Title

Contact-Aware Learning in Robotic Grasping, Manipulation and Sensing

Author

Zhu, Xinghao

Publication Date

2024

Peer reviewed|Thesis/dissertation

Contact-Aware Learning in Robotic Grasping, Manipulation and Sensing

By

Xinghao Zhu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Masayoshi Tomizuka, Chair
Associate Professor Mark W. Mueller
Professor Pieter Abbeel

Spring 2024

Contact-Aware Learning in Robotic Grasping, Manipulation and Sensing

Abstract

Contact-Aware Learning in Robotic Grasping, Manipulation and Sensing

By

Xinghao Zhu

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

In an era where robotic manipulators are increasingly sought after for customized production and household services, this dissertation delves into the development of advanced skills and dexterity in robot manipulations. It proposes an innovative approach by integrating physical principles, particularly contact mechanics, into robot learning. This integration aims to enhance data efficiency and reliability, moving beyond the current paradigm where learning agents heavily depend on vast data and extensive exploration.

The dissertation unfolds in three aspects of contact-aware robot learning. Firstly, it pioneers the development of robust and sample-efficient grasping frameworks. In Chapter 2, a contrastive grasp planning module is introduced to mitigate the effects of camera noise and simulation-to-reality gap, thereby improving grasp robustness. Chapter 3 presents the Maximum Likelihood Grasp Sampling Loss, achieving an eightfold reduction in training sample requirements compared to existing methods. Additionally, Chapter 4 explores grasping planning for multi-fingered hands, expanding the versatility of robotic manipulators. The second aspect delves into the integration of contact planning into a spectrum of manipulation tasks. Chapter 5 proposes contact-aware learning from demonstrations. This approach allows robots to assimilate skills by observing human demonstrations, effectively accelerating robotic skill acquisition. Chapter 6 explores the concept of safe contact in robotic operations, investigating strategies that permit contact with obstacles while ensuring safety. In Chapter 7, an intelligent robotic assembly framework is introduced, featuring multi-level reasoning that combines sequence reasoning transformers and meticulous planning of contact points. The third aspect focuses on the sensing of contact through vision-based tactile sensors, as discussed in Chapter 8. This chapter presents a method for reconstructing contact profiles from image imprints captured by these sensors, providing a crucial feedback mechanism during manipulation tasks.

Collectively, these contributions present a suite of novel grasping, manipulation, and assembly strategies. They are designed to reduce reliance on hand-engineering, thereby improving

the efficiency and stability of robotic systems in diverse applications. This comprehensive body of work demonstrates the feasibility and benefits of integrating contact into robot learning. The effectiveness and practical applicability of these methods are rigorously validated through a series of simulations and real-world experiments involving various manipulators and hands.

To My Family and Friends

Contents

List of Figures

List of Tables

Acknowledgments

It has been an extraordinary journey spanning five and a half years at Berkeley. This period has been marked by profound learning, discovery, and the privilege of encountering remarkable individuals.

Foremost, I extend my deepest and most sincere gratitude to my advisor, Prof. Masayoshi Tomizuka. His exceptional professionalism and dedication as a scholar, coupled with his enthusiasm and patience as a mentor, have been pivotal in my academic journey. Prof. Tomizuka's humour and optimism have been beacons during challenging times, while his unwavering support and trust have empowered me to venture into uncharted territories and persevere through setbacks. The completion of this dissertation owes a great deal to his consistent support and encouragement.

I am also profoundly grateful to my dissertation and qualifying committee members: Prof. Mark Mueller, Prof. Pieter Abbeel, Prof. Kameshwar Poolla, Prof. Somayeh Sojoudi, and Prof. Anil Aswani. Their insightful counsel and invaluable guidance have significantly shaped the direction of my research and the fruition of this dissertation.

My heartfelt thanks are extended to the FANUC Corporation for their generous funding of my research. The engaging and insightful discussions with the researchers at the FANUC Advanced Research Laboratory, particularly Mr. Tetsuaki Katou, Mr. Kaimeng Wang, Dr. Yongxiang Fan, Dr. Te Tang, Dr. Yu Zhao, and Dr. Hsien-chung Lin, have been instrumental throughout these years.

The opportunity to spend enriching summers at Google X and Mitsubishi Electric Research Laboratories (MERL) is also deeply appreciated. Collaborating with Dr. Bodi Yuan and Prof. Wenzhao Lian at Google X was enlightening; their visionary perspectives and passion for innovation have left an indelible mark on me. Similarly, working alongside Dr. Jeroen van Baar, Dr. Siddarth Jain, Dr. Anoop Cherian, Dr. Devesh K. Jha, and Dr. Diego Romeres at MERL was an extraordinary experience, thanks to their sharp intellect and warm camaraderie.

I extend a special acknowledgment to my collaborators for their invaluable contributions and thought-provoking discussions. Profound appreciation goes to Prof. Jianyu Chen, Prof. Lin Shao, Prof. Wenzhao Lian, Prof. Cewu Lu, Dr. Yongxiang Fan, Dr. Te Tang, Dr. Bodi Yuan, Dr. Daniel Freeman, Dr. Shiyu Jin, Dr. Changhao Wang, Dr. Chen Tang, Dr. Jeroen van Baar, Dr. Siddarth Jain, Dr. Anoop Cherian, Dr. Devesh K. Jha, Dr. Diego Romeres, Lingfeng Sun, Mingyu Ding, Ran Tian, Chengfeng Xu, Chenran Li, Xiang Zhang, Yefan Zhou, Jinghan Ke, Zhixuan Xu, Zhixin Sun, Bizhe Bai, Jun Lv, Qingtao Liu, Yuwei Zeng, Qi Ye, and Wu-Te Yang. Their insights have substantially enriched my research experience.

My time as a member of the Mechanical Systems Control (MSC) laboratory over the past five years has been invaluable. I am grateful for the advice and support from both former and current members of the group, including Prof. Changliu Liu, Dr. Daisuke Kaneishi, Dr. Wei Zhan, Dr. Zining Wang, Dr. Kiwoo Shin, Prof. Jiachen Li, Dr. Yeping Hu, Dr. Zhuo Xu, Dr. Yujiao Cheng, Dr. Saman Fahandezhsaadi, Dr. Jessica Leu, Dr. Yiyang Zhou, Dr. Huidong Gao, Dr. Hengbo Ma, Jining Li, Catherine Weaver, Akio Kodaira, Qiyang

Qian, Jen-Wei Wang, Yichen Xie, Wei-Jer Chang, Keita Kobashi, Yiheng Li, Yuxin Chen, Boyuan Liang, Yixiao Wang, Chensheng Peng, Shuqi Zhao, Yutaka Shimizu, Mingrui Yu, and Mingxiao Huo. Their collective support has been a cornerstone of my academic pursuit.

I would also like to express my heartfelt appreciation to all the friends and colleagues I encountered at Berkeley, Google, and MERL. The moments shared have been both memorable and enjoyable, contributing significantly to my personal and professional growth.

Lastly, but most importantly, my deepest gratitude is reserved for my parents, Mr. Zhongzhi Huang and Mrs. Huilan Zhu, and my beloved girlfriend, Yiling Fang. Your unwavering presence, whether in times of triumph or adversity, has been my greatest source of strength and inspiration. It is your unconditional love and support that have been the bedrock of my accomplishments.

Chapter 1

Introduction

1.1 Background and Motivations

Robotic manipulators, a cornerstone in the automation industry for decades, have been traditionally deployed in mass production environments, excelling in tasks like painting, welding, and machining due to their reliability and efficiency. These manipulators, however, face significant challenges in the emerging domains of customized production and household services. The inherent unpredictability and diversity of these unstructured environments render traditional, hand-engineered systems inadequate. Moreover, the demand for manipulators capable of performing multifaceted tasks, including unforeseen ones, necessitates a broader and more adaptable skill set.

Amidst these challenges, the field of robot learning has gained prominence, aiming to endow robots with a versatile array of skills that are both efficient and robust. The core of robot learning revolves around developing control policies – essentially, skills that enable a robot to execute appropriate actions based on observed system states to fulfill specific tasks.

There are multiple hierarchies to develop control policies, as shown in Fig. 1.1. Some researchers advocate for end-to-end learning to deduce direct control signals. However, this black-box approach often compromises reliability due to reduced system transparency and interoperability. Alternatively, object-centric planning methods prioritize generating object motions and subsequent robot controls in a sequential manner, simplifying the planning process and circumventing the need for robot-specific datasets. While these strategies, typically leveraging neural networks, have shown promise in handling complex tasks, they are often data-intensive and necessitate extensive exploration to tackle out-of-distribution scenarios.

In contrast, physics-inspired methodologies, grounded in long-established theorems, offer structured knowledge and explanations of system dynamics, prioritizing stability and safety. These mathematically derived controllers, however, struggle with high-dimensional problems due to the curse of dimensionality, often resulting in inefficiency.

This dissertation posits that integrating physics-inspired concepts with data-driven control policies represents a promising avenue for enhancing data efficiency in robotics. It

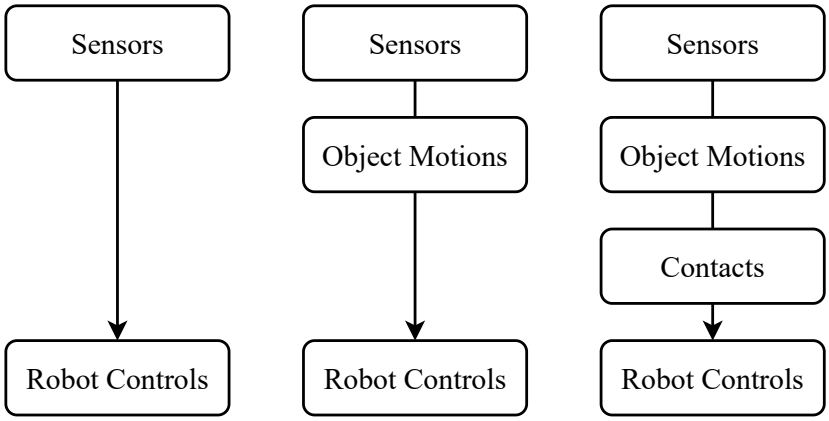


Figure 1.1: Hierarchies of control policies. Left: end-to-end robot manipulation. Middle: object-centric robot manipulation. Right: contact-aware robot manipulation.

focuses on incorporating contact models in robot manipulation learning. This approach eschews direct planning of atomic control signals from raw sensor data or predetermined object motions. Instead, it introduces a contact-aware methodology, incorporating additional contact planning for more nuanced control, as shown in Fig. 1.1.

Consider the task of in-hand object reorientation: The multi-fingered robotic hand must rotate the object, creating continuous yaw motions. The contact planning involves identifying feasible contact points on the object's surface and enabling the robot to apply contact forces for rotation. The robot then computes joint torque commands to reach these contact points and exert the necessary forces for object reorientation.

This contact-aware approach, distinct from other methods, seamlessly integrates analytical contact models into control policies. The inference of object motions remains consistent across various actuators and can be learned from extensive human demonstrations, such as those available on YouTube. The contact points and forces are efficiently determined from desired object motions through contact planning, exemplified by force closure grasps in robotic bin-picking or finger gaits for in-hand manipulation. The execution of these contacts leverages established algorithms in motion planning and optimal control.

Moreover, this dissertation acknowledges the crucial role of contact sensing in manipulation. Despite the prevalent use of force-torque sensors, recent advancements in vision-based tactile sensors offer heightened resolution and sensitivity, crucial for discerning subtle contact differences and shear forces. This research explores the synthesis of contact profiles from such sensors, enhancing feedback accuracy in contact-aware manipulations.

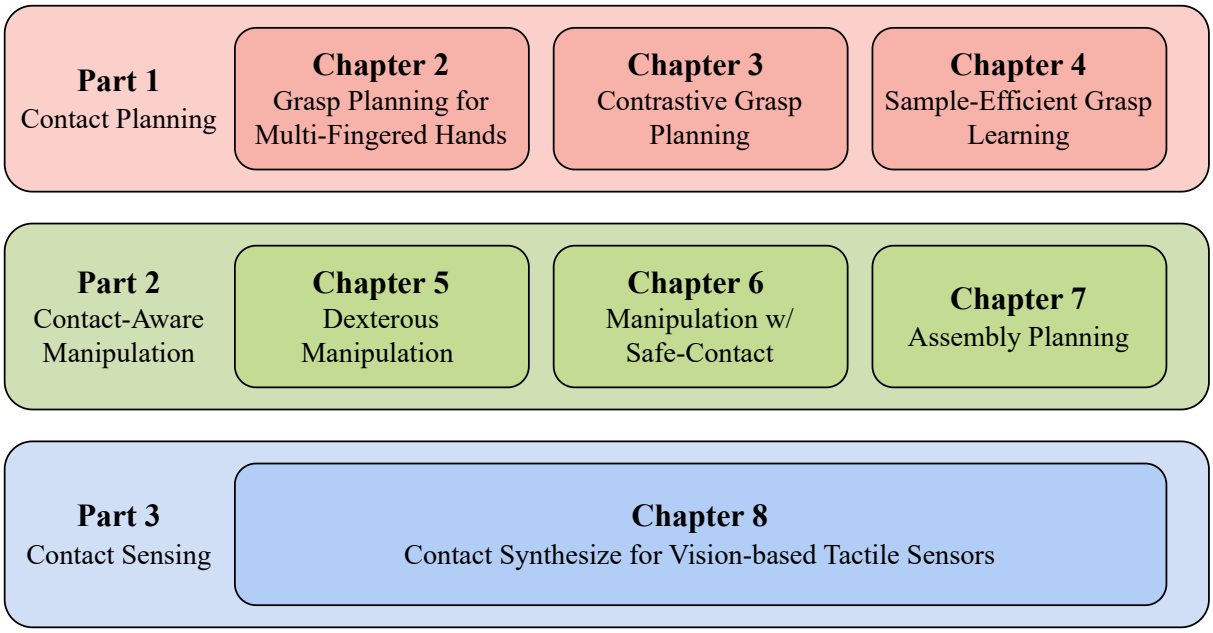


Figure 1.2: Structure of the dissertation.

1.2 Dissertation Outlines and Contributions

In general, the objective of this dissertation is to incorporate contact modeling in learning robot manipulations. Contact planning is first discussed and explored from Chapter 2 to Chapter 3, indicating the reasoning from object motions to contacts. Various scenarios of contact-aware manipulations are demonstrated from Chapter 5 to Chapter 7 with experiments on different manipulators and end-effectors. Contact sensing is illustrated in Chapter 8. Figure 1.2 shows the structure overview of the dissertation. The chapter outline is as follows.

Robust Grasp Planning with Contrastive Representation Learning

While the existing grasp planning algorithm demonstrates proficiency with high-quality observations, its performance is notably compromised under noisy visual observations. Such conditions are frequently encountered due to factors like robot vibration and camera miscalibration. To address this limitation, Chapter 2 introduces a novel contrastive grasp proposal network specifically designed to enhance the robustness of the grasp planner in the face of visual noise. This grasp planer is trained using a synthetic grasp dataset, employing contrastive learning techniques. A key feature of this training process is the deliberate augmentation of each dataset sample to replicate real-world camera noise and extract noise-invariant features. The result is a network capable of reliably identifying 6D collision-free grasps from a

single-view depth image. Rigorous experiments conducted with a physical robot validate its effectiveness, underscoring its potential to improve grasp planning reliability in real-world scenarios where visual noise is an unavoidable challenge. This research was published in [138].

Sample-Efficient Grasp Learning by Maximum Likelihood Sampling

The training of grasp planning networks, as explored in Chapters 2, typically demands a substantial volume of data, encompassing scene images and annotations of successful grasps. While simulation-based synthetic data is commonly employed in many studies, the intricate constraints of industrial bin-picking scenarios, such as specific grasp locations and workpiece-picking sequences, necessitate the involvement of human operators for annotating real-world successes. This requirement presents a significant challenge for efficient learning with minimal reliance on human-labeled data. Chapter 3 addresses this challenge by introducing a novel maximum likelihood grasp sampling loss. This method postulates that successful grasps can be viewed as samples drawn from a predicted grasp distribution. The primary objective is to maximize the likelihood of observing these successful grasps. Empirical results validate the efficacy of this method, demonstrating an 8-fold reduction in the need for training samples compared to existing approaches. Furthermore, the practical applicability of this methodology is confirmed through physical robot experiments, which yield a 90.7% success rate in grasp execution on household objects. This research has been published in [142].

Grasp Planning for Multi-Fingered Hands

The multi-fingered robotic hand, with its applications spanning industrial manufacturing and household services, presents a unique blend of opportunities and challenges. Its human-like morphology enables seamless integration into environments and tasks originally designed for human hands, including the use of conventional tools. However, this anthropomorphic design also introduces complexities in decision-making and control, attributed to its high degree of freedom. In addressing these complexities, Chapter 4 introduces an innovative grasp pose detection algorithm tailored for multi-fingered robotic hands. The algorithm takes a single-view point cloud of the scene as input, generates grasp candidates, and assesses them with an evaluation model. The top-ranked candidates undergo a local refinement process, enhancing their practicality by mitigating collision risks and bolstering robustness. The effectiveness of this approach is rigorously validated through extensive experiments utilizing the three-fingered Barrett Hand.

Robot Dexterous Manipulation by Model-Based Learning from Demonstrations

Chapters 2 to 4 advance the field of robotic manipulation by developing sophisticated grasping algorithms for both parallel jaw grippers and multi-fingered hands, focusing primarily on enhancing kinematic dexterity. However, the scope of broad-scale robotic manipulation extends beyond grasping to encompass post-grasp skills. In order to obtain extensive manipulation skills efficiently, Chapter 5 introduces an innovative Learning from Demonstration (LfD) framework. This framework enables robots to acquire manipulation skills by observing human-performed actions in videos, thereby circumventing the need for labor-intensive reward design. The proposed LfD framework is underpinned by two pivotal components. The first is a self-supervised pose and shape estimation module that employs differentiable rendering techniques. The second component involves the generation of contact sequences, achieved through an iterative optimization process that uses differentiable simulation to refine contact points and forces. Empirical experiments show the remarkable capability of the LfD framework, object tracking, and contact sequence inference. Notably, the framework maintains its performance even in environments with significant noise, underscoring its practical applicability in real-world scenarios. This research was published in [140] as an oral presentation.

Manipulation with Safe-Contact by Null Space Impedance Control

While Chapter 5 and preceding efforts have made significant strides in developing manipulation planning algorithms, they predominantly operate under the paradigm of avoiding contact between robot arms and their environment. This approach, however, diverges from the intricacies of human manipulation, where contact with obstacles is not only common but often strategically utilized. Chapter 6 marks a paradigm shift in this domain, delving into the potential benefits of incorporating safe contact within robotic manipulation. This chapter introduces an approach that advocates for the generation and tracking of compliance reference signals in the operational and null spaces of the robot. This dual-space consideration is pivotal for enhancing the safety and efficiency of robotic manipulation in contact-rich environments. To optimize trajectories where collisions are permitted, the research proposes a hybrid solver, combining the strengths of sampling- and gradient-based methods. The practical effectiveness of this method is evaluated through a series of goal-reaching tasks set in five distinct environments, both simulated and real-world, each presenting different collisional challenges. The results from these evaluations demonstrate that enabling safe contact not only improves the efficiency of goal-reaching tasks but also provides viable solutions in scenarios heavily constrained by collisions. Furthermore, the inclusion of null space planning alongside operational space planning emerges as a key factor in enhancing the safety of the generated trajectories. The findings of this study can be found in [139].

Contact-Aware Robotic Assembly Planning

Beyond the primitive manipulation tasks explored in previous chapters, Chapter 7 addresses the complex and industrially significant challenge of automating the assembly process of objects from their individual components. This task, with its vast applications in manufacturing, maintenance, and recycling, requires a nuanced approach that goes beyond the scope of existing research, which predominantly focuses on target segmentation, pose regression, or the utilization of fixed target blueprints. This chapter presents a holistic multi-level framework that integrates part assembly sequence inference with advanced part motion planning and robot contact optimization techniques developed in Chapters 6 and 5, respectively. Central to this framework is the introduction of the Part Assembly Sequence Transformer (PAST), an innovative sequence-to-sequence neural network designed to recursively infer assembly sequences from a given target blueprint. Following sequence inference, the framework employs a motion planner and contact optimization algorithms to generate the movements and contacts necessary for assembling the parts. To facilitate the training of PAST, Chapter 7 introduces D4PAS, a large-scale Dataset for Part Assembly Sequences. This dataset is unique in its provision of physically valid assembly sequences for a variety of industrial objects. Empirical evaluations of the proposed framework underscore its enhanced generalization capabilities and reduced computational burden compared to previous works. This research has been published in [144].

Contact Synthesize for Tactile Sensors by Graph Neural Network

In the progression from Chapter 5 to Chapter 7, where contact planning plays a pivotal role in robotic manipulations, Chapter 8 introduces an innovative approach to contact synthesis using vision-based tactile sensors. These sensors, characterized by a deformable elastomer and an overhead camera, provide high-resolution visual observations of contact events. A critical advancement in utilizing these sensors is the accurate reconstruction of volumetric meshes corresponding to the elastomer's deformation, which offers direct contact information crucial for subsequent manipulation tasks. Chapter 8 is dedicated to the development of a method for synthesizing these volumetric meshes from the sensor's image imprints. This process involves collecting synthetic image-mesh pairs generated using 3D finite element methods (FEM) alongside real-world images captured from physical sensors. To learn the complex image-to-mesh mappings, the chapter introduces graph neural networks (GNN) trained through supervised learning using synthetic and real-world data. Recognizing the challenges in transferring theoretical models to practical applications, the chapter proposes a self-supervised adaptation method and image augmentation techniques. These strategies are essential for adapting the network from simulated to real-world settings, enabling it to handle a diverse range of contact scenarios and different sensor types. The findings of this study appear in [143].

Part I

Contact Planning

Chapter 2

Robust Grasp Planning with Contrastive Representation Learning

2.1 Introduction

Robotic grasping in unstructured environments can benefit applications in manufacturing, retail, service, and warehousing. Grasping unseen objects is, however, highly challenging due to the limitations in perceptions. When objects are cluttered in a bin, the exact geometry and position of objects are obscured. Sensing imprecision and deficiency then leads to poor grasp planning execution.

Model-based and learning-based methods could be used to plan grasps across a wide variety of objects. Existing physical grasp analysis techniques, such as grasp quality metrics [100], template matching [109], and wrench space analysis [77], can be used to search for the optimal grasp. These approaches, however, can be less robust in practice due to perception limitations. Incompletion of the object surface can lead to flawed analysis. An alternative approach is to plan grasps with supervised deep learning. Current methods show that it is preferable to learn to grasp quality functions and optimize them at the runtime [68, 67, 75, 76, 91, 26, 29, 27, 57]. Learning intermediary information, such as grasp qualities and success rate, can improve training efficiency and prediction accuracy. However, the requirement of the sampling or optimization makes the algorithm time-consuming. To tackle this, other methods use end-to-end learning to infer grasp poses from the sensor inputs directly [82, 110]. Nevertheless, these algorithms require larger datasets and elaborate hyperparameter tuning to reduce the training variance.

Ideally, the grasp planning algorithm generates 6 degrees of freedom (DoF) grasps. Previous works have proposed models that can detect top-down grasps using depth images [68, 44, 90]. However, the top-down nature of such grasps does not allow robots to pick up objects from different orientations, which limits its application in cluttered environments. In this chapter, we propose to use six variables $(x, y, \theta, \gamma, z, \beta)$ to represent a 6-dimensional grasp in a depth image, as shown in Fig. 2.1. The planar pose (x, y, θ) in Fig. 2.1(a) is the

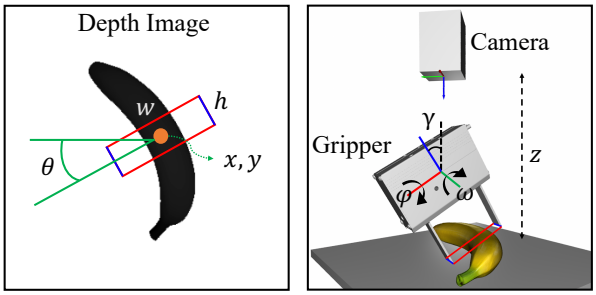


Figure 2.1: Grasp Representation $(x, y, \theta, \gamma, z, \beta)$. The planar pose (x, y, θ) in (a) represents the center position and orientation of the projected bounding box on the camera plane. The bounding box's width w and height h are used in training but not in representing grasps. The other 3 grasp parameters are shown in (b): tilt angle γ is the rotation among axis ω, z is the depth of grasp, and gripper angle β is the rotation among the grasp axis φ.

center position and orientation of the bounding box in the image plane. The bounding box's width w and height h are used in training but not in representing grasps. Three spatial grasp parameters shown in Fig. 2.1(b) are the tilt angle γ among axis ω, the rotation angle β among the grasp axis φ, and the depth of the grasp z.

In previous works [68, 67, 75, 76], the strategy to train on synthetic datasets and apply them to reality has been heavily used. It has been shown that rendered images with numerically computed grasp qualities can ease the data preparation process. The simulation-to-real (sim-to-real) gap, however, is still an open problem for grasp planning. Synthetic data have better resolution and less noise than real images. To tackle this problem, we introduce contrastive learning with sim-to-real-depth image processing in this chapter. Contrastive learning aims to extract invariant features from augmented images, which improves the overall performance of modeling under vision noise.

In this chapter, we propose an end-to-end 6-DoF contrastive grasp proposal network (CGPN). The general framework is shown in Fig 2.2. When an object is presented in the scene, a stereo camera captures a depth image; CGPN rapidly generates 6-DoF robust grasps, which are executed with the Fanuc robot. CGPN is trained with synthetic grasp images with variant augmentation techniques to bridge the sim-to-real gap.

2.2 6-DoF Contrastive Grasp Proposal Network

General Framework

This section introduces the framework of the 6-DoF contrastive grasp proposal network, as shown in Fig. 2.3. The input to the whole pipeline is a single-view depth image of the scene.

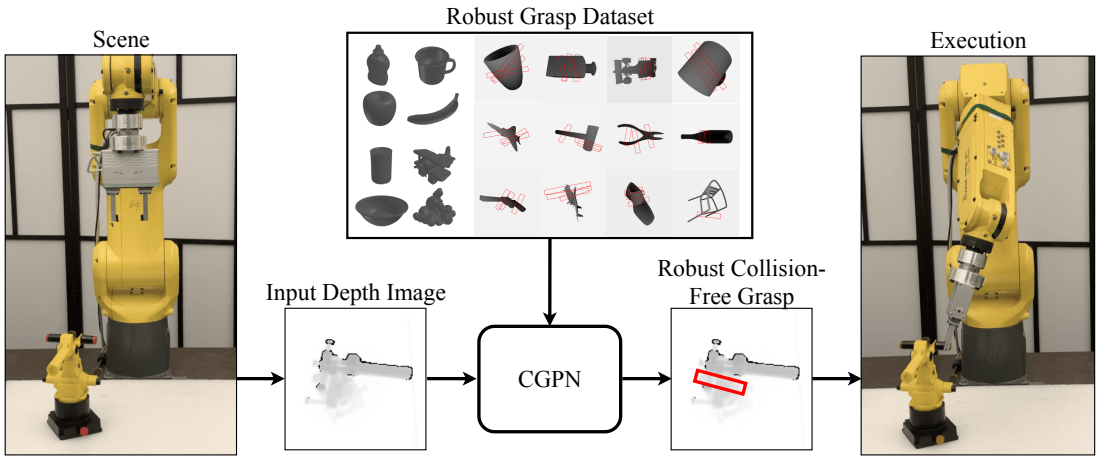


Figure 2.2: CGPN Architecture. The 6-DoF contrastive grasp proposal network (CGPN) is trained offline to infer grasps from depth images using a dataset of synthetic images and grasps. When an object is presented to the robot, a stereo camera captures a depth image; CGPN could rapidly generate 6-DoF robust collision-free grasps, which is executed with the Fanuc robot.

Segmentation is first applied to the image to separate objects. The CGPN model generates grasps for the object using its segmented depth image.

Training phase. During training, the CGPN algorithm works as follows. First, two separate data augmentation operators t, t' are sampled from the augmentation family $\mathcal{T}$. Augmentations are applied to the segmented depth image to obtain two correlated views x_q, x_k regarded as a positive pair in the contrastive learning module. Similar to [39], the other inputs in the same batch are viewed as negative samples. Second, the positive and negative samples are fed into the contrastive encoder. We get query q from x_q with query encoder and keys $\{k_i\}_1^N$ from x_k and other negative samples with key encoder. The key encoder is a slowly updated query encoder with no gradient update. Contrastive loss is calculated using these queries and keys, with more information introduced in the loss section. The purpose of this module is to maximize the agreement of encoded feature maps q, k. We assume the encoder can learn invariant representations of the depth images under different augmentation operations, and the query q is the feature we use for downstream tasks. Third, a rotated region proposal network (RRPN) is introduced to propose 3-DoF grasp regions based on the feature map q. The output of RRPN determines 3-DoF grasp (x, y, θ) and a box shape (w, h). We still need local features to determine other grasp parameters. Fourth, a rotated region pooling (RRPooling) module extracts feature vectors from the feature map q using the predicted rotated bounding box from RRPN. These local feature maps contain depth and other features around the proposed grasp position. Finally, a grasp refinement network (GRN) is designed to infer the tilt angle γ and the depth z using the local feature vectors.

