y)$
    $d^* = g(q_t, y^*)$
    if $d^* < d^*_{max}$ **then**
        add y^* to $Y'[t]$
    end if
end for
for $t = 0, \ldots, T$ **do**
    for $n_s \in N_s$ **do**
        for $q_t[i] \in q_t$ **do**
            $q_t^+ \leftarrow q_t$ with n_s added to $q_t[i]$
            $y^{*+} = arg\,min_{y \in Y_t} g(q_t^+, y)$
            $d^{*+} = g(q_t^+, y^{*+})$
            $q_t^- \leftarrow q_t$ with n_s deleted from $q_t[i]$
            $y^{*-} = arg\,min_{y \in Y_t} g(q_t^-, y)$
            $d^{*-} = g(q_t^-, y^{*-})$
            for y', d' in zip($[y^{*+}, y^{*-}], [d^{*+}, d^{*-}]$) **do**
                if y' is not in $Y'[t]$ and $d' < d^*_{max}$ **then**
                    if $\|y' - y\| > d^*_{min}\ \forall y \in Y_t^k$ **then**
                        add y' to $Y'[t]$
                    end if
                end if
            end for
        end for
    end for
end for
for $t = 0, \ldots, T$ **do**
    for $t' = t - n_t, \ldots, t + n_t$ **do**
        if $0 \leq t' \leq T$ **then**
            for y' in $Y'[t]$ **do**
                if $\|y' - y\| > d^*_{min}\ \forall y \in Y_t^k$ **then**
                    add y' to Y_t^k
                end if
            end for
        end if
    end for
end for

entirely on the input trajectory may not be the most effective approach of selecting index points. To address this, we introduce the following two strategies designed to mitigate this issue:

Spatial Disturbance (SD). Recognizing that the current solution for q differs from the optimal solution yet is likely to be in its vicinity, we propose introducing perturbations to the current solution to discover possible active contact points. These perturbations may aid in identifying index points crucial for establishing the optimal solution. Consequently, in line 11-16 of Alg. 4.3, q_t is perturbed with disturbance n_s along each dimension of q_t separately. An illustration of adding perturbation to rotation in 2D is shown in Fig. 4.2(e). In a 3D scenario, with a single n_s, this approach would result in the detection of 12 additional y^*, accounting for both increases and decreases in x, y, z and $roll$, $pitch$, yaw.

Time Smoothing (TS). Considering that the closest or most deeply penetrating points at time step t may be active not just at t, but also during a surrounding interval, in line 17 of Alg. 4.3, the closest points detected within the adjacent time steps from $t - n_t$ to $t + n_t$, governed by a parameter n_t, are added to the index set of time step t. The effect of using TS is illustrated in Fig. 4.2(c).

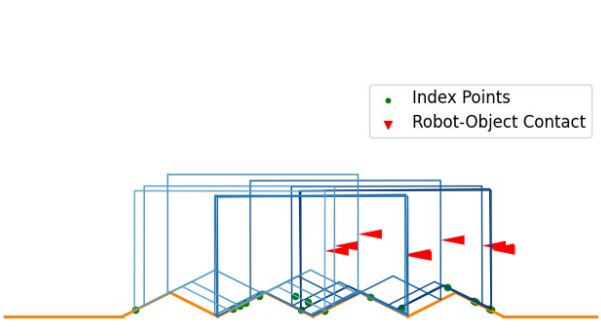
4.4 EXPERIMENTS AND DISCUSSION

The proposed methods are implemented in Python using the optimization interface and the SNOPT solver [46] provided by Drake [68].

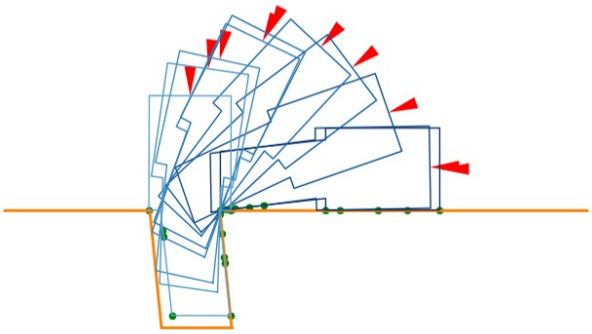
4.4.1 Experiments in 2D

First, we evaluate STOCS in a 2D setting with TAMVO, focusing on testing its applicability with complex-shaped objects and environments. In this experiment, we set $n_t = 1$ and $N_s = [1e^{-2}]$ as the parameters for TAMVO, and we designate these as the default values for TS and SD separately. Four tasks are designed for this evaluation: pushing a dented object across uneven terrain, inserting a tilted peg into a correspondingly angled hole, and maneuvering a bean-shaped object across two distinct curvilinear terrains. The trajectories planned for these tasks are depicted in Fig. 4.3, and the detailed information regarding the objects' geometries and the solve of the trajectories are presented in Table 4.1.

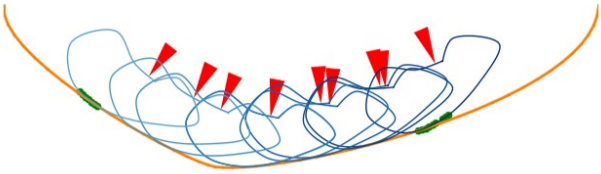
$T = 10$, $\Delta T = 0.1s$, $\mu_{mnp} = 1$ and $\mu_{env} = 0.5$ are used for all the experiments in 2D.



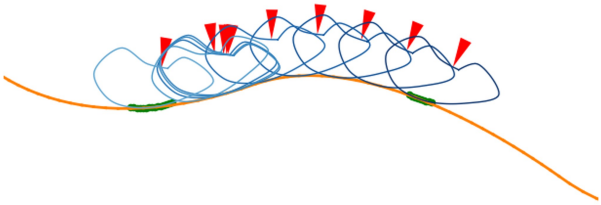
(a) Push dented object on uneven terrain



(b) Push tilted peg into angled hole



(c) Slide bean on curve 1



(d) Slide bean on curve 2

Figure 4.3: This figure illustrates trajectories planned by STOCS on 2D examples involving complex shaped manipulands and environments. The gradient coloring of the object, transitioning from dark to light, signifies the progression from the start to the end of the trajectory. (a) demonstrates the pushing of a dented object on uneven terrain. (b) shows the pushing of a tilted peg into an angled hole. (c) and (d) presents the sliding of a bean-shaped object on two curvilinear terrains. For clarity, only the selected contact points on the manipuland at the first and the last time steps along the trajectories are depicted, and the manipulator's contact with the manipuland is only depicted at the first time step. [Best viewed in color.]

Table 4.1: Numerical optimization results of STOCS in 2D. Number of points in the object's representation (# Point), solve time (Time), outer loop iteration number (Outer iters), and average active index points for each iteration (Index points) are reported in the table.

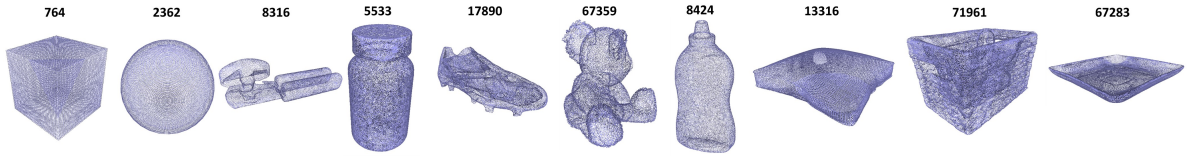
Environment	Object	# Point	Outer iters.	Index points	Time (s)
Uneven	Dented	543	4	9.05	30.83
Tilted Hole	Tilted Peg	214	4	12.93	40.11
Curve 1	Bean	100	7	9.43	72.74
Curve 2			14	14.29	636.89

STOCS selects only a small amount of points from the total number of points in the objects' representation on average, which greatly decreases the dimension of the instantiated optimization problem and makes the solve tractable.

4.4.2 Experiments in 3D

Next, we evaluate STOCS in 3D by performing numerical experiments with object geometries from the YCB dataset [69], the Google Scanned Objects [70], and 3D models found online [71, 72]. All the objects used in the study are shown in Fig. 4.4, and all the environments used in the study are shown in Fig. 4.5. Klampt [53] and the code in [15] are used to find the closest points between two complex-shaped geometries.

(a) Objects



(b) Point Clouds

Figure 4.4: All the objects (first row) alongside their respective point clouds with the number of points in the points cloud (second row) used in the experiments. The relative sizes of the objects depicted in the image do not accurately represent their actual proportions. [Best viewed in color.]

To demonstrate the effectiveness of STOCS in planning with high-fidelity geometric representations, we conduct experiments on ten different objects represented by dense point clouds sampled on the surface of the objects' meshes, and five different environments represented by Signed Distance Field (SDF). The SDFs are calculated offline on a grid that encloses the corresponding environment given a polygonal mesh of the environment, and values off of the grid vertices are approximated via trilinear interpolation.

Using STOCS, we plan for pushing, pivoting, rolling and rotating trajectories on these objects. The resulting planned trajectories are illustrated in Fig. 4.6, while detailed information regarding the objects' geometries and the solve of the trajectories are presented in Table 4.2. Same as the experiment in 2D, we set $n_t = 1$ and $N_s = [1e^{-2}]$ as the default parameters for TAMVO.

Following the initial assessments, we further evaluated the efficacy of the TAMVO alongside the SD and TS techniques through a set of comparative experiments. These experiments utilized STOCS to plan trajectories for the same tasks mentioned earlier, with the primary variation being the specific oracle employed in each scenario.

Figure 4.7 presents the success rates of all tested Oracles. The data shows that the

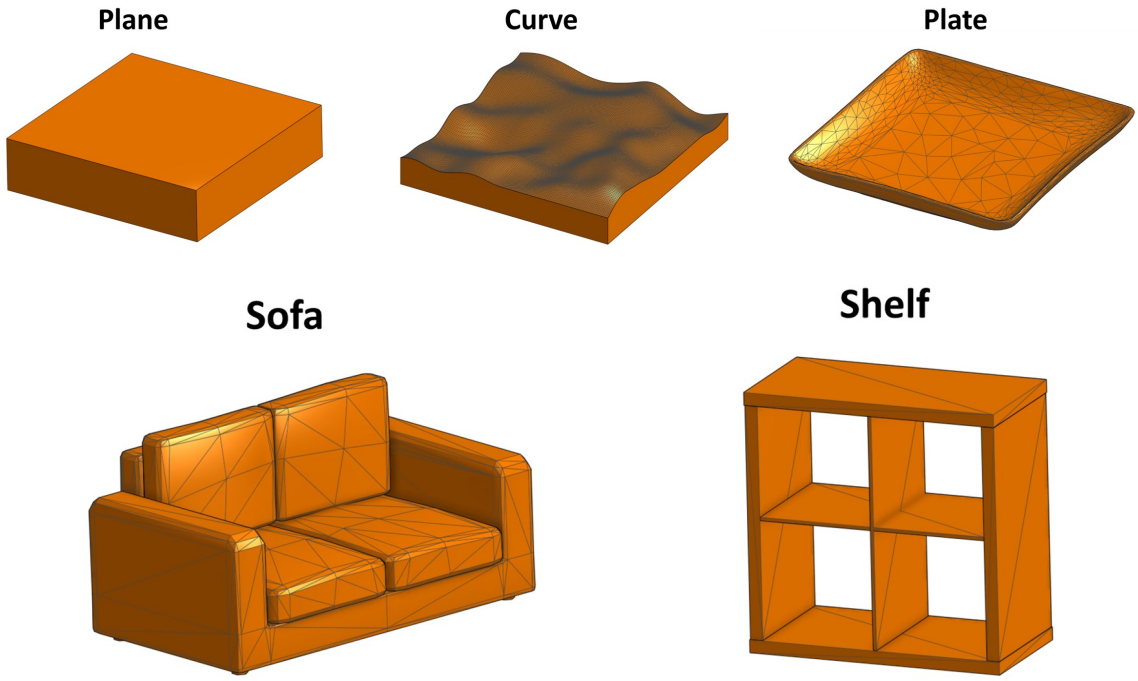
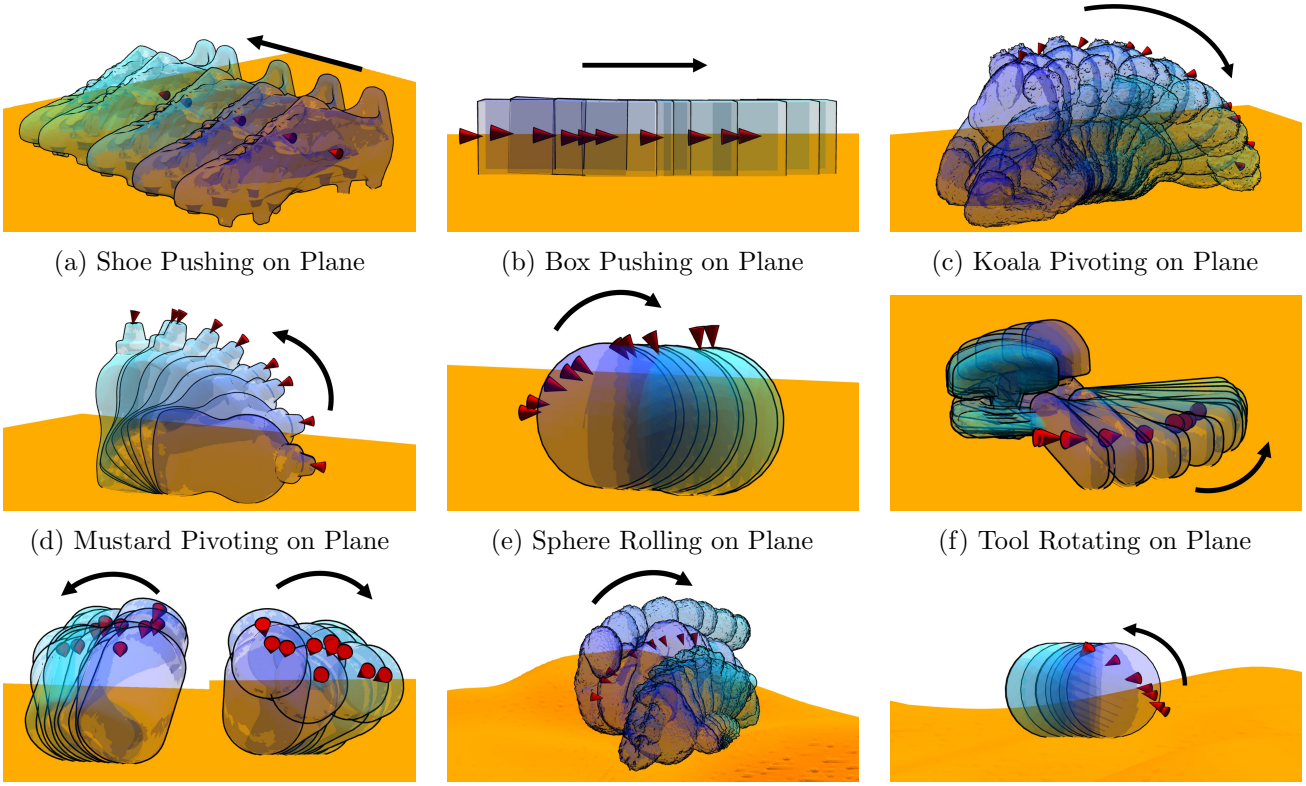


Figure 4.5: All the environments used in the experiments. The relative sizes of the objects depicted in the image do not accurately represent their actual proportions. [Best viewed in color.]

MVO achieves a 67% success rate. In contrast, TAMVO without SD and TS exhibits worse performance than MVO; this is particularly evident in 3D scenarios where relying solely on the nearest object-to-environment point is inadequate for fulfilling the object's balance constraints. The SD and TS techniques, introduced to address this challenge, both demonstrated enhanced performance when combined with TAMVO, surpassing the success rate of TAMVO alone. Furthermore, the integration of SD and TS with TAMVO consistently achieved successful trajectory planning for all tasks.

Figure 4.8 displays the average number of index points selected at each time step by the different Oracles assessed in our study, focusing on tasks that were successfully solved by all Oracles. As depicted in Fig. 4.8(a), the MVO selects a larger number of points than TAMVO and all its variations. Notably, TAMVO combined with SD and TS can successfully plan trajectories for all tasks while selecting fewer index points compared to MVO. These findings validate our hypothesis regarding TAMVO: an index point identified at time step t is most valuable within a temporal vicinity of t. Furthermore, these results substantiate our rationale for introducing SD and TS, affirming that a localized exploration in both temporal and spatial dimensions offers a more efficient strategy than incorporating index points identified at distant time steps along the trajectory.



(a) Shoe Pushing on Plane (b) Box Pushing on Plane (c) Koala Pivoting on Plane

(d) Mustard Pivoting on Plane (e) Sphere Rolling on Plane (f) Tool Rotating on Plane

(g) Drug Bottle Rolling on Plane (h) Koala Pivoting on Curved Surface (i) Sphere Rolling on Curved Surface

(j) Pillow Pivoting on Sofa (k) Basket Pushing on Shelf (l) Plate Sliding on Plate

Figure 4.6: This figure demonstrates trajectories planned by STOCS. The gradient coloring of the object, transitioning from dark to light, signifies the progression from the start to the end of the trajectory. Additionally, the arrow also indicates the object's movement direction. The red cone marks the contact location of the manipulator on the object. Surrounding this contact point, 3 to 5 points on the object's surface are sampled to approximate a patch contact. [Best viewed in color.]

Table 4.2: Numerical optimization results of STOCS in 3D. Number of points in the object's representation (# Point), solve time (Time), outer loop iteration number (Outer iters), and average active index points for each iteration (Index points) are reported in the table.