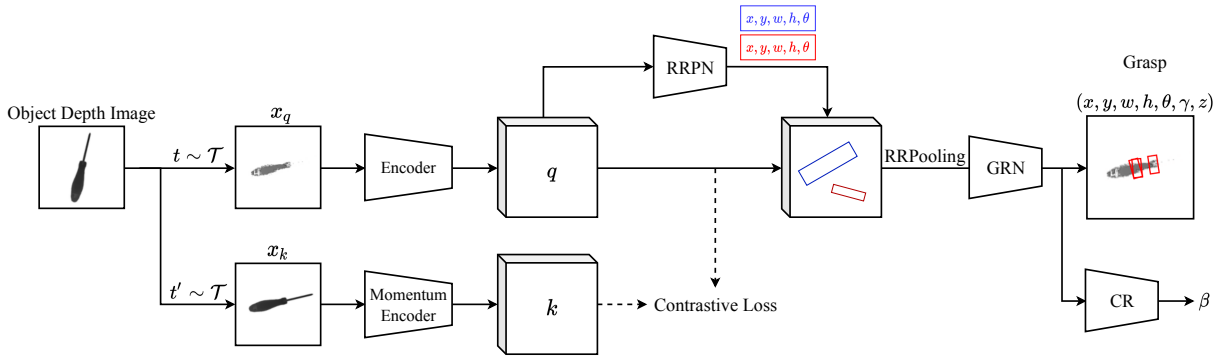


Figure 2.3: CGPN Network Architecture. During the training, a synthetic depth image of the object is fed into the network. First, two separate data augmentation operators t, t' are applied to the segmented depth image to obtain x_q, x_k, which are then input to the contrastive encoders. Second, extracted feature maps q, k are parsed to the rotated region proposal network (RRPN) to generate grasp regions. Third, a rotated region pooling (RRPooling) module extracts feature vectors from the feature map q using the generated grasp regions. Finally, a grasp refinement network (GRN) infers the tilt angle γ and the depth z using the local feature vectors. A collision refinement is further added to search the rotation angle β.

This completes a 5-DoF grasp pose together with (x, y, θ). The last DoF β is searched with a collision refinement (CR) module and thus is neglected in the training process. The training loss is the weighted combination of contrastive loss, region proposal loss, and grasp refinement loss. After training, we get a query encoder, an RRPN module, and a GRN module used for online testing.

Testing phase. During the testing phase, we do not need to create positive pairs for the contrastive modules. Instead, we directly put the input depth image into the query encoder. The following process is the same as training. The output of the CGPN model is valid grasp poses for each object. After that, the 5-DoF grasps for each object are projected back to the original cluttered scene. The last DoF β is determined according to the collision constraints.

Data Augmentation and Contrastive Learning

The instability in real-world grasping is usually brought by occlusion in cluttered scenes, background noise in the environment, and the sim-to-real gap. To overcome this, we design multiple data augmentation operations $\mathcal{T}$ for the input depth image.

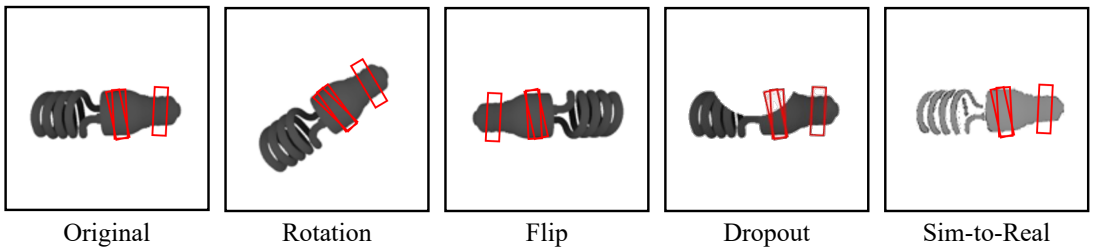


Figure 2.4: Illustrations of the available data augmentation operators in $\mathcal{T}$. Each augmentation can transform the original input with some internal parameters (e.g. rotation degree, flip axis). We use a combination of the first four operations and the sim-to-real process as a complete augmentation for a single image.

Spatial/geometric transformation. Spatial operations include rotation, flip, and dropout on the original synthetic depth image. Each augmentation can transform the input image stochastically with some internal parameters. We use a composition of these operations to create different observations for the same grasp object. Note that, unlike classification labels in vision, the ground-truth high-quality grasp poses would change along with these augmentation operations.

Sim-to-real processing. Similar to [114], we design a sim-to-real transfer operation by leveraging several image processing techniques. We randomly paint the pixels black in areas of high Laplacian gradient and edges of the object detected by Canny edge detector[2]. Then we inject realistic noise into the image. The operation is parameterized by a threshold of black painting areas. Compared to spatial transforms, the sim-to-real process does not change the ground-truth pose in the image.

Illustrations of the data augmentation operators are shown in Figure 2.4. Using the introduced operations, we extend the training dataset and create positive pairs from the same input depth image for the contrastive training process. The transformation family $\mathcal{T}$ is created by a random combination of the spatial operations and the sim-to-real transfer as the last operation. All of the operations have an execution probability. In this way, we can train on synthetic data generated in the simulator and use the model to predict valid grasp in real-world scenarios.

Regarding the contrastive learning module in the pipeline, as described in previous sections, we use separate encoders for query input x_q and other positive or negative samples as suggested in [39]. Note that some operations in the data augmentation change the ground truth in the downstream grasping tasks (e.g. spatial operations), directly maximizing the similarity between q and k^+ might not be suitable for grasping regression. As suggested in [12], we add a multi-layer perceptron (MLP) after the query and key encoders as a projecting head and get $\hat{q}, \hat{k_i}$. This is neglected in the network architecture but implemented in experiments. The query and key vector q, k_i can keep the differences in grasping, but

the projecting head would catch the invariant properties among positive pairs. We use q for downstream grasping tasks but use $\hat{q}$ and $\hat{k_i}$ for calculating contrastive loss.

Rotated Region Proposal

Another core part of our architecture is the rotated region proposal network. Similar to the architecture in [66], which output detected a bounding box for scene text, our network receives feature maps from the learned contrastive encoder and output candidate region proposals with class labels and parameters of the positions. The class label determines if the proposal fits any robust grasp and the parameters (x, y, w, h, θ) gives a rotated bounding box.

Unlike text detection, we do not have a fixed number of non-overlapping labeled bounding boxes for each input image. For each object and its depth image, the high-quality grasp poses may overlap with others. When matching the ground-truth grasps with proposed regions, overlapped grasps may have similar skew intersection over union (IoU) results. Directly choosing the one with the highest IoU may cause all proposals to match the "largest" grasp (i.e., large w, h). To avoid such mode collapse, we introduce random selection among top-k grasps based on IoU during matching.

For parameters of the proposed region, although only (x, y, θ) is used in a grasp pose, the box's width w and height h indicate the range of local features used in the following GRN. The GRN model first aligns, and extracts rotated regions of interest by projecting proposals from the RRPN onto the feature map and then uses the local features to predict the tilt angle and relative depth of the grasp, which finalizes the 5-DoF grasp $(x, y, \theta, \gamma, z)$.

Collision Refinement

The grasp proposal models mentioned above are used for 5-DoF grasps for a single object. To use the proposed grasps in a cluttered scene, we use collision constraints to infer all 6 DoFs. The proposed 5-DoF grasp $g = (x, y, \theta, \gamma, z) \in \mathcal{G}$ are frozen, and a posterior optimization process is used to search for the last rotation angle β:

$$\min_{\beta} \quad \sum_{i}^{N} C(g, \beta, x_i) \tag{2.1}$$

where C is the collision check score for a 6-DoF grasp (g, β) and surrounding objects. $\{x_i\}_{i=1}^{N}$ is the segmented depth image for the N object in the scene.

Sign-distance field is introduced in this chapter to model the collision score:

$$C(g, \beta, x_i) = -SD(FK(g, \beta), x_i) \tag{2.2}$$

where $FK(\cdot)$ denotes the forward kinematics function of the robot. $SD(\cdot)$ denotes the signed-distance function of the robot and the object x_i. Given a 5-DoF grasp, there are

infinitely many grasp candidates since the rotation among the grasp axis φ is free-floating. By minimizing the negative signed distance between the robot and the object, a unique 6-DoF grasp can be determined.

Loss Design

Region proposal loss. Despite the randomness we introduced in positive region matching, most of the models we use in RRPN are similar to [66]. The loss function for the proposal takes the form of:

$$L_p = -\log s_{\text{pos}} + \sum_{v \in \{x,y,\theta,h,w\}} \lambda_i \text{smooth}_{L_1}(v^* - v) \tag{2.3}$$

where s_{pos} is the matching IoU for positive pairs, v^* is the ground-truth value for corresponding variable. The smoothed L_1 loss is:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{2.4}$$

The output (x, y, θ) is the grasp parameter, and (w, h) are parameters for local feature ranges in the original feature map. Therefore we assign more weight on λ_x, λ_y and λ_θ and less on λ_w, λ_h.

Grasp refinement loss. To find the best grasp pose, we need not only the rotated planar grasp position proposed by RRPN but also the tilt angle γ and depth z of grasp. While RRPN gives us a rough estimation of grasp positions, GRN uses the proposed region of interest (ROIs) to generate accurate grasp poses with local features. We formulate a weighted refinement loss in (2.5) to minimize the L1 error of tilt and depth.

$$L_r = \lambda_\gamma \|\hat{\gamma} - \gamma\|_1 + \lambda_z \|\hat{z} - z\|_1 \tag{2.5}$$

Contrastive loss. Unlike regression loss, the contrastive loss does not have supervised signals. For simplicity, we use q, k_i to represent $\hat{q}, \hat{k_i}$ after the projecting head. Consider a query q encoded from sample x and a dictionary of N keys $\{k_1, k_2, ..., k_N\}$ encoded from different samples. Among all keys, there is one positive key k^+ encoded from x_k similar to x_q and $N-1$ negative keys from other samples in the batch. Using dot product as the similarity metric, the loss function is defined as (2.6), where τ is a temperature hyper-parameter.

$$L_q = -\log \frac{\exp(q \cdot k^+)}{\sum_{i=1}^{N} \exp(q \cdot k_i)} \tag{2.6}$$

This is the log loss of a N-way softmax-based classifier that tries to classify q as k^+, introduced as InfoNCE in [85].

The overall loss function used to train the contrastive grasp proposal model is:

$$L_{\text{overall}} = \lambda_p L_p + \lambda_r L_r + \lambda_q L_q \tag{2.7}$$

We adjust the relative weight of the three losses at different stages of training. In the beginning, we set the loss of RRPN and GRN low to reduce the contrastive loss. After we get a stable encoder, we increase the weight of RRPN loss but keep the GRN loss weight low to train the 3-DoF grasp position and local feature bounding box. We then gradually increase the weight of GRN to train the overall pipeline for the final 5-DoF grasp. The high weight of GRN loss at an early stage would affect RRPN's training since the models work in series.

2.3 Experiment

Dataset Generation

To generate the grasp sets, we need to sample a large number of grasps for single objects and label the top-ranked robust grasps as ground truths. This is unrealistic for real robots but not hard in simulation environments. We train our CGPN model on the generated single object grasp dataset and use it on real robots. Similar to [68], 1,366 objects are selected from the 3DNet [129] as the object set. 100 antipodal grasps are evenly sampled among the surface of each object. Each grasp is labeled with the robust force closure metric and is represented by its contact points (c_1, c_2) in 3D. Since the CGPN algorithm requires depth images as input, objects, and grasps are projected to the image plane. For each selected object, 20 synthetic depth images are rendered from different angles. The object is placed at the center of a regular icosahedron; cameras are placed at each face's center and point to the origin. The distance between the camera and the object is sampled from $\mathcal{U}(\sqrt{3}r_{obj}, 2r_{obj})$, where $\mathcal{U}$ denotes the uniform distribution and r_{obj} is the object bounding ball's radiance. Such selection makes sure that the full object is visible in the camera.

Each grasp (c_1, c_2) is then projected to the depth image using projective transformations:

$$\begin{bmatrix} u_i & v_i & f \end{bmatrix}^T = K \cdot \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} X_i & Y_i & Z_i & 1 \end{bmatrix}^T \tag{2.8}$$

where X_i, Y_i, Z_i are positions of the contact point $\{c_i\}_{i=1}^2$ in the camera frame, $K, \begin{bmatrix} R & t \end{bmatrix}$ are the camera's intrinsic and extrinsic matrices, respectively. u_i, v_i are pixel's locations at the image for point c_i. The bounding box's other parameters are then computed. The width w and the height h of the box is set to $\|[u_2 - u_1, v_2 - v_1]\|_2$ and 20 respectively. Grasp depth z, rotation angle θ, and tilt angle γ are computed as $z = \frac{1}{2}(Z_1 + Z_2)$, $\theta = \tan^{-1}\frac{u_2 - u_1}{v_2 - v_1}$ and $\gamma = \tan^{-1}\frac{Z_2 - Z_1}{w}$ respectively.

Ground truth grasps for each image are selected as the top 20% from 100 samples. In this chapter, we limit the tilt angle's range to $[-30°, 30°]$. To give rotation angle θ and tilt angle γ unique definition, we set constraints on the grasp points in the image plane, such

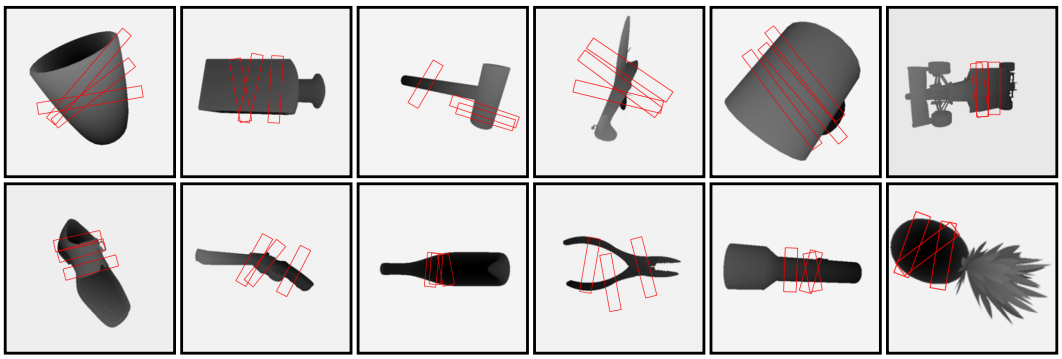


Figure 2.5: Dataset Samples. This figure shows 12 samples from the generated grasp dataset. 3D grasps are projected to the image plane (red rectangles). The tilt angle γ and the distance z are neglected in the plot for simplicity.

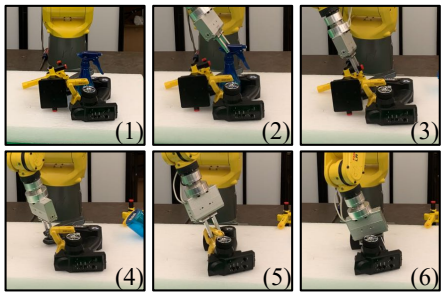


Figure 2.6: (1-6) shows a sequence of proposed grasps in a cluttered scene.

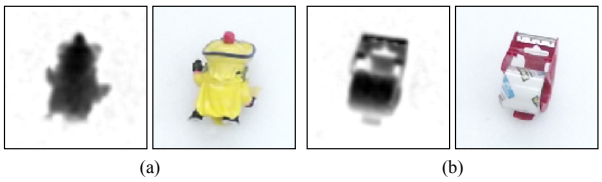


Figure 2.7: Two failure modes of CGPN. (a) shows the unmodeled sim-to-real gap on the object's surface, and (b) shows the limitation of the depth image in representing 6-DoF grasps.

that $v_2 > v_1$. In other words, the point $[u_2, v_2]^T$ is always on the right of the point $[u_1, v_1]^T$. Then, we bound θ in the range of $[-90°, 90°]$, and γ is relabeled with the corresponding sign. The generated training dataset has 24,723 depth images with labeled ground truth grasps. Fig. 2.5 shows some samples from the dataset.

Experiment Results

The proposed CGPN is run on a desktop with GTX2080Ti GPU, 32GB RAM, and a 4.0GHz CPU. For the experiment, we use a FANUC LRMate 200iD/7L industrial manipulator with a SMC LEHF20K2-48-R36N3D parallel-jaw gripper for grasping. A Kinect v2 camera is used to capture depth images of the scene. The point cloud library [103] implementation of the region growing method is utilized to pre-process and segment the object.

We leverage state-of-the-art model architectures for each submodel. ResNet-50 [38] and rotated region proposal networks [66] are utilized as the encoder and downstream models.

Table 2.1: Performance analysis of the CGPN and baselines on single object grasping tasks.

	Success Rate	Time (sec/grasp)
GPD	72.2%	1.84
CGPN w/o Contrastive	69.4%	0.46
CGPN w/o Data Augmentation	66.7%	0.46
CGPN	**75.0%**	**0.46**

The hyper-parameter for RRPN and contrastive learning are mostly the same as introduced in [66, 39] and $\lambda_x = \lambda_y = \lambda_\theta = 5$ and $\lambda_w = \lambda_h = 1$. We design the anchor aspect ratio as $[0.5, 2]$ to fit to grasp poses better. During the first 20 epochs, $\lambda_p, \lambda_r, \lambda_q$ take values of $(1, 1, 5)$ respectively. After that, they are modified to $(5, 5, 2)$ to stabilize the training.

As introduced above, the distance between the object and the camera is drawn from $\mathcal{U}(\sqrt{3}r_{obj}, 2r_{obj})$. In reality, such a condition is hard to achieve since the camera is usually fixed at a particular point. To tackle this, we propose to re-project the depth image into a virtual camera. The object's point cloud is first generated based on the depth image. r_{obj} is then computed as the radiance of the point cloud's bounding ball. Next, the virtual camera's position is determined by the sampled camera-object distance. Finally, the generated point cloud is projected to the virtual camera to obtain the normalized depth image.

Fig. 2.8 shows the grasp planning and grasp execution results on 12 different objects with a single stereo camera. The top two rows of the figure show the captured RGB-Depth images. Located grasps are labeled in the depth image with red rectangles. The tilt angle γ and the distance z are neglected in the plot. The physical grasp poses and the execution result of the planned grasp are shown in the bottom two rows. The algorithm is able to find robust grasps for a) small objects close to the ground, b) large objects with graspable regions, and c) objects with complex surfaces.

Table 2.1 compares CGPN with the grasp pose detection (GPD) [91] algorithm, which also focuses on 6-DoF grasp planning with a single camera. To adopt GPD, we train a point-cloud-based grasp evaluation network with the same dataset as CGPN. Each object is grasped three times, resulting in 36 trials for each algorithm. From experiments, we observe that CGPN outperforms GPD in both the grasp success rate and the computation time. One reason behind this might be the robustness of our model under the various camera angles we set during the experiment. Regarding time cost, CGPN generates valid grasps in an end-to-end manner, it does not require the sampling and evaluation procedure and therefore takes less time to plan a grasp.

For the ablation study on the data augmentation and contrastive learning module, Table 2.1 also compares the effectiveness of adding augmentation operations on training data and using contrastive loss in the sense of object grasp success rate. As can be seen, the sim-to-real gap significantly affects the performance of the grasp proposal network. Ignoring vision noises and training on the synthetic dataset may yield poor results in practice.

Figure 2.6 shows a sequence of proposed grasps in a cluttered environment. The grasp

sequence is selected according to the graspability, or whether a collision-free grasp exists for a particular object.

Figure 2.7 displays two typical failures for CGPN, in which no grasp is proposed. The first failure mode occurs because of the unmodeled sim-to-real gap. The object surface's resolution and distance noise are not appropriately handled. The second type of failure occurs when no robust grasp exists with $\gamma \in [-30°, 30°]$. Compared to 3D representations, the image includes less geometric information, making the end-to-end model hard to infer 6-DoF grasp. It appears that the performance could be improved with comprehensive data augmentations and other grasp representations.

2.4 Chapter Summary

This chapter presents a 6-DoF contrastive grasp proposal network (CGPN) to generate robust grasps on single-view depth images. CGPN is composed of a grasp planning module for 6-DoF grasp detection and a contrastive learning module for sim-to-real gap reduction. The grasp planning module infers 6-DoF grasps based on detected robust grasp regions. An image encoder is used to extract the feature map, followed by a rotated region proposal network to propose planar grasps. Feature vectors are then extracted and refined to 6-DoF grasps. To transfer grasp skill trained in simulation, a contrastive learning module and variant depth image processing techniques are introduced during the training. CGPN can locate 6-DoF collision-free grasps using a single-view depth image within 0.5 seconds. Experiment results show that CGPN outperforms previous grasping algorithms. The experimental videos are available at [120].

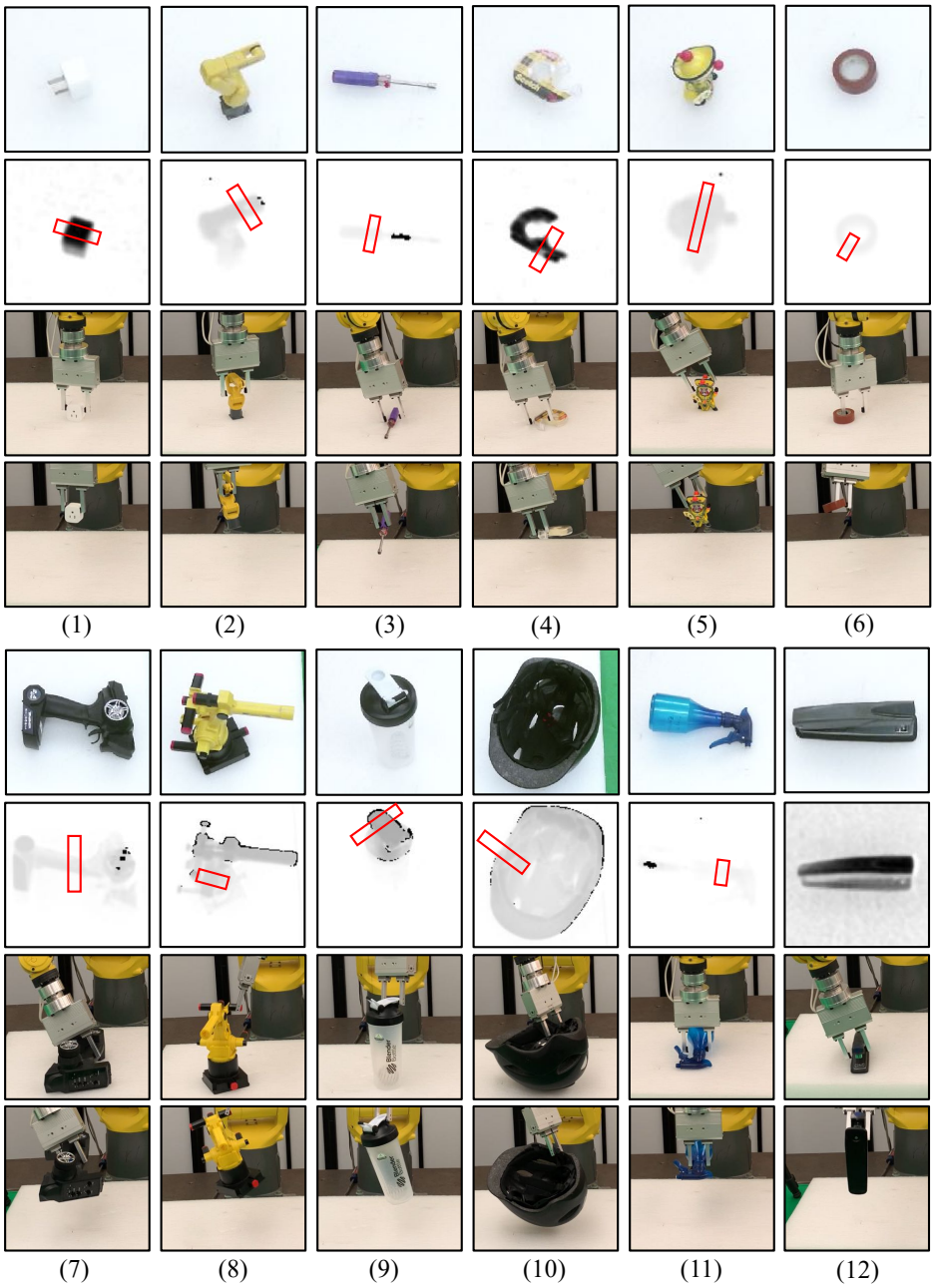


Figure 2.8: (1-12) The grasp planning and execution results on 12 objects with a single depth image. For each column, the top two rows show perceived RGB and depth images with planned grasps, the third row shows physical grasps reaching the target grasp, and the bottom row shows the execution results. The tilt angle γ and the distance z are neglected in the plot for simplicity.

Chapter 3

Sample-Efficient Grasp Learning by Maximum Likelihood Sampling

3.1 Introduction

In Chapter 2, we introduce a novel approach to grasp planning utilizing contrastive representation learning. This method notably enhances both robustness and success rates in grasp executions. However, a significant drawback is its dependency on extensive training data. This requirement is not unique to our approach but is a common characteristic shared by other supervised learning methods in grasp planning, as evidenced in studies such as [68, 67, 75, 76, 110, 73, 74, 134, 107]. These approaches necessitate large datasets comprising sensor inputs and corresponding grasp annotations, which can be either synthesized[68, 67] or collected empirically [75, 76, 20, 52].

A notable challenge in end-to-end training models is the requirement for densely labeled ground truth samples, a condition scarcely met by existing datasets [20, 52]. Addressing the issue of label scarcity, some researchers have proposed a 'center-third grasp generation' technique [73, 74]. This method operates under the assumption that grasp positions proximate to successful labels are inherently robust, thereby incorporating these near-miss grasps into the ground truth. Furthermore, it presumes that areas without labels are invalid for grasping. However, this technique may inadvertently introduce inaccuracies. On one hand, it risks creating false positives, where generated grasps, despite being near successful labels, may actually be unstable. On the other hand, it may lead to false negatives, overlooking robust grasps that do not align closely with existing labels. This problem is exacerbated in datasets with a limited number of grasp labels, thus raising concerns about the reliability of the generated labels.

This chapter proposes a maximum likelihood grasp sampling loss (MLGSL) to improve the data efficiency in training generative grasp planners. The proposed method recovers dense ground-truth grasp distributions from sparse labels. We use this method to train an efficient grasping model based on Fully Convolutional Networks (FCN) that has recently

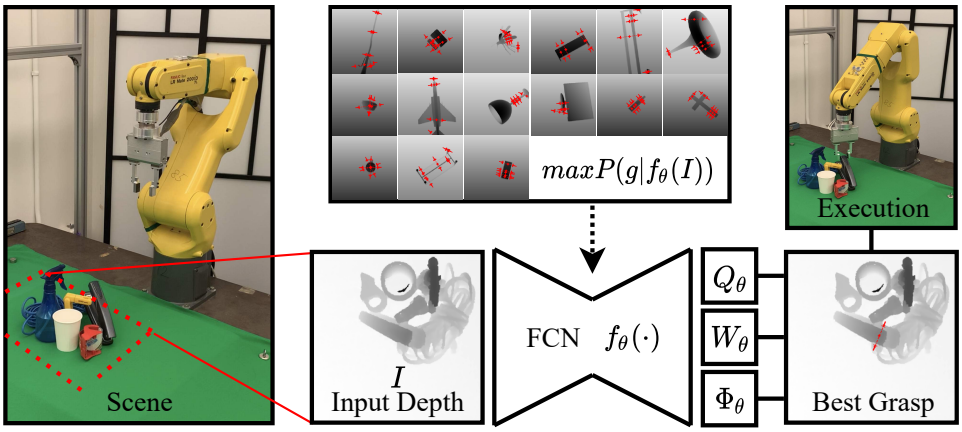


Figure 3.1: Grasp planning and execution pipeline. When an object is presented in the workspace, a stereo camera captures a depth image; a trained generative model $f_\theta(\cdot)$ rapidly computes grasp configuration maps Q_θ, W_θ, and Φ_θ. The best grasp is generated based on configuration maps and executed with the robot manipulator. The grasp model is trained offline with empirical datasets and proposed maximum likelihood grasp sampling loss.

shown promising results for learning to grasp [73, 74, 107]. We develop a novel variant of the proposed loss and model architecture that predicts planar grasps with a single-view depth image. To improve the models' collision avoidance ability, we construct a cluttered dataset consisting of multiple-object scenes and collision-free success grasp labels.

We evaluate the proposed loss in training and physical experiments. Fig. 3.1 shows the experimental grasp planning pipeline. Training results demonstrate that models based on this loss can learn to grasp with datasets consisting of two labels per image, which implies that it is 8× more data-efficient than previous methods [73, 74]. Meanwhile, physical experiments show a similar grasp success rate in single and cluttered scenes.

3.2 Grasp Planning with Maximum Likelihood Grasp Sampling Loss

This section first introduces notations and states the problem. The proposed loss function is then illustrated and compared with previous works. Finally, the network architectures and training datasets are presented.

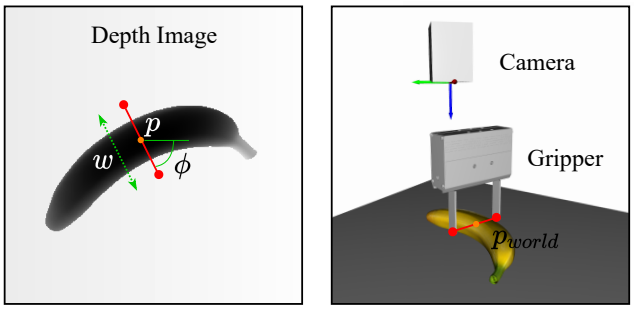


Figure 3.2: Grasp Representation $g = (p, \phi, w)$. The planar pose (p, ϕ, w) in (a) represents the grasp's centre position, orientation, and width in the image plane. Grasp g in (a) has quality $q = 1$ since it is labeled success. g is executed perpendicular to the image plane at point p_{world} in the Cartesian frame, as shown in (b), where p_{world} is p in the world frame. The gripper moves ϵ cm below p_{world} in the direction of the camera's z-axis, shown by the blue arrow.

Notations

As in previous literature, the grasp planning problem is detecting a grasp configuration that allows the robot to pick up objects. Moreover, no explicit knowledge of the object is given beyond camera readings.

Grasp. Let $I \in \mathbb{R}^{H \times W}$ define a given depth image with height H and width W. The i-th grasp is defined in the image I and denoted by

$$g_i = (p_i, \phi_i, w_i) \tag{3.1}$$

where $p_i = (u_i, v_i)$ is a pixel in the image representing the grasp centre. $\phi_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is the gripper's rotation, and $w_i \in [0, 150]$ is the gripper's width in the image frame (Fig. 3.2). Each grasp has a quality measurement $q_i \in [0, 1]$ indicates the success rate. Each grasp is executed perpendicular to the image plane at point $p_{world,i}$ in the Cartesian frame, where $p_{world,i}$ is computed by transforming p_i to the world frame with the object's surface distance. During grasp executions, the gripper attempts to move ϵcm below $p_{world,i}$ in the direction of the camera's z-axis.

Grasp configuration maps. Similar to [73], instead of selecting grasps at specific pixels to evaluate, we compute a grasp for each pixel of I, which results in dense grasp configuration maps $G = (Q, \Phi, W) \in \mathbb{R}^{3 \times H \times W}$. In other words, Q, Φ, W contain values of q_i, ϕ_i, w_i at each pixel of I. Because antipodal grasps are symmetric, we use two components $\Phi_s = \sin(2\Phi)$

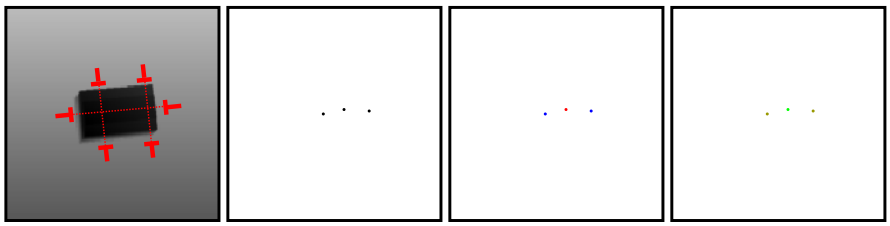


Figure 3.3: Dataset sample. (a) shows the input depth image I with $k = 3$ success grasp labels (red). (b-d) show transferred grasp configuration maps $Q_{label}, \Phi_{label}, W_{label}$ respectively. Each colored pixel represents a successful grasp label $g_{label,i}$ with different colors for different values, while white pixels mean there is no label that exists. Note that scarce labels exist in this sample, referring to sparse label maps.

and $\Phi_c = \cos(2\Phi)$ to resolve the ambiguity as suggested in [73]. The following sections use Φ to represent angle maps for simplicity.

Grasp planning models. We assume there exists a ground-truth grasp $g_{GT,i}$ of quality $q_{GT,i}$ at each pixel of I, and $g_{GT} = \{g_{GT,i}\}$ for $i \in [1, ..., H \times W]$, which can be transferred to dense configuration maps $G_{GT} = (Q_{GT}, \Phi_{GT}, W_{GT})$. Ground-truth configuration maps G_{GT} are approximated by a grasp model $f_\theta(\cdot)$, with θ being the parameters of the model. The model predicts $G_\theta = (Q_\theta, \Phi_\theta, W_\theta)$ by $G_\theta = f_\theta(I)$. Predicted grasps are $g_\theta = \{g_{\theta,i}\}$ for $i \in [1, ..., H \times W]$. At runtime, a pixel is sampled based on quality distribution Q_θ, corresponding ϕ_θ, w_θ are extracted from Φ_θ, W_θ.

Datasets. Empirical datasets typically consist of depth image I and k success grasp labels $g_{label} = \{g_{label,i}\}$ for $i \in [1, ..., k]$. Note $q_{label,i} = 1$ since they are guaranteed to succeed. Labeled grasp configuration maps are denoted by $G_{label} = (Q_{label}, \Phi_{label}, W_{label})$. Different from [73], this chapter does not require local padding to create dense ground-truth labels. Fig. 3.3 shows a sample of G_{label} used in training.

For readability, the following sections use abbreviated variables

$$
\begin{aligned}
G_{GT} \mapsto G = (Q, \Phi, W), \quad & g_{GT,i} \mapsto g_i = (q_i, \phi_i, w_i) \\
G_{label} \mapsto \tilde{G} = (\tilde{Q}, \tilde{\Phi}, \tilde{W}), \quad & g_{label,i} \mapsto \tilde{g}_i = (\tilde{q}_i, \tilde{\phi}_i, \tilde{w}_i) \\
G_\theta \mapsto \hat{G} = (\hat{Q}, \hat{\Phi}, \hat{W}), \quad & g_{\theta,i} \mapsto \hat{g}_i = (\hat{q}_i, \hat{\phi}_i, \hat{w}_i)
\end{aligned}
$$

Maximum Likelihood Grasp Sampling Loss

The ground-truth maps G are different from the labeled maps $\tilde{G}$. The former maps are supposed to be densely labeled, while the latter maps have inadequate labels, as shown in Fig. 3.3. The objective of the grasp model is to approximate ground-truth maps G by $\hat{G} = f_\theta(I)$. This chapter assumes the labeled grasps $\{\tilde{g}_i\}$ are a subset of the dense ground-truth grasps $\{g_i\}$, and $\{\tilde{g}_i\}$ are sampled from $\{g_i\}$ based on a probability $P(G)$ parameterized by ground-truth quality map Q. This implies that at pixels where labels exist, ground-truth maps G and label maps $\tilde{G}$ share the same values.

The probabilities of observing a grasp label $\tilde{g}_i$ from ground-truth maps G and predicted maps $\hat{G}$ are described by $P(\tilde{g}_i|G)$ and $P(\tilde{g}_i|\hat{G})$, respectively. In order to approximate the ground-truth maps G, one choice is to maximize the agreement between $P(\tilde{g}_i|G)$ and $P(\tilde{g}_i|\hat{G})$. This objective requires configuration maps to agree at specific pixels where labels exist. Since unlabeled pixels are ambiguous, it can be too radical to assign grasps in those areas, making sense to provide supervision only in labeled areas. Each successful grasp label $\tilde{g}_i$ has quality $\tilde{q}_i = 1$, leading large values for Q at the labeled pixels, thus resulting in a high probability for $P(\tilde{g}_i|G)$, where $P(\tilde{g}_i|G)$ denotes the probability of $\tilde{g}_i$ being selected as a label. Therefore, the objective becomes a maximum likelihood estimation (MLE):

$$
\begin{aligned}
&\max_{\theta} && P(\tilde{g}_i|\hat{G}) && \\
=&\max_{\theta} && P(\tilde{p}_i, \tilde{\phi}_i, \tilde{w}_i|\hat{G}) && \text{(3.2a)} \\
=&\max_{\theta} && P(\tilde{p}_i|\hat{G}) \cdot P(\tilde{\phi}_i|\tilde{p}_i, \hat{G}) \cdot P(\tilde{w}_i|\tilde{\phi}_i, \tilde{p}_i, \hat{G}) && \text{(3.2b)} \\
\approx&\max_{\theta} && P(\tilde{p}_i|\hat{Q}) \cdot P(\tilde{\phi}_i|\tilde{p}_i, \hat{\Phi}) \cdot P(\tilde{w}_i|\tilde{p}_i, \hat{W}) && \text{(3.2c)}
\end{aligned}
$$

In (3.2a), $\tilde{g}_i$ is replaced with grasp configurations. Chain rules are then applied to obtain (3.2b). For (3.2c), grasp configurations are regarded as random variables and have different distributions. Grasp location $\tilde{p}_i$ is conditioned on the quality map $\hat{Q}$. $P(\tilde{p}_i|\hat{Q})$ represents the likelihood of sampling a robust grasp at pixel $\tilde{p}_i$. Categorical distribution is used to describe its probability, where each pixel is a category with a discrete-event probability that is proportional to the predicted quality $\hat{q}_i$. The second and third terms maximize the chance of observing $\tilde{\phi}_i$ and $\tilde{w}_i$ at labeled pixels in predicted $\hat{\Phi}$ map and $\hat{W}$ map. Moreover, this chapter assumes that networks can encode conditioned probability by sharing layers and simplifies the second and third terms.

Equation (3.2c) is jointly optimized for all grasp labels. We further assume each label is

independent, which leads to

$$\max_{\theta} \quad \prod_{i=1}^{k} P(\tilde{g}_i|\hat{G}) \tag{3.3a}$$

$$\approx \max_{\theta} \quad \prod_{i=1}^{k} P(\tilde{p}_i|\hat{Q}) \cdot P(\tilde{\phi}_i|\tilde{p}_i, \hat{\Phi}) \cdot P(\tilde{w}_i|\tilde{p}_i, \hat{W}) \tag{3.3b}$$

$$\propto \max_{\theta} \quad \sum_{i=1}^{k} \log P(\tilde{p}_i|\hat{Q}) + \log P(\tilde{\phi}_i|\tilde{p}_i, \hat{\Phi}) + \log P(\tilde{w}_i|\tilde{p}_i, \hat{W}) \tag{3.3c}$$

Three terms need to be maximized in (3.3c). The first term detects the best grasp pixel after applying the $\log(\cdot)$ operation. The second and third terms are simplified to minimize the mean square error (MSE) between predictions and labels, as [136] suggests. Such modifications stabilize the training process without loss of generality. Different from the previous work, gradients are computed only at labeled pixels instead of the whole map, i.e.,

$$\max_{\theta} \quad \sum_{i=1}^{k} \log P(\tilde{p}_i|\hat{Q}) - \text{MSE}(\hat{\phi}_i, \tilde{\phi}_i) - \text{MSE}(\hat{w}_i, \tilde{w}_i) \tag{3.4}$$

From all above, the grasp model $f_\theta(\cdot)$ is trained to minimize the negative of the objective function in (3.4), i.e.

$$\theta = \underset{\theta}{\text{argmin}} \ \mathcal{L}(\tilde{G}, \hat{G}) \tag{3.5}$$

where

$$\hat{G} = (\hat{Q}, \hat{\Phi}, \hat{W}) = f_\theta(I)$$

$$\mathcal{L}(\tilde{G}, \hat{G}) = \sum_{i=1}^{k} -\log P(\tilde{p}_i|\hat{Q}) + \text{MSE}(\hat{\phi}_i, \tilde{\phi}_i) + \text{MSE}(\hat{w}_i, \tilde{w}_i)$$

As can be seen, the maximum likelihood grasp sampling loss (MLGSL) minimizes a pixel classification loss and two pixel-wise regression losses. Previous works [73, 74, 99, 135] use regression or spatial cross-entropy losses for all three objectives, making networks estimate pixel-wise grasp stability. In contrast, the proposed MLGSL predicts grasp success distributions using pixel classifications and aims to locate the most robust grasp in the image.

Model Architectures

The grasp planning model is used to predict dense ground-truth configuration maps G, consisting of Q, Φ_c, Φ_s, W, which have the same sizes as the input depth image I. Note that angle maps Φ is calculated by $\Phi = \frac{1}{2}\tan^{-1}\frac{\Phi_s}{\Phi_c}$. The model uses a fully convolutional topology identical to [74]. The architecture includes four downsampling layers, two dilated layers, and two upsampling layers. Downsampling layers use kernel size of $[11, 5, 5, 5]$ respectively,

activated by ReLU and max-pooling. Two dilated layers apply $[5, 5]$ kernels with dilation $[2, 4]$. Upsampling layers employ transpose convolutional kernels with size 3 and striding 2.

Despite FCNs, SAM blocks are added after convolution layers as described in [130]. SAM utilizes both max-pooling and average-pooling along the channel axis and forwards them to a convolution layer. Outputs are then integrated into input features.

Dataset

Single object dataset. Jacquard [20] is a large-scale empirical dataset for robotic grasp detection. We directly adopt Jacquard as our single object dataset. The dataset contains more than 50k images of 11k objects and 1 million unique success grasp labels. We split the dataset into 90% and 10% for training and validation. We apply random rotation and zoom to each data and resize the image to 300×300.

Cluttered object dataset. A cluttered dataset is generated based on Jacquard. We randomly select a few images from the single object dataset and combine them into a cluttered sample. Before combining, each single object data is segmented, rotated, zoomed, and translated in the image plane. Success grasp labels are then combined and pruned according to collision constraints. Since data in Jacquard includes images from different viewpoints, such operation well reflects the geometry of cluttered scenes.

3.3 Training Experiments

We trained a series of models to test the proposed approach. The goals of the experiments are three-fold: 1) to demonstrate that the proposed loss function can increase the grasp performance with fewer labels and samples, 2) to determine whether attention modules help in learning dense grasp configurations, and 3) to inspect the collision-avoidance ability in cluttered scenes.

Evaluation Metrics

Two metrics are utilized to evaluate models' performance: predictions' success rate and predictions' accuracy and recall. For prediction success rate, a predicted grasp $\hat{g}_i$ is considered success if

$$\exists \tilde{g}_j \in \tilde{g},\ \ni \text{IoU}(\tilde{g}_j, \hat{g}_i) \geq \delta_{IoU};\ \text{and}\ |\tilde{\phi}_j, \hat{\phi}_i| \leq \delta_\phi$$

where $\text{IoU}(\cdot)$ represents the intersection over union ratio between two grasps. $\tilde{g}$ is the set of success grasp labels. In this chapter, we select top one (Top-1) and top five (Top-5) grasps to measure the success rate and choose $\delta_{IoU} = 25\%$ and $\delta_\phi = 30^\circ$ as in [73, 74, 20]. These criteria predict the models' precision. In other words, they evaluate the ratio of true positives among grasps with predicted positive labels and thus offer an estimation of grasp distributions.

Despite the grasp success rate, we propose to measure predictions' accuracy and recall. We pre-train a grasp quality discriminator [68] to evaluate the robustness of certain grasps. For each validation data, n predicted grasps $\hat{g}_i$ for $i \in [1, ..., n]$ are first uniformly sampled in the image, including predicted quality $\hat{q}_i$. Then, the discriminator evaluates quality for $\hat{g}_i$, obtaining ground-truth quality label q_i. Prediction accuracy and recall are measured based on q_i and $\hat{q}_i$ for $i \in [1, ..., n]$. This chapter chooses $n = 100$. The classification threshold for the binary label is selected as $\delta_{cls} = 0.5$, in which we observe the best classification results; accuracy and recall are averaged among the whole validation batch.

Baseline Methods

We compare the training performance of MLGSL in (3.5) to the following baseline approaches:

Image-wise MSE (ImgMSE). ImgMSE is introduced in [73] that uses the same predicted grasp configuration maps $\hat{G}$ as our method. This baseline resolves the label sparsity problem by local padding. It uses the center third of each positive grasping label as the training ground truth $\bar{G}$. Note $\bar{G}$ is not the same as G introduced in this chapter. Instead, it is an approximation of G. The loss used to train the model is

$$\mathcal{L}_{ImgMSE} = \text{MSE}(\hat{Q}, \bar{Q}) + \text{MSE}(\hat{\Phi}, \bar{\Phi}) + \text{MSE}(\hat{W}, \bar{W})$$

Maximum likelihood sampling with LogMSE (MLGSL+Log). MLGSL+Log can be derived from (3.3c), which uses $\text{MSE}(\cdot)$ to replace $P(\cdot)$ instead of $\log P(\cdot)$, i.e.

$$\mathcal{L}_{MLGSL+Log} = \sum_{i=1}^{k} -\log P(\tilde{p}_i|\hat{Q}) + \log \text{MSE}(\hat{\phi}_i, \tilde{\phi}_i) + \log \text{MSE}(\hat{w}_i, \tilde{w}_i)$$

Pixel-wise MSE (PixMSE). PixMSE removes the maximum likelihood sampling term from (3.5). Instead, it applies supervision on labeled pixels with MSE loss, i.e.

$$\mathcal{L}_{PixMSE} = \sum_{i=1}^{k} \text{MSE}(\hat{q}_i, \tilde{q}_i) + \text{MSE}(\hat{\phi}_i, \tilde{\phi}_i) + \text{MSE}(\hat{w}_i, \tilde{w}_i)$$

Results

For comparisons, we train 110 models with different loss functions and architectures. Each model is trained with different seeds for 50 epochs to select the best one.

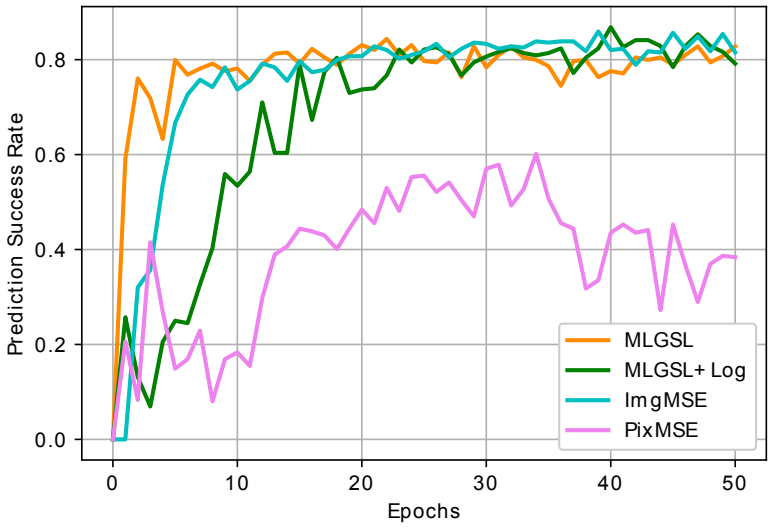


Figure 3.4: Comparing the Top-1 prediction success rate of MLGSL with baseline methods. Models are trained with densely-labeled datasets (16 labels per image).

Table 3.1: Training performance of MLGSL and baselines (Mean %)

Method	Top-1	Top-5	Accuracy	Recall
MLGSL	82.8	91.0	80.2	75.7
MLGSL (2 labels)	81.6	90.3	77.3	73.4
ImgMSE [74]	82.5	89.7	85.2	92.3
ImgMSE [74] (2 labels)	42.4	45.2	41.9	17.4

Baseline comparisons. Our first experiment compares MLGSL to three baseline methods with a single object dataset, in which each training sample includes 16 success grasp labels. Top-1 and Top-5 prediction success rates are shown in Fig. 3.4 and Table 3.1. We see that MLGSL has similar performances compared to previous ImgMSE, while MLGSL shows a higher convergency rate at first a few epochs. MLGSL with logarithm converges to a similar point as the previous two methods but with a slower rate, which might occur because the $\log(\cdot)$ operation lowers the gradient in each training step. PixMSE performs the worst among the four approaches. This is likely due to it only applying supervision on specific pixels, resulting in unbounded other areas.

We also compare MLGSL to ImgMSE on prediction accuracy and recall. Results are shown in Table 3.1. It is interesting to observe that models trained with MLGSL have lower accuracy and recall. To seek reasons for such phenomenon, we plot several predicted $\hat{G}$ in

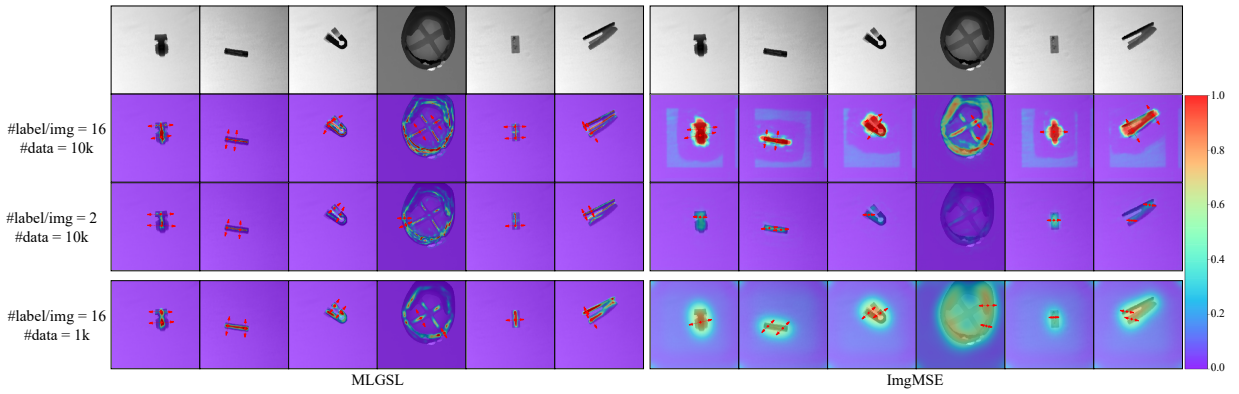


Figure 3.5: Predicted grasp distributions with variant models. Predicted grasp qualities are painted as heatmaps with color listed in the right sidebar. Detected grasps are labeled with red lines in each image. (First row) input depth images, (Left) results from models trained with MLGSL, (Right) results from models trained with ImgMSE, (Second row) results from datasets consisting of 16 labels per image and 10k data, (Third row) results from datasets consisting of 2 labels per image and 10k data, (Bottom) results from datasets consisting of 16 labels per image and 1k data.

the second rows of Fig. 3.5. As can be seen, models trained with MLGSL have a conservative estimation of the grasp quality. They prefer to grasp each part's center, which is consistent with labels' distributions in training datasets. Such behavior leads to false-negative labels in measurements and, thus, the observed results.

Less training labels per sample. We then investigate whether our method can learn grasping with fewer labels. For this study, we down-sample success grasp labels to $[2, 4]$ in each training data and still use all labels for validation. It is a more difficult setting; the grasp planning model learns to effect change only through inadequate demonstrations. We report results in Fig. 3.6 and Table 3.1. In the figure, models trained with MLGSL are evaluated with the Top-1 prediction success rate, indicated by solid lines. Dashed lines indicate the performances of models trained with ImgMSE.

From these results, we see that MLGSL is capable of learning to grasp with 2 labels per image, achieving prediction success rates at 81.6% for Top-1 and 90.3% for Top-5, which is similar to models trained with 16 labels. The third row in Fig. 3.5 shows predicted $\hat{G}$ by models trained with 2 labels per image. We also notice that ImgMSE under-performances MLGSL in such settings. This is attributed to the center-third padding used by ImgMSE, which generates false negatives. Padded $\bar{G}$ may mistakenly label high-quality grasps as negatives since they are not close to existing success labels. Moreover, due to the fact that ImgMSE applies gradients to each pixel in the image, grasp planning models can be

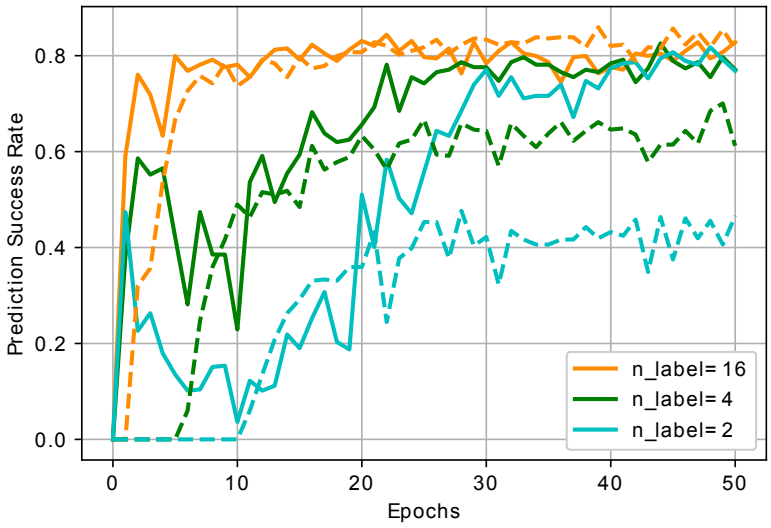


Figure 3.6: Comparing the Top-1 prediction success rate of MLGSL to ImgMSE with different numbers of labels (n_label). Success grasp labels are down-sampled to $[2, 4]$ for each training data. Solid lines indicate models' performance trained with MLGSL, and dashed lines indicate that they trained with ImgMSE.

confused for ambiguous labels. For MLGSL, supervisions are only applied to specific pixels, while other pixels are regulated indirectly with the probabilistic objective. This procedure minimizes assumptions toward unlabeled areas and, thus, does not suffer from insufficient labels.

Less training samples. We next train models with a smaller dataset (1k data) using MLGSL and ImgMSE and report their performances in Fig. 3.7. The results suggest that less training data makes it harder to learn grasping strategies for both methods. However, we still observe our MLGSL outperforms ImgMSE by about 10%. We visualize prediction results at the bottom in Fig. 3.5. Compared to ImgMSE, MLGSL predicts more reasonable grasp distributions with less training data. The less accurate predictions from ImgMSE may be due to false training labels. The center third method can construct false positives. Models may require more data to compensate for such errors; thus, we observe better prediction results in previous experiments with more training data.

Different attention module integration. Besides loss designs, we compare the effectiveness of attention modules by adding SAM blocks to 1) downsampling convolutional layers (DsATT), 2) upsampling transpose convolutional layers (UsATT), 3) both downsampling

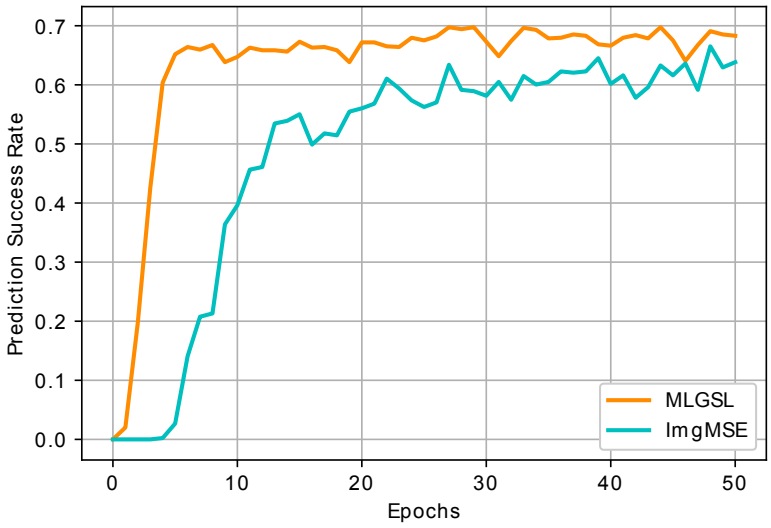


Figure 3.7: Comparing the Top-1 prediction success rate of MLGSL to ImgMSE with fewer training samples (1k data).

Table 3.2: Training performance of MLGSL with single and cluttered datasets (Mean %)

Training Datasets	Collision-Free Ratio	Top-1
Single Datasets	67.3	78.9
Cluttered Dataset	84.2	74.4

and upsampling layers (DsUsATT), and 4) no attention modules (NoATT). Interestingly, results in Fig. 3.8 suggest that the attention module is not a contributor to the performance of grasp planning. This could be because the backbone FCN architecture is simple, consisting of 10 layers, which do not allow SAM blocks to take effect. Furthermore, as suggested in [35, 34], attention modules help to reduce the action sampling complexity by providing a region of importance, which might have a similar effect to quality map $\hat{Q}$ in our FCN models.

Collision avoidance with collision-free cluttered datasets. Networks in [73, 74] are trained with single object datasets and demonstrate promising grasp success rates in the clutter. However, we observe collisions during grasp prediction in the clutter. To improve this, we train models with MLGSL and collision-free cluttered object datasets. Results are shown in Table 3.2. Both metrics are computed using cluttered object datasets. Although we observe a 5% down on prediction success rate, models trained with proposed datasets improve collision detection ability by 25%. It suggests that models trained with MLGSL are

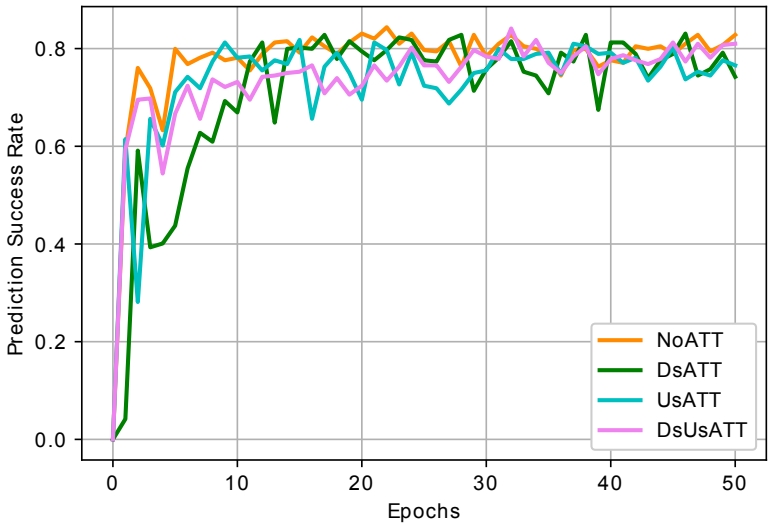


Figure 3.8: Comparing the Top-1 prediction success rate of FCN with MLGSL and different attention module integration.

capable of both predicting grasp robustness and detecting collisions.

3.4 Real-World Experiments

Trained models are run on a laptop with GTX1060 GPU, 16GB RAM, and 2.5GHz CPU. Experimental setups are shown in Fig. 3.9(a), we use a FANUC LRMate 200iD/7L manipulator with an SMC LEHF20K2-48-R36N3D parallel-jaw gripper for grasping. An Ensenso N35 camera is used to capture depth images of the scene. Depth images are cropped and resized to 300×300. We paint invalid depth values using OpenCV [9]. During experiments, the algorithm selects one best grasp $\hat{g}$. Selected $\hat{g}$ is projected to the Cartesian space with calibrated camera matrices. Robots then execute the grasp with an offset $\epsilon = 1$cm along the camera's z-axis.

17 household and 1 adversarial object were selected to test the grasp success rate of our approach (Fig. 3.9(b)). Household objects contain items of varying sizes and shapes. Most of the items (staple, tape, cube, robots, sprayers, glue stick) appear in previous works. We used several additional objects that are deformable (cable) and perceptually challenging (thin edges on the cup, helmet, board eraser, scissor, and reflective zinc container). We also added an adversarial object, which is proposed in [68], to verify models' robustness with a complex geometry.

Figure 3.9: (a) shows experimental setups. Our robot operates over a workspace observed by a statically mounted stereo camera. (b) shows objects used for single/cluttered grasping experiments.

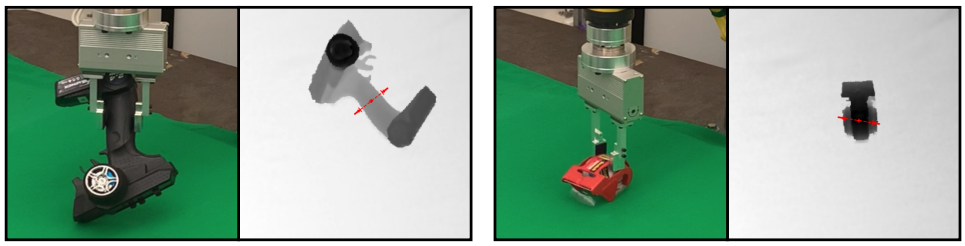
Table 3.3: Real-World performance of MLGSL and Baselines

Method	Required Data	SR (%)	CTPG (ms)
MLGSL	10k × 2	91.8 (78/85)	21
GGCNN [74]	10k × 16	92.1 (47/51)	21
FC-GQCNN [107]	7m × 1	94.1 (32/34)	29

Require data represents the size of required training datasets (number of images × number of labels per image); SR and CTPG represent grasp success rate and computation time per grasp, respectively.