Environment	Object	Task	# Point	Outer iters.	Index points	Time (s)
Plane	Box	Push	764	3	5.75	22.91
	Shoe	Push	17890	5	4.20	92.41
	Koala	Pivot	67359	3	4.91	37.93
	Mustard	Pivot	8424	3	9.15	95.44
	Sphere	Roll	2362	5	3.78	61.54
	Tool	Rotate	8316	6	4.95	135.70
	Drug	Roll	5533	13	5.67	361.72
Curve	Koala	Pivot	67359	9	11.72	277.36
	Sphere	Roll	2362	4	7.16	89.73
Sofa	Pillow	Pivot	13316	7	10.19	424.5
Shelf	Basket	Push	71961	3	13.11	42.86
Plate	Plate	Slide	67283	3	26.61	116.6

Figure 4.8(b) presents the solve times for STOCS employing various Oracles across all successful tasks. The results indicate that the quantity of index points selected by an Oracle does not necessarily correlate with the solve time. For instance, in the sphere rolling on plane task, TAMVO+TS chooses a larger number of index points than TAMVO+SD, yet the solve time for TAMVO+TS is much faster than that of TAMVO+SD. Similarly, in the case of the drug bottle rolling on plane task, it can be seen that both TAMVO+SD+TS and TAMVO+TS select a comparable number of index points, yet the solve time for TAMVO+TS is much faster than TAMVO+SD+TS. This phenomenon underscores that a larger number of index points does not invariably lead to longer solve time. Although TAMVO+SD+TS provides the best performance in terms of success rate, it is not guaranteed to give the best solve time for all different tasks.

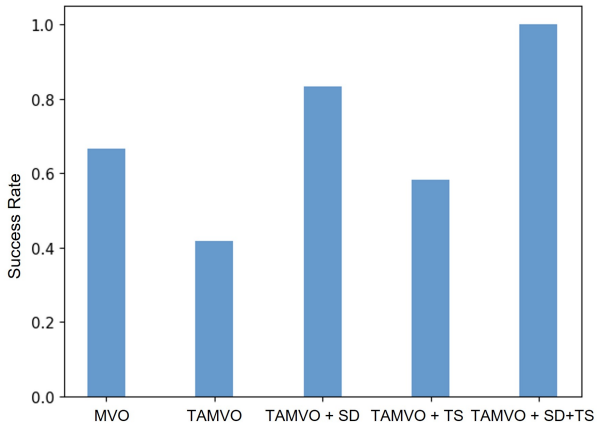
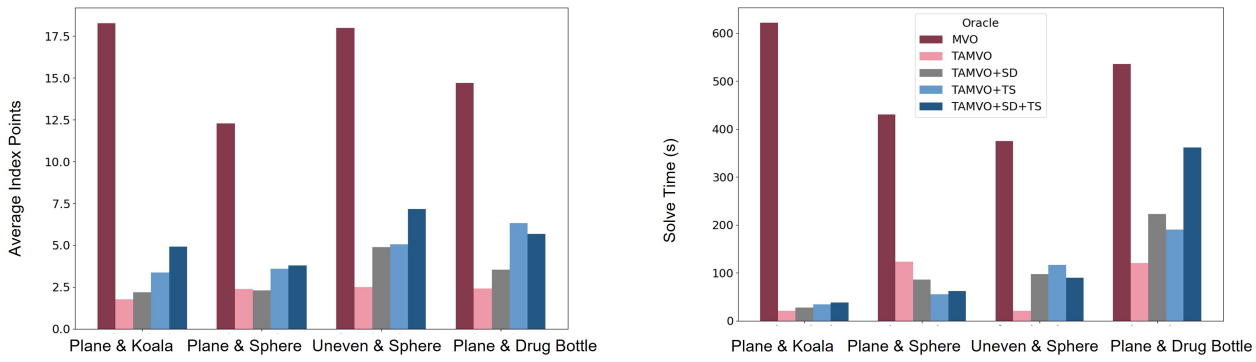


Figure 4.7: Success and failure rates of all the Oracles on all the tasks. [Best viewed in color.]



(a) Average Index Points

(b) Solve Time

Figure 4.8: The average index points selected at each time step by the different Oracles (a) and the solve time (b) for tasks that were successfully solved by all Oracles. [Best viewed in color.]

CHAPTER 5: MULTI-MODAL MANIPULATION PLANNING USING STOCS

In Chapter 4, we introduced Simultaneous Trajectory Optimization and Contact Selection (STOCS), which is contact-implicit for object-environment contact, but it still requires a pre-selected robot-object contact state. In this chapter, we introduce a sampling-based planner, Multi-Modal Manipulation Planner (MMMP) [56], that uses STOCS as a local optimizer to incrementally construct a manipulation tree, which helps it solve for longer-horizon trajectories with changes of manipulation mode between pushing, pivoting, and grasping. MMMP's capability of generating plans involving changes of manipulator contact state is validated on several manipulation tasks in simulation and on a real robot. [1]

5.1 INTRODUCTION

Generating multi-modal behaviors through optimization-based motion planners presents challenges due to the inherently discontinuous dynamics caused by contacts. An illustrative instance is depicted in Figure 5.1, where the robot is tasked with assembling a gearbox. Since the blue gear is wider than the gripper's opening, the robot needs to manipulate the gear into an upright pose before it can grasp it. A possible solution is to pivot the gear with a point contact against the external surface before grasping it. However, existing planning and optimization techniques struggle to identify such solutions.

In Chapter 4, we introduced the simultaneous trajectory optimization and contact selection (STOCS) algorithm to address the scaling problem in contact-implicit trajectory optimization. It applies an infinite programming (IP) approach to dynamically instantiate possible contact points between the object and environment inside the optimization loop, and hence the resulting MPCCs become far more tractable to solve. Experimental results demonstrate that it can solve for manipulation trajectories involving changes of contact between object and environment.

Although STOCS is contact-implicit for object-environment contact, it still requires pre-selected robot-object contact state (i.e., a manipulation mode). It is also still a local trajectory optimization method, so its output is sensitive to the specified temporal resolution and initial trajectory. We address these limitations in this study by introducing a sampling-based planner that uses STOCS to incrementally construct a manipulation tree, which helps

[1] This chapter is adapted from Mengchao Zhang, Devesh K. Jha, Arvind Raghunathan, and Kris Hauser, "Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning." In Robotics: Science and Systems (RSS), 2023.

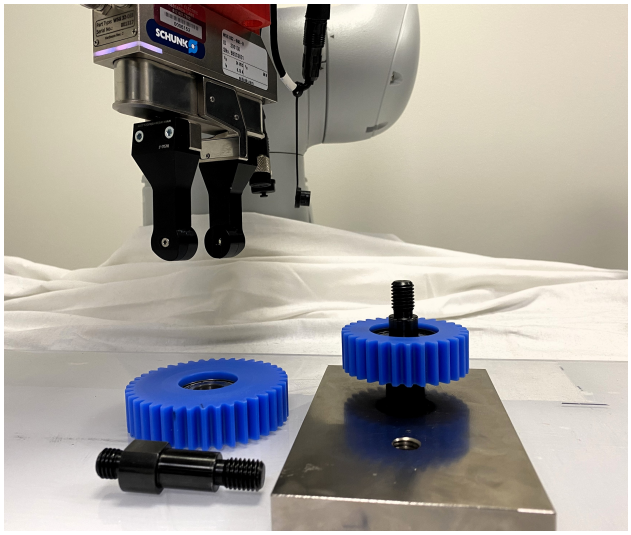


Figure 5.1: A robot equipped with a two-fingered gripper may need to re-orient and grasp parts to assemble a gear box. This problem is difficult as the algorithm has to reason about performing a non-prehensile grasp, rotating, and/or sliding before the pick and place action. [Best viewed in color.]

it solve for longer-horizon trajectories that change manipulation mode between pushing, pivoting, and grasping. Each extension of the tree uses STOCS as a local planner to reach as far as possible toward randomly sampled subgoals. The workflow of the proposed multi-modal manipulation planner is illustrated in Fig. 5.2.

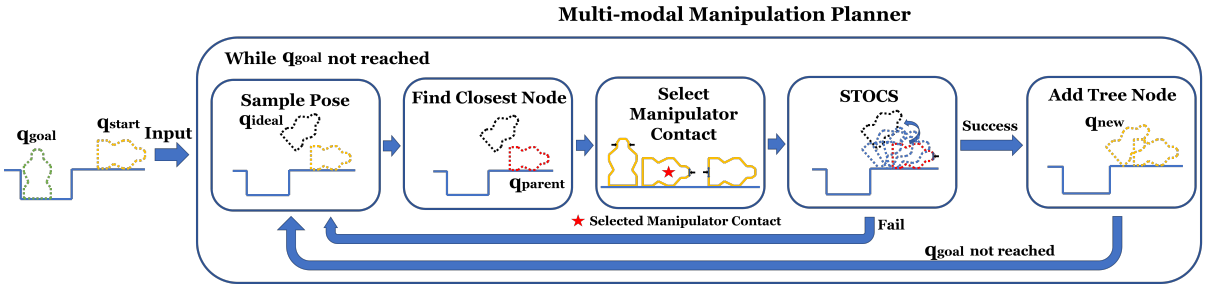


Figure 5.2: Illustrating the workflow of the proposed multi-modal manipulation planner, which could plan for manipulation behaviors consisting a sequence of manipulation modes. Given the start and goal pose of an object as input, the planner incrementally constructs a manipulation tree through using STOCS to steer the object to reach as far as possible toward randomly-sampled subgoals until the goal pose is reached. [Best viewed in color.]

5.2 RELATED WORK

Task and Motion Planning (TAMP) solves long-horizon planning problems [73] by integrating a search over plan skeletons and the satisfaction of constraints over hybrid action parameters. The complexity of planning through contacts is reduced by introducing predefined motion primitives. In [74], TAMP shows the potential to solve sequential manipulation and tool-use planning problems. However, all objects are assumed to have a sphere-swept convex geometry to make trajectory optimization fast and differentiable, but this assumption is a severe limitation on applicable objects. Also, most of the TAMP methods require expert knowledge and engineering efforts to predefine states and manipulation primitives. On the contrary, our proposed method applies to more general object and environment geometries, and can also discover different motion primitives involving pivoting and sliding within the STOCS optimizer.

5.3 METHOD

Our objective is to devise plans for contact-rich manipulations that may include a change of manipulator contact state for arbitrary-shaped object and environment geometries, e.g., reorientation and translation of large parts for assembly, peg-in-hole, and object packing. Three example tasks with corresponding initial and goal poses are shown in Fig. 5.3.

Our approach uses a multi-modal sampling-based planner that constructs a manipulation tree to guide the planning toward the goal using STOCS to perform each extension of the tree.

5.3.1 Problem Description

Our method requires the following information as inputs:

1. Object initial pose: $q_{init} \in SE(2)$.
2. Object goal pose: $q_{goal} \in SE(2)$.
3. Object properties: a rigid body $\mathcal{O}$ whose geometry, mass distribution, and friction coefficients with both the environment μ_{env} and the manipulator μ_{mnp} are known.
4. Environment properties: rigid environment $\mathcal{E}$ whose geometry is known.
5. Manipulation primitives: all the allowable contact states between the manipulator and the object.

Our method will output a trajectory τ which includes the following information at time t:

1. Object configuration: q_t.
2. Manipulation primitive and manipulator's configuration: c_t^{mnp}, and q_t^{mnp}.
3. Manipulation force: u_t.
4. Object-environment contact state y_t.
5. Object-environment force z_t.

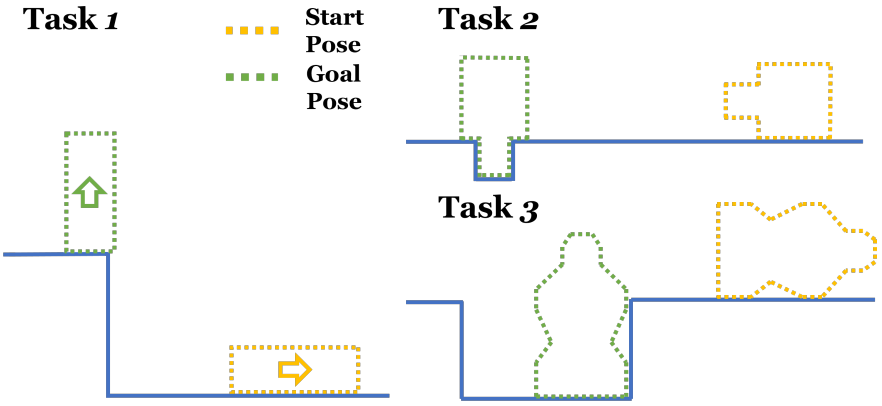


Figure 5.3: The start and goal pose of the multi-modal manipulation tasks. In Task 1, the arrow illustrates the orientation of the object. All the objects are not graspable at the start pose. [Best viewed in color.]

We make the following assumptions:

1. All objects and environments are rigid.
2. A set of manipulation primitives are pre-specified as a set of manipulator contact points/surface, specified in object-relative coordinates.
3. Change of manipulator contact state (i.e., regrasping) is only allowed when the object can stably be supported by the environment alone.
4. Quasi-Static: the motion of the manipulated object is slow enough that the forces acting on it are in equilibrium at each instant of time along the trajectory.

5.3.2 Multi-Modal Manipulation Planner

The multi-modal manipulation planner uses sampling to enable robot-object contact state switches, while STOCS is used to optimize changes of contact between the object and envi-

ronment. Alg. 5.1 presents the proposed planner, which combines STOCS with a T-RRT [75] approach to guide tree expansion toward the goal.

Algorithm 5.1 Multi-modal Manipulation Planner

Input q_{init}, q_{goal}
Output tree $\mathcal{T}$
$\mathcal{T} \leftarrow$ initialize tree(q_{init})
while $q_{goal} \notin \mathcal{T}$ **do**
  $q_{ideal} \leftarrow$ sample random configuration($\mathcal{C}$)
  $q_{parent} \leftarrow$ find nearest neighbor($\mathcal{T}$, q_{ideal})
  if transition test(q_{parent}, q_{ideal}, q_{goal}) **then**
    stable $\leftarrow$ stability test(q_{parent})
    if *stable* **then**
      $c^{mnp} \leftarrow$ sample mnp contact state(q_{parent})
    else
      $c^{mnp} \leftarrow$ parent mnp contact state(q_{parent})
    end if
    $q_{new} \leftarrow$ STOCS(q_{parent}, q_{ideal}, c_{mnp})
    if $q_{new} \neq$ null **then**
      add node q_{new} to $\mathcal{T}$
      add edge $q_{parent} \rightarrow q_{new}$ to $\mathcal{T}$
    end if
  end if
end while

The **sample random configuration** function has a probability p_1 of returning a random sample q_{ideal} from the configuration space $\mathcal{C}$, a probability p_2 of returning a random sample whose rotation angle is sampled from the angles of all the possible stable poses of the object on a plane, and a probability $1-p_1-p_2$ of returning the goal configuration q_{goal}. Without p_2, the probability of a stable pose to be sampled is 0, and the switch of manipulation contact state will never be triggered. The **find nearest neighbor** function returns the nearest neighbor of q_{ideal} in the tree $\mathcal{T}$ using the weighted $SE(2)$ metric defined as:

$$dist(q_1, q_2) = \sqrt{w_1 \cdot (d_x^2 + d_y^2) + w_2 \cdot d_\theta^2} \tag{5.1}$$

where $d_x = q_1^{(x)} - q_2^{(x)}$, $d_y = q_1^{(y)} - q_2^{(y)}$, $d_\theta = min(|q_1^{(\theta)} - q_2^{(\theta)}|, 2\pi - |q_1^{(\theta)} - q_2^{(\theta)}|)$, and w_1 and w_2 are the weights that balance the importance between translation and rotation.

The **transition test** function follows T-RRT [75] to decide whether to accept to propagate the tree from q_{parent} towards the newly sampled configuration q_{ideal} or not. This loosely guides the propagation of the tree toward the goal while still allowing the tree to steer away from the goal with lower probability. We define the cost C_q of a configuration q as

$dist(q, q_{goal})$, and the transition test is done by comparing the cost of q_{parent} and q_{ideal}. Define $\Delta C = \frac{C_{q_{ideal}} - C_{q_{parent}}}{dist(q_{parent}, q_{ideal})}$ as the normalized change in cost. The new sample will be discarded if C_{ideal} exceeds a maximum bound C_{max}, and the new sample will be accepted if $\Delta C \leq 0$. But if $\Delta C > 0$, then q_{ideal} is accepted with probability

$$p(q_{parent}, q_{ideal}) = exp\left(\frac{\Delta C_{(q_{parent}, q_{ideal})}}{KT}\right), \tag{5.2}$$

where K is a normalization factor defined as the average of $C_{q_{ideal}}$ and $C_{q_{parent}}$, and T is the temperature parameter that is used to control the difficulty level of transition tests. T is adaptively tuned during sampling as in T-RRT.

The **stability test** function checks if the object $\mathcal{O}$ is stable at the input configuration in the environment $\mathcal{E}$ without any manipulation force. This is accomplished by solving an IPCC problem. If q_{parent} is stable, then the **sample mnp contact state** function will randomly select an allowable manipulator contact state c^{mnp} at this configuration. If q_{parent} is unstable, then the manipulation contact state will be inherited from q_{parent}.

The permissible manipulation modes may be defined in a problem-dependent manner to reflect the manipulation primitives available to the robot. The modes used in our experiments are illustrated in Fig. 5.4. Two-point contact is only possible when the object is at certain upright rotation angles, and one-point contact is permissible at surfaces not in contact with the environment. For each extension of the tree, STOCS is configured with an objective function $\tilde{f}(q, \dot{q}, u, z) = W \sum_t dist(q_t, q_{goal})^2$ to guide the object toward q_{goal} as close as possible. When a stable angle of the object is sampled by **sample random configuration**, we set $w_1 = 0$ and $w_2 = 1$ to disregard the translation components in the objective function, and STOCS will try to steer the object to the target angle. Otherwise, we use $w_1 = w_2 = 1$. $W = 5$ is used in the experiments.

5.4 EXPERIMENTS AND DISCUSSION

We evaluate the proposed multi-modal manipulation planner by performing several numerical experiments and some physical experiments. The STOCS method used in this study is implemented in Python using the PYROBOCOP framework [76], which uses the IPOPT solver for optimizations [77]. All experiments were run on a single core of a 3.6 GHz AMD Ryzen 7 processor with 64 GB RAM.