We performed physical grasping experiments with two arrangements: 1) isolated single objects and 2) cluttered objects. For single-object grasping, we trained a model with MLGSL and a dataset consisting of 2 labels per image. We compared our approach with two baselines [74, 107]. Results are reported in Table 3.3, which are consistent with training results. Note that for [74] and [107], they require larger well-labeled datasets to achieve listed performances.

For cluttered scenarios, objects were randomly placed inside the workspace. The robot attempted one grasp each time, and the grasped object was removed from the scene. This procedure continues until all objects are removed or consecutively failed five times. We ran this experiment 10 times to measure performance. Models trained with MLGSL and cluttered datasets achieved an object removal rate of 90%, compared to 70% in models trained with



(a) (b)

Figure 3.10: Two failure cases. (a) shows the object slippage when the robot grasps curved parts, and (b) shows models mistakenly generate grasps toward deformable thin covers.

single datasets, mainly due to undetected collisions. Compared to [74], we observed similar results that models trained with cluttered datasets outperform those with single datasets.

Figure 3.10 displays two common failures of MLGSL. One failure mode occurs when the robot grasps curved surfaces. Utilized models can only predict planar grasps. Grasp depth is set as a hyperparameter. Such limitations make it hard to grasp ball-shaped objects with a uniform grasp depth; models can generate tenuous grasps. The second type of failure occurs when a thin deformable layer is on top of the object's main body. It is challenging to distinguish thin layers from solid cubes with a single vision sensor. Such ambiguity tricks the model into generating a grasp toward those unfavorable areas.

3.5 Chapter Summary

This chapter proposes a Maximum Likelihood Grasp Sampling Loss (MLGSL) to tackle the data sparsity issue in grasp planning. The proposed method regards successful grasp labels are sampled from a ground-truth grasp distribution and aims to recover such dense distribution. Training results suggest that FCN models based on MLGSL can learn to grasp with datasets composed of 2 labels per image, considering 8× more data-efficient than current state-of-the-art techniques. Meanwhile, physical robot experiments demonstrate a 91.8% grasp success rate on household objects. Experimental videos are available at [122].

Chapter 4

Grasp Planning for Multi-Fingered Hands

4.1 Introduction

Earlier chapters have delineated the development of grasp planning networks specifically tailored for parallel grippers. While parallel grippers exhibit commendable reliability in mass-production scenarios, their compatibility is limited when it comes to customized production and household services. In contrast, the domain of grasp planning for multi-fingered hands assumes critical importance in robotic grasping and manipulation, primarily due to its potential to augment dexterity and facilitate human-robot collaboration. The primary advantage of multi-fingered hands lies in the increased number of joints, offering a greater range of degrees of freedom (DOFs). This enhancement not only elevates the manipulability of the system but also ensures a more dexterous grasp compared to parallel-jaw grippers. Additionally, the incorporation of multiple contact points allows these multi-fingered grasps to withstand larger disturbances during various grasping and manipulation tasks. The development of a versatile, multi-fingered hand would significantly streamline operational procedures and bolster adaptability to a diverse array of tasks. However, the task of devising grasp planning strategies for general-purpose multi-fingered hands is fraught with challenges. These challenges stem primarily from the extensive variation in object shapes and sizes, the intricate coupling dynamics between the hand and objects, and the high-dimensional complexity inherent in the hand-object system. Addressing these issues is imperative for the advancement of robotic manipulation.

In general, there are mainly three categories of approaches to solving the grasp planning problem: grasp optimization in an online manner [28, 30, 27], grasp candidate sampling with learning-based grasp evaluation [68, 76, 75], and end-to-end learning-based grasp generation from raw inputs [82, 110]. While the above approaches have demonstrated promising performance on parallel grippers with low DoFs, some works leverage these pipelines with multi-fingered hands. First, optimization methods are proposed to maximize a manually

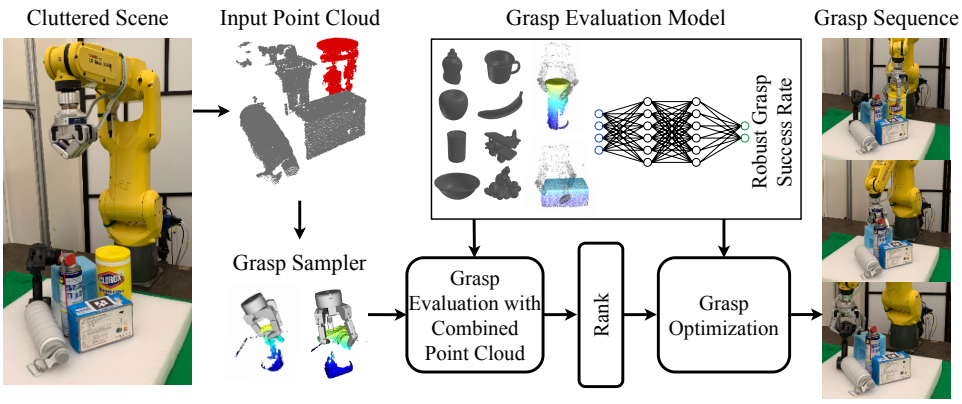


Figure 4.1: MF-GPD Architecture. Given the objects' single-view point cloud, the cross-entropy sampler generates candidates among the object surface. Candidates are assessed with the evaluation model, which takes in the point cloud representation of the grasp. A local grasp optimization is introduced to avoid collisions in complex environments and minimize the grasp quality loss. MF-GPD could rapidly determine the robust grasp candidates, which is executed with a close-and-clamp strategy using the BarrettHand.

designed grasp quality while avoiding collisions in an online manner [26, 29]. However, the grasp quality heuristics cannot reflect the real-world grasp success, and such online optimization typically requires dense surface information and extensive computation. Second, some methods try to recognize the best grasp among sampled candidates with learned quality metrics (sample-then-evaluate) [63, 62]. Nevertheless, grasp representations used to estimate the grasp quality are hard to design, and the grasp candidate sampler can be inefficient and prone to collisions. Third, end-to-end grasp planners have been used to infer grasp poses directly from raw sensor inputs [110, 58]. These methods, however, always require delicate parameter tuning and large datasets and suffer from the curse of dimensionality.

This chapter aims to detect multi-fingered hand grasps *reliably* and *efficiently*. To enable *reliable* grasp evaluation, we consider a novel representation of the multi-fingered grasps. Some works [91, 57] represent grasps by their local contact areas. Despite good performance for parallel-jaw grippers, experiments in this chapter show that this representation is not sufficient for multi-fingered grasps. Compared to parallel-jaw grippers that grasp objects by clamping, multi-fingered grippers typically have higher DoFs and more contact points. While higher DoFs offer more dexterity, it is difficult to find proper contact points and exert balanced contact forces that stabilize the object. This chapter combines the object point cloud with the rendered gripper as the input to evaluate grasps. To improve the prediction accuracy for grasping with multi-fingered hands, we add extra features, including the point normals and a binary object-gripper mask. We demonstrate in experiments that these features are crucial for reliable grasp evaluation.

To detect multi-fingered grasps *efficiently*, we consider a method that combines the

sample-then-evaluate and optimization methods. On one hand, the sample-then-evaluate approach can learn accurate grasp quality, but it is time-consuming to provide collision-free samples in high dimensions. On the other hand, the optimization method is not sensitive to dimensionality and can easily adjust grasp away from collisions, but it suffers from the inaccurate quality metric design. This chapter leverages an evaluation model that combines the sample-then-evaluate and optimization methods for efficient and high-quality grasp evaluation, as shown in Fig. 4.1. An evaluation network takes the grasp samples (represented by the aforementioned grasp representation) from a sampling module as inputs and learns the highly accurate grasp qualities as outputs. We further design a local grasp optimization to address the time complexity to sample feasible grasps in complex environments. The optimization utilizes the evaluation network as the cost function to avoid inaccuracy from the manually designed heuristics [26]. Moreover, the optimization is defined in the entire robots' configuration space, making the problem harder to solve compared to local search as in [75, 76].

The contributions of this chapter are two-fold. First, we introduce a point-cloud-based grasp representation to embed multi-fingered grasps. The grasp representation incorporates normals and binary masks to find *reliable* contacts that can exert effective and balanced forces to stabilize the object. Experiments demonstrate the proposed representation outperforms the feature representations from previous multi-fingered grasping works. Second, we propose a grasp detection structure that combines an evaluation network and an optimization model for *efficient* collision avoidance and grasp refinement. The evaluation-refinement structure can take advantage of both the learning-based evaluation and the optimization for accurate quality prediction and efficient grasp adjustment. Experiments suggest such a structure significantly reduces the computation time.

4.2 Grasp Planning using Point Cloud

Notation

We define the grasp planning problem as finding a grasp configuration that allows the robot to pick up the object without collisions. We focus on scenarios where the target object is isolated or in the clutter with others. Object segmentations are computed based on raw camera readings. Some notations are introduced here.

Point cloud. Let $o \in \mathbb{R}^{3\times n}$ denotes the point cloud captured by the depth camera. $o_i \in \mathbb{R}^{3\times n_i}$ for $i = 1, 2, ..., m$ denote the point clouds for m different objects in the scene. n_i represents the number of points for the ith object. The point cloud of the target object is o_{target}.

Grasp. A grasp g is determined by the palm pose (R, t) and the gripper joint angles θ. The palm pose (R, t) is given in $SE(3)$, specifying the 3D orientation and 3D translation of

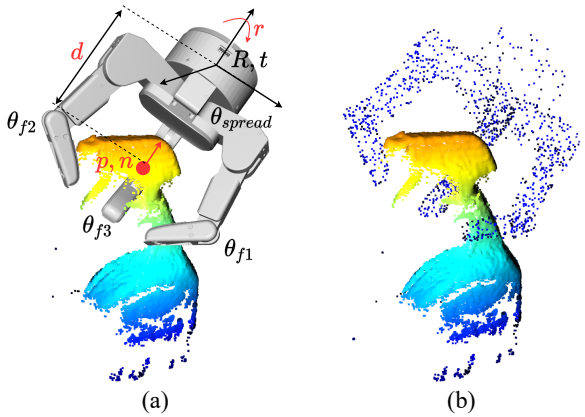


Figure 4.2: (a) shows the definition of a grasp g, (b) shows the combined point cloud representation o_{input} of the same grasp.

the gripper's base. The joint angles $\theta \in \mathbb{R}^j$ represent the angle of j joints in the gripper. We further define the palm pose $(R, t) = (p, n, d, r)$ as Fig. 4.2(a) shows. Here $p = (p_x, p_y, p_z) \in \mathbb{R}^3$ is the grasp point on the point clouds o_{target}. $n = (n_x, n_y, n_z) \in \mathbb{R}^3$ and $d \in \mathbb{R}$ are the unit approaching vector and the offset distance, respectively. To simplify the representation, the approaching direction n is computed as the object's surface normal at point p similar to [82, 91, 57]. $r \in \mathbb{R}$ is the rotation angle of the palm.

For joint angles θ, some assumptions are made in this chapter. First, we assume that the gripper joint angles are determined by the spread angle $\theta_{spread} \in \mathbb{R}$ and the finger joint angles θ_{fi} for $i \in [1, ..., k]$, where k is the number of fingers. Second, we assume each finger only has one degree of freedom such that $\theta_{fi} \in \mathbb{R}$ for $i \in [1, ..., k]$. This assumption resolves the ambiguity: since finger joint angles can be uniquely determined after contact without ambiguity, their values are neglected for simplification.

Grasp quality metric. Numerical grasp quality metrics are used to measure how likely the grasp can succeed. In this chapter, we directly utilize the simulated grasp success rate as our metric Q_{rate}. Since approximating a continuous value is challenging, we formulate the quality estimation as a classification task with a binary metric Q_{succ}:

$$Q_{succ} = \begin{cases} 1 & \text{if } Q_{rate} \geq \delta \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Cross-Entropy Grasp Sampler

The grasp candidate sampler iteratively generates grasp candidates g, which is parameterized by $(R, t), \theta$. The grasp point p is iteratively fitted with Gaussian distributions while d, r, θ are uniformly sampled from the configuration space.

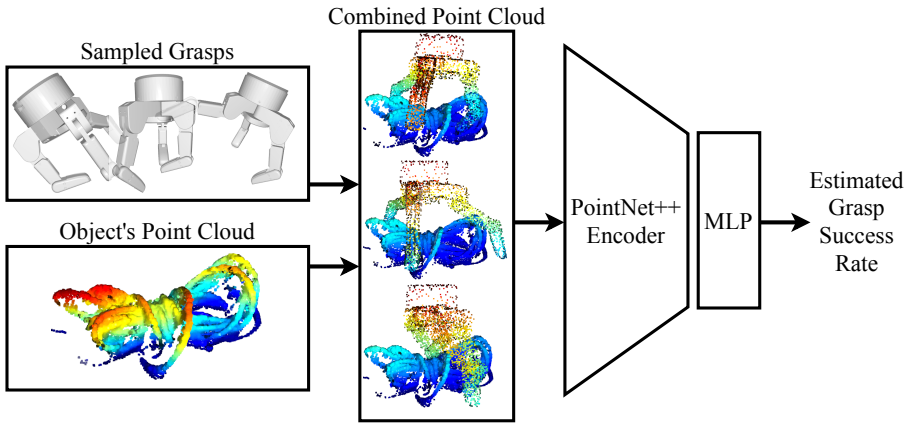


Figure 4.3: The architecture of the grasp evaluation network based on PointNet++ [96]. Given sampled grasps and the point cloud, the grasp is represented by the combined point cloud. Point normal directions and binary gripper-object masks are estimated as extra features. After three set-abstraction layers, the global feature vector is classified with three fully-connected layers (MLP).

The procedure is described as follows: First, given the object's point cloud, the normal direction n of each point is estimated. Second, a probability grid is constructed to model each point's sampling chance with a Gaussian distribution. Grasp points are then sampled based on the probability grid. Third, for each grasp point, d, r, θ are drawn from the configuration space with a pre-defined uniform distribution. A grasp g can be determined by combining grasp point p with d, r, θ. Fourth, each sampled grasp is assessed and ranked with the evaluation network. Top 10% grasp samples are used to update the grasp point distribution. Such a procedure repeats until a robust grasp has been proposed.

For the joint angles $\theta = [\theta_{spread}, \theta_{f1}, ..., \theta_{fk}]^T$, this chapter assumes each finger has one DoF, suggesting finger joints $\{\theta_{fi}\}$ can be uniquely determined once the spread angle θ_{spread} is chosen. In other words, when the palm poses and the spread angle are fixed, each finger is closed until contact to find $\{\theta_{fi}\}$. Finding such exact contact joints, however, is time-consuming in practice. Each finger should be iteratively tested for collisions in order to find the contact configuration. To tackle this problem, we propose to set finger joints $\{\theta_{fi}\}$ at zero as shown in Fig 4.4(b). Consequently, the grasp evaluator is designed to estimate the grasp quality in such a non-exact contact scenario. The exact contact configuration is determined afterward.

Grasp Evaluation with Point Cloud

The grasp sampler fits the posterior distribution $P(g|o_{target})$ to propose grasp candidates g. These contain both poor and good grasps. Poor samples need to be identified and pruned

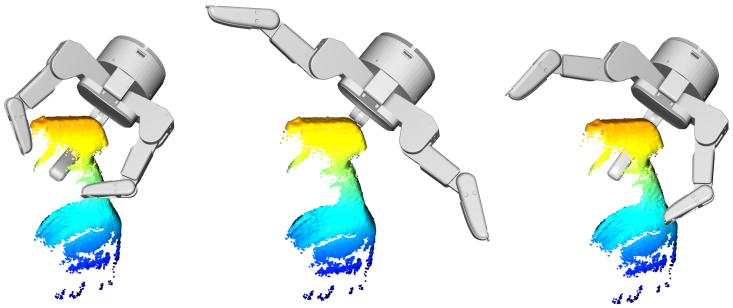


Figure 4.4: Given a grasp (i.e., R, t, θ_{spread}), (a) shows fingers' configuration at the exact contact (i.e., $\{\theta_{f_i}\}$ at contact), (b) shows the zero-finger-joint contact (i.e., set $\{\theta_{f_i}\} = 0$), and (c) shows the random-finger-joint contact (i.e., randomly sample $\{\theta_{f_i}\}$).

out. To achieve this, we need an evaluation model to assign a grasp success rate to each grasp candidate as $P(s|g, o_{target})$, where s is the execution success rate. The evaluation model takes in the object point cloud o_{target} and the sampled grasp g. We use the PointNet++ architecture to classify the grasp with Q_{succ} metric as Fig. 4.3 shows.

There are multiple ways to represent the grasp of the object. First, the gripper contact areas of each grasp can be extracted [91, 57]. The contact area is an intersection of the gripper's closing volume with the object point cloud. Second, the pose of the grasp g can be associated with the point cloud o_{target} in a latent space [63, 62]. Our training results show that such two encoding methods lead to a relatively worse classification accuracy. Instead, we represent a grasp g more tied to the object point cloud: robot gripper's point cloud $o_{gripper}$ is rendered according to the grasp configuration g. The gripper's point cloud $o_{gripper}$ and the object point cloud o_{target} are combined to o_{input} as the input to the network (Fig. 4.2(b)). Besides the point position, the estimated normal direction for each point and binary masks are used as extra features. The binary mask indicates whether a point belongs to the object or the gripper. The PointNet++ processes extra features according to the spatial relationship of points. In other words, neighbor points are encoded together, making it natural to use all the relative information.

As discussed above, we would like the grasp evaluator to predict Q_{succ} with a non-exact contact. To do so, gripper configurations g in the training data (Fig. 4.4(a)) are perturbed in two ways. First, finger's joint angles $\{\theta_{fi}\}$ are set to zero with the probability ξ, as shown in Fig. 4.4(b). Second, the finger's joint angles $\{\theta_{fi}\}$ are uniformly drawn from zero to the exact-contact angles (Fig. 4.4(c)). The grasp evaluation model is trained with the standard cross-entropy loss:

$$L_{eval} = -y \cdot log(s) - (1 - y) \cdot log(1 - s) \tag{4.2}$$

where y is the binary grasp quality metric Q_{succ} and s is the predicted success rate by the evaluator.

Local Grasp Optimization

Although the sampling module and the evaluation network recognize plausible grasps, collisions and object potential slippages during execution are not considered. The sampler alone can perform poorly in narrow spaces. It may produce grasps that collide with the ground or the surrounding objects. With small objects and cluttered environments, the collision-free regions around the target are narrow, making many sampled grasp candidates infeasible. Besides, the gripper may slip on object surfaces if the planned grasps are executed by close-and-clamp strategy. Different fingers may have asynchronous contacts without a complete point cloud. The clamp approach may further exert non-internal forces on the object, which causes unexpected object movements. To tackle this, we introduce grasp optimization.

If there exists one top-ranked candidate who is collision-free, the local refinement will be bypassed, and that grasp will be executed. Otherwise, grasp optimization will be used to avoid collisions and object slippages:

$$\begin{aligned} \max_{g=(R,t,\theta)} \quad & P(s|g, o_{target}) \\ \text{s.t.} \quad & SD(FK(g), o_i) \geq 0, \text{for } i = 1, ..., m \\ & \|\theta - \theta_{center}\|_2 \leq \epsilon \\ & \|R - R_0\|_2 \leq \varsigma_R, \|t - t_0\|_2 \leq \varsigma_t \end{aligned} \tag{4.3}$$

where g is the top-ranked grasp candidate, o_{target} is the point cloud of the target object, $P(s|g, o_{target})$ predicts the grasp success rate. $FK(\cdot)$ denotes the forward kinematics function. $SD(\cdot)$ denotes the signed-distance function of the robot and the surrounding object o_i. θ_{center} is the center of the gripper's joints and bounds the gripper's configuration as suggested in [26]. R_0, t_0 represents the palm pose of the grasp candidate, which serves as the initial state in the optimization. The optimization is solved locally to guarantee the final grasp is still close to the robust candidate.

Dataset Generation

To generate multi-fingered grasp sets, we use the PyBullet [17] physics engine to simulate grasps. 556 objects from the 3DNet [129] are selected as the object set. Comparing to newer datasets [68, 67], selected datasets contain household objects with suitable dimensions for multi-fingered grippers. These objects have demonstrated promising grasp learning performance in [63, 62]. To generate the object point cloud o_{target}, 12 synthetic depth images are rendered from different angles. The object is placed at the center of a regular dodecahedron; cameras are placed at each face's center and point to the origin. Depth images are then projected to 3D space to obtain the point cloud. 295,403 grasps are sampled with uniform distributions and simulated in the physics engine.

The simulation has a free-floating multi-fingered gripper and free-floating objects. The simulator's gravity is set to $[0, 0, -10]^T m/s^2$ with a $1m/s^2$ variance in each direction. Surface friction and the object mass are kept constant. To simulate a grasp, the gripper first reaches

the sampled contact configuration R, t, θ, and grasp forces are then applied to the object. Instead of tracking constant contact forces, we close each finger by $\Delta\theta_{fi}$ to clamp the object. A grasp is labeled success if the object is kept in the gripper after 3 CPU seconds with Gaussian white external forces. As in [68, 75], such a procedure injects dynamical noise into the simulation and generates a robust label for each grasp, narrowing the dynamical simulation-to-reality (sim-to-real) gap. The same procedure is repeated for 5 times to obtain the grasp success rate Q_{rate}. Overall, 103,561 grasps have Q_{rate} larger or equal to 0.8.

4.3 Training Experiments

We trained a series of evaluation models with variant grasp representations. The goals of the experiments are three-fold: 1) to determine whether combined point cloud representation is efficient for multi-fingered grasps, 2) to demonstrate whether point normals and masks can help in evaluating multi-fingered grasps, and 3) to inspect the influence of in-exact contacts.

Training Details

The object point cloud o_{target} is selected from 12 candidates according to the grasp configuration. We choose the point cloud to have the smallest angle between its corresponding camera view angle and the gripper's z-axis n. The position of the object o_{target} is jittered and rotated to simulate the sensor noise. A zero-mean Gaussian noise with a 2 mm variance is applied to simulate the visual sim-to-real gap. The object point cloud is then down-sampled to 2048 points. The gripper's point cloud is uniformly sampled from the robot mesh file, denoted as $o_{gripper} \in \mathbb{R}^{3\times512}$. Normalization is applied to all the training data. Combined with the estimated point normals and the binary mask, the input data has 7 features: $o_{input} = o_{gripper} \bigcup o_{target} \in \mathbb{R}^{7\times2560}$.

The evaluation network takes the 7-dimensional combined point cloud o_{input} as input and predicts a binary grasp success rate. The grasp evaluator leverages the PointNet++ architecture with 3 set-abstraction (SA) layers and 3 fully connected (FC) layers as in [96]. Each SA layer samples and groups 512, 128, and all points within the radius of 10cm, 40cm, and 1m in the input space. Standard PointNet [95] is used to extract grouped inputs followed by 3 FC layers. The dimension of the FC layers are $[128, 128, 256]$, $[256, 256, 512]$, and $[512, 512, 512]$, respectively. Another 3 FC layers with the dimension of $[512, 256, 2]$ are utilized to predict the grasp success label. ReLU is used as the activation function in the whole network.

Baselines

We compare the training performance of the proposed combined point cloud representation with the following baseline methods:

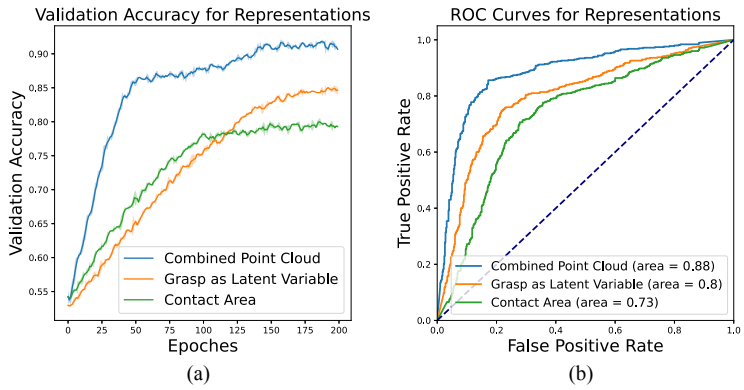


Figure 4.5: (a,b) shows the learning curves and the ROC curves for baseline comparisons.

Contact area. Contact area representation is widely used for parallel-jaw grippers as in [68, 91, 57]. We extract contact area points for each grasp to adopt such a method. Moreover, we add point normals and object-gripper masks to the input points. The resulting 7-dimensional point clouds are fed into evaluation models to predict the grasp success rate.

Grasp as latent variable. Instead of combining grasps with objects at the input space, [62] propose to concatenate encoded objects and grasps in a latent space. It is not clear whether this representation can perform better than interpretable representations (i.e., combining in the input space). We adopt similar model architectures in [62] to inspect such a question. We replace the original voxel encoder with a PointNet++ encoder for objects and keep other layers the same for a fair comparison.

Results

We trained multiple evaluation models with different input designs. Each input design was trained with different seeds for 200 epochs to select one with the best performance. We used 1,200 samples that were held out during the training for validation. Models were optimized with SGD and a constant learning rate of 0.001. Each model took 6 hours to converge with two GTX1080Ti GPUs. Prediction accuracy and receiver operating characteristic (ROC) curves are used to measure the performance.

Baseline comparisons. Our first experiment compares the combined point cloud representation with baseline methods. Fig. 4.5(a) shows the validation accuracy across the training epochs. Fig. 4.5(b) shows the ROC curves. As can be seen, the combined point cloud performs the best among all three embedding methods. As suggested by geometric quality metrics [26], the gripper's configuration is related to grasp success. By combining the point cloud, we encode the gripper's kinematics information into the input. Furthermore, the

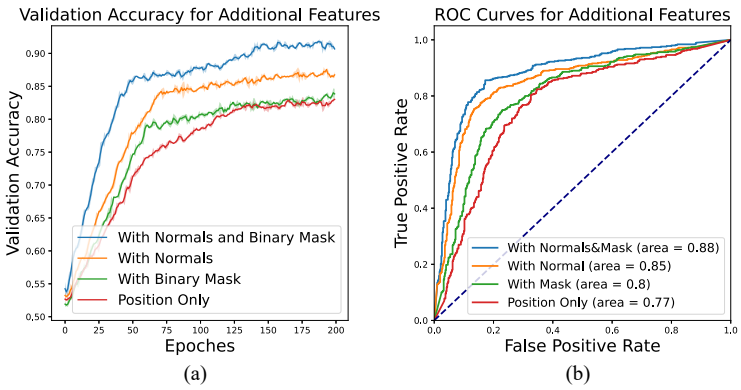


Figure 4.6: (a,b) shows the learning curves and the ROC curves for additional features.

combined representation intrinsically includes the contact area. In the PointNet++ architecture, there is a neighbor grouping operation at every SA layer. Thus, the evaluation model would take advantage of the contact area using the combined representation, together with the geometric heuristics. Moreover, we observe the latent representation performs worse than the combined point cloud. This suggests contact information, which interpretable representations encode, contributes to the prediction performance, and is essential for multi-fingered grasp planning.

Additional features. We then investigate whether point normal directions and binary gripper-object mask can improve the evaluation performance. As analytical grasp metrics [100] suggest, the grasp success rate is strongly related to the contact force and the object's normal directions. For example, the grasp map matrix [77] is essential to analyze a grasp's wrench space. Contact normal and forces are required to compute the grasp map matrix and characterize the force-closure constraint. Experiments support such reasoning as shown in Fig. 4.6. Extra features improve the prediction accuracy with point cloud representations. Moreover, adding normal directions yields better performance than only adding binary masks. This might be because the network can learn to segment the point cloud, while it is harder to regress normal directions for each point [96].

In-exact finger contacts. We next train models with in-exact finger contacts as shown in Fig. 4.4. The objective of using in-exact contacts is to speed up the sampling procedure and increase the evaluator's robustness. In this experiment, finger joints θ are jittered in the training dataset. Fig. 4.8 demonstrates the results. The exact contact scenario is added for comparison. The proposed two jittering techniques have similar performances in the sense of prediction accuracy. Although they both take a long time to converge and underperform

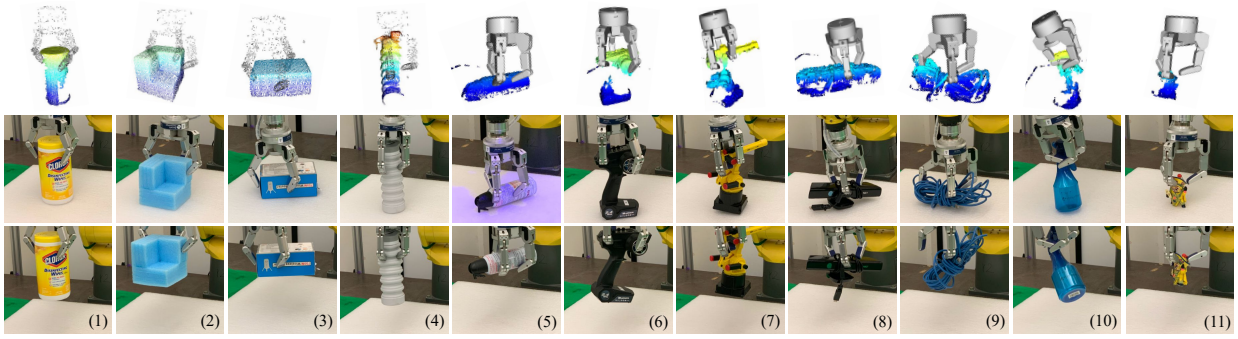


Figure 4.7: (1-11) The grasp planning and execution results on 11 objects with single-view point clouds. For each column, (Top) shows the perceived point cloud and planned grasps, (Middle) shows physical grasps reaching the target grasp, and (Bottom) shows the execution results.

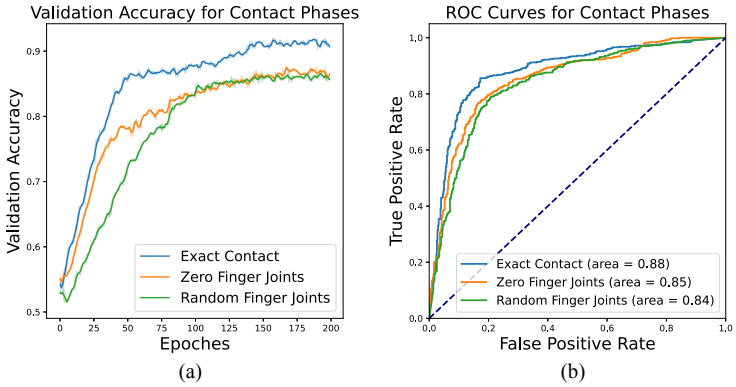


Figure 4.8: (a,b) shows the learning curves and the ROC curves for different finger contact phases.

the exact contact, they have lower time complexity in practice. The slow convergence might be due to the irrelevant variables that confuse the network.

4.4 Real-World Experiments

The proposed MF-GPD is run on a desktop with a GTX1080Ti GPU, 32GB RAM, and a 4.0GHz CPU. For the experiment, we use a BarrettHand BH8-282 multi-fingered hand attached to a FANUC LRMate 200iD/7L manipulator for grasping. An Ensenso N35 camera is used to capture the point cloud. The region-growing approach implemented in the point cloud library [103] is utilized to pre-process and segment objects. The sign-distance field

Table 4.1: Performance analysis of MF-GPD and baselines on single object grasping tasks.

	Success Rate	Time (sec/grasp)
GPD (position only) [57]	57.6% (19/33)	2.89
GPD (position and normal) [57]	65.5% (36/55)	2.89
GPI [62]	69.1% (38/55)	1.37
PPO-JPO [26]	66.7% (22/33)	3.26
MF-GPD w/ Exact Contact	**74.7% (74/99)**	2.82
MF-GPD w/o Exact Contact	71.7% (71/99)	**1.07**
MF-GPD w/o Extra Features	63.6% (21/33)	2.82

in (4.3) is computed using the flexible collision library [88]. Hyperparameters are chosen as $\delta = 0.8$, $\epsilon = 30$, $\varsigma_R = 30, \varsigma_t = 0.05$, $\xi = 0.5$, and $\Delta f_{fi} = 0.1$.

Fig. 4.7 shows the grasp planning and grasp execution results on 11 different objects with the zero-finger-joint contact evaluation model. The point cloud is captured using one depth camera. The object's point cloud and the located grasp are shown on the top of each subfigure. The physical grasp poses, and the execution result of the planned grasp are shown in the middle and bottom, respectively. The algorithm is able to find collision-free grasps for a) small objects, b) objects with complex surfaces, and c) objects with sharp edges. Comparing to experimental results of parallel grippers in previous chapter 2, it can be seen that multi-fingered grippers have advantages in grasping objects with curved and complex surfaces, such as cylinders and toys. This might be due to the fact that the extra finger provides additional contact and thus makes the grasp stable.

Table 4.1 compares MF-GPD with baseline methods in single object grasping tasks. Grasp pose detection (GPD [57]) and grasp pose probability inference (GPI [62]) were implemented and compared. Each object was grasped for even times. Besides these two baselines, the palm pose optimization joint pose optimization (PPO-JPO) [26] is implemented and compared, which solves the grasp planning problem using runtime optimization. In the experiments, grasp candidates are sampled with the cross-entropy sampler. Grasps in collisions are pruned before the execution. Table 4.1 also compares different input feature selections and contact phases. Normal directions and the binary mask are added by default. The results show that additional point normals and masks improved the evaluation accuracy; the non-exact contact reduces the computation complexity.

Fig. 4.9 displays two typical failures of MF-GPD. One failure mode occurs when the object's point cloud is significantly incomplete. The exact contact configuration is hard to determine under such circumstances, which yields asynchronous contact. The second type of failure occurs when the gripper is placed in sharp regions of the object. The gripper might exert unbalanced contact forces on the object when in those areas and cause object motions, yielding object slippage. It appears that the performance could be improved with an object completion and adaptive clamp policy.

Fig. 4.10 shows a sequence of detected grasps in a cluttered environment. The robot

Figure 4.9: Two failure modes of MF-GPD. (a) shows the object slippage due to lack of contact detection with incomplete point clouds, and (b) shows the net object movement when unbalanced forces are applied.

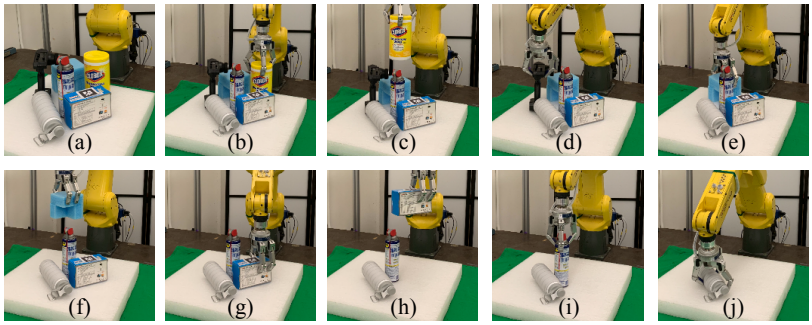


Figure 4.10: (a-j) show a sequence of detected multi-fingered grasps in a cluttered scene.

attempts one grasp each time, and the grasped object is removed from the scene. This procedure continues until all objects are grasped and removed or consecutively failed five times. The grasp sequence is selected according to the graspability or whether a collision-free grasp exists for a particular object. The local grasp optimization (4.3) is solved using the gradient descent algorithm, and the grasp trajectory is planned using the RRT-Connect [50]. Table 4.2 shows the task completion rate, the required computation time, and the grasp success rate for 10 trials. A trial is completed if all objects are removed from the scene, and the grasp success rate is measured for each grasp execution. MF-GPD uses inexact contact and extra features to evaluate each candidate. As can be seen, MF-GPD with local grasp optimization has the highest task completion rate and requires the least computation time. From experiments, we observe the major bottleneck for sample-then-evaluate methods is the collision. GPD and GPI both require numerous iterations and samples to find a collision-free robust grasp, while MF-GPD resolves such a cumbersome procedure by local optimization. Moreover, finding exact contact configurations takes time for multi-fingered grippers: we observe longer computation time for GPD and GPI than MF-GPD, which uses in-exact

Table 4.2: Performance analysis of MF-GPD and baselines on cluttered objects removing tasks.

	Time (sec/grasp)	CR	SR
GPD [57]	3.2	70%	60.4% (32/53)
GPI [62]	3.0	80%	63.5% (33/52)
PPO-JPO [26]	3.3	60%	58.7% (27/46)
MF-GPD w/o Opt.	2.9	80%	64.3% (36/56)
MF-GPD	1.7	90%	72.4% (42/58)

CR for scene removal completion rate and SR for grasp success rate.

contact to reduce the sampling complexity. The performance of PPO-JPO is compromised due to the inaccurate grasp quality estimation and incomplete point cloud.

4.5 Chapter Summary

In this chapter, we propose a multi-fingered grasp pose detection (MF-GPD) algorithm to plan grasps in clutter. The algorithm generates grasp candidates using a cross-entropy sampler and assesses them with an evaluation model. Gripper's point cloud is rendered and combined with that of the object before feeding into the evaluation model. Top-ranked grasps are refined with a local grasp optimization to avoid collisions in clutter. MF-GPD can locate a collision-free grasp in the clutter with 72.4% success rate in reality.

Part II

Contact-Aware Manipulation

Chapter 5

Robot Dexterous Manipulation by Model-Based Learning from Demonstrations

5.1 Introduction

Chapters 2 through 4 present a series of algorithms focused on robust and sample-efficient robotic grasp planning and execution. To extend the capabilities of robots to more complex and general tasks, there is a necessity to equip them with advanced manipulation skills. These skills include actions such as pushing, pivoting, and in-hand reorientation of objects. Recent advancements in reinforcement learning (RL) algorithms have shown promising outcomes in mastering these complex tasks. However, a significant limitation of RL is its dependence on extensive reward engineering, which challenges its scalability in learning a broad spectrum of skills. In this context, Learning from Demonstration (LfD) emerges as a potent alternative. LfD enables robots to assimilate policies from expert demonstrations, including those sourced from platforms like YouTube [141], potentially diminishing the human labor required in the robotic skill acquisition process [112, 113].

This chapter specifically focuses on the development of a model-based LfD framework that utilizes raw RGB video data as input. Although model-based learning approaches are recognized for their superior sample-efficiency and generalization capabilities over model-free counterparts [86, 45], the domain of model-based LfD has not been extensively explored and remains a burgeoning field. We identify and discuss several major challenges that currently impede the widespread adoption of model-based LfD in real-world applications.

One challenge is how to automatically and efficiently develop a model that scales to high-dimensional input such as raw images or videos [43]. To tackle this, we introduce a self-supervised modelling pipeline that leverages recent advancements in differentiable rendering and signed distance functions. This pipeline estimates both the geometric shape of the object and its associated 6D poses, forming an explicit representation. A second challenge lies in

enabling robots to effectively utilize physical models to generate efficient policies. This is particularly critical for robots operating in real-world contact-rich manipulation tasks where the physical interaction between the robot and its environment is a key factor. To address this, we develop a hierarchical LfD framework that integrates low-level modules for contact-point localization and contact-force optimization with a high-level module for contact sequence planning. These modules work in concert to plan manipulative actions. To ensure robust and real-time deployment, we further incorporate a neural policy designed to imitate the outcomes of planning algorithms. This enables the robot to execute complex tasks with high reliability and efficiency.

We have evaluated our pipeline on two datasets, including the sth-sth dataset [33] containing basic manipulation actions on various objects and a small recorded video dataset showing a human performing dexterous in-hand manipulation with primitive objects. The results, derived from rigorous simulation and real-world experiments, bear testament to the effectiveness of our proposed pipeline, available on our website [125].

5.2 Diff-LfD: Contact-aware Model-based Learning from Visual Demonstration

The overall framework is illustrated in Fig. 5.1. Given a demonstrated RGB video consisting of N frames denoted as $\mathcal{V} = \{\mathcal{I}_t\}_{t=1}^{N}$, we prepossess the video to segment and identify the most relevant objects with masks $\{\mathcal{M}_t\}_{t=1}^{N}$, exploiting the SAM [49]. The local frame of the object is randomly defined at the first frame. Our *Diff-LfD* calculates the object's relative pose transformation in the demonstration and jointly estimates the object's mesh $\mathcal{O}$ and the associated 6D poses $\{\mathcal{P}_t\}_{t=1}^{N}$ at each frame. If the robot is provided with a similar but different object from the object recorded in the video, we align the pose of the provided manipulated object with the reconstructed object. Next, our pipeline infers the *wrench* (a combination of external forces and torques) required to complete the pose transformation across two consecutive time steps and generates feasible robot actions to accomplish the pose transformation. This planning includes both the low-level contact-point localization and contact-force optimization and high-level contact sequence planning to chain the whole (long-horizon) manipulation sequences. The manipulation actions are then utilized to train a neural network for robust real-world execution and generalization.

5.3 Pose and Shape Estimation with Differentiable SDF

This subsection introduces the pipeline for pose estimation and shape reconstruction from raw videos. We adopt the differentiable SDF (*Diff-SDF*) [119] to represent the object geometry, which has a large representation capability to model diverse objects with various

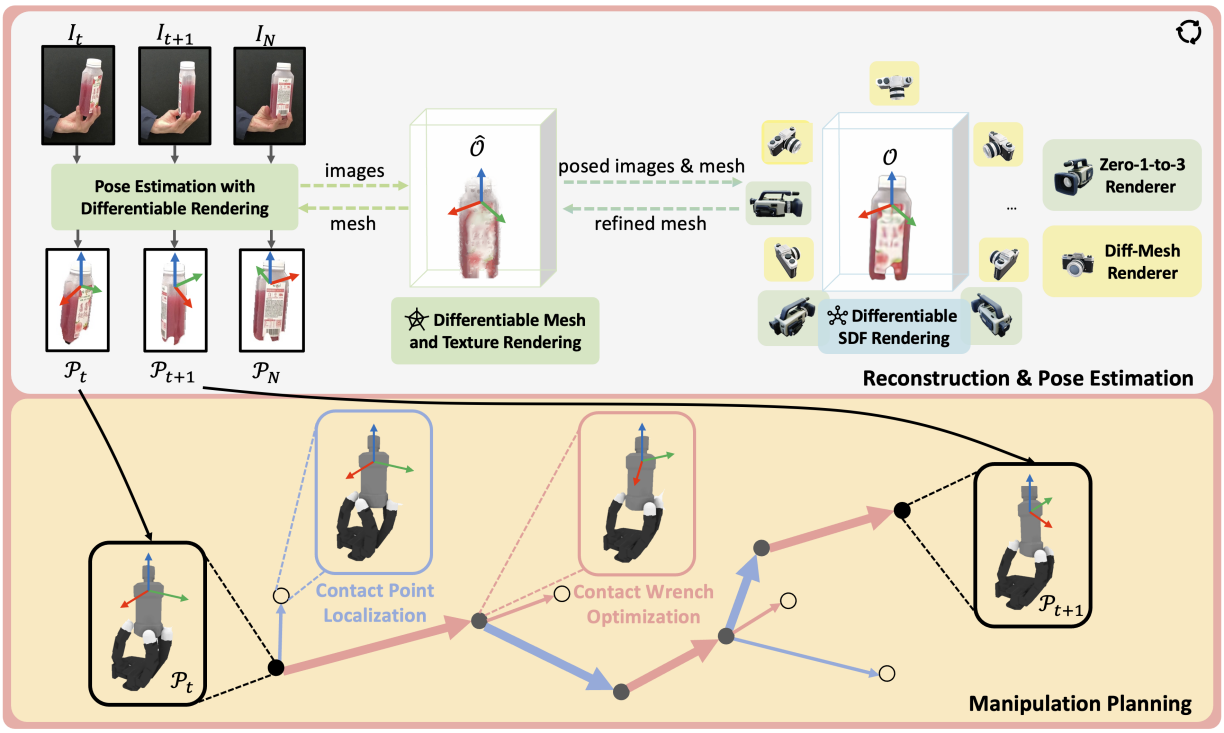


Figure 5.1: The proposed model-based learning from demonstration (LfD) pipeline can be divided into two primary components. The top part focuses on object shape reconstruction and pose estimation, employing differentiable mesh rendering and signed distance function (SDF). The bottom part illustrates the process of contact-aware hierarchical manipulation planning involving contact point localization and differentiable wrench optimization.

topology structures. Moreover, Diff-SDF enables smooth image-based optimization to the images due to its inherent convexity [119]. The explicit surface mesh $\mathcal{O}$ can be extracted from the SDF using the marching cube method [61].

Ideally, given an initialization of an SDF parameterized by ϕ and its associated 6D poses at each frame $\{\mathcal{P}_t\}_{t=1}^N$, the differentiable renderer $\mathcal{R}$ produces a sequence of images $\{\mathcal{I}(\phi, \mathcal{P}_t)_t\}_{t=1}^N = \{\mathcal{R}(\phi, \mathcal{P}_t)\}_{t=1}^N$. *Diff-SDF* optimizes the SDF parameters to reconstruct the object shape by reducing the reconstruction loss:

$$\mathcal{L}^R = \sum_{t=1}^{N} ||\mathcal{I}(\phi, \mathcal{P}_t)_t - \mathcal{I}_t|| \tag{5.1}$$

However, this approach encounters difficulties when applied to real-world videos due to the following reasons: *unknown camera poses* and *lack of views for unseen regions*. Because the SDF optimization presumes that camera poses of $\mathcal{I}_t$ are known in advance, which is not valid for real-world videos where camera poses are not provided. To estimate camera poses

$\{\mathcal{P}_t^{-1}\}_{t=1}^N$, we employ differentiable rendering to hierarchically produce an explicit surface mesh $\hat{\mathcal{O}}$ with texture denoted as $\hat{\mathcal{T}}$, and jointly estimate the objects poses $\{\hat{\mathcal{P}}_t\}_{t=1}^N$ over multiple images.

According to the first frame $\mathcal{I}_1$, our pipeline estimates the object shape at the coarse level denoted as $\mathcal{O}^{coarse}$ and selects the local frame as the geometry center of its bounding box. Based on the reconstructed mesh $\mathcal{O}^{coarse}$, we estimate the sequence of object poses among each frame denoted as $\{P_t\}_{t=1}^N$. With the reconstructed mesh $\mathcal{O}^{coarse}$ and associated poses $\{P_t\}_{t=1}^N$, we can render the sequences of images denoted as $\{\hat{\mathcal{I}}_t\}_{t=1}^N$. Starting from $\mathcal{O}^{coarse}$, we then refine the mesh so that the loss between rendered video and real video is reduced. In this stage, the mesh is refined according to loss between the whole sequence of images. With the updated mesh $\hat{\mathcal{O}}$, we render a sequence of new images and refine the poses to minimize the loss $\mathcal{L}^R$. We iteratively refine the object meshes and poses until the loss $\mathcal{L}^R$ is smaller than a given threshold or the process reaches the max iteration number. An overview of our pipeline is summarized in Alg 1. Details are available on the project website [125].

Algorithm 1 Shape and Pose Estimation

1: **Input:** $\{\mathcal{I}_t\}_{t=1}^N$ RGB video and $\{\mathcal{M}_t\}$ segmented object masks, loss threshold ϵ, max iteration L, current iteration l
2: **Output:** reconstructed mesh $\hat{\mathcal{O}}$ and the sequence of object poses $\{\mathcal{P}_t\}_{t=1}^N$
3: Estimate the coarse object mesh denoted as $\mathcal{O}^{coarse}$ based on the first image and define the object's local frame (for example, the geometry center of the bounding box)
4: Estimate the sequence of object poses denoted as $\mathcal{P}_t$ based on $\mathcal{O}^{coarse}$. Render the whole sequence of images and compare the loss $\mathcal{L}^R$
5: **while** $\mathcal{L}^R \geq \epsilon$ and $l < L$ **do**
6: Refine object mesh $\hat{\mathcal{O}}$ to minimize the loss $\mathcal{L}^R$
7: Refine object poses $\{\mathcal{P}_t\}_{t=1}^N$ to mimimize the loss $\mathcal{L}^R$
8: $l \leftarrow l + 1$
9: **end while**
10: Return $\hat{\mathcal{O}}$ and the sequence of object poses $\{\mathcal{P}_t\}_{t=1}^N$

We denote the optimized mesh, textures, and poses from differentiable rendering as $\hat{\mathcal{O}}^*, \hat{\mathcal{T}}^*, \{\hat{\mathcal{P}}_t^*\}$, respectively, and their associated rendered images $\hat{\mathcal{I}}_t^* = \mathcal{R}(\hat{\mathcal{O}}^*, \hat{\mathcal{T}}^*, \{\hat{\mathcal{P}}_t^*\})$ as:

$$\hat{\mathcal{O}}^*, \hat{\mathcal{T}}^*, \{\hat{\mathcal{P}}_t^*\} = \arg\min_{\hat{\mathcal{O}}, \hat{\mathcal{T}}, \{\hat{\mathcal{P}}_t\}} \sum_{t=1}^{N} \|\hat{\mathcal{I}}_t - \mathcal{I}_t\| \quad \text{where} \quad \hat{\mathcal{I}}_t = \mathcal{R}(\hat{\mathcal{O}}, \hat{\mathcal{T}}, \{\hat{\mathcal{P}}_t\}) \tag{5.2}$$

The quality of mesh $\hat{\mathcal{O}}^*$ is usually not satisfying. We then optimize the Diff-SDF to get an optimized SDF ϕ^* by setting the camera pose to be $\{\hat{\mathcal{P}}_t^{-*}\}$ to reduce the projection loss:

$$\phi^* = \arg\min_{\phi} \sum_{t=1}^{N} \|\mathcal{I}(\phi, \hat{\mathcal{P}}_t^*)_t - \hat{\mathcal{I}}_t^*\| \quad \text{where} \quad \hat{\mathcal{I}}_t^* = \mathcal{R}(\hat{\mathcal{O}}^*, \hat{\mathcal{T}}^*, \{\hat{\mathcal{P}}_t^*\}) \tag{5.3}$$

After the Diff-SDF optimization, the resulting surface mesh $\hat{\mathcal{O}}^{**}$ is then extracted from the SDF ϕ^*. The $\hat{\mathcal{O}}^{**}$ is then leveraged to optimize the object poses $\{\hat{\mathcal{P}}_t^{**}\}$ as below. The process above iterates until we get a small loss below a given threshold or reach the maximum iteration number.

$$\{\hat{\mathcal{P}}_t^{**}\}, \hat{\mathcal{T}}_t^{**} = \arg\min_{\{\hat{\mathcal{P}}_t\}, \hat{\mathcal{T}}} \sum_{t=1}^{N} \|\hat{\mathcal{I}}_t - \mathcal{I}_t\| \quad \text{where} \quad \hat{\mathcal{I}}_t = \mathcal{R}(\hat{\mathcal{O}}^{**}, \hat{\mathcal{T}}, \{\hat{\mathcal{P}}_t\}) \tag{5.4}$$

Although each video contains multiple frames, there are still cases that lack sufficient views, resulting in poorly reconstructed unseen regions of the object. To address the incomplete views, we adopt a diffusion model [59] to infer the unseen areas. Our model takes the first real image $\mathcal{I}_1$ with a known camera pose and synthesizes images from different viewpoints around the object. We then combine these synthetic views with others for a complete object-shape reconstruction.

After solving the optimization, we apply $\mathcal{P}^{-1}$ to the tracked object pose $\hat{\mathcal{P}}_t^{**}$ to obtain the pose of the manipulation object for the contact-aware manipulation $\mathcal{P}_t$.

5.4 Contact-Aware Manipulation Policy

Building on the estimated object's pose and shape, this subsection delves into the process of manipulating an object between two consecutive poses $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$. If the robot is provided with a similar but different object from the object recorded in the video, we align the pose of the provided manipulated object with the reconstructed object. The alignment is solved as a pose estimation problem:

$$\arg\min_{\mathcal{P}} \mathrm{d}(\hat{\mathcal{P}}_0^{**}\hat{\mathcal{O}}^{**}, \mathcal{P}\mathcal{O}) \tag{5.5}$$

where d is the chamfer distance between two meshes, $\hat{\mathcal{P}}_0^{**}$ is the pose of the reconstructed mesh at the first frame, $\hat{\mathcal{O}}^{**}$ is the reconstructed mesh, $\mathcal{P}$ is a transformation to the manipulation object, and $\mathcal{O}$ is the manipulation object.

Our framework employs a hierarchical structure consisting of low-level modules for contact point localization and contact-force optimization, as well as high-level contact sequence planning. The low-level modules serve dual purposes: contact-point localization allows the robot to establish new contacts while keeping the object stationary, whereas contact-force optimization enables the robot to manipulate the object toward its target and maintain stable contact. These low-level actions are then orchestrated by the high-level contact sequence planning module to form a cohesive sequence of actions. To facilitate efficient and robust real-time deployment, we also incorporate a neural policy designed to imitate the planned trajectories.

Contact point localization. Contact point localization enables the robot to change contact points on the object, which encompasses two critical steps: *the generation of the transition target* and *the execution of the transition.* As shown in Fig. 5.2, the transition target

is calculated analytically with the desired object transformation wrench $\mathcal{W}$. Wrench $\mathcal{W}$ is located at the objects' center and represents the necessary wrench to facilitate the transformation from $\mathcal{P}_t$ to $\mathcal{P}_{t+1}$. It is determined using a Proportional-Derivative (PD) controller: $\mathcal{W} = k_p * (\mathcal{P}_{t+1} - \mathcal{P}_t) - k_d \dot{\mathcal{P}}_t + g$, where k_p, k_d are the proportional and derivative gains, and g signifies the gravitational and external forces acting upon the object. Following this, we use an enumeration process to identify all plausible contact combinations that can generate the desired wrench $\mathcal{W}$. Initially, all potential contacts are assessed to single out those capable of producing the desired object wrench $\mathcal{W}$ through contact points $\{p_i\}$ for $i \in [1..n]$. The number of contact points n is pre-determined based on the manipulation task. Further filtering processes are implemented to ascertain contacts that meet kinematic and stationary constraints: the inverse kinematics are solved to verify kinematic feasibility, and only one contact point can move at a given time while the remaining contact points hold the object immobile. Although multiple contact points could theoretically move while maintaining the object stationary, we found that planning is considerably more complex due to the enlarged search space, and the objects are prone to unintended movement due to execution errors. To execute the transition, we use the Rapidly-exploring Random Tree (RRT) motion planner to generate a feasible trajectory for the moving contact and the gravity compensation wrench on the remaining contacts to hold the object.

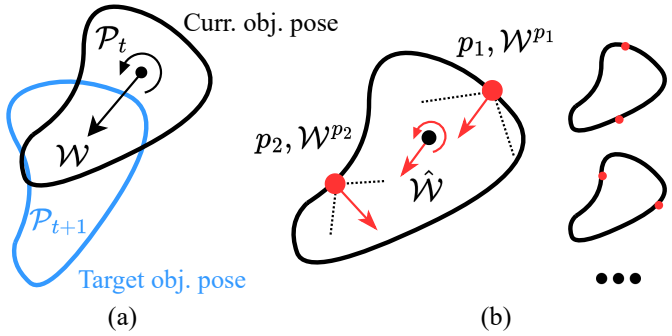


Figure 5.2: Contact points inference in the 2D case with $n = 2$ contact points. (a) Given object current and target poses $\mathcal{P}_t, \mathcal{P}_{t+1}$, the desired object wrench $\mathcal{W}$ is computed to produce the transformation. (b) Enumerate all contact points combinations $\{p_i\}$ (red dots) to find those that can produce the target wrench by applying contact wrenches $\{\mathcal{W}^{p_i}\}$ (red arrows) within the friction cone (black dashes). The contact wrenches $\{\mathcal{W}^{p_i}\}$ are equivalently transferred to the object coordinate as $\hat{\mathcal{W}}$ to compare with the desired object wrench $\mathcal{W}$.

Contact wrench optimization. Once the contact points $p = \{p_i\}$ are determined, the robot exerts contact wrenches $\mathcal{W}^p = \{\mathcal{W}^{p_i}\}$ at contact points to manipulate the object toward its target. The objective and loss functions to optimize the contact wrenches are defined in (5.6).

$$\min_{\mathcal{W}^p} \mathcal{L}(\mathcal{W}^p) = \lambda_{\mathcal{P}} \left\| \mathcal{P}' \ominus \mathcal{P}_{t+1} \right\| + \lambda_v \left\| \dot{\mathcal{P}}' \right\| + \lambda_{\mathcal{W}} \left\| \mathcal{W}^p \right\| \quad \text{subject to} \quad \mathcal{P}' = \mathcal{F}(\mathcal{P}, \mathcal{W}^p) \tag{5.6}$$

where $\mathcal{F}$ represents the contact dynamics, $\mathcal{P}'$ is the 6D object pose after applying the wrench $\mathcal{W}^p$ from the initial object pose $\mathcal{P}$. $\mathcal{P}_{t+1}$ is the target object pose, $\ominus$ represents subtraction for 6D poses. $\dot{\mathcal{P}}'$ represents the object's velocity and is added to damp the object's speed, making the manipulation more stable [97]. $\lambda_{\mathcal{P}}, \lambda_v, \lambda_{\mathcal{W}}$ are hyperparameters standing for loss weights.

We propose using gradient-based methods to optimize the objective function, (5.6), with differentiable simulation in Nimble Physics[83] to approximate the forward dynamics $\mathcal{F}$. We use $s = (\mathcal{P}, p)$ as the concatenated state of the system in the following of this chapter. The gradient of the objective can be computed as $\nabla = \frac{\partial \mathcal{L}}{\partial \mathcal{W}^p}$ from the simulation. However, gradients near the contact are often nonlinear, sensitive, and discontinuous, posing challenges for vanilla gradient descent optimization methods. To address this issue, this chapter draws inspiration from [89, 102, 1, 139] and proposes computing the gradient expectation at each point with Gaussian noises, as shown in (5.7). The contact wrench is then updated using a step size α along the gradient direction. In this chapter, we use the analytical contact wrench computed during the contact point localization as the initial solution point for the optimization process.

$$\nabla = \mathbb{E}_{n_s, n_{\mathcal{W}} \sim \mathcal{N}} \left[\frac{\partial \mathcal{L}(\mathcal{F}(s + n_s, \mathcal{W}^p + n_{\mathcal{W}}), \mathcal{W}^p + n_{\mathcal{W}})}{\partial \mathcal{W}^p} \right] \tag{5.7}$$

Algorithm 2 Global Planning for Manipulation Sequences

Input: $s_0 = (\mathcal{P}_t, p_t)$, target object pose $\mathcal{P}_{t+1}$
Output: $\mathcal{R} = \{s\}$
$\mathcal{Q} \leftarrow \{s_0\}$, $\mathcal{R} \leftarrow \{\varnothing\}$ ▷ Init.
while $\mathcal{Q}$ is not empty **do**
  $s \leftarrow$ `SelectNode`$(\mathcal{Q})$
  if `IsSuccess`$(s, \mathcal{P}_{t+1})$ **then**
    Return $\mathcal{R}$ ▷ Exit if success
  end if
  $s' \leftarrow$ `OptWrench`$(s, \mathcal{P}_{t+1})$ ▷ Opt. wrench
  if `OptIsSuccess`(s, s') **then**
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup s'$; $\mathcal{R} \leftarrow \mathcal{R} \cup s'$
  else
    $\mathcal{S} \leftarrow$ `ContactLoc`$(s, \mathcal{P}_{t+1})$ ▷ Loc. contacts
    for $s' \in \mathcal{S}$ **do**
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup s'$, $\mathcal{R} \leftarrow \mathcal{R} \cup s'$
    end for
  end if
end while
Return $\mathcal{R}$

High-level planning. Our global contact sequence planning, as detailed in Algorithm 2, employs hierarchical planning to identify viable manipulation sequences, utilizing the previously introduced contact point localization (`ContactLoc`) and contact wrench optimization (`OptWrench`). While the exertion of contact wrenches allows the robot to perform manipulation tasks involving nearby target poses, switching between multiple contacts is necessary when dealing with distant targets due to kinematic limitations.

Every node s in the planning tree encapsulates the object pose $\mathcal{P}$ and contact p. The tree begins with a start node that represents the initial object pose and robot contact, with the goal of reaching the target object pose $\mathcal{P}_{t+1}$. At each iteration, a node s is chosen and expanded using `ContactLoc` or `OptWrench` following A* search [37]. If the object has been successfully manipulated through the exertion of an optimized contact wrench (`OptIsSuccess`), the resulting node s', containing the manipulated object pose and contact points, is expanded. Conversely, if the exertion fails, contact localization is performed by identifying a new set of contacts and expanding them in the tree. This planning algorithm continues until either the target object pose is reached (`IsSuccess`) or all nodes within the tree have been explored. The search procedure we propose focuses on optimizing contact wrenches at first and resorts to locating new contacts only if the exertion fails. Although this approach narrows down the search space, it may also prune potentially valid and optimal paths. For instance, transiting contacts before reaching the kinematic limit might result in a shorter trajectory with fewer contact switches. To address this, we introduce a

random chance for each node to transition contacts, regardless of the wrench optimization outcome. This design promotes exploration within the planning process, enabling a more comprehensive discovery of the entire search space.