We test the proposed multi-modal manipulation planner on tasks requiring one or more changes of manipulation mode. Parameters used in the experiments include:

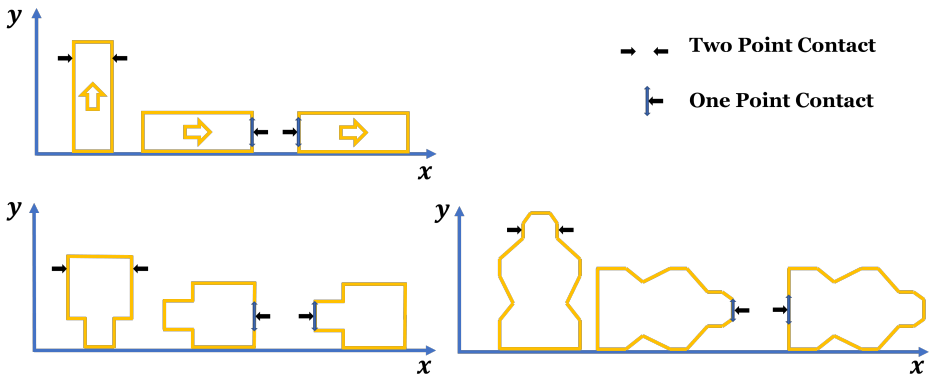


Figure 5.4: Allowable manipulator contact states for objects used in the experiments. One-point contact allows the manipulator to slide on a designated surface of the object, and two-point contact is a fixed contact location relative to the object's frame. [Best viewed in color.]

- **Physical parameters**: Object mass $m = 0.1$ kg, environment friction coefficient $\mu_{env} = 1.0$, manipulator friction coefficient $\mu_{mnp} = 1.0$.
- **STOCS parameters**: $N^{max} = 10$, $\epsilon_x = \epsilon_{gap} = \epsilon_s = \epsilon_p = 1e^{-4}$, $S = min(30 + 10 * k, 200)$, $T = 5$ and $\Delta t = 0.1$.
- **Multi-modal planner parameters**: $C_{max} = 2$. Runs are terminated after a maximum of 500 extensions.

All experiments in this section are evaluated under 10 different random seeds.

Results on the 3 tasks of Fig. 5.3 are illustrated in Fig. 5.5. The solution trajectories demonstrate that the planner discovers the changes of manipulation mode from one point contact to two point contact or vice versa, and can switch from pivoting to grasping to sliding. The sampled trees are also plotted, illustrating that very few nodes are actually sampled and the planner makes quite direct progress toward the goal.

Timing and success rates on the same tasks as well as their *reversed* versions, in which the start and goal pose are interchanged, are shown in Table 5.1 . We see that STOCS is only called a few dozen times at most, and the transition test is effective at rejecting ineffective pose samples. We also explored how the planner performs as the total number of time steps T in STOCS is varied. As can be seen in Table 5.2, the success rate slightly increases as T increases and the number of nodes in the tree and solution path decreases. This can be explained by the longer time horizon enabling STOCS to make larger steps in the state space, giving a better chance to connect to the goal pose within the sample limit. However, a larger T increases solve times overall, since each call of the local planner solves a larger optimization problem.

Table 5.1: Success rate, number of nodes in the tree and the path, the planning time, and the number of STOCS called of the proposed planner on 3 tasks with different initial and goal pose. Forward direction has the same start and goal pose as shown in Fig. 5.3, and reverse direction has the start and goal pose interchanged.

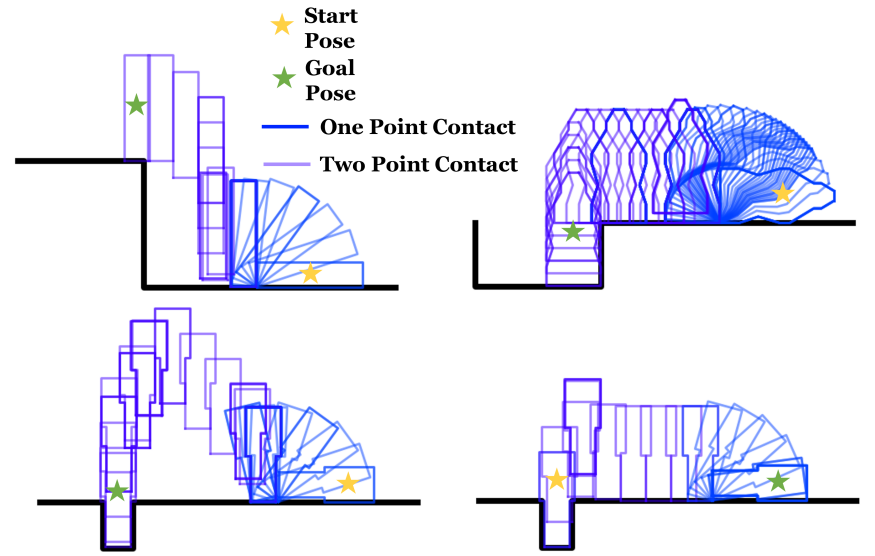
Direction		Forward			Reverse		
Task		1	2	3	1	2	3
Success		9/10	9/10	8/10	10/10	10/10	10/10
Nodes	in tree	26	21	18	8	14	10
(median)	in path	13	14	13	7	11	6
Time (s)	min	334	633	401	99	327	316
	median	1300	1310	1048	425	3360	2014
	max	3712	4472	1837	3747	5882	7712
STOCS calls (median)		25	24	27	7	15	10

Table 5.2: Success rate, number of nodes in the tree and the path, and the planning time of the proposed planner on 3 tasks with different total number of time steps T for STOCS.

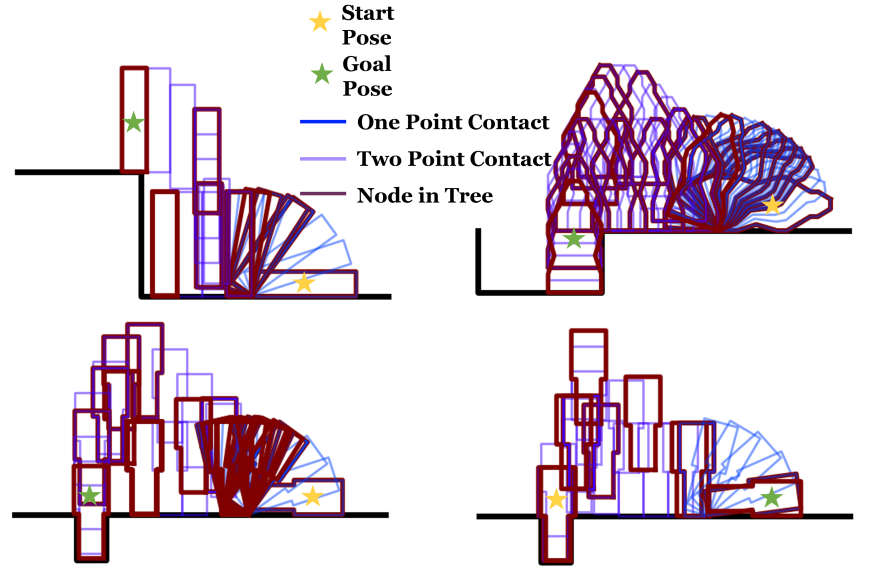
T		3			5			10		
Task		1	2	3	1	2	3	1	2	3
Success		9/10	8/10	8/10	9/10	9/10	8/10	8/10	10/10	10/10
Nodes	in tree	39	28	18	26	21	18	18	12	6
(median)	in path	14	20	13	13	14	13	12	9	5
Time (s)	min	422	410	221	334	633	401	595	926	606
	median	915	906	385	1300	1310	1048	2072	1726	1648
	max	1884	1752	1275	3712	4472	1837	6994	2959	5590

Hardware experiments are performed to evaluate the planned trajectories. A Mitsubishi Electric Assista industrial position-controlled arm with a F/T sensor mounted at the wrist of the robot is used in the experiment. The default stiffness controller of the robot is used to execute the planned force trajectories. To implement the optimal force trajectory on the object, we design a reference trajectory for the robot that presses into the object such that the robot would apply the desired force for the estimated stiffness constants for the low-level robot position control.

Since the computed trajectory is executed without object pose feedback, execution error can accumulate. Thus, we use AprilTags [78] (Fig. 5.6) to track the pose of the object, and after a single-mode manipulation trajectory is completed, the object pose feedback is used to adjust the execution of the next mode's trajectory. Some trajectories recorded during the robot experiments are shown in Fig. 5.6.

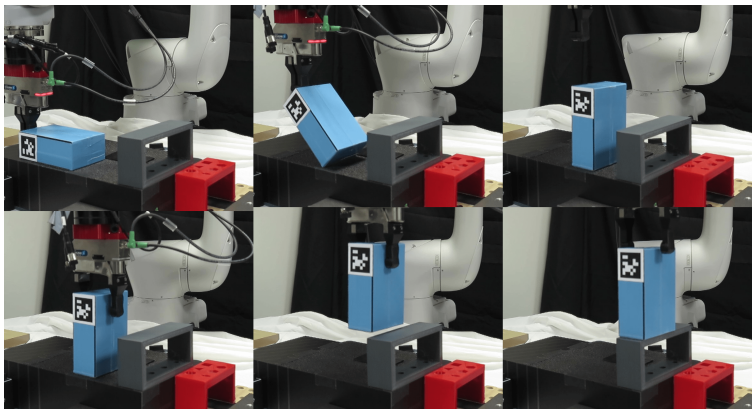


(a) Trajectories

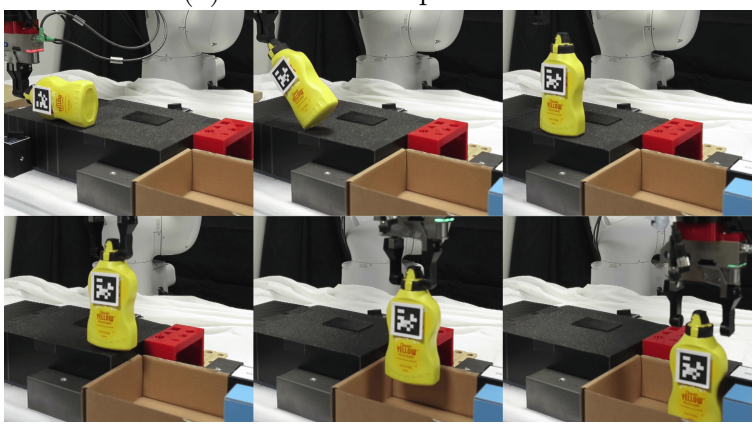


(b) Trees

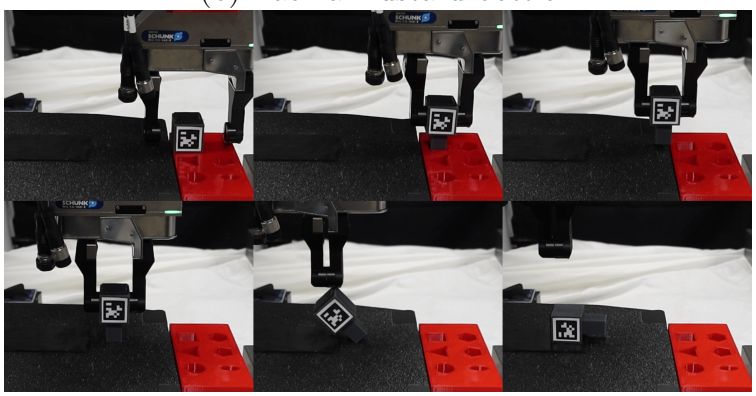
Figure 5.5: Plans generated by the multi-modal manipulation planner for four tasks. (a) Waypoint object poses along solution trajectories, with colors representing different manipulation modes. (b) Trees explored by the planner corresponding to the above trajectories. Nodes are highlighted in bold red and waypoints along edges are colored in the same manner as above. [Best viewed in color.]



(a) Reorient and place a box



(b) Pack a mustard bottle



(c) Unplug and lay down a peg

Figure 5.6: Snapshots along the trajectories executed on the real robot. (a) Reorient and place a box. (b) Pack a mustard bottle. (c) Unplug a peg and lay it down on a plane. AprilTag pose feedback is used to adjust trajectories only when the manipulator contact state changes. [Best viewed in color.]

CHAPTER 6: PLAN-GUIDED REINFORCEMENT LEARNING FOR WHOLE-BODY MANIPULATION

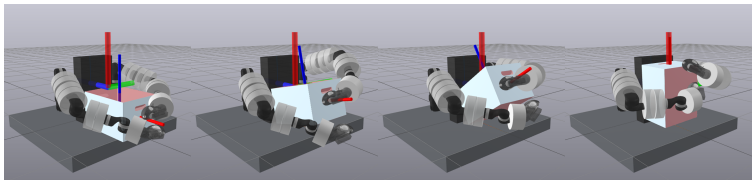
In Chapter 5, we introduced the Multi-Modal Manipulation Planning (MMMP) approach, capable of planning contact-rich manipulation trajectories that encompass changes in contact between the object and environment, as well as between the robot and object. In the experimental section, we successfully executed these planned trajectories using an open-loop approach. However, successful execution in open-loop mode requires precise alignment of planning parameters with their real-world analogs, including matching physical parameters and initial conditions for both the robot and the manipuland. To enhance the robust execution of planned trajectories, this study presents a novel framework, Plan-Guided Reinforcement Learning (PGRL) [79]. This framework combines the strengths of model-based planning and reinforcement learning (RL), where the planned trajectory from a model-based planner acts as a demonstration, enabling the RL agent to mimic the desired motion style in the motion plan. Additionally, the integration of domain randomization within the framework ensures the policy's robustness against uncertainties in model parameters and the initial pose of manipulands. The efficacy of this approach is validated on a whole-body large object manipulation task using Toyota Research Institute's Punyo robot [80]. [1]

6.1 INTRODUCTION

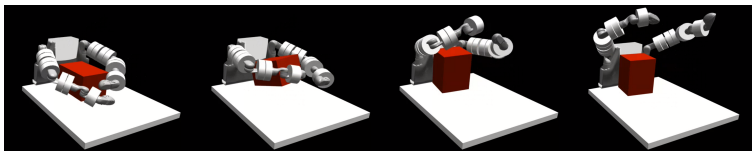
This study draws inspiration from Guided Reinforcement Learning (Guided RL), a method that leverages existing knowledge to enhance the efficiency and effectiveness of the reinforcement learning process [81]. In particular, example (or demonstration)-guided RL that aims to combine motion imitation with RL has been a popular concept in robotics and in animation to aid exploration by instilling a desired motion style, accelerate learning, and/or ease reward shaping [82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92].

Peng et al. [93] proposed an example-guided RL framework based on adversarial imitation learning, named Adversarial Motion Priors (AMP). This approach does not require designing imitation objectives or motion selection mechanisms, and it can automatically synthesize a policy that completes a desired high-level task given a set of unstructured example motions. Due to its promise to impose motion characteristics on the RL policy without the burden of reward engineering, the AMP idea, which was originally proposed for animation, has

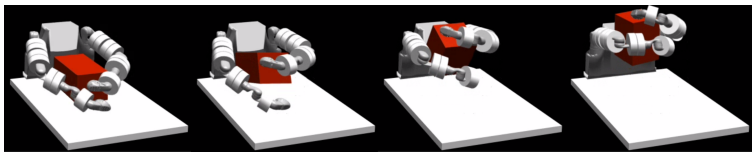
[1] This chapter is reproduced from Mengchao Zhang, Jose Barreiros, and Aykut Özgün Önol. "Plan-Guided Reinforcement Learning for Whole-Body Manipulation", in the Workshop on Leveraging Models for Contact-Rich Manipulation at IROS 2023.



(a) GQDP plan



(b) RL policy in simulation



(c) PGRL policy in simulation

(d) PGRL policy on hardware

Figure 6.1: GQDP plan, and snapshots from simulation and hardware rollouts. This highlights the style difference between the plan and the RL and PGRL policies as well as the effective sim-to-real transfer for the latter.

caught the attention of the robotics community and led to impressive results for quadruped locomotion, e.g., [94, 95, 96, 97, 98]. These are compelling examples showing that the AMP approach can generate policies: (i) with a desired style even from infeasible, scarce demonstrations; and (ii) that can directly transfer to the real world.

For trajectory generation, we adopt the Global Quasi-Dynamic Planner (GQDP) [99] in this study. The rationale behind this choice stems from GQDP's adeptness in synthesizing long-horizon behaviors with complex intermittent contact interactions. In contrast to our infinite programming approach, GQDP compromises geometric fidelity in favor of achieving longer-horizon planning. However, the resultant plans exhibit fragility, particularly when executed via open-loop control, even in the presence of the same model used during planning. This susceptibility aligns with our objective to test the AMP's ability to convert rough, and potentially infeasible plans into feedback policies that can be deployed on the hardware.

The proposed framework first uses the GQDP to synthesize quasi-dynamic plans, which

are not necessarily feasible, given a desired pose for the manipuland. Then, the AMP is used to complete the gaps in the plan and derive a closed-loop policy with similar motion characteristics to the plan. We test this approach to address a complex whole-body manipulation task employing Toyota Research Institute's Punyo robot [80]. Our preliminary findings show that this approach can enable generating policies efficiently even from a single, infeasible example plan. Facilitated by domain randomization and the inherent robustness stemming from Punyo's passive compliance, the trained policies can be seamlessly transferred to real hardware without necessitating subsequent processing.

While we present an instance of this approach using GQDP, we emphasize that this approach is not limited to a certain source for the demonstration and could be used with other types of planners.

6.2 RELATED WORK

Model-based trajectory optimization methods [3, 100], while effective in simulated environments, face challenges in real-world manipulation because contacts lead to stiff, non-smooth numerics with an excessive number of discrete contact modes. Although recent advancements in planning with contacts have exhibited promising outcomes in manipulation planning [99, 101, 102, 103, 104, 105], the robust execution of the planned trajectory in hardware remains an unresolved issue. Often, these planners take a considerable amount of time to converge and run offline when trying to discover complex contact sequences. Open-loop execution manifests susceptibility to uncertainties in model parameters and initial pose [101], so achieving robust execution necessitates closing the loop.