Sim2Real: closed-loop policy with domain randomization. Despite its efficacy in generating viable manipulation trajectories, the above-described planning algorithm is computationally demanding as it requires online enumeration of contacts and wrench optimization, rendering it unsuitable for real-time applications. To surmount this challenge, we utilize deep learning to approximate the manipulation policy. We leverage a fully connected network to learn the robot control commands that were derived from the high-level planning algorithm. This network ingests the object pose and joint angles as inputs and outputs of robot joint torques. These torques are obtained by mapping the contact wrenches into joint torques, courtesy of the Jacobian. Our training dataset is generated by solving the planning problem under conditions of noisy initial and target positions and perturbed system dynamics. This process results in a set of state-torque training pairs. We further augment each sample by introducing noise into the states and optimizing the joint torques to adhere to the planned trajectory. It's noteworthy that for domain randomization, we optimize the contact wrench to reach the next state along the trajectory rather than solving the original planning problem with a distant target. This makes the data augmentation process more efficient. In a bid to further enhance performance, we fine-tune the network within a Markov Decision Process (MDP) framework using the REINFORCE algorithm [126]. During the fine-tuning process, the state and action spaces maintain the same setup as described earlier, while the reward function is defined as $r(s, \tau) = -\lambda_{\mathcal{P}} \left\| s \ominus \mathcal{P}_{t+1} \right\| - \lambda_v \left\| \dot{s} \right\| - \lambda_{\mathcal{W}} \left\| \tau \right\|$, where $\mathcal{P}_{t+1}$ is the target object pose, and $\dot{s}$ denotes the object velocity [97]. $\lambda_{\mathcal{P}}, \lambda_v, \lambda_\tau$ are hyperparameters.

5.5 Experiments

This section offers both quantitative and qualitative assessments of our proposed methodology. Our experiments are designed to address the following research questions: 1) How does our Diff-LfD framework compare to baselines that also rely on visual demonstrations? 2) What is the efficacy of our contact-aware manipulation algorithm in generating long-horizon trajectories? 3) Is our approach feasible for deployment in real-world scenarios? 4) How accurate is our self-supervised object reconstruction and tracking? 5) What is the utility of the views synthesized by the diffusion model? 6) What impact do gradient-based optimization, global planning, and random contact transition have on performance? 7) How robust are the generated trajectories and the closed-loop policy?

Experimental setups and hyperparameters. In this work, we evaluated the proposed algorithms through two experimental settings: 1) basic manipulation actions on primitive objects and 2) in-hand object manipulation. For the basic manipulation tasks, we created environments using the Nimble Physics engine based on the 20BN-something-something

(sth-sth) dataset [33]. The dataset contains labeled video clips of human demonstrations performing pre-defined basic manipulations. These environments were equipped with a Kinova MOVO robotic manipulator and primitive objects placed on a table. For the in-hand manipulation, we utilized the Allegro Hand to rotate objects. The Allegro Hand has four fingers, each with four joints controlled independently. The palm of the robotic hand is maintained in a fixed position, and fingers grasp the object. In comparison to previous studies such as [89, 11], our experiments involved lifting the object off the palm rather than having it lie on the palm, thus increasing the challenge for the hand to maintain its grasp on the object during the manipulation process. For the baseline comparison, the human demonstrated an in-hand object rotation of 180 degrees.

We do not derive the physical parameters, such as mass, from observations. Instead, for each planning trial, they are sampled from a uniform distribution. In real-world experiments, we employ these randomized physical parameters during planning to amass a dataset, which is then used to train the closed-loop policy.

All computations were performed using an Intel 3.6GHz CPU with 64GB RAM, with the exception of the closed-loop policy, which was trained using a GTX3080 GPU. The simulation time step was set to 1e-4 in the Nimble simulation for dynamical stability and accuracy. Each experiment was conducted seven times with different seeds to report the mean and variance in the following experiments. The hyperparameters are as follows: $k_p = 200$ and $k_d = 10$ were used for desired wrench computation; $n = 1$ was used for manipulation of primitive objects and $n = 4$ for in-hand manipulation. A threshold of $\delta = 0.01$ was set for wrench projection error. We set $\lambda_{\mathcal{P}} = 2.0$, $\lambda_v = 0.3$, and $\lambda_{\mathcal{W}} = \lambda_\tau = 0.5$. Gaussian noise was injected to smooth the gradient with $n_s \sim \mathcal{N}(0, 0.01)$ and $n_{\mathcal{W}} \sim \mathcal{N}(0, 0.1)$. Contact wrench optimization was deemed successful (`OptIsSuccess`) if $|\mathcal{P} \ominus \mathcal{P}'| \geq 0.05$, and the search algorithm was successful (`IsSuccess`) if $|\mathcal{P} \ominus \mathcal{P}_g| \leq 0.02$. During each iteration of the tree search, we randomly selected a node to transit contact with a probability of 0.3 to balance exploration and exploitation.

To account for the fragility and time-sensitive nature of in-hand manipulation, we trained the closed-loop policy specifically for this task. For the manipulation of primitives, we used the search algorithm for iterative replanning. Our trajectory dataset comprises 200 trajectories for each initial and target pose and $87,300$ state-wrench pairs. To ensure variability, object initial rotations were noised, with rotational offset drawn around the demonstrated initial pose within $10°$. We also determined the initial contact to stably grasp the object. Each transition was augmented by uniformly distributed noise with $5°$ on rotation and $0.005cm$ on translation. The closed-loop policy takes the state input $s \in \mathbb{R}^{6+16}$ and determines the contact wrench $\mathcal{W}^p \in \mathbb{R}^{4*6}$, as the Allegro Hand has 16 DoFs and 4 fingers. We fine-tuned the policy for $20,000$ steps in the simulation to obtain a robust policy. During training and fine-tuning, we used the Adam optimizer with a learning rate of $1e-4$ and a linear schedule.

Baseline comparisons on LfD framework. We compare our model-based approach with the method introduced in [92], which presents an optimization-based method to estimate a coarse 3D state representation, using a cylinder for the hand and a cuboid for the

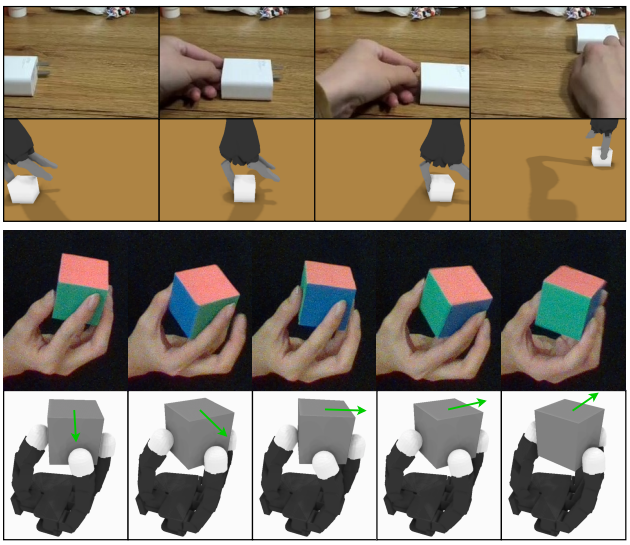


Figure 5.3: Experimental results from sth-sth (1st & 2nd rows) and in-hand object manipulation (3rd & 4th rows).

Table 5.1: Baseline comparisons on LfD framework. Each cell represents the success rate of the manipulation.

	Pull Right	**Pull Left**	**Push Right**	**Push Left**
Baseline [92]	0.976	0.992	0.994	0.946
Ours	1.000	1.000	1.000	1.000

manipulated object(s). Such coarse approximation limits the representation capability and the quality of the state estimation. We utilize our object reconstruction and tracking to estimate the object trajectory and use contact planning to find a path. We select videos of 4 classes from the sth-sth dataset [33]: "Pull Right" with 164 videos, "Pull Left" with 130 videos, "Push Right" with 89 videos, and "Rush Left" with 253 videos. We report the results as in Fig. 3. Our approach successfully finished all the classes and slightly outperformed the method introduced in [92]. One explanation is that these four types of videos are simple for our proposed pipeline to imitate. Thus, we also apply our method for in-hand manipulation tasks from raw videos to test the limits of our proposed framework. Fig. 5.3 shows the manipulation trajectory in two environments generated by our method.

Baseline comparisons on shape reconstruction and pose estimation. In contrast to other learning-based approaches for shape reconstruction and pose estimation, such as Neural Radiance Fields (NeRF), our perception module operates under a distinct task setting. Specifically, our input consists of a single-object RGB video featuring objects that undergo both rotation and translation. Most NeRF-based methods, on the other hand, rely

Table 5.2: Baseline comparisons on contact-aware manipulation policy. The first element in each cell is the mean/variance for the computation time (s); the second is the difference between the target and final object rotation (°).

	RRT		**CITO**		**PGDM**		**iLQR**		**Ours**	
Ball	122 ± 20	7.2°	52 ± 8	16.7°	2.14 ± 0.4	2.4°	57 ± 10	11.2°	62 ± 12	2.6°
Cube	136 ± 16	9.0°	60 ± 7	18.5°	2.16 ± 0.3	4.1°	70 ± 19	13.3°	78 ± 16	3.8°
Capsule	127 ± 24	8.4°	63 ± 4	15.2°	2.18 ± 0.3	9.3°	80 ± 6	12.2°	81 ± 8	7.7°

on multiple static object poses with known camera positions. Some NeRF implementations utilize COLMAP [108] to initialize camera poses. However, this approach is less effective in our setting, where the background remains largely unchanged and the frame count is limited. These factors hinder COLMAP's ability to accurately estimate object poses, leading to unstable object surface reconstructions from NeRF. Further experimental comparisons with Nope-NeRF [7], which also employs COLMAP for initialization, are available on the website [125]. Our findings indicate that Nope-NeRF fails to converge in more than half of the test cases (5 out of 9), resulting in empty reconstructions. The remaining cases yielded incorrect pose estimations and reconstructions when compared to our method.

Baseline comparisons on the contact-aware manipulation policy. In this study, we evaluate our contact-aware trajectory planning algorithm against four established baselines within the context of in-hand object rotation tasks. The baselines are as follows: 1) The Rapidly-Exploring Random Tree (RRT) planner, as outlined in [89], employs random sampling within a configuration space defined by both robot joint positions and object poses to identify feasible trajectories. 2) Contact-Implicit Trajectory Optimization (CITO) [11] first establishes a predefined trajectory for the object, then identifies optimal contact points along this path before calculating the requisite control inputs for trajectory tracking. 3) Pre-Grasp Informed Dexterous Manipulation (PGDM) [18] utilizes reinforcement learning to train manipulation agents, incorporating pre-computed grasp data to achieve the desired manipulation trajectory. 4) The Iterative Linear Quadratic Regulator (iLQR) [89] employs local approximations of the dynamical system to iteratively solve for optimal manipulation strategies through quadratic planning. For the purposes of this experiment, our algorithm operates without the closed-loop policy. We apply baselines on three in-hand manipulation tasks associated with the ball, the cube, and the capsule. We adopt two evaluation metrics: the averaging planning time and the difference between the target rotation and the final object rotation. Results are reported in Fig. 4.

While both PGDM and iLQR boast the quickest inference times, it's crucial to highlight that PGDM requires approximately 5 hours of training for each task, and iLQR suffers from a higher tracking error compared to our method. The RRT approach uniformly expands its search tree, thereby increasing the probability of encountering unstable contacts and consequently requiring the most time to complete the task. In contrast, our algorithm and CITO focus on a more constrained search space where stable grasping is feasible, thereby simpli-

Table 5.3: Shape Reconstruction and Pose Estimation Error. CD represents Chamfer Distance

	Chamfer Distance	Translation Error	Rotation Error
After Stage a	4.9	-	-
After Stage b	2.1	2.0	0.27
Iterative Refinement	1.7	0.5	0.25

fying the search complexity. Furthermore, our empirical results indicate that the final error rates for all baseline methods were consistently higher than our algorithm. Specifically, the RRT approach lacks a guarantee for optimal trajectory sampling, CITO overlooks physical dynamics during the planning phase, and iLQR struggles with optimization over nonlinear loss contours. These limitations render the baseline methods susceptible to failure due to dynamic uncertainties and execution errors.

Real-world experiments. We conducted real-world experiments for in-hand object manipulation. The experimental setups are illustrated in Fig. 5.4. We trained the closed-loop policy to imitate a human rotating a cube and deployed it as the robot controller. This network receives the current joint angles of the robot, and the current object poses as input and outputs the joint torques. We performed the in-hand manipulation task with the Allegro Hand for different primitive objects and initial poses. During supervised learning, convergence of the policy is achieved in approximately 9.3 minutes, while fine-tuning takes an average of 50.1 minutes. The results of the difference between the target and final object rotation errors are Cylinder (3.8°), Ball (2.4°), Lemon (6.3°), and Avocado (5.9°), which further underscore the ability of our closed-loop policy to generalize across similar but distinct geometries.

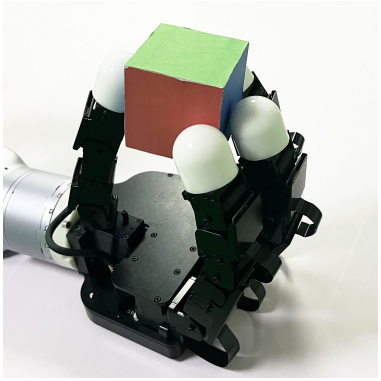


Figure 5.4: Allegro Hand performs in-hand object manipulations.

Ablation studies of shape reconstruction and pose estimation. We randomly select 50 daily objects from Shapenet, generating 6D trajectories and rendering videos as demonstrations. We apply our proposed approach and report the shape reconstruction error and pose estimation error at different stages in Table 5.3. For the 6D pose estimation error, we report both the translation error and rotation error. The rotation error denoted as e_ω is represented as $\min(||q_{pred} - q_{gt}||_2, ||q_{pred} + q_{gt}||_2)$. We adopt the quaternion representation for q_{pred} and q_{gt}, which stands for the predicted and ground truth quaternion, respectively. As the results show, the iterative refinement stage (which jointly optimizes object shape and pose within the whole sequence of images) significantly reduces the shape reconstruction error, rotation error, and translation error.

Table 5.4: Computation time (s) on the in-hand manipulation task with a rotation target of 180°. Ablation on the contact force optimization methods.

	CMA-ES	**GD**	**Ours**
Ball	104 ± 15	84 ± 8	62 ± 12
Cube	121 ± 18	95 ± 11	78 ± 16
Capsule	129 ± 10	103 ± 16	81 ± 8

To evaluate the effect of the synthesized views, we perform an ablation comparison for the diffusion model. Visual demonstrations are available on our website [125]. The hallucinated or unseen side of the object is more reasonable with the diffusion model. Specifically, we propose to utilize the pre-trained Zero 1-to-3 diffusion model in our approach. Zero 1-to-3 is a specialized framework designed for novel view synthesis from a single RGB image. During the inference phase, the Zero 1-to-3 diffusion model takes the input view and a relative viewpoint as conditional information, synthesizing the corresponding novel view. It's important to note that we do not train diffusion models within our pipeline. Instead, we leverage these pre-trained diffusion models, honed on extensive datasets, to provide supervision on unseen views.

Ablation studies of contact-aware planning. To evaluate the benefit of using differentiable simulation for contact force optimization, we compare the smooth gradient technique to the sampling-based and vanilla gradient descent (GD) methods. For the sampling-based solver, we used the covariance matrix adaptation evolution strategy (CMA-ES) to optimize the contact force. For the vanilla GD, we computed the explicit gradient and updated the contact force with a fixed step size $\alpha = 0.1$. These methods have experimented with the in-hand manipulation task with a rotation target of 180 degrees. The closed-loop policy is not employed, and each experiment is repeated seven times with different seeds to report the mean and variance of the results. From experiments, we observed that all three methods yield a similar solution given adequate computation time. A similar phenomenon has been observed in [139]. This is because solvers can converge to a close point after iterative optimization. Thus, Table 5.4 reports the computation time required to converge. An experimental trial planned by our method is displayed in Fig. 5.5.

The results of our experiments indicate that our method requires 1.6× and 1.4× less computation time compared to the sampling-based and vanilla gradient descent (GD) optimizers, respectively. Additionally, the gradient-based methods (i.e., vanilla GD and our proposed method) were found to converge faster than the sampling-based approach due to their ability to leverage the first-order local gradient information. Furthermore, the results show that the trajectories optimized using gradient-based methods are less jerky compared to those optimized using the sampling-based approach. This is likely due to the smooth gradients provided by the optimizers.

In addition, this experiment evaluated the effectiveness of the global planning module

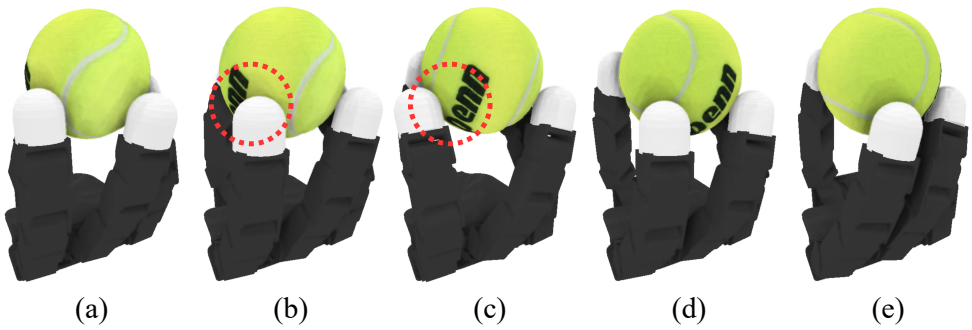


Figure 5.5: (a-e) display a trial of a robotic hand rotating a ball. The rotation of the object can be identified by the position of the logo. The red circles in images (b) and (c) show the change in contact points.

Table 5.5: Error between the target and final object rotation angle (°) on the in-hand manipulation task with a target of 180°. Ablation on whether to include the global planning (GP) module and random contact transition (RCT) module.

	Ours	**Ours w/o GP**	**Ours w/o RCT**
Ball	2.6 ± 2.5	127.2 ± 5.8	27.5 ± 13.5
Cube	3.8 ± 3.2	139.7 ± 7.1	31.6 ± 17.8
Capsule	7.7 ± 10.4	156.8 ± 9.0	35.1 ± 14.4

Table 5.6: Error between the target and final object rotation (°) angle under variant noises.The manipulation trajectory is generated by the closed-loop policy trained to rotate objects.

No Noise	**State**	**Execution**	**Inertial**	**Friction**
4.7 ± 0.3	4.7 ± 0.5	4.8 ± 0.4	5.3 ± 0.7	5.7 ± 0.5

and the random contact transition in the same in-hand manipulation task. Table 5.5 shows the final rotation error for each ablation. The results demonstrate that the global planning module significantly improves the object rotation angle through finger gaiting. Manipulation towards a distant target is challenging with only contact wrench optimization. However, our proposed method addresses this challenge by combining both local optimization and global contact switching. The combination of these two strategies leads to improved results. Additionally, the study shows that the random contact transition improves the results by exploring a larger search space. We hypothesize that the random transition can locate better manipulation sequences with a larger set of visited states, thus increasing the probability of finding feasible paths.

Robustness to the noise. The aim of this experiment is to evaluate the stability and robustness of the closed-loop policy for in-hand manipulation. Despite the availability of adequate trajectories around the manipulation sequence in the training dataset, the closed-loop policy assumes known state and deterministic dynamics, as explained in the previous section. Such an assumption can render the policy susceptible to failure in the face of out-of-distribution uncertainties, such as state observation noises, execution errors, and frictional and inertial uncertainties. State noises arise from object tracking issues and noisy sensor readings, while execution errors stem from robot controller and motor movement errors. Frictional and inertial uncertainties typically result from inaccurate system identification and the gap between simulation and reality.

To assess the robustness of the closed-loop policy, we added artificial Gaussian noises to the system during the manipulation to test the performance, including state noises, contact wrench noises, object friction noises, and inertia noises. Specifically, the state noise was sampled from $\mathcal{N}(0, 0.01)$, the contact wrench noise from $\mathcal{N}(0, 0.1)$, the object friction noise from $\mathcal{N}(0, 0.3)$, and the inertial noises from $\mathcal{N}(0, 1e-5)$. We deployed the closed-loop policy to perform the rotation task with different types of noise. The policy knows neither the noise type nor the noise value during execution. The experiments showed that the closed-loop policy could achieve the manipulation task without the object falling. Table 5.6 presents the average final rotation error for the fine-tuned policy.

The results suggest that the closed-loop policy can handle state and execution noises effectively, with consistent errors across experiments. One possible explanation for this is that we trained the policy on a dataset with randomized states and augmented trajectories with noise. Moreover, the training dataset for the contact strategy was collected using a smooth gradient approximation, which may have contributed to its robustness against execution noises, as demonstrated in [89]. However, we did observe that introducing noise to the friction and inertia coefficients affected the planning results, likely because the policy was not trained to account for such shifts in dynamics. Nonetheless, we believe that fine-tuning the policy could mitigate these covariant shifts, as shown in [97]. While this is beyond the scope of our learning from the demonstration algorithm, we acknowledge that such experiments could be explored in future work.

5.6 Chapter Summary

This chapter investigates the use of model-based learning from demonstrations for robotic manipulation tasks, contributing several significant aspects to the field. First, we introduce a new framework for learning from human visual demonstrations in a self-supervision manner, which has the potential to generate robot skills at a large scale. Second, we utilize differentiable rendering to track object poses in a self-supervised manner. Third, we design a high-level planning framework that employs differentiable simulations to generate long-horizon contact actions. This includes inferring and transitioning contact points, optimizing contact forces, and exerting them. The manipulation trajectories are then approximated

by a neural network. Finally, we conduct experiments to evaluate the effectiveness of our approach from multiple angles. Our results demonstrate the robustness and efficiency of our proposed method to learn from human demonstrations and outperform existing approaches by a large margin. Experimental videos are available on the project website [125].

Chapter 6

Manipulation with Safe-Contact by Null Space Impedance Control

6.1 Introduction

While Chapter 5 and related studies [137, 70] have contributed significantly to the development of manipulation planning algorithms, their primary focus has been on strategies that avoid contact between robot arms and their surrounding environment. This conventional approach, however, does not fully encapsulate the nuances of human manipulation strategies, where contact with obstacles is not merely incidental but often deliberately employed. For instance, in tasks like shelf picking, robots may need to make contact with surrounding obstacles to effectively grasp the target object [22, 135]. Similarly, in agricultural contexts such as harvesting, robots can benefit from pushing aside occluding elements like petioles and leaves to access produce [84].

In this chapter, we advocate for a paradigm shift in robotic manipulation by proposing the integration of contact as a strategic element. This approach aims to enhance robotic efficiency and expand capabilities by broadening the range of feasible actions [94]. The benefits of allowing contact are twofold, as illustrated in Fig. 6.1. Firstly, it facilitates the generation of more efficient trajectories in terms of both length and smoothness, compared to the trajectories produced under strict collision-free constraints. Secondly, in scenarios densely populated with obstacles, where collision-free solutions are unattainable, this approach enables the creation of viable manipulation strategies.

One of the most critical concerns in contact-allowed manipulation is safety. Without considering safety, the robot can hit obstacles heavily and damage the hardware. Safety constraints can be imposed by restricting the contact forces [101], and we impose this constraint by online generating and tracking of the reference signals. For the prior, a constraint can be applied during trajectory optimization to limit the contact forces. For the latter, a robot controller can leverage null space compliance to reduce the contact force while tracking the desired trajectory [104].

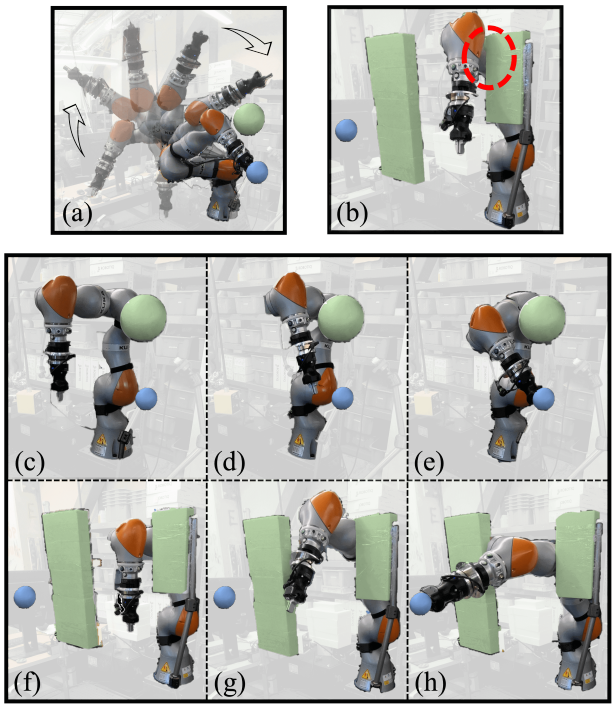


Figure 6.1: **(a)** The robot finds a long and unnatural trajectory to reach the goal (blue), avoiding the obstacle (green). **(b)** The goal-reaching robot fails to find a collision-free trajectory; the red circle indicates collision with the obstacle. By allowing safe contact, the robot reaches the goal more efficiently **(c-e)** and finds a trajectory in the highly collisional scenario **(f-h)**.

This chapter studies the safe contact-allowed robotic goal-reaching problem with two feedback control loops. The outer loop optimizes the time-varying operational and joint trajectories in a receding horizon manner. The dynamics model of the robot and objects is approximated with a differentiable simulator, Brax [31], and utilized to impose contact constraints. The inner loop tracks the trajectory using an impedance controller. This chapter focuses on redundant manipulation, where the robot is kinematically redundant to the task, and includes a null space projector in addition to an operational space control. Compared with other works, our method optimizes joint configuration as extra decision variables and actively explores the null space for a safer motion. To solve the trajectory planning, we propose a hybrid algorithm that integrates sampling- and gradient-based methods, gaining the benefit of scalability and differentiability.

We empirically evaluate the proposed method in various simulation and real-world environments to demonstrate the effectiveness of contact-allowed planning and control. In summary, our work makes the following contributions: First, we state the contact-allowed

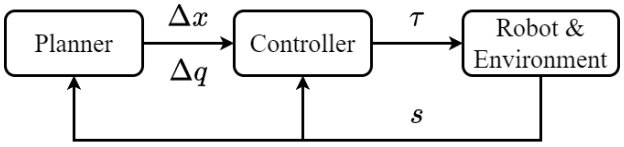


Figure 6.2: The planning loop takes in state s and optimizes reference signals (i.e., operational space references Δx and null space references Δq) for the robot controller, which generates joint torques τ as actuation commands.

robotic goal-reaching task with safety constraints. We provide open-source environments and benchmarks for the task. Second, we propose a time-varying trajectory planner and tracking controller for the contact-allowed problem. The planner optimizes both operational and null space reference signals to achieve the goal efficiently and safely. Third, we present a hybrid optimization solver for the trajectory planner. The superiority of the proposed algorithm is evaluated in diverse experiments ranging from free space to highly collisional.

6.2 Contact-Allowed Robotic Goal-Reaching

Problem Formulation

This chapter chooses goal-reaching as the operational manipulation task. It requires the end-effector to reach a given goal pose, while the manipulator can collide with the environment with a maximum permitted contact force. Relating to the formalisms above, a hierarchical framework is utilized to generate and track robotic motions, as shown in Fig. 6.2. The outer planner takes the robotic and environmental state s as input and optimizes operational and joint space motions $a = \{\Delta x, \Delta q\}$; the input includes robot joints and obstacle information[1]. The inner robot controller computes joint torques τ based on the references to actuate the robot.

The 3D goal pose is denoted as s_g. The success criteria is defined as $\|s_t - s_g\| \leq \delta$ ($\delta = 0.01m$ in this chapter), where $\|\cdot\|$ computes the translational distance from the end-effector to s_g. The system propagates with the transfer function $T(s_t, \tau_t)$ as the robot executes the torque command at t, which computes the successive state s_{t+1} and the contact force. The maximum permitted contact force is written as ε.

[1]Poses for rigid obstacles; node positions for deformable obstacles represented by node graphs, as in Fig. 6.3(c).

Operational and Null Space Impedance Control

In this chapter, we adopt operational space computed torque control [41] for the manipulation task and introduce null space projection [104] for safe contact. Previous works have shown the advantage of planning task-related motions in the operational space over the joint space [70]. Not only the manipulation tasks are typically defined in the operational space, but also it is straightforward to adapt the controller among different robots. Moreover, null space projection allows compliance at the joint level for a redundant robot without disturbing the operational task, thus improving safety by reducing contact forces.

This chapter uses the computed torque control [77] as the operational space controller. It first calculates the wrench for the end-effector based on the desired motion and then maps the wrench to joint torques. The control law is written as (6.1).

$$\tau_{op} = J(q)^T \Lambda(q) \left(-K_p \Delta x - K_d \dot{x}\right) + C(q, \dot{q}) + g(q) \tag{6.1}$$

For a robotic manipulator with n joints, $q \in \mathbb{R}^n$ is the joint angle, $x \in \mathbb{SE}(3)$ is the end-effector pose, $J(q)$ is the task Jacobian, $\Lambda(q) = (J(q)M^{-1}(q)J(q)^T)^{-1}$ is the operational space inertial matrix and $M(q) \in \mathbb{R}^{n \times n}$ is the robot inertial matrix, $C(q, \dot{q}) \in \mathbb{R}^n$ is the Coriolis and Centrifugal forces, $g(q) \in \mathbb{R}^n$ is the robot gravity vector. We use $\Delta x = x \ominus x_d$ to represent the error in $\mathbb{SE}(3)$, where x_d is a desired operational pose. $K_p, K_d \succeq 0$ are PD gains for the controller. The following sections will drop the dependencies in $q, \dot{q}$ for better readability.

The null space control projects joint torques into the null space with the projector $N = I - J^\dagger J$, where $J^\dagger = M^{-1} J^T \Lambda$ is the dynamically consistent inverse of the Jacobian [47, 137]. The projected torque does not affect the motion in the operational space. We implement a joint space PD controller to generate and project the torque:

$$\tau_{null} = N^T \left(-K_{qp} \Delta q - K_{qd} \dot{q}\right) \tag{6.2}$$

, where $\Delta q = q - q_d$ and q_d is a desired joint posture. $K_{qp}, K_{qd} \succeq 0$ are control gains in the joint space.

Combining the operational space control (6.1) with the null space projection (6.2), the final control output is defined as:

$$\tau = \tau_{op} + \tau_{null}. \tag{6.3}$$

Contact-Allowed Motion Planning

Section 6.2 introduces an operational space controller with null space compliance. The control law (6.1, 6.2, 6.3) takes in the reference signals $a = \{\Delta x, \Delta q\}$ to compute the actuation torque. This section introduces how to find the time-varying reference signals for efficient and safe robotic manipulation.

The problem is formulated in a receding horizon manner. For each time step k, the optimization is written as follows.

$$\min_{a_{k:k+H-1}} \sum_{t=k}^{k+H-1} \lambda_1 \|s_{t+1} - s_g\| - \lambda_2 \|s_{t+1} - s_k\| \tag{6.4a}$$

$$\text{s.t.} \quad \tau_t = f(a_t) \qquad \triangleright \ (6.3) \tag{6.4b}$$

$$s_{t+1} = T(s_t, \tau_t) \tag{6.4c}$$

$$\text{contact}(s_t, a_t) \leq \varepsilon \qquad \text{for } t \in [k, ..., k+H-1] \tag{6.4d}$$

The objective function (6.4a) minimizes costs over the horizon H. s_k is the initial state at the current planning horizon. $\lambda_1, \lambda_2 > 0$ are hyper-parameters. $\|\cdot\|$ measures the distance of end-effectors between states. The first term minimizes the distance to the goal, representing the objective to reach the destination efficiently. The second term encourages exploration by forcing the robot away from the initial state. In practice, it helps to escape from the local minimum.

Constraint (6.4b) stands for the control law in (6.3). Constraint (6.4c) represents the transfer function. Constraint (6.4d) encodes the safety requirement, where $\text{contact}(s_t, a_t)$ measures the contact force while executing the robot control; the contact force is constrained by the maximum bound ε.

The transfer function $T(s_t, \tau_t)$ approximates the system propagation in (6.4c) and estimates the contact force in (6.4d). We use the Brax physics engine [31] as the transfer function. Brax is designed for performance and parallelism on accelerators, allowing a large sample size for sampling-based optimization solvers. Moreover, it supports auto-differentiation of the dynamics, which makes it possible to use gradient-based solvers. The next section introduces how we solve the optimization problem by leveraging the parallelism and differentiable properties of the Brax engine.

Solving the Optimization Problem

The optimization (6.4) is first written in a non-constraint form by plugging dynamics (6.4b, 6.4c) into the objective (6.4a) and simplifying contact constraints (6.4d):

$$\min_{a_{k:k+H-1}} \sum_{t=k}^{k+H-1} \lambda_1 \|T(s_t, f(a_t)) - s_g\| - \lambda_2 \|T(s_t, f(a_t)) - s_k\| + \lambda_3 \left(\text{contact}(s_t, a_t) - \varepsilon\right) \tag{6.5}$$

To solve (6.5), this chapter proposes a hybrid solver that integrates the covariance matrix adaptation evolution strategy (CMA-ES, [36]) and the gradient descent method, as outlined in Algorithm 3. The cost function in (6.5) is denoted as $\mathcal{L}$.

Compared to others [36, 1], our method uses a larger population size, bounds the decision variables, and applies an additional gradient descent step. The boundary constrains the robot's movement in operational and joint space. A penalty is added to the cost function,

Algorithm 3 Hybrid Optimization Solver

Initialize $m = 0,\ \sigma = 0.01,\ C = I,\ p_\sigma = p_c = 0$
$k = 200,\ k_{top} = 50$ ▷ Population size
$\gamma,\ \beta_{max} = 1$ ▷ Boundaries & max grad. step
for $step = 1$ to $max\ step$ **do**
  for $i \in \{1, ..., k\}$ **do**
    $a_i \sim \mathcal{N}(m, \sigma^2 C)$ ▷ Sample candidates
    $a_r = \text{clip}(a_i, -\gamma, \gamma)$ ▷ Repair candidates
    $d_i = \|a_i - a_r\|^2$
    $\alpha = \begin{cases} \exp(\log \mathcal{L}(a_i) - \log d_i) & \text{if } d_i > 0 \\ 0 & \text{otherwise} \end{cases}$
    $l_i = \mathcal{L}(a_r) + \alpha \cdot d_i$ ▷ (6.5) w/ penalty
  end for
  $a_{best} = \text{argsort}(l_{1:k})$ ▷ Find k_{top} best candidates
  $m = \text{mean}(a_{best})$ ▷ CMA-ES mean
  $\nabla\mathcal{L} = \frac{\partial \mathcal{L}(m)}{\partial a}$ ▷ Diff. w/ Brax
  $\beta_1, ..., \beta_k \sim \mathcal{U}[0, \beta_{max}]$
  $m = m + \arg\min_{\beta_{1:k}} \mathcal{L}(m + \beta_j \cdot \nabla\mathcal{L}) \cdot \nabla\mathcal{L}$
  $m = \text{clip}(m, -\gamma, \gamma)$
  Update $\sigma,\ C,\ p_\sigma,\ p_c$ ▷ CMA-ES update [36]
end for
Return m

which is the distance to the boundary. A tradeoff weight α is adaptively selected so that the cost and the penalty are similar in magnitude. Moreover, we add a gradient descent step before refitting distributions in CMA-ES. The gradient step optimizes the solution locally. Similar ideas have been utilized in [13, 1] and have shown improved performance over CMA-ES alone. Nevertheless, our method does not iterate the gradient descent until convergence. We sample a population of neighbors in the gradient direction and find the best one as the local optimal, similar to [13]. Such a design gains an advantage in computation speed as the evaluation of the cost is much faster than that of the gradient. Thus, we sample more candidates and only apply the gradient descent once per step. The algorithm is halted if the maximum step has been reached or the cost stagnates for three steps. Readers are referred to [36] for details of the CMA-ES. Although this chapter initializes the solution to zeros for convenience, it can be easily replaced by solutions obtained from other motion planners. However, due to the large population size used in the solver, we did not observe significant improvements by changing the initial values.

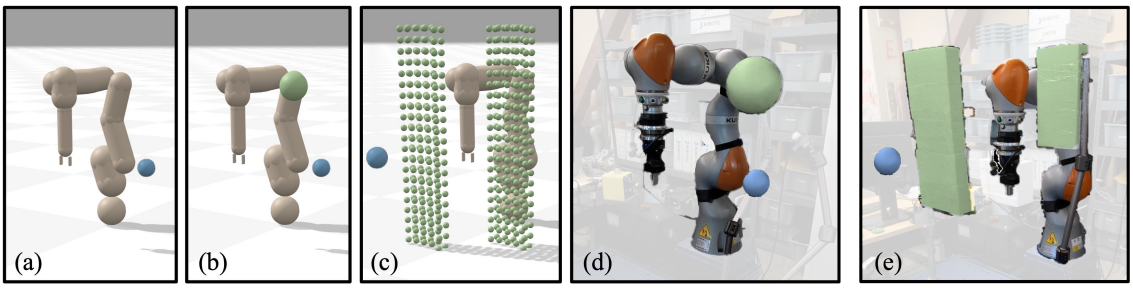


Figure 6.3: Goal-reaching task environments of different collision complexity levels. The target is blue, and the obstacles are green. (a) Free space, (b) ball obstacle, (c) wall obstacle, (d) real-world ball obstacle, and (e) real-world wall obstacle.

Environments

This section introduces the simulation and the real-world environments in which we evaluate and compare different control and planning methods, as visualized in Fig. 6.3. These environments span different levels of collision conditions, ranging from free space to highly collisional.

The simulation environments are built with Brax [31], where the robot is modeled with cylinder links to imitate the kinematics and dynamics of a real one (i.e., Kuka iiwa 14). Such modeling significantly reduces the computation time with simpler contact dynamics. The following sections introduce the detailed configurations for each environment.

Free space. This environment aims to measure the performance of the proposed method in a collision-free condition. The target is randomly placed inside the robot manipulability ellipsoid. Since there is no obstacle, the optimal trajectory in the operational space is a straight line, as defined in (6.4, 6.5).

Ball obstacle. This environment intends to show that it is more efficient to accomplish certain manipulation tasks by allowing safe contacts. In this environment, a ball with a $0.1m$ radius is added as an obstacle. The target is randomly placed within a workspace, varying from $[0.2, -0.3, 0.0]$ to $[0.6, 0.3, 0.5]$ in $[x, y, z]$, respectively. The obstacle is placed to collide with the robot in the free space trajectory. Specifically, the optimal path is first computed without the obstacle. Then the ball is added to hamper the optimal path by colliding with the robot's middle body. Although the obstacle makes the free space trajectory infeasible, other collision-free paths exist.

Wall obstacle. This environment demonstrates that allowing safe contacts enables highly constrained manipulation tasks where collision-free paths cannot be found [101]. The envi-

Table 6.1: Comparison of trajectory efficiency and safety metrics in simulated and real-world environments. In each cell, the first element is the task execution time (s); the second is the maximum contact force on the robot body in simulation (N) or the maximum computed external torque on robot joints in the real-world (Nm).

	Free Space		**Ball Obstacle**		**Wall Obstacle**		**Ball Obstacle Real**		**Wall Obstacle Real**	
Collision-Free	11.4 ± 1.4	0	37.8 ± 3.9	0	Fail		38.4 ± 7.9	0	Fail	
Ref. Posture	11.4 ± 1.4	0	13.1 ± 0.4	4.7 ± 2.3	21.3 ± 2.0	1.9 ± 0.7	15.2 ± 1.7	1.6 ± 0.2	18.7 ± 1.1	1.1 ± 0.1
Ours	11.4 ± 1.6	0	12.8 ± 0.5	2.8 ± 1.2	17.5 ± 1.5	0.5 ± 0.3	14.7 ± 1.6	0.7 ± 0.1	15.0 ± 1.0	0.5 ± 0.1

ronment puts two $1 \times 0.1 \times 0.2m$ walls between the robot and the target. The walls obstruct all collision-free paths.

The deformable wall is modeled as a mass-spring system in the Brax simulator. The wall is tessellated to volumetric finite element meshes, including vertices and edges. We use small rigid spheres to represent the mesh vertices and spring joints to represent the mesh edges. The simulation can mimic the physical deformation by carefully parameterizing the sphere mass and the spring stiffness. The robot's initial joint configuration is fixed through trials, while the target pose is randomized in front of the robot.

Real-world. Besides the simulation, we create physical environments with foam balls and walls to evaluate the proposed method in the real world. The foam ball has a $0.1m$ radius, whose position is generated similarly as in the simulation. The foam walls are positioned with fixtures as shown in Fig. 6.3(e) and have the same effective dimensions as in the simulation. Since tracking the real-world system is not the main focus of this work, we assume the position and deformation of the foam obstacles are known. We manually tuned the coefficients in the dynamics approximator (i.e., Brax) to obtain an accurate transfer function (6.4c). In practice, the transfer function and the system state can be estimated with an additional observer, as suggested in [143]. In our real-world experiments, we utilized joint torque sensors to measure external contact torques and scaled the threshold in (6.4d) to address the gap between simulation and reality.

6.3 Experiments

The goal of the experiments is three-fold. First, we demonstrate the advantage of allowing contacts in multiple collisional scenarios. Second, we show the benefit of generating and tracking both the operational and null space reference signals. Third, we provide empirical evaluations for the proposed hybrid optimization solver.

The hyperparameters used throughout our experiments are set as follows. $\lambda_{1,2,3} = \{1, 0.2, 5\}$, $H = 3$, $\varepsilon = 10N$, $max\ step = 100$, $\gamma = [0.01m, 0.2rad, 0.2rad]$ where the first two represent the end-effector maximum movement and the last limits the null space motion. In this chapter, we do not infer control gains; instead, they are set to constants: $K_p = 880I, K_d = 100I, K_{qp} = 30I, K_{qd} = I$. On the one hand, gains work as a scaling

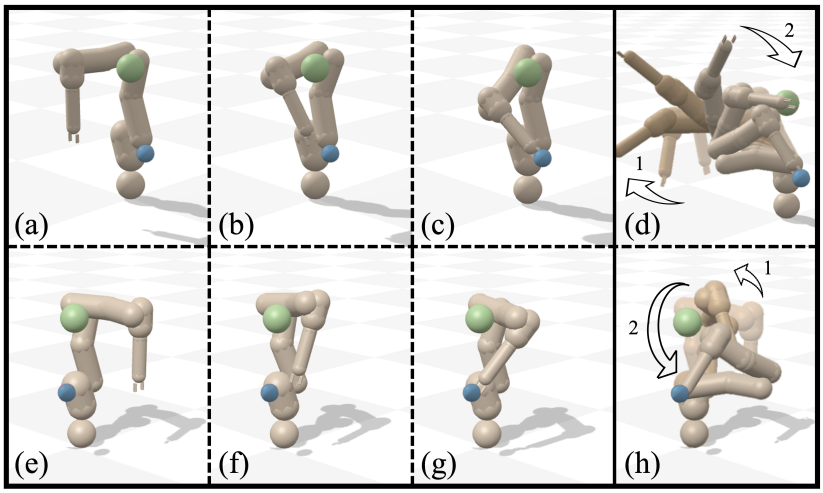


Figure 6.4: Execution trajectories in simulated ball environments. (a-c, e-g) show two trajectories generated by our method, (d, h) show the collision-free trajectories with the same environmental settings.

factor for $\Delta x, \Delta q$ in (6.3), thus implicitly optimized. We did not observe improvement by including gain to the decision variables in our experiments. On the other hand, including gain inference makes the algorithm less stable and harder to solve. The control torque is prone to going unbounded with the inferred scaling.

Collision-free or contact-allowed. This experiment compares the performance of our contact-allowed planner with a collision-free trajectory planner. We used the RRT* [48] to generate the collision-free trajectory in the joint space and our controller to track the path without replanning. Each experiment was repeated three times to record the average task completion time in Table 6.1. The maximum contact force throughout the execution is reported in simulated environments. A torque observer was implemented to measure the external torque in the real world [137].

For contact-allowed goal-reaching, an alternative to the receding horizon planning is to plan the whole trajectory before moving the robot [25, 101]. However, the large search space, from allowing contact and additional null space motion, makes the planning problem intractable. Thus, we do not include such a planning framework in the comparison.

As seen in the ball obstacle experiment in Table 6.1, allowing safe contact improves the task efficiency with the ball obstacle. Although contact forces appeared during the execution, our method reduces the task completion time by 2.6**x**. Fig. 6.4 visualizes two example executions in the simulated environment. In a highly collisional environment, i.e., wall obstacles, a collision-free trajectory cannot be found to reach the target. In contrast,

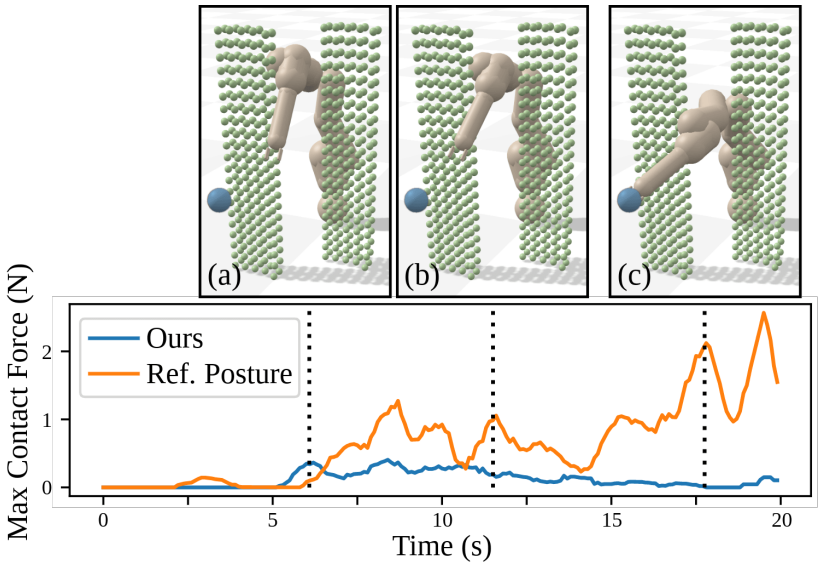


Figure 6.5: Example contact force profile when reaching a goal in the wall environment. Pictures correspond to the robot configuration induced by our method at time stamps marked by dotted vertical lines; (a, b) the robot tracks trajectories to push obstacles in a compliant manner and adjusts its joint configuration in the null space; (c) the robot reaches the goal while maintaining a minimum contact force. Compared to the ablations, our method, which controls both operational and null spaces, results in lower overall contact forces.

our method completes the task by pushing the obstacles away. These results indicate that allowing collision relaxes the optimization constraints and provides a larger feasible motion set. Lastly, in the free-space environment comparison, we observed the same performance, suggesting that allowing contact does not change the behavior in unconstrained scenarios.

Different control laws and decision variables. This experiment provides ablation studies to the control laws and decision variables in (6.4). The ablation method applies a reference posture torque in the null space to track a joint reference. We set the desired joint posture $q_d = 0$ in (6.2). The control law is written as:

$$\tau_{posture} = \tau_{op} + N^T \left(-K_{qp} q - K_{qd} \dot{q}\right) \tag{6.6}$$

The trajectory planner only optimizes the operational space motion Δx and has no control over the null space.

Based on the results in Table 6.1, our method outperforms the reference posture control method in all environments. Fig. 6.5 shows the contact force profile for an experiment in the wall environment. Our method reduces the contact force by almost 4**x** in the simulated

wall environments and 1.7**x** in the ball environments. Our method also reduces the task execution time by generating shorter operational trajectories. The real-world experiments further support the findings that considering additional null space motions increases task safety. These results validate the necessity of optimizing the null space motion. Since the null space behavior is neither optimized nor controlled in the ablation method, it has a tighter search space than ours and often cannot find a trajectory with the same performance as ours.

Another approach to planning robot trajectories is to directly optimize the motor torque. We implemented a joint torque planner and observed similar performance compared to our method. In contrast, our approach involves mapping the planned trajectory to motor torque using a control law defined in (6.3). This structured design can be viewed as a specific instance of direct torque planning. However, optimizing joint torques directly leads to a more complex optimization problem and requires more computation to solve. For instance, in the wall obstacle environment, our method took an average of $0.47s$ to converge for each step, whereas direct torque planning took $1.34s$. Similar results were reported in [70]. Furthermore, as manipulation tasks like our goal-reaching problem are typically defined in the operational space, it is more intuitive and explicit to search for actions in the operational space.

With or without gradient descent. This experiment analyzes the proposed hybrid optimization solver in the wall obstacle environment. We demonstrate the effectiveness of the bounded CMA-ES and single-step gradient descent, line 7-9 and 16 in Algorithm 3. Two baselines were used for comparison. The first only uses the vanilla CMA-ES to search for the optimal reference signals [36], while the second involves multiple gradient steps until convergence, similar to [13].

Interestingly, all solvers yield similar trajectories given the same environmental configuration, suggesting that they have converged to close solutions. Thus, we use the average cost decrease after 10 optimization steps as the comparison metric. Results showed that our method (0.34) outperforms the vanilla CMA-ES (0.22). This validates the benefit of integrating gradient descent with the sampling-based approach; it expedites the convergence by locally refining the solution. Meanwhile, we observed further improved performance with multiple gradient steps (0.40). However, the iterative gradient calculation requires a much longer computation time and significantly reduces the planning frequency. With the consideration of real-time performance, we applied single-step gradient descent in our proposed solver.

6.4 Chapter Summary

This chapter investigates the problem of contact-allowed robotic goal-reaching with operational and null space control and makes several key contributions. Firstly, we formulate the contact-allowed robotic manipulation problem with safety constraints and provide a set

of open-source environments for contact-allowed goal-reaching. These environments have different collision conditions, ranging from free space to highly collisional. Secondly, we propose a receding horizon trajectory planner to generate and track reference signals for collision-allowed motion. This planner optimizes the operational and null space reference signals and tracks the reference using an impedance controller. Finally, we present a hybrid solver to optimize the reference signals. Simulation and real-world experiments demonstrate that by allowing contact, our proposed algorithm enables efficient and safe achievement of the manipulation goal.

Chapter 7

Contact-Aware Robotic Assembly Planning

7.1 Introduction

The preceding chapters have delved into foundational manipulation tasks such as grasping, pushing, in-hand object reorientation, and goal-reaching. While these tasks pose significant challenges in the planning and control aspects for robots, they do not necessitate advanced, complex reasoning. In contrast, this chapter shifts the focus to the intricate task of robotic assembly, which requires a deeper understanding of part geometries, sophisticated reasoning about physical interactions and collisions, and the execution of assembly plans bolstered by robust sensing capabilities.

The task of assembling parts in alignment with a specified target blueprint represents a pivotal area of research within the fields of robotics and machine learning. This endeavor is not only a valuable function that autonomous robots can perform but also encapsulates a multifaceted problem domain fraught with unpredictable complexities. To effectively navigate this complex landscape, robots must acquire a diverse set of competencies that are essential for successful assembly. These competencies include accurately deciphering the sequence of assembly, coordinating the trajectories of the parts, identifying optimal points of contact, and physically executing these processes with precision. Furthermore, it is imperative for robots to not only master these skills in specific scenarios but also to generalize these competencies across a wide range of assembly tasks, adapting to different configurations and requirements.

Previous investigations in robotics and computer vision have tackled this multifaceted challenge of part assembly through diverse methodological lenses. For example, one line of research attempts to sidestep the complexities of physics, electing to focus on more specialized tasks, such as the segmentation of target blueprints [55, 54] or the estimation of part poses [42, 14]. A second vein of research emphasizes the importance of physical interactions, specializing in the assembly of predetermined targets [78]. A third category adopts different

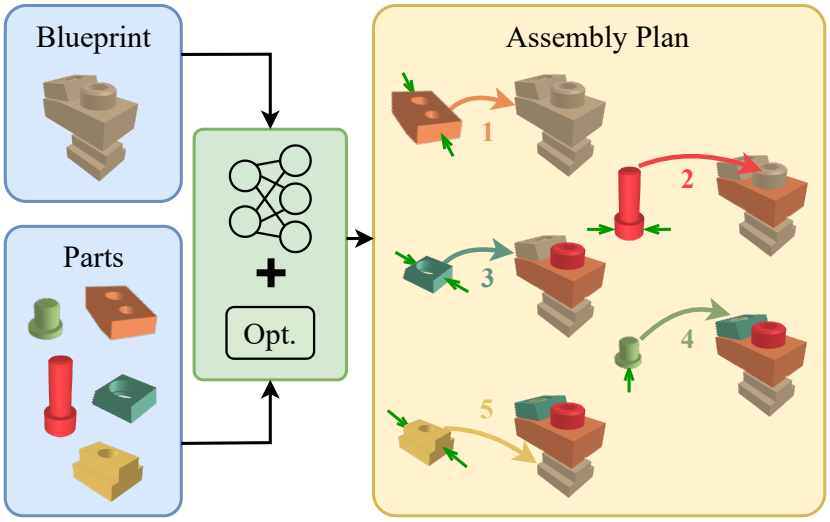


Figure 7.1: Our goal is to facilitate robotic assembly across different target blueprints. Utilizing point clouds from target blueprints and assembly parts, our method identifies feasible assembly sequences (indicated by colored numbers), orchestrates part motions (represented by colored long arrows), and pinpoints contact points (denoted by short green arrows).

target blueprints but imposes simplifying assumptions on part geometries (i.e., blocks) [32] or restricts its scope to a seen set of blueprint categories (e.g., chairs) [132]. The use of reinforcement learning (RL) has seen success in this space [32, 132]. However, RL-based solutions face challenges in terms of computational resources and efficiency. For instance, Ghasemipour et al. [32] requires an elaborate computational infrastructure involving thousands of CPUs and billions of steps for training, raising concerns about the practicality of the system. Our work aims to advance the field of robotic assembly by holistically considering intricate physical interactions between parts and designing a supervised training paradigm, while our approach is applicable to a broad spectrum of practical target blueprints.

To achieve successful robotic assembly, this study breaks down the task into three distinct and key sub-tasks: 1) inferring the sequence in which the parts should be assembled, guided by the target blueprint, part shapes, and assembled poses; 2) coordinating the movements of the individual parts; and 3) identifying viable contact points for robotic manipulation. An illustration of these steps is provided in Fig. 7.1. Addressing the first challenge involves contending the physical interactions between parts and the inherent ambiguities. The former is due to the collision between parts that prevent arbitrary assembly order, and the latter arises due to multiple viable assembly sequences. Can we learn the order of assembling the parts statistically from their geometry and their locations in the target assembly? For example, it is clear in Fig. 7.1 that the red screw can only be inserted if the orange piece is in place – the target blueprint and individual pieces collectively establish a specific order

for assembly. We use this insight towards designing an implicit neural planning network using Transformers [118], dubbed *Part Assembly Sequence Transformer* (PAST) that takes as input point clouds of the target blueprint and the assembled parts and identifies the next parts to be assembled. Then, it is applied to generate the full sequence in an autoregressive fashion. For training our PAST model, we construct a benchmark dataset for part assembly sequences, dubbed D4PAS, by enumerating feasible assembly sequences [117].

To solve the problem of part motion planning for assembly, we leverage the RRT-connect [50] to generate trajectories from each part's resting pose to its assembled pose. Concurrently, we conduct an efficient physics-inspired multi-scale optimization of potential contact points on the part's surface to identify those that are most effective in achieving the desired part movement. Upon generating the assembly plan through the aforementioned steps, prior research has explored the use of reinforcement learning [18], model-predictive control [139], and diffusion policies [15] for its execution. However, the focus of this work is not on physical execution, which is earmarked for future investigation.

In summary, our primary contributions are as follows. First, we present an assembly planning algorithm to generate feasible part assemblies based on target blueprints, including inference of assembly sequences, planning of part movements, and optimization of contact points. Second, we introduce the Part Assembly Sequence Transformer (PAST) to infer assembly sequences in an autoregressive fashion. PAST is designed to generalize to novel, diverse, and practical blueprints and part geometries. Third, we provide a dataset for part assembly sequences (D4PAS), replete with assembly trajectories, enumerated assembly sequences, and viable contact points, thereby providing a foundation for future studies in robotic assembly. The experimental videos are available at [121].

7.2 Assembly Planning

Problem Overview

This work employs a part assembly formulation consistent with [117, 132], as shown in Fig. 7.1. Given M part meshes $\mathcal{M} = \{\mathcal{M}_i\}_{i=1}^{M}$ and their respective 6D assembled poses in the target blueprints $p^{tgt} = \{p_i^{tgt}\}_{i=1}^{M}$, the algorithm plans the assembly trajectories $(p^0, p^1, ...)$ for each part from their resting poses $p^0 = \{p_i^0\}_{i=1}^{M}$. We use p_i^t to represent the pose of part $\mathcal{M}_i$ at time t and use p^t to represent poses of all parts at time t. The algorithm assumes, at each time step, that only one part is in motion while the others remain stationary [117]. The work breaks down the complex task of assembly planning into three sub-tasks: assembly sequence inference, part motion planning, and contact point selection.

For one possible assembly, let the assembly sequence be denoted by $m = (m_k)_{k=1}^{M}$, specifying the order of part assembly. Each m_k belongs to the set $\mathcal{M}$ and identifies the kth part to be assembled. Part movements are represented by $\mathcal{T} = (p_{m_k})_{k=1}^{M}$, where p_{m_k} details the trajectory of parts when part m_k is moving throughout its moving horizon. Contact points facilitating these movements p_{m_k} are indicated by $\mathcal{C} = (c_{m_k})_{k=1}^{M}$. The assembly planning

problem is formulated as follows:

$$\mathbb{P}(p^0, p^1, ..., p^{tgt}) = \mathbb{P}(m, \mathcal{T}, \mathcal{C}) = \mathbb{P}(m) \cdot \mathbb{P}(\mathcal{T}|m) \cdot \mathbb{P}(\mathcal{C}|m, \mathcal{T})$$

where in this work, we assume a multi-level solution approach by sequentially solving for the sub-problems of (i) assembly inference, (ii) motion planning, and (iii) contact selection, in that order.

It is crucial to recognize that feasible assembly sequences are a subset of all possible part permutations. This constraint arises from the potential for part collisions, which precludes arbitrary assembly sequences. For instance, a washer must be in place before tightening a screw. To address this combinatorial inference problem approximately, we introduce the Part Assembly Sequence Transformer (PAST) to learn statistical correlations between the parts and the target blueprint to produce physically viable assembly sequences $\mathbb{P}(m)$. The network ingests both target blueprints and unassembled parts, outputting the next feasible parts for assembly. This iterative process determines the full assembly sequence, as in Fig. 7.2.

Upon establishing the assembly sequence m, part movement can be planned using established motion planning algorithms $\mathbb{P}(\mathcal{T}|m)$. Following that, the algorithm optimizes the contact points that the robot can utilize to execute these movements, $\mathbb{P}(\mathcal{C}|m, \mathcal{T})$, thus culminating in a coherent assembly process.

Part Assembly Sequence Transformer (PAST)

The preceding section outlines our multi-level approach to assembly planning. This section delves into the details of each of the three planning levels.

The transformer ingests the target assembly blueprint and the remaining unassembled parts, outputting a probability for each part's suitability for assembly at the current step. The part with the highest probability is chosen for assembly and removed from the list of remaining parts. This iterative process continues until all parts are assembled, yielding one assembly sequence $m = (m_k)_{k=1}^{M}$. Fig. 7.2 illustrates this recursive planning approach.

PAST takes two branches of inputs: a target assembly blueprint and unassembled remaining parts, both represented using point clouds. For an assembly comprising M parts, the target blueprint is rendered with the 6D assembled poses p^{tgt} of all parts, resulting in a point cloud $PC_{tgt} \in \mathbb{R}^{N_t \times 6}$. This point cloud consists of N_t sampled points, each with positional and normal features. During the kth step of assembly, PAST selects the next part to assemble from the $M-k$ remaining parts. These remaining parts are input as $M-k$ individual point clouds, denoted as $\{PC_{r,i}\}_{i=1}^{M-k}$, where $PC_{r,i} \in \mathbb{R}^{N_r \times 6}$ and contains N_r sampled points.