Imitation learning (IL) emerges as a promising pathway to tackle this challenge, a prospect bolstered by recent progress in gradient-field learning methods [106, 107, 108]. Yet, applying this strategy to whole-body manipulation tasks confronts impediments originating from the limitations of teleoperation methodologies. Predominantly tailored for end-effector tracking, these techniques fall short when tasked with effectively showcasing intricate whole-body maneuvers. Even when resorting to whole-body teleoperation techniques (such as motion-capture-based kinematic retargeting) for generating demonstrations, our empirical observations indicate limitations stem from the absence of comprehensive whole-body haptic feedback [109, 110] available to the teleoperator. Furthermore, the substantial volume of demonstrations required to effectively train a proficient imitation learning policy has emerged as another restrictive factor.

Recent advances in reinforcement learning (RL) have yielded remarkable outcomes in dexterous manipulation [5, 6, 111, 112] that are difficult to replicate with model-based planning.

Notably, these advancements frequently hinge upon the availability of task-specific insights, either in the form of well-defined reward functions or expert guidance. The acquisition of such task-related information, however, can pose difficulties, particularly in the domain of whole-body manipulation.

As a means to streamline the process of reward design, guided RL capitalizes on pre-existing knowledge inferred from data to enhance the efficiency and efficacy of the reinforcement learning process [81]. In particular, example-guided RL that aims to combine motion imitation with task-based rewarding (namely, standard RL) has been a popular concept in robotics (as well as in animation) to aid exploration by instilling a desired motion style, accelerate learning, and ease reward shaping [82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92].

As an example, Generative Adversarial Imitation Learning (GAIL) [113] effectively integrates a Generative Adversarial Network (GAN) [114] with RL to develop a discriminator that evaluates the resemblance between the policy and the example motions. However, GAIL's direct applicability is limited when the demonstrator's actions are not observable. Addressing this, recent work [93] introduced Adversarial Motion Priors (AMP), which leverages the GAIL framework to discern whether a state transition is a sample from the example motions or a sample generated by the agent. This approach does not require designing imitation objectives or motion selection mechanisms, and it can automatically synthesize a policy that completes a desired high-level task given a set of unstructured example motions. Due to its promise to impose motion characteristics on the RL policy without the burden of reward engineering, the AMP idea which was originally proposed for animation, has caught the attention of the robotics community and led to impressive results for quadruped locomotion, e.g., [94, 95, 96, 97, 98]. These are compelling examples showing that the AMP approach can generate policies: (i) with a desired style even from infeasible, partial, and scarce demonstrations; and (ii) that can directly transfer to the real world.

While significant progress has been made in using example motions to guide RL in locomotion tasks, this approach's translation to manipulation tasks, which involve intricate interactions between the robot and the manipuland, remains untested. Specifically, the applicability and effectiveness of the AMP framework in addressing challenges unique to manipulation problems have yet to be established.

6.3 METHOD

In this section, we briefly describe the methods we use for the proposed framework. The implementation details can be found in Section 6.4.2.

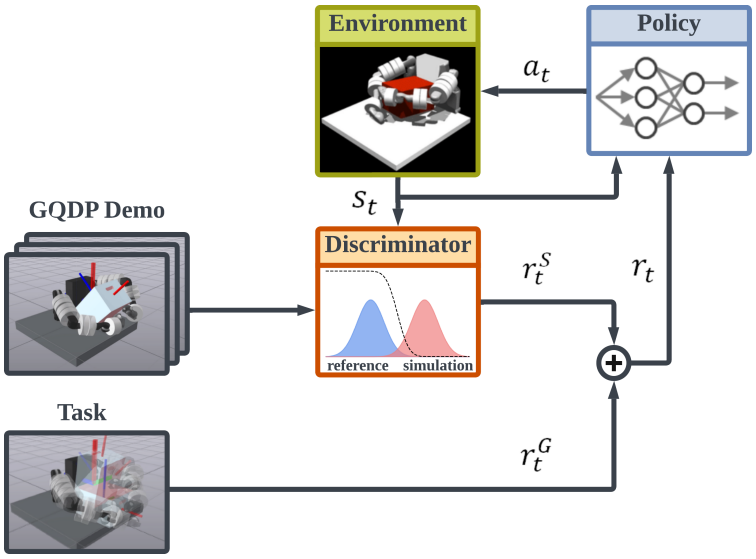


Figure 6.2: A flowchart of the proposed framework: the process commences with the acquisition of a reference motion dataset, generated through the utilization of a model-based planner; subsequently, the system undertakes the training of a discriminator, designed to acquire proficiency in learning an imitation reward. This imitation reward is integrated with a task reward to train a policy, which in turn empowers the robot to replicate the demonstrated motion while simultaneously accomplishing the intended task.

6.3.1 Planning Through Contact

We select GQDP [99] as the contact-implicit planner because of its capability for synthesizing long-horizon behaviors with multiple intermittent contacts. This approach assumes quasi-dynamic motions to reduce the problem into the configuration space and uses a contact smoothing scheme to derive a reachability metric. As a result, the planning through contact problem given initial and desired configurations can be solved by a sampling-based planner, in this case, the rapidly exploring random tree (RRT) method [115]. The path generated by RRT is then refined using a trajectory optimization to output a trajectory denoted as $\mathcal{T}' = \{\mathbf{q}_t^a, \mathbf{q}_t^u, \mathbf{a}_t | t = 0, \cdots, T\}$. Here, $\mathbf{q}_t^a$ and $\mathbf{q}_t^u$ are the configurations of the actuated and unactuated degrees of freedom of the system, and $\mathbf{a}_t$ denotes the robot position commands (or actions) for a joint stiffness controller at each discrete time step t.

It is pertinent to note that the resulting plan may exhibit non-physical behavior because of: (i) the quasi-dynamic assumption breaking when the system moves at non-negligible speeds and (ii) a robot teleport issue when switching contact modes. Due to (ii), the manipuland is not guaranteed to remain stable when the robot transitions from one contact configuration to another. This entails that while the robot follows the planned waypoints $\mathbf{q}_t^a$, the manipuland may not stay at the corresponding $\mathbf{q}_t^u$ for unstable tasks. Consequently, in our approach, we

retain only the robot's configuration sequence $\mathcal{T} = \{\mathbf{q}_t^a | t = 0, \cdots, T\}$ as the demonstration.

6.3.2 Example-Guided Reinforcement Learning

Given a dataset of reference motions and a task objective defined by a reward function, AMP synthesizes a control policy that enables the agent to achieve the task objective, while utilizing behaviors that resemble the motion style of the dataset. It is important to emphasize that the agent's behaviors are not obligated to precisely replicate individual motions from the dataset. Rather, the agent is encouraged to embody the broader characteristics observed within the collection of reference motions. This renders the GQDP a promising candidate for serving as the demonstrator. The emphasis here lies not on the successful task completion through the plan, but rather on utilizing it as a stylistic guide to steer the robot's behavior and increase the training efficiency and effectiveness.

Our policy is trained through a RL framework, wherein an agent engages with an environment in accordance with a policy denoted as π. At each time step t, the agent perceives the state $\mathbf{s}_t \in \mathcal{S}$ of the system. It then proceeds to sample an action $\mathbf{a}_t \in \mathcal{A}$ from the policy, adhering to the probabilistic distribution $\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)$. Subsequently, the agent runs this chosen action, leading to a new state $\mathbf{s}_{t+1}$, accompanied by a scalar reward $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. In accordance with [93], we formulate the reward function as comprising two distinct components: (i) the task reward r^G that quantifies the degree of task accomplishment, and (ii) the style reward that assesses the resemblance between the robot's motion and the reference motions:

$$r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = \lambda r^G(\mathbf{s}_t, \mathbf{a}_t) + (1 - \lambda) r^S(\mathbf{s}_t, \mathbf{s}_{t+1}), \tag{6.1}$$

where $\lambda \in [0, 1]$ determines the task reward weight with respect to the imitation reward weight.

To minimize the incorporation of reward shaping, we choose to employ a straightforward task reward function $r^G = d_{trans} + d_{rot} + p$, where d_{trans} and d_{rot} quantify the translation and rotation distances to the goal manipuland pose, and p encompasses conventional penalty components often integrated for RL, such as the actions and velocities.

r^S is estimated by a generative adversarial network that discriminates whether a state transition belongs to the demonstration distribution or not. Prior to presenting a state transition as input to the discriminator, an observation map $\Phi(\mathbf{s}_t)$ extracts the features, for discerning the attributes associated with a particular motion, from the system state $\mathbf{s}_t$. This is an important aspect since it lets us get away without feasible actions to be mimicked and

use only the equilibrium joint poses, $\mathbf{q}_a$, from plans for imitation, unlike typical behavior cloning methods.

6.4 EXPERIMENTS & DISCUSSION

6.4.1 Experiments

We test our approach on a whole-body manipulation task for pivoting and lifting a box, as depicted in Fig. 6.1. This evaluation is conducted using the Punyo robot [80]: an assemblage comprising of two Jaco robot arms, each with soft visuotactile sensors as end-effectors [116] and an array of 7 air-filled pressure-based sensors that cover each arm. To estimate the manipuland pose, we use an OptiTrack motion-capture system.

The code provided in [99] is used for planning, and we post-process the refined plan to connect segments together for two reasons: (i) to prevent undesired robot-manipuland collisions and (ii) to stabilize the object using a task-specific heuristic that makes the idle left arm hold the box when moving between segments.

For policy learning, we employ the proximal policy optimization (PPO) algorithm [117]. Particularly, we utilize the PPO implementation in [118] and the AMP implementation in [119] to run in the Isaac Gym [120]. Furthermore, we incorporate domain randomization [121] to facilitate the transfer of learned behaviors from simulation to real.

We run simulation experiments to compare PGRL to GQDP and to pure RL (denoted as RL), which is also trained with PPO using only the task reward. Additionally, we test the policy generated by our proposed approach, PGRL, on hardware. When rolling out the GQDP plan in simulation, the robot encounters difficulty in successfully lifting the box due to the teleportation issue described in Section 6.3.1. The inability to achieve success in this scenario, in the absence of any model mismatch, prompted the decision not to proceed with testing under conditions of model mismatch.

Figure 6.1 presents a juxtaposition between key points extracted from the plan employed for training and analogous key points identified during the execution of the RL policies. Please see the accompanying video for the full motions. The visual analysis shows the proficiency of our method in emulating the motion style imposed by the plan, whereas pure RL leads to unnatural behaviors with unsatisfactory task performance. Moreover, the RL training in PGRL helps to bridge the gaps in the GQDP plan (i.e., infeasible transitions during the teleports) and enables task completion. It is also noteworthy that PGRL alters the contact sequence in the plan to a more robust one owing to domain randomization.

To compare PGRL and RL, we train policies by maintaining uniformity across all parameters except for λ, which is set to 1 (i.e., no imitation reward) for the RL case. Fig. 6.3 illustrates the mean task reward curve along with the standard deviation resulting from three training instances with different seeds. Notably, our approach (trained with $\lambda = 0.9$) outperforms RL in terms of task reward even though RL's sole objective is to maximize this metric.

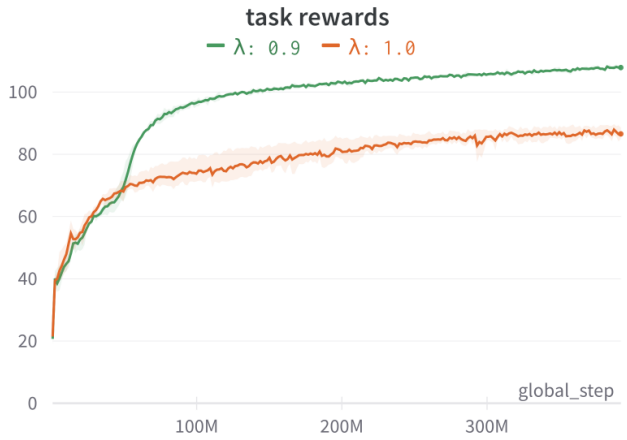


Figure 6.3: Performance of PGRL ($\lambda = 0.9$) and RL ($\lambda = 1$).

To evaluate the trained policies, we roll them out in 1000 distinct environment instantiations, encompassing variations in the physical properties of the manipuland (such as mass, friction, and size) as well as the initial position (extracted from the identical distribution utilized during training). The mean and standard deviation of the accumulated task reward, the smallest translational, and rotational distances of the manipuland to its goal pose for both RL and PGRL are shown in Fig. 6.4.

6.4.2 Implementation Details

Task Reward

The task reward is designed as

$$\begin{aligned} r_t^G = w_{trans}(1/(\|d_{trans}(\mathbf{q}_t^u, \mathbf{q}_{goal}^u)\| + 0.1)) + \\ w_{rot}(1/(\|d_{rot}(\mathbf{q}_t^u, \mathbf{q}_{goal}^u)\| + 0.1)) + \\ w_{action}||\mathbf{a}_t||^2 + w_{velocity}||\dot{\mathbf{q}}_t^u||^2 + w_{termination}\mathbb{1}(\mathbf{q}_t^u). \end{aligned} \tag{6.2}$$

The initial two terms incentivize task completion. The following two terms impose penalties on both the robot's actions and the manipuland's velocity. The final term enforces a

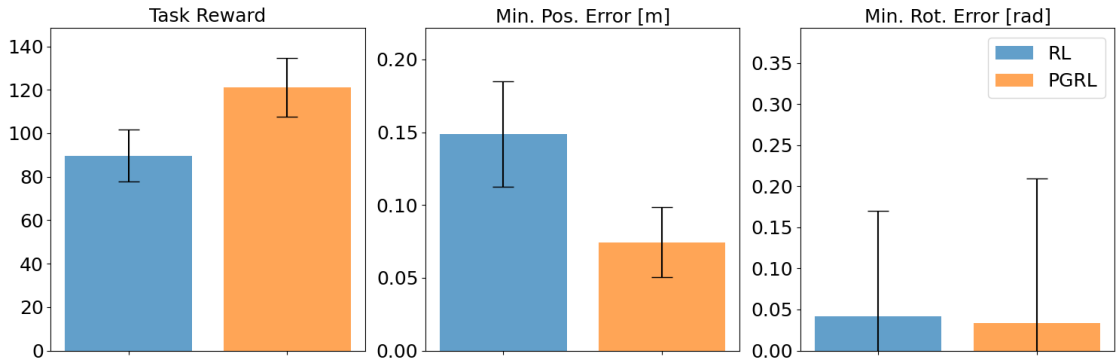


Figure 6.4: Aggregated results of rolling out the RL and PGRL policies in 1000 different environments. The box plots show the mean and standard deviation for the task reward and the minimum translational and rotational distances of the manipuland to the goal along each rollout.

penalty upon the activation of termination conditions. These conditions manifest when the box deviates significantly from the center of the table or experiences a drop. The function $\mathbb{1}(\cdot)$ is an activation function based on termination conditions. We use the weights $w_{trans} = 0.07$, $w_{rot} = 0.03$, $w_{action} = -0.002$, $w_{velocity} = -0.002$, and $w_{termination} = -1$ for all experiments. The task is defined by $\mathbf{q}^u_{goal} = (0.15, 0, 0.4, 0, -\pi/2, 0)$ concatenating the positions in m and roll-pitch-yaw in rad.

Observations

The observation of the discriminator is equilibrium joint position transitions of the robot $(\mathbf{q}^a_t,\ \mathbf{q}^a_{t+1})$. The observation of the policy is $(\mathbf{q}^a_t, \mathbf{q}^u_t, \mathbf{p}^{ee}_t)$ where $\mathbf{p}^{ee}_t$ is the Cartesian end-effector poses.

Training Parameters

The learning networks and algorithm were implemented in PyTorch 1.8.1 with CUDA 12.0. The training procedure encompassed the collection of experiences from 4096 uncorrelated instances of the simulator performed in parallel. The entirety of the experimental work was executed on a desktop equipped with NVIDIA 3090 GPUs. A single run comprising 1,500 iterations, adhering to the aforementioned computational settings and device specifications, was accomplished within approximately 4 hours. Detailed training parameters and network architectures are outlined in Table 6.1 and Table 6.2, respectively.

Table 6.1: Training parameters.

Parameter	Value
γ	0.99
τ	0.95
λ	0.9
parallel training environments	4096
sample per update iteration	4096
batch size	512
learning rate	$5e^{-5}$
KL divergence target	$8e^{-3}$
clip range	0.2
horizon length	64
discriminator weight decay	$1e^{-4}$
discriminator gradient penalty	10

Table 6.2: Network architecture.

Network	Type	Hidden Layers	Activation
policy	MLP	[256,128,64]	Rectified Linear Unit (ReLU)
value function	MLP	[256,128,64]	Rectified Linear Unit (ReLU)
discriminator	MLP	[256,128,64]	Rectified Linear Unit (ReLU)

Domain Randomization

Domain randomization techniques are judiciously employed during the training process to enhance robustness and performance. This involves the systematic application of disturbances to various parameters. Specifically, we initialize the manipuland on the table uniformly within a 5 cm radius of its nominal initial position [0.35. 0, 0.13] m. Additionally, a perturbation sampled from the uniform distribution $\mathcal{U}(-0.05, 0.05)$ is added to its yaw angle. A noise sampled from $\mathcal{N}(0, 0.02)$ rad is injected into the robot's actions (i.e., joint position commands to a stiffness controller), and a disturbance drawn from $\mathcal{U}(0.0, 0.5)$ m/s^2 is added to the gravity. Furthermore, the distributions $\mathcal{U}(0.9, 1.1)$, $\mathcal{U}(0.8, 1.2)$, and $\mathcal{U}(0.8, 1.0)$ are used to scale the dimensions and mass of the manipuland and the friction coefficient for all contacts. The nominal manipuland dimensions and mass are [0.41. 0.315, 0.26] m and 0.45 kg, and the nominal friction coefficient is 1.

CHAPTER 7: GENERALIZABLE MANIPULATION USING A PLANNED-CONTACT INFORMED POLICY

In Chapter 6, we introduced Plan Guided Reinforcement Learning (PGRL), a method that combines model-based manipulation planning and reinforcement learning by using planned trajectories as demonstrations in the training of an RL control policy. In this chapter, we introduce another method, Planned-Contact Informed Policy (PCIP), that uses planned trajectories more explicitly during the training of an RL control policy. Different features, especially the novel-designed planned contact feature, are extracted from the planned manipulation trajectories and given as input to the control policy. The inclusion of planned contact information in the policy's observation allows it to anticipate future contact modes without requiring privileged information or adding sensing modalities to the robot. Experimental results on a peg insertion task and a non-prehensile pivoting task show that PCIP achieves superior performance to standard RL and residual RL [122], and that planned contact information plays a vital role in PCIP's performance. [1]

7.1 INTRODUCTION

Robust and generalizable manipulation continues to be a significant challenge in the field of robotics. The development of dexterous manipulation capabilities, enabling automated reasoning about contact interactions with various objects, the robot's own body, and its surrounding environment, has the potential to greatly expand the range of physical tasks that can be automated in diverse settings. However, this is challenging because the robot often lacks precise knowledge about the manipuland. While its geometry can be detected through sensors such as cameras or LiDAR, other properties like mass and friction are less readily observable, often leaving the robot to rely on approximate estimations.

Take, for example, the scenario depicted in Fig. 7.1, where a robot must reorient a mustard bottle by pivoting it against a corner. Utilizing a model-based planning method, a trajectory can be planned based on estimated mass and friction coefficients, but discrepancies between these estimated properties of the manipuland and their actual values can easily lead to execution failures.

A possible approach to tackle this task is to employ reinforcement learning (RL). During the RL training phase for the pivoting policy, domain randomization can be applied to these

[1] This chapter is adapted from Patrick Naughton, Mengchao Zhang, Devesh Jha, and Kris Hauser, "Generalizable Manipulation Using a Planned-Contact Informed Policy", Under review for Robotics: Science and Systems (RSS), 2024.

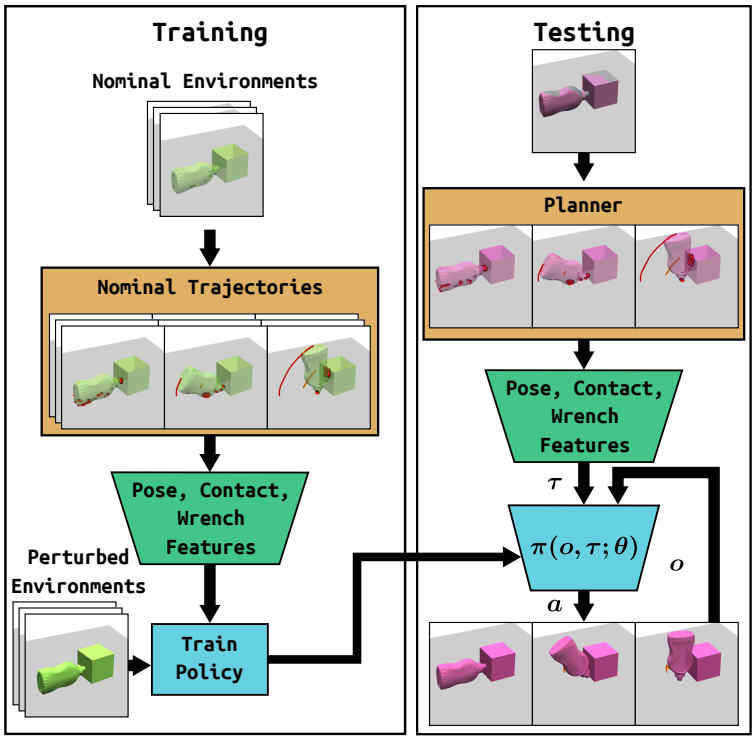


Figure 7.1: Illustration of the workflow of our proposed method on a task that requires the robot to pivot an object against a block. During the training phase (left), we employ trajectories planned for nominal environments and simulate policy rollouts on perturbations of these environments. This policy uses observations of pose, contact, and wrench features extracted from the nominal trajectory to accomplish the task. A red line shows the trajectory of the robot's contact point with the object, and red dots show object-environment contact points at each timestep. In the testing phase (right), a nominal trajectory is planned based on approximate estimates of parameters of the actual environment. The policy combines features of this trajectory with observations of the actual environment to accomplish the task.

uncertain parameters, with the aim of encompassing the true values within the training distribution. However, in practice, these policies often generalize poorly to similar tasks that may, for example, change the position and orientation of the mustard bottle and the block.

The recent trend of integrating model-based planning with RL is gathering attention for its potential to harness the strengths of both approaches. This methodology first gained traction in locomotion research. In [123], a network policy is developed in simulation to follow the optimized footholds determined by a model-based planner, achieving robust performance with sparse rewards in environments where viable footholds are scarce. Similarly, in [124], a policy is trained to imitate reference trajectories from model-based optimal control, offering improved resilience to modeling errors and state estimation inaccuracies compared to model-

based controllers. This concept has also been recently adapted to manipulation tasks. For instance, in [79], a policy is encouraged to mimic a physically infeasible trajectory generated by a model-based planner while still accomplishing the intended task, effectively addressing a challenging problem for either model-based methods or RL alone.

This study introduces Planned-Contact Informed Policy (PCIP), which addresses the robustness challenge posed by unobservable properties of the manipuland by merging model-based planning with RL. Like prior work, we condition manipulation policies on nominal trajectories, but our novel contribution is that we integrate planned contact information into the conditioning. The inclusion of planned contact information in the policy's observation allows it to anticipate future contact modes without requiring privileged information or adding sensing modalities to the robot. The workflow of the proposed method is illustrated in Fig. 7.1. We leverage manipulation planners to generate manipulation trajectories for imperfectly observed nominal models of the environment and train a trajectory-conditioned policy to accomplish the task despite this sensor noise. This allows the policy to robustly accomplish tasks similar to those it has been trained on.

Using a plan to encode the task, rather than seeking a global task encoding of goals, environments, and object geometries, also offers superior generalization performance by providing the policy with more task-invariant observations. Rather than reasoning about the full task space, the policy only needs to reason about a region around the planned trajectory.

Our experiments evaluate PCIP on two task types: peg insertion and non-prehensile pivoting. We show that PCIP achieves superior performance to standard RL and residual RL, and that planned contact information plays a vital role in PCIP's performance.

7.2 RELATED WORK

Model-based trajectory planning methods are appealing for addressing contact-rich manipulation challenges, due to their generalizability and the extensive research in this domain [8, 103, 105, 125, 126, 127, 128]. However, planning in the presence of contacts is notoriously difficult due to the stiff, non-smooth optimization landscape and the large number of discrete contact modes to consider. Various studies have approached this challenge in different ways: one approach [2] enumerates all possible contact modes; another [129] applies a convex, differentiable, and quasi-dynamic formulation of contact dynamics; while a third [56] employs complementarity constraints to model contact, adaptively selecting potential contact points on the object to maintain a manageable size for the optimization problem. Despite these recent advancements in contact-rich manipulation planning showing promise, the robust execution of these planned trajectories on hardware remains a significant hurdle.

These planners often require considerable time to converge, rendering them unsuitable for use within a Model Predictive Control (MPC) [130] framework to achieve robust execution through closed-loop control. Additionally, executing the planned trajectory in an open-loop fashion is vulnerable to model parameter uncertainties and inaccuracies in initial positioning.

RL is known for its robustness against uncertainties, largely attributed to the prevalent use of domain randomization, which exposes the policy to diverse instantiations of the environment during training, thereby enhancing adaptability during testing. In recent years, RL has achieved impressive successes in the realm of dexterous manipulation [5, 87, 111, 112, 131, 132, 133, 134]. One study [131] developed a versatile object reorientation controller capable of dynamically adjusting complex and unfamiliar objects to any orientation in real-time. This was achieved using a two-stage teacher-student training approach, where the teacher policy leveraged privileged information accessible only in simulation. A parallel approach [135] incorporated tactile sensing as an additional sensory input, following a similar training structure. Yet another work [136] demonstrated that robust object reorientation could be achieved solely through tactile feedback. However, the effectiveness of these methods often relies on the availability of task-specific insights, whether through meticulously crafted reward functions or convoluted training regimes.

As previously discussed, methods integrating model-based planning with RL offer the potential to harness the strengths of both techniques. Deep Tracking Control (DTC) [123] and Motion Imitation from Model-Based Optimal Control (MIMOC) [124] have demonstrated enhanced performance using this combined approach compared to either method alone on locomotion tasks. MIMOC imitates reference trajectories produced by model-based optimal control algorithms, which are generated offline. During the training process, only the proprioceptive states present in the example trajectory are utilized. In contrast, DTC employs a model-based planner to create trajectories online throughout the training phase. The DTC policy incorporates a broader set of inputs, encompassing not only the robot's proprioceptive information but also additional features such as the desired foothold locations. Our approach draws parallels with the MIMOC through its utilization of trajectories generated offline, while also aligning with the DTC framework in terms of leveraging contact-related features. However, to address manipulation problems effectively, we introduce and incorporate features that are more pertinent to contact dynamics and demonstrate their importance in solving manipulation problems that require high accuracy.

Combining model-based planning with RL to solve manipulation problems has also been explored in the literature. In [79], such synergy is applied to a whole-body large object manipulation problem, which is challenging for either strategy to tackle alone. Compared to the approach in [79], where a planned trajectory is used as an exemplar motion for the policy

to mimic in terms of motion style, our method adopts a more direct approach. We allow the policy to observe segments of the planned trajectory, thereby utilizing it more explicitly in our framework.

7.3 METHOD

7.3.1 Problem Formulation

We consider the problem of manipulating a single rigid object from a start pose to a goal pose. For example, inserting a peg into a socket or pivoting an object against a wall. We assume that the robot knows the geometry of the manipuland and environment, but has imperfect sensing of several key parameters of the problem, such as the manipuland's pose, target pose, mass, and friction coefficients with the robot and the environment. Additionally, each of these parameters may vary significantly between episodes so that the robot must adjust its policy to consistently achieve the task.

Formally, we cast this problem as a Contextual Markov Decision Process (CMDP)

$$M = \langle S', A, O, R, T, C, \phi, p(\mathbf{s}'|\mathbf{c}), p(\mathbf{c}), \gamma \rangle, \tag{7.1}$$

[137], with state space $S = S' \times C$, action space A, observation space O, observation function $\phi : S' \times C \to O$, reward function $R : S' \times C \to R$, and transition function $T((\mathbf{s}', \mathbf{c}), \mathbf{a})$. $\mathbf{s}' \in S'$ denotes a state in the "underlying" state space (for example, the configuration of the robot), and may change throughout an episode as the agent takes actions, while $\mathbf{c} \in C$ denotes a context in the "context" state space (for example, the mass of an object in a particular episode) which remains constant throughout each episode (but may change between episodes). $p(\mathbf{s}'|\mathbf{c})$ denotes the initial state distribution given the context vector $\mathbf{c}$, $p(\mathbf{c})$ denotes the distribution over context vectors (corresponding to variation in the setup of the manipulation task, such as initial poses of objects and their physical parameters), and γ denotes the discount factor. We make several assumptions about the CMDP:

1. Every context vector $\mathbf{c}$ designates a start state for the manipulation problem $s'_0(\mathbf{c}) \in S'$, and a zero-velocity region of the underlying state-space as a goal set $G(\mathbf{c}) \subset S'$ where the maximum reward of the MDP is given.

2. For any given context, we can simulate the corresponding MDP.

3. The space C is continuous and we receive one observation $\hat{\mathbf{c}}$ of the drawn context $\mathbf{c}$ corrupted with noise at the beginning of each episode. We refer to this noise distribution

as $\psi(\mathbf{c})$. $\hat{\mathbf{c}}$ is assumed to also be a physically valid context.

We additionally assume access to a transition function $\hat{T}((\mathbf{s}', \mathbf{c}), \mathbf{a}) = \widetilde{\mathbf{s}}'$ given a context $\mathbf{c}$ that approximates the true transition function $T((\mathbf{s}', \mathbf{c}), \mathbf{a})$. This allows us to compute a nominal trajectory from $s'_0(\hat{\mathbf{c}})$ to $G(\hat{\mathbf{c}})$.

Our goal is then to find a policy which maps a history of observations and observed context to an action, $\pi(\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_t, \hat{\mathbf{c}}) = \mathbf{a}_t$ that maximizes the long-term expected discounted return in the CMDP over a horizon T, $E_{\mathbf{c}\sim p(\mathbf{c})}[\sum_{t=1}^{T} \gamma^{t-1} r_t | s_0(\mathbf{c}), \pi]$, where r_t is the reward received at timestep t.

7.3.2 Planned-Contact Informed Policy

Achieving high reward in CMDPs representing manipulation problems is difficult in part because they involve many sources of variation, such as the stochastic transition function $T((\mathbf{s}', \mathbf{c}), \mathbf{a})$, the variation in contexts $p(\mathbf{c})$, and the noise corrupting the observation of $\mathbf{c}$, $\psi(\mathbf{c})$. Here, we use a nominal planner to adapt to variations in the context so that the policy only needs to learn to adapt to sensing errors. Our method trains a policy conditioned on the nominal trajectory using reinforcement learning to account for this source of error by training using nominal trajectories generated for the corrupted context $\hat{\mathbf{c}}$ rather than $\mathbf{c}$. To process the rich data available from the nominal trajectory into a form the policy can use to achieve the task, we introduce a pose-difference encoder (PDE) neural network and a contact-patch encoder (CPE) neural network. We show that these novel features improve the policy's performance compared to several baseline methods.

Nominal Trajectories

In this work, we consider manipulation problems involving just one movable object (manipuland) where the goal of the manipulation is to move the manipuland from a start pose to a set of goal poses. We assume that the robot, manipuland, and environment are all rigid, and the geometry of the robot and manipuland are known. After noisily observing the context vector, we build a model of the corresponding environment and compute a nominal trajectory using this model. We use two different methods to compute this trajectory according to the task type: a hand-specified context-parameterized controller, and a trajectory optimizer (STOCS) [56]. This nominal trajectory serves two purposes: it provides nominal commands to guide the robot's exploration during training, and more informative features than the raw observed context $\hat{\mathbf{c}}$ on which to condition the policy. The core question of this

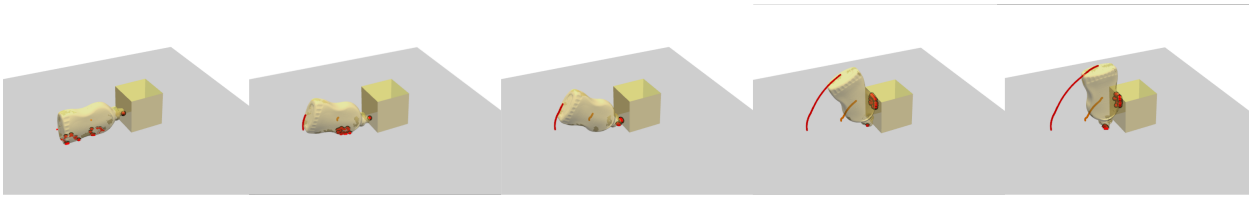


Figure 7.2: Snapshots of an example planned pivoting trajectory (from left to right), highlighting the path of the robot's contact point with the object in red. The points of contact between the object and the environment are depicted as red dots.

work is how this nominal trajectory can be encoded to enable robust and generalizable task completion.

Observation Representation

The core contribution of this work is how the nominal trajectory is encoded as an observation for the policy. Some of the features of the nominal trajectory, such as robot pose, can be tracked easily through a simple feedback controller. Aggregate wrenches, given appropriate force/torque sensors, can be regulated in response to unintended observed forces using force and impedance control. However, other potentially informative aspects of a manipulation plan, like the object contact force distribution, cannot be directly observed and regulated through available sensors. However, as shown in Fig. 7.2, in the nominal trajectory this information is readily available and can inform the policy of whether it should expect contact between the manipuland and the environment in the near future. Moreover, if there is an inconsistency in contact state, such as loss of contact, this information and an appropriate corrective action can be potentially inferred from the observed object movement.

Using this intuition, we design three categories of trajectory features to inform the robot of its progress along the nominal trajectory and how it can advance towards the goal: **Pose and Geometry** features, **Contact Patch** features, and **Wrench** features. Pose and Geometry features relate the manipuland's current pose to its nominal future poses to inform the policy of how the manipuland should move in the near future. Contact Patch features encode the nominal contacts the robot and manipuland will experience in the future. Although these features are not directly sensed from the true environment, they can still inform the policy of likely contact points so that it can determine how its actions will affect the manipuland. Finally, Wrench features encode the nominal wrenches on several links of the robot to inform the policy of the magnitudes of forces and torques it should experience.

Rather than encoding these features along the entire nominal trajectory, we only encode them for timesteps in a temporal window around the policy's current progress point along

the nominal trajectory, since these features are most relevant to the policy's current choice of action. We use the pose of the manipuland and find the closest pose along the nominal trajectory to determine the policy's "progress point" along the trajectory, as shown in Fig. 7.3. This yields a more informative notion of progress than the time since the episode began if, for example, the robot loses contact with the manipuland and it falls back to its initial pose. Specifically, the progress point k^* is computed as

$$k^* =_{k \in [K]} d(T_m[t]P_m, \hat{T}_m[k]P_m), \tag{7.2}$$

where P_m denotes a (object-local) subsampled point cloud of the manipuland, $T_m[t]$ the current pose of the manipuland at time t, $\hat{T}_m[k]$ the nominal pose of the manipuland at time k, pose-point cloud multiplication transforms the point cloud according to the given pose, and $d(P_a, P_b)$ computes the average distance between the points with known correspondences in point clouds P_a and P_b, which differ only by a rigid transformation. This point-cloud based computation removes dependence on the arbitrary choice of object origin for finding the "closest" pose and further eliminates the need to weigh between positional and rotational distances. Using this progress point, we examine the nominal trajectory in the range $[k^* - n_p, k^* + n_f]$, where n_p and n_f are integers indicating how far into the past and future of the nominal trajectory to look. In our implementation, we set $n_p = 0$ and $n_f = 1$ so that the trajectory window only contains features from the current and next time steps of the nominal trajectory. For each time index i in this range, we then compute the three classes of feature vectors, and concatenate them to form the input to the policy.

Pose and Geometry Features

We represent the object motion in a way that is invariant to the arbitrary choice of the manipuland's coordinate reference frame, and ideally that will allow the same policy to manipulate multiple geometries. To achieve this, we compute a "point-cloud difference" $P_d \coloneqq T_e[t]^{-1}(\hat{T}_p[i]P_m - T_p[t]P_m)$ that specifies the translation between each manipuland keypoint in the current timestep and its nominal position. We then pass P_d through a PointNet [138] to obtain a $\mathbf{d}_{\text{pcd}} = 8$-dimensional encoding, as shown in Fig. 7.3. In addition, we encode the transform between the current end-effector pose ($T_e[t]$) and its nominal poses ($\hat{T}_e[i]$) and nominal commanded poses ($\hat{T}_c[i]$) for each timestep in the window. These quantities are expressed in the current end effector frame ($T_e[t]^{-1}\hat{T}_e[i]$ and $T_e[t]^{-1}\hat{T}_c[i]$) and vectorized by concatenating the axis-angle representation of the transform's rotation with its translation before being input into the policy.

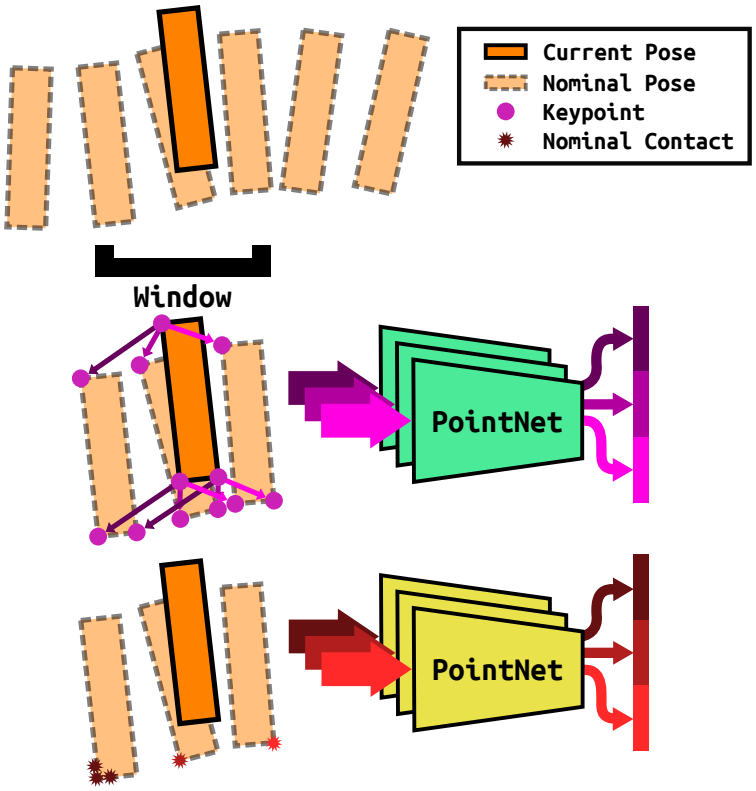


Figure 7.3: Our feature extractor first determines a time window of relevant information in the planned trajectory by finding the index of the closest pose to the current pose (top). Pose and geometry, contact patch, and wrench features for each timestep in the window are extracted. Geometry keypoints (middle) and nominal contact points (bottom) are processed through two PointNet architectures. Each PointNet generates an 8-dimensional feature vector, and all feature vectors are concatenated as policy inputs.

Contact Patch Features

We encode the plan's contact region information in a manner that accepts variable distribution and numbers of contact points. To encode this information, we sample points on the manipuland and robot end effector, and during each timestep in the nominal trajectory determine which of these points are in contact, marking points as in contact if they are within $d_{\text{cmax}} = 1$ mm of another object. These contact points form a "contact point-cloud," $P_c[i]$. The 3D coordinates of these points are transformed to the nominal end effector frame and passed through a PointNet to yield a $\mathbf{d}_{\text{contact}} = 8$-dimensional encoding, as shown in Fig. 7.3.

Table 7.1: Observations used by PCIP. Features computed at nominal progress point i (e.g. nominal command, nominal pose difference, etc.) are computed for all i in a temporal window around the computed progress point k^*, $i \in [k^* - n_p, k^* + n_f]$. Note that the "Planned command" is indexed by the current time t in the trajectory rollout while the "Nominal command" is indexed by i.

Observation Name	Definition
Robot config	$q[t]$
Manipuland pose	$T_e[t]^{-1}T_m[t]$
Manipuland goal	$T_e[t]^{-1}T_g$
Planned command	$\hat{T}_e[t]^{-1}\hat{T}_c[t]$
Nominal command	$\hat{T}_e[i]^{-1}\hat{T}_c[i]$
Nominal pose difference	$T_e[t]^{-1}\hat{T}_e[i]$
Manipuland PC difference	$\text{PDE}(T_e[t]^{-1}(\hat{T}_p[i]P_m - T_p[t]P_m))$
Nominal contact PC	$\text{CDE}(\hat{T}_e[i]^{-1}P_c[i])$
Nominal wrench	$[W_r[i], W_\ell[i], W_w[i]]$
Command residual	$T_e[t]^{-1}T_c[t-1]$

Wrench Features

We include the nominal wrenches on the robot's gripper fingers (left and right respectively denoted as W_ℓ and W_r), as well as at the robot's wrist (W_w) for timestep i. We also include the "command residual," i.e. the pose difference $T_e[t]^{-1}T_c[t-1]$ between the previously commanded pose and the true current pose of the end-effector. Since the commanded pose is tracked using an impedance controller, this feature provides a signal for the true force on the end effector.

Standard Features

To benchmark against standard RL algorithms, we assume that a constant-dimension observation vector is provided for each task, which we call the "standard" observation vector. This includes elements such as the current and goal poses of the manipuland ($T_m[t]$ and T_g), and the joint configuration of the robot ($q[t]$).

All the features used during the training in this paper are summarized in Table 7.1.

Action Representation

The action of the robot is a 6-dimensional vector representing the desired residual twist of the robot's end-effector, interpreted in the robot's current end-effector frame. This vector

is added to the desired twist given by the nominal trajectory at the current timestep to compute a total twist. This total twist is then forward-integrated using Euler integration, and the resulting target pose is tracked by a task-space impedance controller. All task types considered here can be completed without changing the robot's gripper state (open or closed), so no action is provided to modify it.

7.4 EXPERIMENTS AND DISCUSSION

We characterize three key properties of the PCIP method:

1. Its in-distribution generalization.
2. Its out-of-distribution generalization.
3. Its robustness to perturbations of the nominal environment (ψ).

We furthermore show that each component of PCIP is necessary to achieve higher task-success rates by individually ablating the pose, contact patch, and wrench features. We test PCIP on two different manipulation task types using a simulated Franka-Emika Panda robot arm and gripper: peg-in-hole insertion (the **Insert** task type) and 90° vertical pivoting (the **Pivot** task type), shown in Fig. 7.4. Within each of these task types, we generate individual tasks that specify the manipuland's geometry, start and goal poses, and physical parameters. We use IsaacGym [139] to simulate policy rollouts and its SDF geometry representation [140] to simulate contact-rich interactions. We evaluate policies trained by our method under contexts drawn from the same distribution observed in training (ID) and out-of-distribution (OOD) contexts. We also used different methods to generate nominal trajectories, depending on which was most appropriate for the task type. In both task types, the nominal trajectories are generated using perturbed measurements of the context (e.g. the poses of the objects, physical properties).

7.4.1 Task Types and Nominal Trajectories

For the Insert task type, we use the peg and socket assets from IndustRealKit [134], which include rectangular and circular pegs of width ranging from 8 to 16 mm, with peg-socket clearances ≤ 0.6 mm. The pegs and sockets are shown in Fig. 7.5. Each task starts with the robot grasping the peg, and the goal is to fully insert the peg into the socket. Within this task type, each task varies the geometry of the peg (and correspondingly the geometry of the socket), the pose of the socket, the initial pose of the peg with respect to the socket,

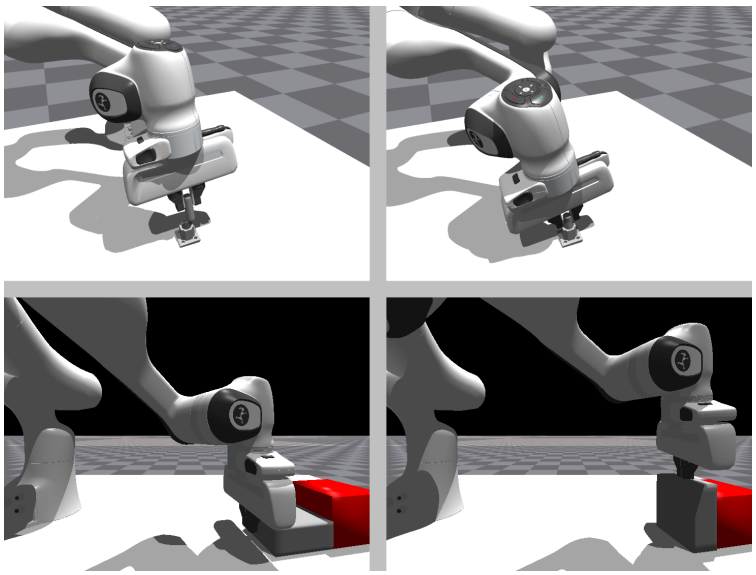


Figure 7.4: Example task initial configurations (left) and completed states (right) for the Insert (top) and Pivot (bottom) task types.

the coefficients of friction between the peg and robot, and peg and socket, and the density of the peg. To generate nominal trajectories for this task type, we wrote a custom controller that commands the simulated robot in IsaacGym to move the peg to a pre-insertion pose above the socket, wait for the arm to stop moving, and then insert the peg straight into the socket. The task is considered successfully completed if the peg is touching the bottom of the socket at the end of execution.

The Pivot task type requires the robot to push the target object against a static block while it is laying on a table and rotate it 90° so that it stands upright. Within this task type, the tasks vary the geometry of the pivoted object (the rectangular 16 mm IndustRealKit peg, a large box, and a mustard bottle, shown in Figure 7.5), the pose (XY position and Z rotation) of the static block on the table, the density of the object, and the coefficient of friction between the object and environment. To generate nominal trajectories, we use the STOCS planner [56] to compute 2D pivoting trajectories, including the desired contact force between the robot and object at each timestep. We then use the known kinematics of the robot, geometries of the objects, and gains of the impedance controller to translate this into a full 3D pivoting trajectory. The task is considered successfully completed if the object is within 5° of vertical at the end of execution.

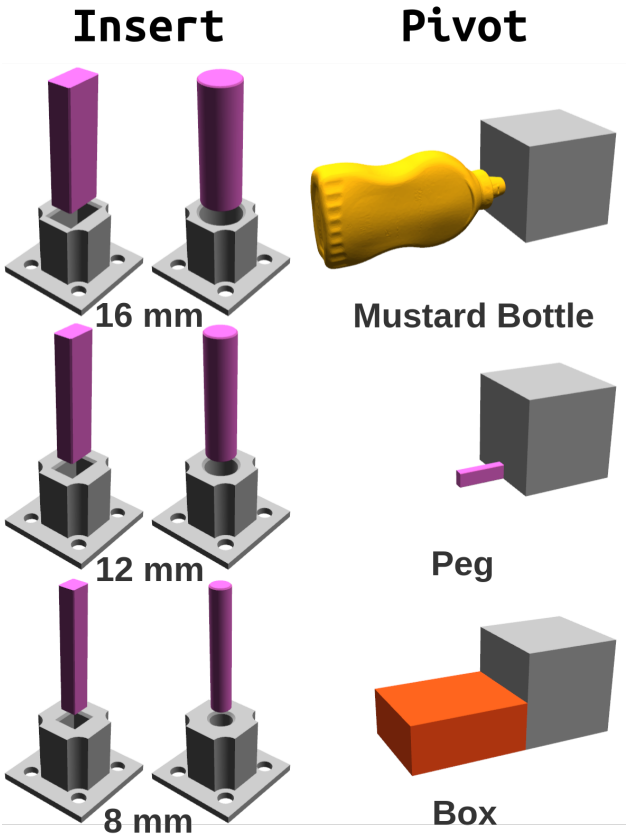


Figure 7.5: The geometries used for the different task types. In the Insert task type, two distinct classes of pegs are utilized: round and rectangular. Each class comprises three sizes, with the maximum dimension of the cross-section being 8 mm, 12 mm, and 16 mm, respectively. For the Pivot task type, a mustard bottle, the same 16 mm rectangular peg as in the Insert task type, and a box are employed.

7.4.2 Training

We parameterize the manipulation policy with a neural network. Table 7.6 details the architecture and training parameters of this network. To train the PCIP method, we draw a context vector $\mathbf{c}$ that determines the setup of the task. We then corrupt this vector with noise to obtain $\hat{\mathbf{c}}$, which is passed to the appropriate planner to generate a nominal trajectory. Table 7.5 describes the distribution of this noise. The current policy is then rolled out on the true environment $\mathbf{c}$ and the agent observes a sequence of states, actions, and rewards. We then update the policy using PPO [141]. The critic network receives the standard observation vector augmented with additional features; the full set of critic features is listed in Table 7.7. We employ the simulation-aware policy update introduced by [134], which downweights policy updates from episodes that result in high (non-physical) interpenetration between geometries to prevent policies from achieving high reward by exploiting weaknesses

of the simulator. We train a separate policy for each task type but use the same policy for all instances of that type. The PDE and CPE are optimized end-to-end with the policy. For the Insert task type, we follow Tang and Lin et al. [134] in using a curriculum that initially samples initial peg poses that are partially inserted into the socket and gradually shifts this distribution as the policy achieves higher success rates until the pegs all start outside of the socket. The Pivot task type does not use a curriculum.

7.4.3 Reward Function

The reward function for each CMDP consists of three components: a control penalty, a tracking reward, and a task reward. The control penalty is given by

$$R_{\text{control}} = -\alpha_{\text{control}} \mathbf{u}[t]^\intercal \mathbf{u}[t], \tag{7.3}$$

where $\mathbf{u}[t]$ is the current applied torque (minus the gravity compensation torque) and α_{control} is a scaling factor. The tracking reward applies a penalty to the agent for each timestep it deviates significantly from the nominal trajectory. The reward is given by

$$R_{\text{track}} = -\alpha_{\text{track}} \max(\|\mathbf{p}_e[t] - \hat{\mathbf{p}}_e[k^*]\|_2 - d_{\text{track}}, 0), \tag{7.4}$$

where $\mathbf{p}_e[t]$ and $\hat{\mathbf{p}}_e[k^*]$ are the current and nominal positions of the end effector, d_{track} is a hyperparameter specifying the maximum allowable tracking deviation, and α_{track} is a scaling factor. While tracking the nominal trajectory is often helpful, it is generally not sufficient to complete the task because of the discrepancy between $\mathbf{c}$ and $\hat{\mathbf{c}}$. Thus, we clip the penalty so that the robot receives no penalty within d_{track} of the nominal trajectory and include a task specific reward. For the Insert task type, we use the signed-distance field (SDF) reward of [134], given by

$$R_{\text{sdf}} = -\log\left(\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\right), \tag{7.5}$$

where x_i are points sampled from the mesh of the target object in its current pose, N is the number of sampled points, and $\phi(\cdot)$ is the SDF of the target object in its goal pose (clamped to return only nonnegative values). For the Pivot task type, we only care about the angle the object makes with the vertical, and so use a different task reward given by

$$R_{\text{angle}} = z_m^\intercal z_{\text{block}}, \tag{7.6}$$

Table 7.2: Reward Hyperparameters

Hyperparameter	Insert	Pivot
α_{control}	1	0
α_{track}	0	100
d_{track}	0	2 cm

where z_m is the local z-axis of the manipuland and z_{block} is the local z-axis of the static block (vertical).

The total reward is simply the sum of the control penalty, tracking reward, and appropriate task reward. Depending on the task, different reward hyperparameters were used; their values are shown in Table 7.2. We introduced a control penalty to the Insert task to promote smoother execution trajectories, and introduced a tracking reward in the Pivot task to encourage the robot to maintain contact with the object during execution. The same reward is used for all methods within a task type.

7.4.4 Evaluation and Metrics

To evaluate PCIP's in and out-of-distribution generalization, we hand-specified two context distributions: a training distribution $p_{\text{train}}(\mathbf{c})$ and an approximately disjoint testing distribution $p_{\text{test}}(\mathbf{c})$. Table 7.4 describes the training and testing context distributions for each task type. During training we draw contexts from $p_{\text{train}}(\mathbf{c})$ and collect data from rollouts on these environments. To evaluate the policies, we test their in-distribution generalization (ID) by drawing novel contexts from $p_{\text{train}}(\mathbf{c})$, and their out-of-distribution generalization (OOD) by drawing contexts from $p_{\text{test}}(\mathbf{c})$ and rolling the policy out on these environments. Figure 7.6 shows the training and testing distributions over the objects' initial poses and the resulting distribution of nominal trajectories. In these experiments, we perturb the context vector $\mathbf{c}$ using the "Narrow" distribution (Table 7.5) to generate the observed context vector $\hat{\mathbf{c}}$ and nominal trajectories. We evaluate each policy by the proportion of tasks it successfully completes in each distribution.

We additionally evaluate PCIP's tolerance to sensing errors ($\psi(\mathbf{c})$) by testing each policy under "No Perturbation," "Narrow Perturbation," and "Wide Perturbation" conditions, where the variance of $\psi(\mathbf{c})$ increases under each condition. Table 7.5 shows the precise $\psi(\mathbf{c})$ distribution used for the Narrow and Wide conditions (under the No Perturbation condition we simply have $\mathbf{c} = \hat{\mathbf{c}}$). This introduces larger errors between the nominal and actual environments, which tends to make the nominal trajectory less representative of successful

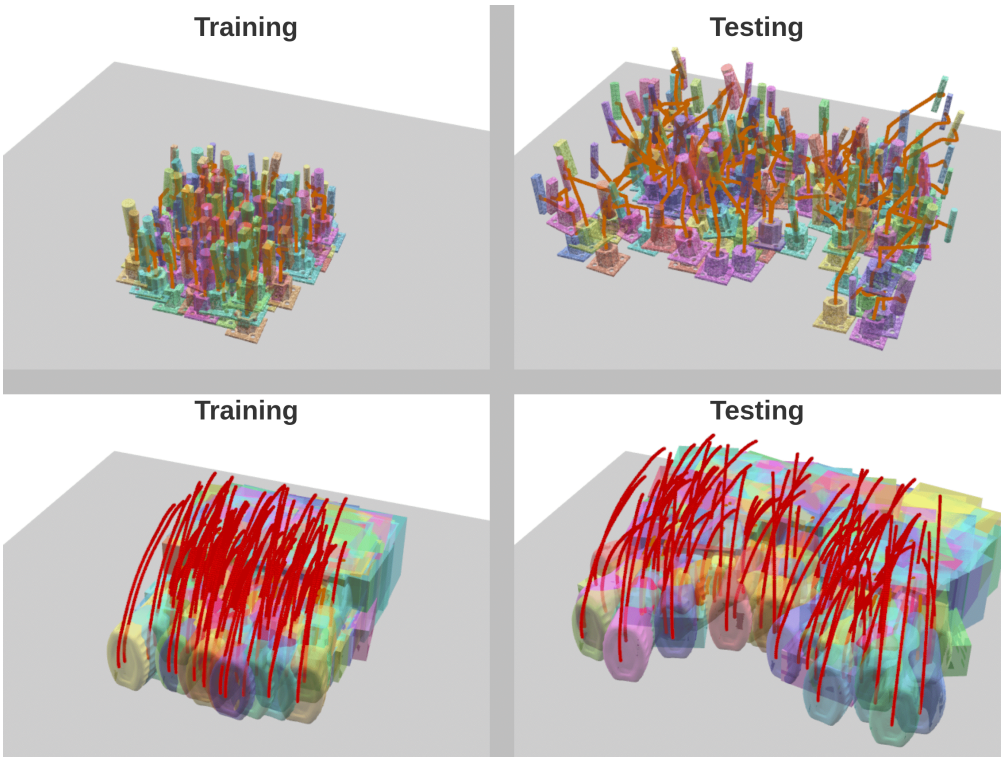


Figure 7.6: Training (left) and testing (right) setup distributions for the Insert (top) and Pivot (bottom) task types. We show the nominal initial pose of the object and trajectory; the trajectory of the center of the object is shown in orange while the trajectory of the contact between the robot and object is shown in red for the pivot task. When drawing samples for the testing distribution, samples in the support of the training distribution (1σ for Gaussian training distributions) are rejected.

trajectories. In these experiments, we draw context vectors from $p_{\text{train}}(\mathbf{c})$.

7.4.5 Baseline Methods

We compare PCIP against three baseline methods: Open Loop control, Residual Reinforcement Learning (RRL), and vanilla Reinforcement Learning (RL). Open loop control simply issues the planned commands that are given by the nominal trajectory in each timestep. Vanilla RL learns a policy mapping the standard observation vector to a robot action while RRL learns a mapping from the same observation vector to a residual action that is added to the planned command at each timestep. RRL additionally observes the planned command at each timestep.

We also test three ablations of PCIP: PCIP (NP), which removes the pose observations, PCIP (NC), which removes the contact observations, and PCIP (NW), which removes the

Table 7.3: Features used by different methods.

	PCIP	RL	RRL	PCIP(NP)	PCIP(NC)	PCIP(NW)
Robot config	✓	✓	✓	✓	✓	✓
Manipuland pose	✓	✓	✓	✓	✓	✓
Manipuland goal	✓	✓	✓	✓	✓	✓
Planned command	✓		✓	✓	✓	✓
Nominal command	✓				✓	✓
Nominal pose difference	✓				✓	✓
Manipuland PC difference	✓				✓	✓
Nominal contact PC	✓			✓		✓
Nominal wrench	✓			✓	✓	
Command residual	✓			✓	✓	

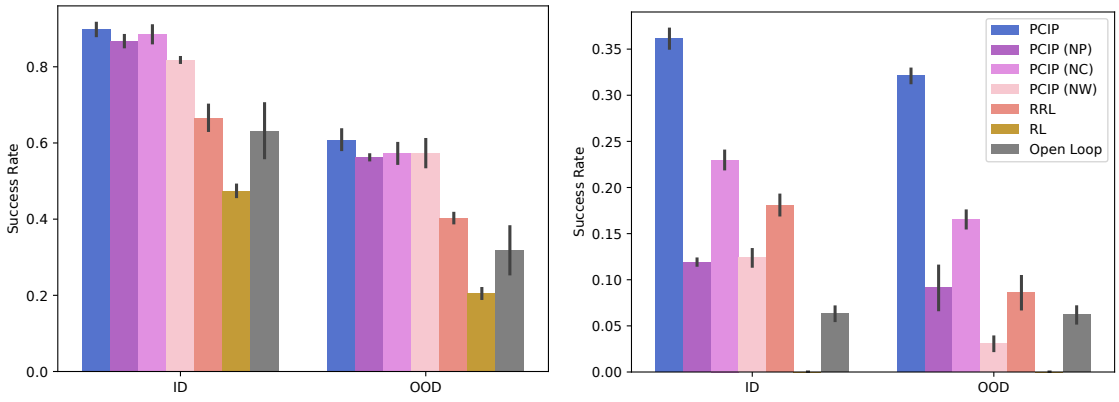
wrench observations. The features used by each policy are shown in Table 7.3.

7.4.6 Results and Discussion

Figures 7.7 and 7.8 show the success rate of PCIP across testing scenarios compared with the baseline and ablated methods. For each method, we run each experiment under three different seeds and report the average and standard deviation of the method's success rate.

As shown in Fig. 7.7, PCIP outperforms all baseline and ablated methods across both task types under both in and out-of-distribution contexts. Considering the Insert task type, we note that, as expected, the RL baseline's performance steeply declines when moving from in to out-of-distribution task types as it encounters unseen observation vectors. Although PCIP also experiences a decline in performance, it is able to use the nominal trajectory to adapt to the new contexts while offering superior robustness compared to simply executing the trajectory in an open-loop. PCIP also significantly outperforms RRL, showing that the detailed information in the nominal trajectory, not just its desired actions, is vital for successful adaptation. Similarly, in the Pivot task type, PCIP's performance degrades very little when moving from in-distribution to out-of-distribution contexts compared to the performance loss of RRL.

As shown in Fig. 7.8, PCIP achieves robust execution of both the Insert and Pivot task types across a range of perturbation distributions. This suggests that while using the nominal trajectory to extract features, PCIP does not become overly reliant on it. Particularly in the Insert task type, even though the Open Loop performance drops sharply as the perturbation distribution widens, PCIP maintains a high success rate. In fact, PCIP outperforms open loop execution by a wide margin across all tested perturbation distributions, indicating



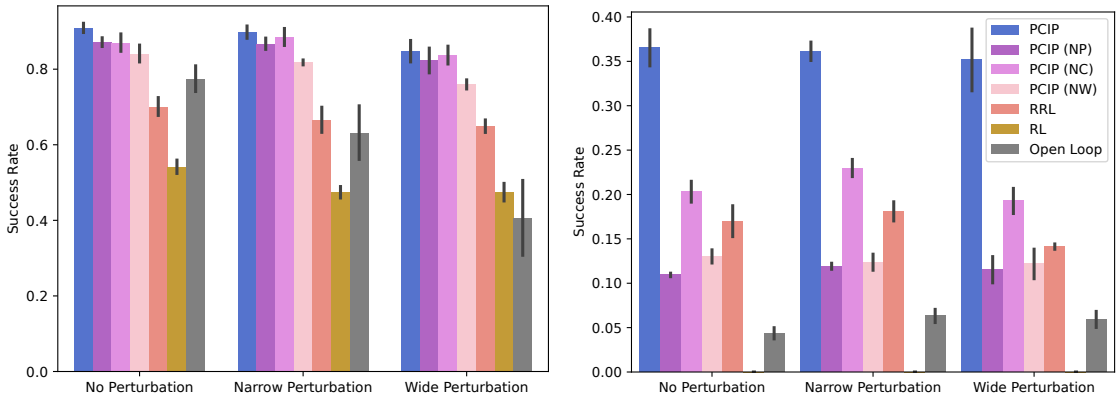
(a) Pivot (b) Insert

Figure 7.7: In and out-of-distribution generalization performance of each method. PCIP outperforms all methods under both in and out-of-distribution contexts, demonstrating how it can effectively use a nominal trajectory to generalize over a wide task distribution. We report the average and standard deviation of each method's success rate when tested across three seeds.

that even when accurate models of the environment are available, PCIP can improve the robustness of plan execution.

We note that PCIP outperforms PCIP (NC) across all testing scenarios, supporting the idea that nominal contact information can inform the policy of how to robustly achieve manipulation tasks. Interestingly, the performance gap between them is much larger on the Pivot task type than the Insert task type, suggesting that detailed contact information is more vital for robust execution of pivoting than insertion. We hypothesize that this is because pivoting trajectories involve more changes of contact as the object transitions from horizontal to vertical, requiring the policy to reason about the object's changing axis of rotation. Comparatively, insertion only requires a single change of contact as the peg makes contact with the socket. The inherent instability of pivoting compared to insertion also makes the consequences for suboptimal actions more severe. Knowing the nominal contacts of the object during the pivot thus allows the policy to more robustly control the object's motion.

Surprisingly, PCIP (NP) and PCIP (NW) tend to perform worse on the Pivot task type than RRL, despite having more information. This suggests that depending on the feature, adding more information from a nominal trajectory may actually make it more difficult to learn a successful policy. In this case, it suggests that the nominal contact information must be combined with other features to yield useful information. This makes sense since only nominal contact patches are observed: this feature must be combined with features



(a) Pivot

(b) Insert

Figure 7.8: Robustness of each method to different levels of sensing noise. Even though PCIP uses the nominal trajectory to extract features, it is significantly more robust than Open Loop and RRL policies under a wide range of perturbations. We report the average and standard deviation of each method's success rate when tested across three seeds.

referenced to the observed state to usefully inform the policy.

7.4.7 Limitations

Although PCIP has shown promising results, there are several limitations and potential avenues for future research.

In this study, we have established that PCIP offers superior OOD generalization compared to several baseline methods. However, such generalization is confined to variations within a given task type, rather than extending across different task types. Future research will focus on exploring the feasibility of policy generalization to more dimensions within each task type, such as to unseen geometries, and even across different task types.

In the proposed method, the robot's policy uses information extracted from nominal trajectories, without integrating advanced sensory inputs such as tactile sensors. In the future, we will investigate how these execution-time observations can be integrated with their nominal counterparts to maximize task performance.

7.4.8 Training Details

Table 7.4: Training and testing context distributions. Samples drawn for the testing distribution that are within the support of the training distribution are rejected (within 1 standard deviation for normally distributed training distributions). $U[\cdot]$ denotes a uniform distribution over the given domain. $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. Position distributions are in units of meters. Rotations are drawn by sampling a moment vector from the given distribution (in radians). Objects used in the tasks are given "standard" mass and friction values, and these standard values are multiplied by the sampled values.

Task Type	Physical Parameter	Train/Test	Distribution
Insert	Socket Position	Train	$U[[-0.10, 0.10] \times [-0.10, 0.10] \times [0.00, 0.05]]$
		Test	$U[[-0.10, 0.20] \times [-0.20, 0.20] \times [0.00, 0.10]]$
	Socket Rotation	Train	$N([0, 0, 0], [0.05, 0.05, 0.05])$
		Test	$N([0, 0, 0], [0.10, 0.10, 0.10])$
	Socket Friction Multiplier	Train	$U[0.8, 1.2]$
		Test	$U[0.6, 1.4]$
	Peg Position	Train	$U[[-0.01, 0.01] \times [-0.01, 0.01] \times [0.00, 0.01]]$
		Test	$U[[-0.05, 0.05] \times [-0.05, 0.05] \times [0.00, 0.05]]$
	Peg Rotation	Train	$N([0, 0, 0], [0.05, 0.05, 0.05])$
		Test	$N([0, 0, 0], [0.10, 0.10, 0.10])$
	Peg Friction Multiplier	Train	$U[0.9, 1.1]$
		Test	$U[0.8, 1.2]$
	Peg Mass Multiplier	Train	$U[0.9, 1.1]$
		Test	$U[0.8, 1.2]$
Pivot	Block Position	Train	$U[[0.0, 0.1] \times [-0.1, 0.1] \times [0.0, 0.0]]$
		Test	$U[[0.0, 0.2] \times [-0.2, 0.2] \times [0.0, 0.0]]$
	Block Rotation	Train	$N([0, 0, 0], [0.1, 0.1, 0.1])$
		Test	$N([0, 0, 0], [0.2, 0.2, 0.2])$
	Block Friction Multiplier	Train	$U[0.9, 1.1]$
		Test	$U[0.8, 1.2]$
	Object Mass Multiplier	Train	$U[0.9, 1.1]$
		Test	$U[0.8, 1.2]$

Table 7.5: Perturbation distributions (ψ) used to introduce discrepancies between the true and nominal environments ($\mathbf{c}$ and $\hat{\mathbf{c}}$). $U[\cdot]$ denotes a uniform distribution over the given domain. $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. Position distributions are in units of millimeters. Rotations are drawn by sampling a moment vector from the given distribution (in radians). Friction perturbations are added to their nominal values, masses are perturbed by multiplying the nominal value by the sampled value.

Task Type	Physical Parameter	Narrow/Wide	Distribution
Insert	Socket Position	Narrow	$U[[-0.5, 0.5] \times [-0.5, 0.5] \times [0.0, 0.5]]$
		Wide	$U[[-1.0, 1.0] \times [-1.0, 1.0] \times [0.0, 1.0]]$
	Socket Rotation	Narrow	$N([0, 0, 0], [0.001, 0.001, 0.001])$
		Wide	$N([0, 0, 0], [0.01, 0.01, 0.01])$
	Socket Friction	Narrow	$N(0, 0.03)$
		Wide	$N(0, 0.08)$
	Peg Position	Narrow	$U[[-0.5, 0.5] \times [-0.5, 0.5] \times [0.0, 0.5]]$
		Wide	$U[[-1.0, 1.0] \times [-1.0, 1.0] \times [0.0, 1.0]]$
	Peg Rotation	Narrow	$N([0, 0, 0], [0.001, 0.001, 0.001])$
		Wide	$N([0, 0, 0], [0.01, 0.01, 0.01])$
	Peg Friction	Narrow	$N(0, 0.03)$
		Wide	$N(0, 0.08)$
	Peg Mass Multiplier	Narrow	$N(1, 0.03)$
		Wide	$N(1, 0.08)$
Pivot	Block Position	Narrow	$U[[0.0, 5.0] \times [-1.0, 1.0] \times [0.0, 0.0]]$
		Wide	$U[[0.0, 10.0] \times [-1.0, 1.0] \times [0.0, 0.0]]$
	Block Rotation	Narrow	$N([0, 0, 0], [0.0, 0.0, 0.001])$
		Wide	$N([0, 0, 0], [0.0, 0.0, 0.01])$
	Object Position	Narrow	$U[[0.0, 0.0] \times [-0.5, 0.5] \times [0.0, 0.0]]$
		Wide	$U[[0.0, 0.0] \times [-1.0, 1.0] \times [0.0, 0.0]]$
	Block Friction	Narrow	$N(0, 0.03)$
		Wide	$N(0, 0.08)$
	Object Mass Multiplier	Narrow	$N(1, 0.03)$
		Wide	$N(1, 0.08)$

Table 7.6: PPO parameters and network architectures used in each policy.

	Insert	Pivot
MLP network size (Actor)	[512, 256, 128]	[512, 256, 128]
MLP network size (Critic)	[256, 128, 64]	[256, 128, 64]
LSTM network size (Actor)	[256, 2]	[256, 2]
PointNet pointwise channels (Actor)	[64, 64, 64]	[64, 64, 64]
PointNet MLP size (Actor)	[64, 64]	[64, 64]
Horizon length (T)	128	128
Adam learning rate	0.001	0.001
Discount factor (γ)	0.998	0.998
GAE parameter (λ)	0.95	0.95
Entropy coefficient	1.0	0.0
Critic coefficient	2.0	2.0
Minibatch size	8192	8192
Minibatch epochs	8	8
Clipping parameter (ϵ)	0.2	0.2

Table 7.7: Observations used by the PPO critic.

Observation Name	Definition
Robot config	$q[t]$
Manipuland pose	$T_e[t]^{-1}T_m[t]$
Manipuland goal	$T_e[t]^{-1}T_g$
Joint velocity	$\dot{q}[t]$
End effector twist	$\dot{T}_e[t]$

CHAPTER 8: CONCLUSION

This thesis makes foundational contributions to address key challenges in the perception, planning, and control workflow of automatic robotic manipulation. It introduces a novel computational model that employs a semi-infinite/infinite programming approach to effectively address the intricate nature of pervasive contact scenarios and demonstrate its applicability in various contexts including object's 6D pose estimation within cluttered environments, contact-rich stable grasp pose optimization, and manipulation trajectory optimization involving complex-shaped objects. Additionally, it introduces methods that integrate model-based planning with reinforcement learning to develop controllers capable of executing contact-rich manipulation tasks with enhanced robustness.

8.1 FUTURE DIRECTIONS

Imagine a scenario in which a user captures a scene as an RGB-D image, and through a digital interface, simply selects and relocates an object to specify its desired goal pose. In the background, a trajectory planning algorithm engages, planning a trajectory that instructs the robot on moving the object from its original position to the designated target state. Upon completing the planning phase, a feedback controller, skilled in stabilizing manipulation trajectories, takes over. This controller takes the planned trajectory as input, guiding the robot to execute the task with high precision. While the methods introduced in this thesis contribute foundational elements to this envisioned scenario, realizing such a seamless sensor-to-action pipeline necessitates additional work. Below, we identify several areas for further exploration that could facilitate the development of this integrated framework:

Active Sensing and RGB-D Sensor Data Post-Processing Pipeline In Chapters 2 and 3, the proposed pose and trajectory optimization methods utilize high-fidelity geometric representations of objects, specifically dense point clouds sampled from the surfaces of object meshes. These meshes are assumed to be available and are typically sourced from datasets such as Google Scanned Objects and the YCB dataset. Such datasets are generated in laboratory settings with carefully designed hardware setups and undergo extensive post-processing from raw sensor data to achieve high-quality meshes. However, in real-world scenarios, a robot must be capable of constructing such geometric representations from raw sensor data. This necessitates the development of an automated active sensing algorithm that allows the robot to comprehensively observe the object, alongside a specialized sensor data

post-processing pipeline that transforms raw data obtained during the active sensing phase into a high-quality geometric representation that can be used for the following planning step.

Robust Planning under Uncertainty In Chapters 2 and 3, the pose and trajectory optimization methods presented presuppose the knowledge of specific physical properties of the object, such as mass, center of mass, and friction coefficients. However, accurately measuring these parameters in practice can be challenging, and discrepancies between the nominal values used during planning and their actual real-world counterparts may lead to failures in execution. While Chapters 6 and 7 introduce techniques to enhance the robustness of executing planned trajectories, integrating robustness directly into the trajectory planning phase remains preferable. It would be of considerable interest to explore the incorporation of robust optimization techniques into the STOCS framework to develop manipulation trajectories that maintain robustness against uncertainty in physical parameters.

Universal Control Policy In Chapter 7, we demonstrate that the Planned-Contact Informed Policy significantly outperforms several baseline methods in out-of-distribution generalization. However, this generalizability is currently limited to variations within a specific task type, rather than extending to different task types. Future research should aim at developing a universal control policy capable of stabilizing various manipulation trajectories across a broader spectrum of tasks. This endeavor would involve expanding policy generalization to encompass more dimensions within each task type, including adapting to unseen object geometries and different task objectives. Achieving this will likely need the exploration of conditioning the policy on different relevant features extracted from the planned trajectories, thereby enabling the policy to adapt to changes in object geometry and the objectives of the task.

8.2 CONCLUSION

This thesis makes foundational contributions towards advancing automatic robotic manipulation, with the aim of bridging the gap between human-like contact interactions and the capabilities of current robotic systems.

To enhance the estimation of an object's 6D pose in cluttered environments—a common challenge in real-world settings—we have integrated the non-penetration among rigid objects into the Iterative Closest Points algorithm. In an effort to reduce the need for expert knowledge or human intervention in determining the appropriate level of simplification for objects' geometric representations needed for manipulation planning algorithms, we have de-

veloped efficient planning methods that directly utilize high-fidelity geometric representation converted from raw sensor output. Moreover, to bolster the robustness of executing planned trajectories and address the challenge of incorporating contact-rich manipulation plans into a feedback control loop—owing to their long solve time—we have formulated methods to train reinforcement learning controllers that either mimic the motion style of the planned trajectory or condition on features extracted from the trajectories to stabilize the execution.

Looking forward, the concurrent development of advanced sensing technologies and efficient numerical solvers, in conjunction with the methods proposed in this thesis, holds the promise of equipping robots with the capability for autonomous dexterous manipulation in unstructured, open-world environments.

REFERENCES

[1] K. Chao and P. Hur, "Generalized contact constraints of hybrid trajectory optimization for different terrains and analysis of sensitivity to randomized initial guesses," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1435–1440.

[2] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact Mode Guided Motion Planning for Quasidynamic Dexterous Manipulation in 3D," in *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9811872/ pp. 2730–2736.

[3] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2012, pp. 137–144.

[4] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[5] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray et al., "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[6] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," in *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[7] F. R. Hogan and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 755–773, 2020.

[8] A. Aydinoglu, A. Wei, and M. Posa, "Consensus complementarity control for multi-contact mpc," *arXiv preprint arXiv:2304.11259*, 2023.

[9] E. Huang, X. Cheng, and M. T. Mason, "Efficient contact mode enumeration in 3d," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2021, pp. 485–501.

[10] R. Reemtsen and S. Görner, "Numerical methods for semi-infinite programming: a survey," in *Semi-Infinite Programming*. Springer, 1998, pp. 195–275.

[11] Y. Ding, M. Zhang, C. Li, H.-W. Park, and K. Hauser, "Hybrid sampling/optimization-based planning for agile jumping robots on challenging terrains," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2839–2845.

[12] M. Zhang and K. Hauser, "Non-penetration iterative closest points for single-view multi-object 6d pose estimation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1520–1526.

[13] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.

[14] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[15] K. Hauser, "Semi-infinite programming for trajectory optimization with non-convex obstacles," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1106–1122, 2021.

[16] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6d object pose and predicting next-best-view in the crowd," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3583–3592.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[18] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3343–3352.

[19] "Benchmark for 6d object pose estimation," Accessed on 2023-09-04. [Online]. Available: https://bop.felk.cvut.cz/home/

[20] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. GlentBuch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis et al., "Bop: Benchmark for 6d object pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[21] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee, 2010, pp. 998–1005.

[22] T. Birdal and S. Ilic, "Point pair features based object detection and pose estimation revisited," in *2015 International Conference on 3D Vision*. IEEE, 2015, pp. 527–535.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[24] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[25] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *European Conference on Computer Vision*. Springer, 2020, pp. 577–594.

[26] C. Mitash, A. Boularias, and K. Bekris, "Physics-based scene-level reasoning for object pose estimation in clutter," *The International journal of robotics research*, vol. 41, no. 6, pp. 615–636, 2022.

[27] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[28] K. Hauser, "Semi-infinite programming for trajectory optimization with nonconvex obstacles," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 565–580.

[29] R. Hettich and K. O. Kortanek, "Semi-infinite programming: theory, methods, and applications," *SIAM review*, vol. 35, no. 3, pp. 380–429, 1993.

[30] T. Hodaň, J. Matas, and Š. Obdržálek, "On evaluation of 6d object pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 606–619.

[31] B. Drost, M. Ulrich, P. Bergmann, P. Hartinger, and C. Steger, "Introducing mvtec itodd-a dataset for 3d object recognition in industry," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2200–2208.

[32] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold et al., "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3364–3372.

[33] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *European Conference on Computer Vision*. Springer, 2020, pp. 574–591.

[34] M. Zhang and K. Hauser, "Semi-infinite programming with complementarity constraints for pose optimization with pervasive contact," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6329–6335.

[35] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[36] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[37] A. T. Miller and P. K. Allen, "Graspit!: A versatile simulator for grasp analysis," in *in Proc. of the ASME Dynamic Systems and Control Division*. Citeseer, 2000.

[38] I.-M. Chen and J. W. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 507–512, 1993.

[39] M. Liu, Z. Pan, K. Xu, and D. Manocha, "New formulation of mixed-integer conic programming for globally optimal grasp planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4663–4670, 2020.

[40] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Robotics Research*. Springer, 2020, pp. 455–472.

[41] E. J. Anderson and A. B. Philpott, *Infinite Programming: Proceedings of an International Symposium on Infinite Dimensional Linear Programming Churchill College, Cambridge, United Kingdom, September 7–10, 1984*. Springer Science & Business Media, 2012, vol. 259.

[42] M. López and G. Still, "Semi-infinite programming," *European Journal of Operational Research*, vol. 180, no. 2, pp. 491–518, 2007.

[43] Z.-Q. Luo, J.-S. Pang, and D. Ralph, *Mathematical programs with equilibrium constraints*. Cambridge University Press, 1996.

[44] M. Anitescu, "On solving mathematical programs with complementarity constraints as nonlinear programs," *Preprint ANL/MCS-P864-1200, Argonne National Laboratory, Argonne, IL*, vol. 3, 2000.

[45] R. Fletcher* and S. Leyffer, "Solving mathematical programs with complementarity constraints as nonlinear programs," *Optimization Methods and Software*, vol. 19, no. 1, pp. 15–40, 2004.

[46] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[47] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[48] R. J. Vanderbei and D. F. Shanno, "An interior-point algorithm for nonconvex nonlinear programming," *Computational Optimization and Applications*, vol. 13, no. 1-3, pp. 231–252, 1999.

[49] P. T. Boggs, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, Jan. 1995.

[50] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[51] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 15, pp. 2673–2691, 1996.

[52] N. Wu, G. Kenway, C. A. Mader, J. Jasa, and J. R. R. A. Martins, "pyoptsparse: A python framework for large-scale constrained nonlinear optimization of sparse systems," *Journal of Open Source Software*, vol. 5, no. 54, p. 2564, 2020. [Online]. Available: https://doi.org/10.21105/joss.02564

[53] "Klampt – Intelligent Motion Laboratory at UIUC," Accessed on 2023-09-04. [Online]. Available: http://motion.cs.illinois.edu/klampt/

[54] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Proceedings Shape Modeling Applications, 2004.* IEEE, 2004, pp. 167–178.

[55] "Ycb benchmarks object and model set," Accessed on 2023-09-04. [Online]. Available: http://ycbbenchmarks.org

[56] M. Zhang, D. K. Jha, A. U. Raghunathan, and K. Hauser, "Simultaneous Trajectory Optimization and Contact Selection for Multi-Modal Manipulation Planning," in *Robotics: Science and Systems*, Daegu, South Korea, July 2023.

[57] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," 2012.

[58] C. Eppner and O. Brock, "Planning grasp strategies that exploit environmental constraints," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4947–4952.

[59] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[60] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on mechatronics*, vol. 15, no. 5, pp. 783–792, 2009.

[61] M. P. Kelly, "Transcription methods for trajectory optimization: a beginners tutorial," *arXiv preprint arXiv:1707.00284*, 2017.

[62] K. Harada, K. Hauser, T. Bretl, and J.-C. Latombe, "Natural motion generation for humanoid robots," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 833–839.

[63] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[64] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "Planar in-hand manipulation via motion cones," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 163–182, 2020.

[65] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[66] Z. Manchester, N. Doshi, R. J. Wood, and S. Kuindersma, "Contact-implicit trajectory optimization using variational integrators," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1463–1476, 2019.

[67] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[68] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: https://drake.mit.edu

[69] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols," *arXiv preprint arXiv:1502.03143*, 2015.

[70] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2553–2560.

[71] Open Robotics. [Online]. Available: https://app.gazebosim.org/OpenRobotics/fuel/models/Sofa

[72] Sketchfab. [Online]. Available: https://sketchfab.com/3d-models/burlap-pillow-1ef567278bba4db68ac5488d6b4bc851

[73] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 265–293, 2021.

[74] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Systems*, 2018.

[75] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.

[76] A. U. Raghunathan, D. K. Jha, and D. Romeres, "Pyrobocop: Python-based robotic control & optimization package for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 985–991.

[77] I. Dikin, "Iterative solution of problems of linear and quadratic programming," in *Doklady Akademii Nauk*, vol. 174, no. 4. Russian Academy of Sciences, 1967, pp. 747–748.

[78] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 3400–3407.

[79] M. Zhang, J. Barreiros, and A. O. Onol, "Plan-guided reinforcement learning for whole-body manipulation," *arXiv preprint arXiv:2310.12263*, 2023.

[80] A. Goncalves, N. Kuppuswamy, A. Beaulieu, A. Uttamchandani, K. M. Tsui, and A. Alspach, "Punyo-1: Soft tactile-sensing upper-body robot for large object manipulation and physical human interaction," in *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2022, pp. 844–851.

[81] J. EEßerer, N. Bach, C. Jestel, O. Urbann, and S. Kerner, "Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics," *IEEE Robotics & Automation Magazine*, 2022.

[82] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[83] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[84] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[85] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[86] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas et al., "Reinforcement and imitation learning for diverse visuomotor skills," *arXiv preprint arXiv:1802.09564*, 2018.

[87] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3651–3657.

[88] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," *arXiv preprint arXiv:1910.04281*, 2019.

[89] S. Christen, S. Stevšić, and O. Hilliges, "Demonstration-guided deep reinforcement learning of control policies for dexterous human-robot interaction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2161–2167.

[90] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[91] A. Nair, A. Gupta, M. Dalal, and S. Levine, "AWAC: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[92] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto, "Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation," in *2023 ieee international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 5954–5961.

[93] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[94] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5120–5126.

[95] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[96] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger, and G. Martius, "Learning agile skills via adversarial imitation of rough partial demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 342–352.

[97] C. Li, S. Blaes, P. Kolev, M. Vlastelica, J. Frey, and G. Martius, "Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2944–2950.

[98] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, 2023.

[99] T. Pang, H. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *arXiv preprint arXiv:2206.10787*, 2022.

[100] V. Kumar, Y. Tassa, T. Erez, and E. Todorov, "Real-time behaviour synthesis for dynamic hand-manipulation," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6808–6815.

[101] X. Cheng, E. Huang, Y. Hou, and M. T. Mason, "Contact mode guided motion planning for quasidynamic dexterous manipulation in 3d," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2730–2736.

[102] M. Zhang, D. K. Jha, A. U. Raghunathan, and K. Hauser, "Simultaneous trajectory optimization and contact selection for multi-modal manipulation planning," *arXiv preprint arXiv:2306.06465*, 2023.

[103] R. Natarajan, G. L. Johnston, N. Simaan, M. Likhachev, and H. Choset, "Torque-limited manipulation planning through contact by interleaving graph search and trajectory optimization," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8148–8154.

[104] A. Ö. Önol, R. Corcodel, P. Long, and T. Padır, "Tuning-free contact-implicit trajectory optimization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1183–1189.

[105] M. Wang, A. Ö. Önol, P. Long, and T. Padır, "Contact-implicit planning and control for non-prehensile manipulation using state-triggered constraints," in *The International Symposium of Robotics Research*. Springer, 2022, pp. 189–204.

[106] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *arXiv preprint arXiv:2303.04137*, 2023.

[107] S. Haldar, J. Pari, A. Rai, and L. Pinto, "Teach a robot to fish: Versatile imitation from one minute of demonstrations," *arXiv preprint arXiv:2303.01497*, 2023.

[108] M. Du, S. Nair, D. Sadigh, and C. Finn, "Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets," *arXiv preprint arXiv:2304.08742*, 2023.

[109] J. Barreiros, L. Tianshu, M. Chiaramonte, K. Jost, Y. Menguc, N. Colonnese, and P. Agarwal, "Hyfar: A textile soft actuator for haptic clothing interfaces." ACM, 2022.

[110] C. Rognon, S. Mintchev, F. Dell'Agnola, D. Atienza, and D. Floreano, "Flyjacket: An upper body soft exoskeleton for immersive drone control." IEEE, 2018, pp. 2362–2369.

[111] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," in *Conference on Robot Learning*. PMLR, 2022, pp. 297–307.

[112] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*. PMLR, 2020, pp. 1101–1112.

[113] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[114] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[115] S. M. LaValle et al., "Rapidly-exploring random trees: A new tool for path planning," 1998.

[116] N. Kuppuswamy, A. Alspach, A. Uttamchandani, S. Creasey, T. Ikeda, and R. Tedrake, "Soft-bubble grippers for robust and perceptive manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9917–9924.

[117] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[118] D. Makoviichuk and V. Makoviychuk, "rl-games: A high-performance framework for reinforcement learning," Accessed on 2023-09-04. [Online]. Available: https://github.com/Denys88/rl_games

[119] IsaacGymEnvs, Accessed on 2023-09-04. [Online]. Available: https://github.com/NVIDIA-Omniverse/IsaacGymEnvs

[120] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

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

[122] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[123] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024.

[124] A. Miller, S. Fahmi, M. Chignoli, and S. Kim, "Reinforcement learning for legged robots: Motion imitation from model-based optimal control," *arXiv preprint arXiv:2305.10989*, 2023.

[125] V. Kurtz, A. Castro, A. Ö. Önol, and H. Lin, "Inverse dynamics trajectory optimization for contact-implicit model predictive control," *arXiv preprint arXiv:2309.01813*, 2023.

[126] I. Mordatch, Z. Popovic, and E. Todorov, "Contact-Invariant Optimization for Hand Manipulation," *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, p. 8 pages, 2012. [Online]. Available: http://diglib.eg.org/handle/10.2312/SCA.SCA12.137-144

[127] Y. Shirai, D. K. Jha, and A. U. Raghunathan, "Robust pivoting manipulation using contact implicit bilevel optimization," *arXiv preprint arXiv:2303.08965*, 2023.

[128] M. Toussaint, J. Harris, J.-S. Ha, D. Driess, and W. Hönig, "Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 13 753–13 760.

[129] T. Pang, H. T. Suh, L. Yang, and R. Tedrake, "Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models," *IEEE Transactions on Robotics*, 2023.

[130] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[131] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," *arXiv preprint arXiv:2211.11744*, 2022.

[132] C. Higuera, J. Ortiz, H. Qi, L. Pineda, B. Boots, and M. Mukadam, "Perceiving extrinsic contacts from touch improves learning insertion policies," *arXiv preprint arXiv:2309.16652*, 2023.

[133] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's Cube with a Robot Hand," Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.07113

[134] B. Tang, M. A. Lin, I. A. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. S. Narang, "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[135] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik, "General in-hand object rotation with vision and touch," in *Conference on Robot Learning*. PMLR, 2023, pp. 2549–2564.

[136] Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, and X. Wang, "Rotating without seeing: Towards in-hand dexterity through touch," *arXiv preprint arXiv:2303.10880*, 2023.

[137] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A Survey of Zero-shot Generalisation in Deep Reinforcement Learning," *Journal of Artificial Intelligence Research*, vol. 76, pp. 201–264, Jan. 2023. [Online]. Available: http://jair.org/index.php/jair/article/view/14174

[138] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *arXiv:1612.00593 [cs]*, Apr. 2017. [Online]. Available: http://arxiv.org/abs/1612.00593

[139] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_

[140] Y. Narang, K. Storey, I. Akinola, M. Macklin, P. Reist, L. Wawrzyniak, Y. Guo, A. Moravanszky, G. State, M. Lu, A. Handa, and D. Fox, "Factory: Fast Contact for Robotic Assembly," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[141] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017. [Online]. Available: http://arxiv.org/abs/1707.06347