A key design question for PAST is which neural model to use for representing the input point clouds. Among point cloud encoders, such as PointNet and its variants [95, 96], it was shown in [14] that dynamic graph CNN (DGCNN) [124] offers superior efficiency and representational capabilities in assembly segmentation. To this end, we employ DGCNN to derive target features $v \in \mathbb{R}^{N_t \times h}$ from the target blueprint (one feature for every sample

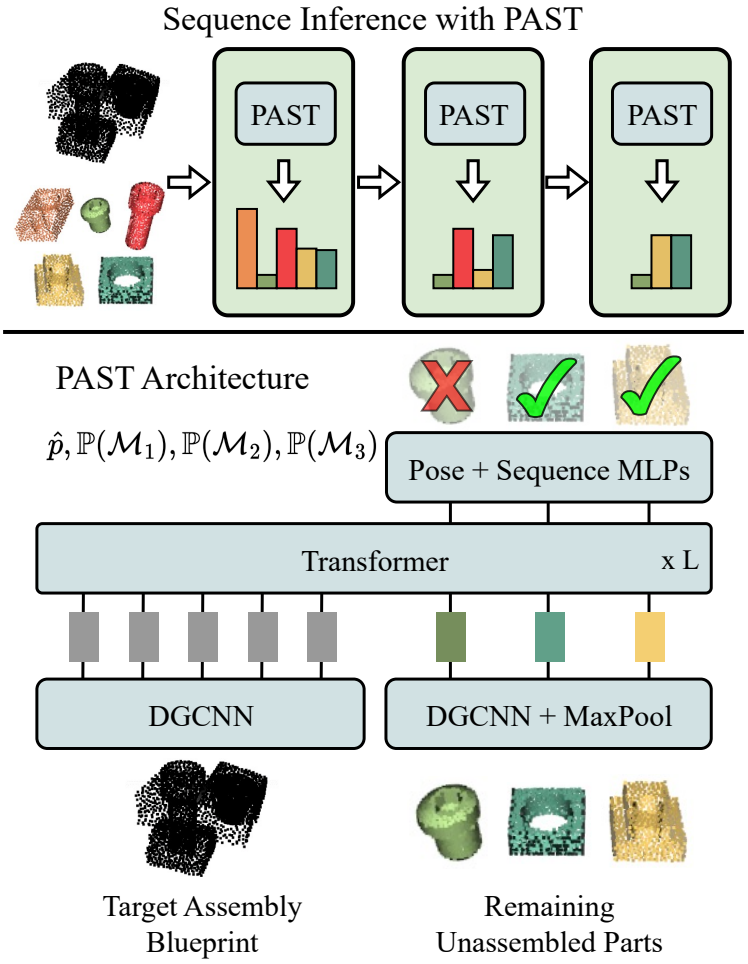


Figure 7.2: Sequence inference pipeline and PAST architecture. (Top) PAST operates sequentially to estimate the assembly probability $\mathbb{P}(\mathcal{M}_i)$ for each remaining part. The part with the highest probability is chosen for assembly. (Bottom) Using the third block as an example, PAST selects one part for assembly from the three remaining options. PAST also performs pose regression for each part $\hat{p}$ as an auxiliary task.

point) and part features $u_i \in \mathbb{R}^h$ from each remaining part (i.e., one feature for every part after max-pooling the features from all samples belonging to that part). Here, h is the hidden feature dimension and $i \in \{1, ..., M-k\}$.

Once the features are extracted, our PAST model then jointly refines these features through L transformer blocks, as illustrated in Fig. 7.2. As is well-known, transformers use self- and cross-attention to learn correlations between their inputs and have demonstrated state-of-the-art performances in generating sequential outputs (e.g., language). Our key in-

sight is to use such attention to learn physically plausible assembly sequences in a supervised setting. Mathematically, suppose for a *query* set $\mathbf{q}$ and a *key* set $\mathbf{k}$, let the transformer dot-product attention operator be defined as $\text{Attention}(\mathbf{q}, \mathbf{k}) = W_v(\mathbf{k})^T \text{softmax}\left(\frac{W_k(\mathbf{k})W_q(\mathbf{q})}{\sqrt{h}}\right)$, where W_q, W_k, W_v are matrices embedding the query and the key sets in a common latent space. Our PAST transformer blocks utilize a two-stage approach for feature refinement between the target and parts. The first stage independently processes and updates the features with self-attention; that is,

$$v = \text{Attention}(v, v) \text{ and } u_i = \text{Attention}(U, u_i),$$

where $U = \{u_i\}_{i=1}^{M-k}$ denotes all part point cloud features. The second stage applies cross-attention between the target features and part features [10], updating them to:

$$\hat{v} = \text{Attention}(U, v) \text{ and } \hat{u}_i = \text{Attention}(v, u_i).$$

where $\hat{v}$ and $\hat{u}_i$ are fed into the next block. We use the same attention expression for all blocks except for replacing u and v updated to $\hat{u}$ and $\hat{v}$ from the previous block. In the final step, PAST calculates the assembly probability for each part using the formula $\mathbb{P}(\mathcal{M}_i) = \text{MLP}(u_i)$. The part to be assembled next is then selected based on the maximum probability, denoted as $m_k = \arg\max_i \mathbb{P}(\mathcal{M}_i)$. In addition to predicting the assembly sequence, the transformer also estimates the 6D assembled pose p_i^{tgt} for each part as an auxiliary task, represented as $\hat{p}_i = \text{MLP}_p(u_i)$. The inclusion of this auxiliary task enhances the network's capability to comprehend the geometric interrelations between parts, a technique that has proven effective in [14, 55].

We use supervised learning for training PAST via estimating the predicted assembly probability $\mathbb{P}(\mathcal{M}_i)$ with mean squared error loss $\text{MSE}(y_i, \mathbb{P}(\mathcal{M}_i))$ with the feasibility of assembling a part $\mathcal{M}_i$ at a given step is denoted y_i. Additionally, the pose regression task aims to minimize the difference between the actual 6D assembled pose p_i^{tgt} and the predicted pose $\hat{p}_i$. The difference is calculated with $\sum_i \left\|p_{i,tra}^{tgt} - \hat{p}_{i,tra}\right\| + \left\|p_{i,rot}^{tgt} - \hat{p}_{i,rot}\right\|$, where $p_{i,tra}^{tgt}, \hat{p}_{i,tra}$ indicate target and predicted translation for each part and $p_{i,rot}^{tgt}, \hat{p}_{i,rot}$ represent the axis-angle.

Dataset for Part Assembly Sequences (D4PAS)

To train PAST, we introduce a new *dataset for part assembly sequences* or D4PAS. Each sample in the assembly sequence dataset comprises multiple components, namely: (i) the target blueprint point cloud PC_{tgt}, (ii) point clouds for $M-k$ remaining parts $\{PC_{r,i}\}_{i=1}^{M-k}$, and (iii) the feasibility of assembly for each part $\{y_i\}_{i=1}^{M-k}$. The feasibility y_i specifies whether a part $\mathcal{M}_i$ can be assembled at the current step and can subsequently lead to a successful final assembly. Note that there can be many viable part candidates at every step, derived from all possible sequence enumerations using the scheme in assembly-by-disassembly [117], as described in Algorithm 4 and illustrated in Fig. 7.3.

Algorithm 4 Disassembly Planning

1: **Input:** part meshes $\{\mathcal{M}_i\}_{i=1}^M$ and inertias $\{I_i\}_{i=1}^M$
2: **Input:** target part pose p^{tgt}, empty queue $\mathcal{J}$
3: **Output:** sequence of disassembly
4: $\mathcal{J}$.`enq`(p^{tgt}, f_i^j) for $f_i^j \propto I_i$ and $i \in \{1, \cdots, M\}$
5: **while** not finish **do** ▷ In parallel
6: $(p^t, f_i^j) = \mathcal{J}$.`deq` ▷ BFS
7: **if** `success`(p^t) **then**
8: return `GetSequence`(p^{tgt}, p^t)
9: **end if**
10: $p^{t-1} =$ `simulate`(p^t, f_i^j) ▷ Disassembly attempt
11: **if** `isNovel`(p^{t-1}) and `isExec`(p^t, p^{t-1}) **then**
12: $\mathcal{J}$.`enq`(p^{t-1}, f_i^j) for all unassembled part i
13: **end if**
14: **end while**

The disassembly planning algorithm operates in a search-based manner, where the set of parts poses' at time t, $p^t = \{p_i^t\}_{i=1}^M$, serves as the search state. At each step, the algorithm selects an unassembled part i (line 6) and attempts to remove it from the blueprint with a physical simulation (line 10). The selection of the moving part follows a breadth-first search (BFS) scheme, ensuring a comprehensive enumeration of all possible disassembly sequences. The chosen part i is moved in the direction of its moment of inertia I_i with force f_i^j, calculated as $f_i^j = \Lambda_i e_j I_i$, where Λ_i is the mass of the part and e_j is a one-hot vector with nonzero element at $j \in \{1, 2, 3\}$. Torque is also applied to enable rotational movement and is computed similarly to f_i^j. The search queue is expanded only if the disassembly attempt results in novel part poses and is executable by a robot (line 11). Part poses are regarded as novel (`isNovel`) if they lead to a new location for one of the parts. The feasibility of execution (`isExec`) is confirmed by determining whether there exists collision-free grasp or push points on the part's surface. After each disassembly attempt, the remaining unassembled parts are added back to the queue, considering all possible dragging forces (line 12).

Compared to [117], the dataset generation algorithm in this work incorporates several advancements. First, we employ a parallel simulation (Isaac Gym [69]) to expedite the dataset generation and allow future studies on assembly planning with RL [32]. Second, unlike [117] that use arbitrary force directions, our algorithm applies disassembly forces along the part's moment of inertia, aligning with the physical properties of the parts. Third, we incorporate additional constraints to ensure that the generated plans are executable by robots (`isExec`), making our dataset more practical for further robotic applications.

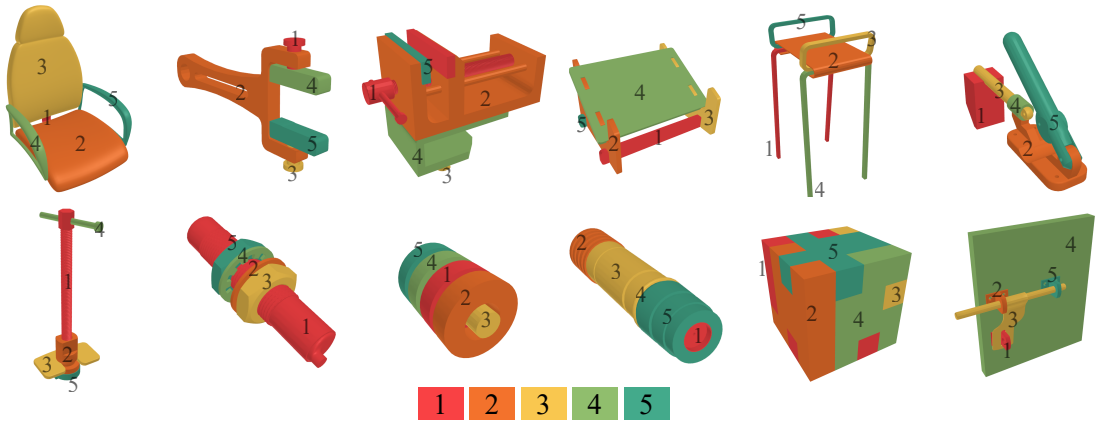


Figure 7.3: Example assembly sequences in our dataset. For each target blueprint, we enumerate all feasible assembly sequences and present one representative sequence here. The color coding and the numbers beside each part signify the assembly sequence, as indicated by the color bar.

Part Motion and Contact Planning

Once the sequence is determined by the PAST transformer, the next steps involve planning the movements of the parts and identifying suitable contact points for robotic manipulation. For part movements, we use the RRT-connect [50] to search for a collision-free path for each part in line with the inferred assembly sequence.

For contact points planning, this work first plans robust grasps for each part with two contact points and filters out those in collision with other parts. To do this, we enumerate all grasp pairs from the part point cloud $PC_{r,i}$ and use the Ferrari Canny metrics to determine their robustness [26]. We then identify the feasibility of execution using FCL [88] to check the collision between grasp points and the assembled parts along the assembly trajectory. If no feasible grasps are found, we resort to optimizing for a single pushing point c by solving the optimization [140]:

$$\begin{aligned} \min_{c,F_c} \quad & F_c \\ s.t. \quad & F_c \in FC(c) \\ & G(c)F_c = W \end{aligned} \tag{7.1}$$

where F_c is the pushing force at c. $FC(c)$ is the friction cone at point c and is expressed by $F_{c,1}^2 + F_{c,2}^2 \leq \mu F_{c,3}^2$ with μ being the friction coefficient. $G(c) \in \mathbb{R}^{6\times 3}$ is the grasp map which maps the contact force to the part movement force [8]. W is the part movement force derived from the part movement [140]. We observe that the optimization problem (7.1) can be cast as a semi-definite program (SDP) once the pushing point c is specified. To solve this

problem, we employ a hybrid approach that combines sampling with an SDP solver [140]: the pushing point c is sampled from the part's point cloud and held constant during the optimization of F_c as SDP. The outcomes are iteratively updated across all sampled pushing points to identify the optimal solution.

7.3 Experiments

This section offers details of our model, dataset, and empirically validates our approach against the related methods.

Dataset and Network Details

Both disassembly planning and PAST training were performed using a single RTX3090 GPU. During the disassembly planning, properties of parts Λ_i, I_i were extracted from the simulation. The friction $\mu = 0.2$. Our dataset comprises 8,670 target blueprints and a total of 84,326 assembly sequences, broken down as follows: 7,278 are 3-step sequences, 54,612 range from 4 to 7 steps, and 22,436 have more than seven steps. These sequences can be further augmented with varying choices of the remaining (unassembled) parts. Specifically, for an assembly sequence with M parts, we randomly split the sequence into two segments with size k and $M - k$, respectively. Parts in the second segment are regarded as unassembled, and the first part in the second segment is the part that can be assembled with $y_{M-k} = 1$. Additionally, we gather segments from all viable sequences to identify all potential parts that can be assembled for a specific unassembled segment.

During the training of PAST, two-part assemblies were only used to train the auxiliary pose regression. The target blueprint was sampled at $N_t = 1024$ points, and each part mesh was sampled at $N_r = 512$ points. The DGCNN encodes point clouds with dimension $h = 256$. PAST uses $L = 8$ transformer blocks and was implemented in PyTorch and was trained with the AdamW optimizer using the One-Cycle learning rate scheduler. The target blueprints were re-centered and normalized to a unit ball and randomly rotated and jittered as augmentation. We allocate 500 multi-part assemblies for the test set, reserving others for training.

Baselines and Ablations

The baseline and ablation studies aim to evaluate the efficacy of PAST and the overall algorithm.

Metrics. We employ two key metrics to assess the performance of assembly sequence inference: one-step prediction accuracy (`1-Acc`) and sequence prediction accuracy (`Seq-Acc`). A one-step prediction is deemed correct if the selected part, sampled based on predicted assembly probabilities, belongs to the set of possible parts for assembly. In addition, we apply PAST in an autoregressive fashion to generate a full assembly sequence, which is

Table 7.1: Quantitative results on assembly sequence inference. We report three metrics: one-step prediction accuracy (`1-Acc`), full sequence prediction accuracy (`Seq-Acc`), and the computation time (CT).

	`1-Acc` (%)	`Seq-Acc` (%)	CT (ms)
NSM [14]	75.0	57.8	58.5
DGL [42]	77.4	54.1	104.4
ATA [117]	NA	NA	24312.6
Seg-PAST	90.3	80.4	**52.2**
NoAux-PAST	79.1	58.4	**52.2**
PAST	**91.7**	**82.9**	**52.2**

considered correct if it aligns with any of the possible assembly sequences for that object in our dataset. In addition, we also report the computational time (CT) taken for full sequence inference.

Compared algorithms. Given that no existing solutions are tailored specifically for part assembly sequence inference, we adapt methods from part segmentation and pose regression as baselines, and compare against various ablations.

- NSM (Neural Shape Mating [14]) uses a transformer to address two-part shape mating. We adapt this network to accommodate multiple part inputs.
- DGL (Dynamic Graph Learning [42]) employs a graph neural network to perform assembly pose regression. We use a global node to represent target assembly [3], which facilitates assembly sequence inference.
- ATA (Assemble-Them-All [117]) solves assembly through runtime physical simulation. The assembly process aligns with Algorithm 4.
- Seg-PAST: We substitute the final pose regression layer in PAST to predict blueprint segmentation during the pretraining, as advised in [55].
- NoAux-PAST: We eliminate the auxiliary pose regression task, focusing solely on predicting the assembly sequence using the same network.

Experimental Results

Table 7.1, Fig. 7.4, and Fig. 7.5 summarize the quantitative and qualitative results, which are elaborated upon in this section. For more results, we refer readers to our video.

Multi-level assembly planning. Our methodology breaks down assembly planning into three distinct phases. This approach achieved assembly planning in 1565.9 ms, with an 82.9% success rate across our 500 multi-part assemblies test set. Figure 7.4 illustrates an example of assembling parts from scratch. Specifically, part movement planning averaged

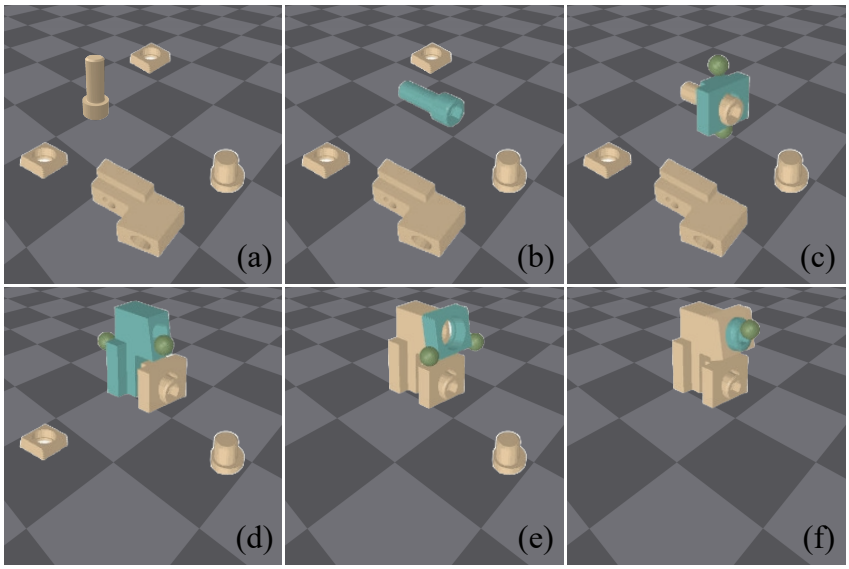


Figure 7.4: Assembly planning results (a-f). In each step, the part colored blue indicates the one in motion, while the yellow parts signify those that are stationary. In (c, d, e), the dual green spheres denote feasible grasp points. In (f), a solitary green sphere highlights the designated pushing point.

1476.9 ms with a 100% success rate, while contact point optimization took 36.8 ms with 100% success rate using multi-threaded computation and the CVXPY solver [21]. Unlike our method, RL-based assembly planning [132] exhibits inferior performance when evaluated in our setting, achieving a 63.9% assembly success rate. While prior works showed promise with simpler geometries [132, 32], we posit that end-to-end policies may struggle to handle complex geometries and assembly reasoning simultaneously.

Generalization to novel assemblies. From Table 7.1, we see that PAST consistently outperforms other neural sequence inference models, such as NSM, DGL, Seg-PAST, and NoAux-PAST in both one-step and full-sequence prediction tasks. Unlike NSM, which employs a discriminator for target shape understanding and fails to capture the assembly geometries' distribution, PAST leverages the target assembly blueprint as input and can extract direct features from the target assembly. DGL, which represents parts as a graph and updates features at the node level, struggles to model geometries from other parts and the target shape. In contrast, PAST aggregates features at the point level, thus enhancing geometric understanding, consequently yielding superior learning outcomes.

Further, from Table 7.1, we also see from the ablated PAST variants that incorporating auxiliary tasks, such as pose regression or part segmentation, significantly improves network performance. This improvement is attributed to the additional guidance these tasks offer, enhancing the network's understanding of part interactions, which are key for assem-

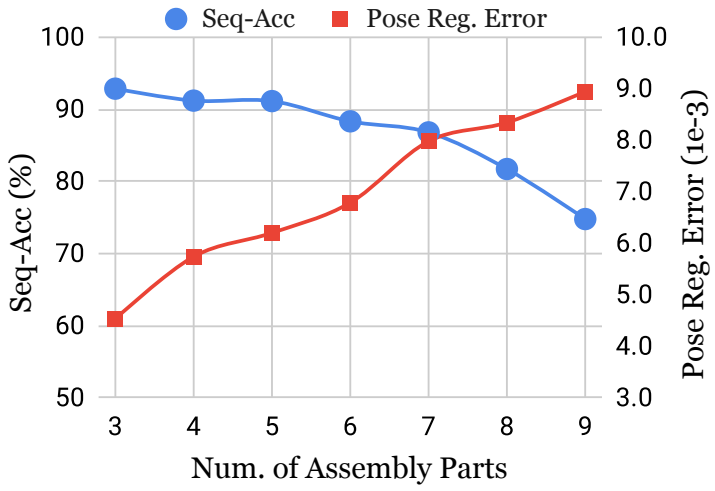


Figure 7.5: Sensitivity analysis for sequence inference accuracy and auxiliary pose regression error.

bly sequence inference. Interestingly, target segmentation underperforms compared to pose regression, possibly because some points in the target assembly, like those corresponding to assembled parts, are not supervised during training. Further, as is expected, ATA takes significant computing as it uses enumeration and simulation for assembly sequences. Instead, PAST, which shows promising accuracy, has a dramatically short computing time, making it well-suited for real-world robotic assembly. In Figure 7.5, we analyze the impact of the number of parts on sequence inference accuracy and auxiliary pose regression error. The results indicate that increased part count leads to reduced sequence accuracy and higher pose error, corroborating results from [32]: the complexity of the assembly problem increases with the number of parts in the target blueprint.

Disassembly Planning. As noted earlier, while our D4PAS construction bears similarities to [117], we enhance disassembly planning through (i) employing GPU-accelerated simulation and (ii) applying part-centric forces. Enabling parallel computations, thereby reducing compute time for disassembly planning to 3592.0 ms against 24312.6 ms. This efficiency not only allows for the enumeration of feasible assembly sequences within a manageable timeframe but also establishes a performance benchmark for future research in end-to-end robotic assembly learning [32]. Second, we apply part-centric disassembly forces aligned with the parts' inertia axes, as supported by findings in [127, 128]. Although this adjustment in the force application coordinate system may seem minor, it led to a notable increase in planning success rate: 92.7% in our approach versus 83.8% in [117].

7.4 Chapter Summary

This chapter makes several key contributions to robotic assembly. First, we introduce a multi-level framework for generating assembly plans, encompassing part sequences, motions, and contact points. Second, we unveil the Part Assembly Sequence Transformer (PAST) for inferring feasible assembly sequences based on target blueprints and part geometries. Third, we offer a large-scale benchmark dataset for part assembly sequence (D4PAS) featuring thousands of physically validated sequences. Post-sequence inference, we employ motion planning and contact optimization to complete part assembly. Our evaluations show that PAST and the overall algorithm match previous simulation-based methods but with significantly reduced computation time.

Part III

Contact Sensing

Chapter 8

Contact Synthesize for Tactile Sensors by Graph Neural Network

8.1 Introduction

Feedback signals are essential in control theory, closing the loop and significantly enhancing robustness. Throughout the progression in previous chapters, where contact planning is central to robotic manipulations, the direct sensing and utilization of contact signals have been notably absent. This chapter focuses on sensing contact with vision-based tactile sensors in robotic applications. Tactile sensors play a crucial role in providing direct and tangible information about contacts during the processes of robotic grasping and manipulation. Among various tactile sensor designs, vision-based tactile sensors represent a unique and innovative category [133, 24, 65, 115, 71, 72, 51, 87]. These sensors employ a camera to capture high-resolution images of the contact-induced deformations on a specialized sensing surface. This surface typically consists of an elastomeric gel coated with an opaque material designed to effectively visualize and measure contact interactions. The operational principle and design of these sensors are illustrated in Fig. 8.1 (a) and (b). Vision-based tactile sensors stand out for their ability to provide detailed spatial information about the contact, a feature that is essential for enhancing the precision and adaptability of robotic manipulations in a variety of contexts.

Obtaining a mesh representation of the contact elastomer can advance the development of applications with vision-based tactile sensors since meshes can provide accurate contact information. For instance, meshes of the elastomer have enabled in-hand object localization [4, 80, 5], vision-free manipulation [23, 40, 111], and contact profile reconstruction [65, 115, 53, 64, 123]. Also, meshes can be used for precise dynamics simulation [93, 105, 106] and future state estimation [6, 56].

Previous simulation studies [65, 115] for vision-based sensors focus on reconstructing the *surface mesh* by tracking markers on the sensor. This can provide the surface displacement fields of the elastomer. However, to better simulate the dynamics, a *volumetric mesh* is pre-

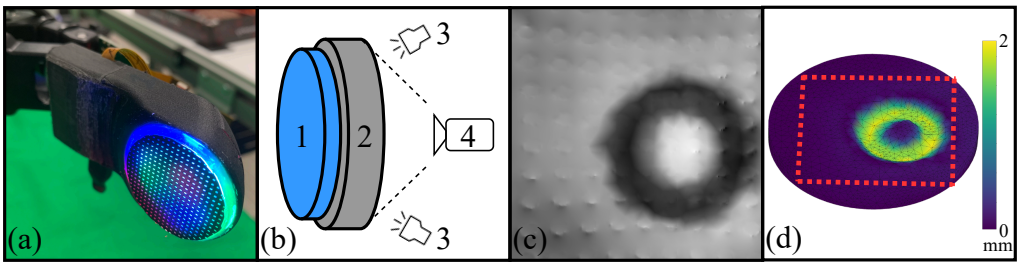


Figure 8.1: **(a)** the GelSlim visual-tactile sensor, **(b)** the construction of the sensor, with the elastomer (1), the transparent lens (2), the lights (3), and the camera (4). **(c)** a depth image observation obtained from the sensor, and **(d)** the corresponding reconstructed volumetric mesh with our method. The red rectangle denotes the camera's view range, and the color represents the displacement level.

ferred [105]. Compared to the *surface mesh*, the *volumetric mesh* contains internal vertices and edges, thus can better encode the dynamics and estimate the contact profile with the Finite Element Method (FEM) [81, 116, 93]. Nevertheless, internal elements also challenge the reconstruction of the *volumetric mesh* due to additional dimensions. This chapter addresses that challenge and proposes a method to directly predict the *volumetric mesh* from images using vision-based tactile sensors, such as the GelSlim [115], in a sim-to-real setting. Moreover, our approach does not rely on fiducial sensor markers to synthesize a *volumetric mesh*.

We first employ 3D FEM simulations of the GelSlim sensor's elastomer to collect image-mesh data pairs. The FEM simulations compute volumetric deformation fields for the elastomer with arbitrary contacts. The depth image observation is then rendered with synthetic cameras. The contact experiments are also executed in the real world with physical GelSlim. However, real-world contacts only provide images since ground-truth meshes are unprocurable. We then learn mappings from real-world images to mesh deformations (as shown in Fig. 8.1 (c) and (d)) by leveraging supervised pre-training and self-supervised adaptations. Specifically, we learn an image-to-mesh projection in latent space with synthetic data pairs.

Sim-to-real approaches have to overcome the distribution differences between the two domains, that is the sim-to-real gap. We propose data augmentation of the synthetic images together with a self-supervised adaptation method on real-world images to address this gap. The adaptation uses a differentiable renderer to project the network output into images and minimize the difference between projected and input images. We demonstrate that this adaptation can transfer networks for sim-to-real, seen contact objects to novel contact objects and between different GelSlim sensor instances. In this chapter, our goal is to introduce the synthesis of volumetric meshes from tactile imprints and will address applications with our approach in future work.

Our work makes the following contributions: first, We provide a FEM model for GelSlim tactile sensors with a GPU-based simulator and propose a method to calibrate the FEM model with physical GelSlim sensors. Second, we collect contact datasets from synthetic and real-world contact experiments for GelSlim sensors. Third, we present an image-to-mesh projection network to reconstruct the volumetric mesh of the elastomer without the need for fiducial sensor markers. Fourth, we further propose a self-supervised adaptation method and image augmentation techniques to mitigate the domain shift of sensor readings.

8.2 Learning to Synthesize Volumetric Meshes

This section first introduces the problem statement and preliminaries. Next, the image-to-mesh projection and self-supervised adaptation methods are discussed. Finally, the datasets are described, including synthetic labeled data, real-world unlabeled data, and data augmentation techniques.

Problem Statement and Preliminaries

This chapter focuses on the problem of reconstructing an elastomer's volumetric mesh with image observations for vision-based tactile sensors. The non-injective projection (or mapping) from surface images to volumetric vertex positions makes this problem nontrivial. Some preliminaries are described below:

Image observations. Visual tactile sensors typically contact objects with a silicone elastomer and use a camera to capture the deformation of the surface, as shown in Fig 8.1. The captured RGB image can be used to construct a depth map of the contact surface using shape from shading [46, 115]. It establishes a mapping from the RGB color to the surface normals with a marble of known dimension. During runtime, surface normals are retrieved and integrated into the depth map I. Compared to raw RGB images, depth maps contain 2.5D information and can better represent the geometry of the contact surface [138]. Moreover, depth maps are much easier to simulate using synthetic cameras and thus have less sim-to-real gap. Therefore, in this chapter, we use (128×128) depth maps I as the image observations.

Volumetric meshes with FEM. The FEM is a mathematical tool to solve complex partial differential equations (PDEs) [81]. In the FEM, geometrical shapes are represented by volumetric meshes $\mathcal{M}$, which consist of 3D elements, such as tetrahedrons and hexahedrons. With high-resolution meshes and small computation steps, FEM can estimate the forward dynamics of soft bodies [116, 93].

This chapter uses graphs to represent volumetric meshes. Specifically, volumetric meshes are defined as a set of vertices and edges, $\mathcal{M} = (\mathcal{V}, \mathcal{A})$, with n vertices in 3D Euclidean

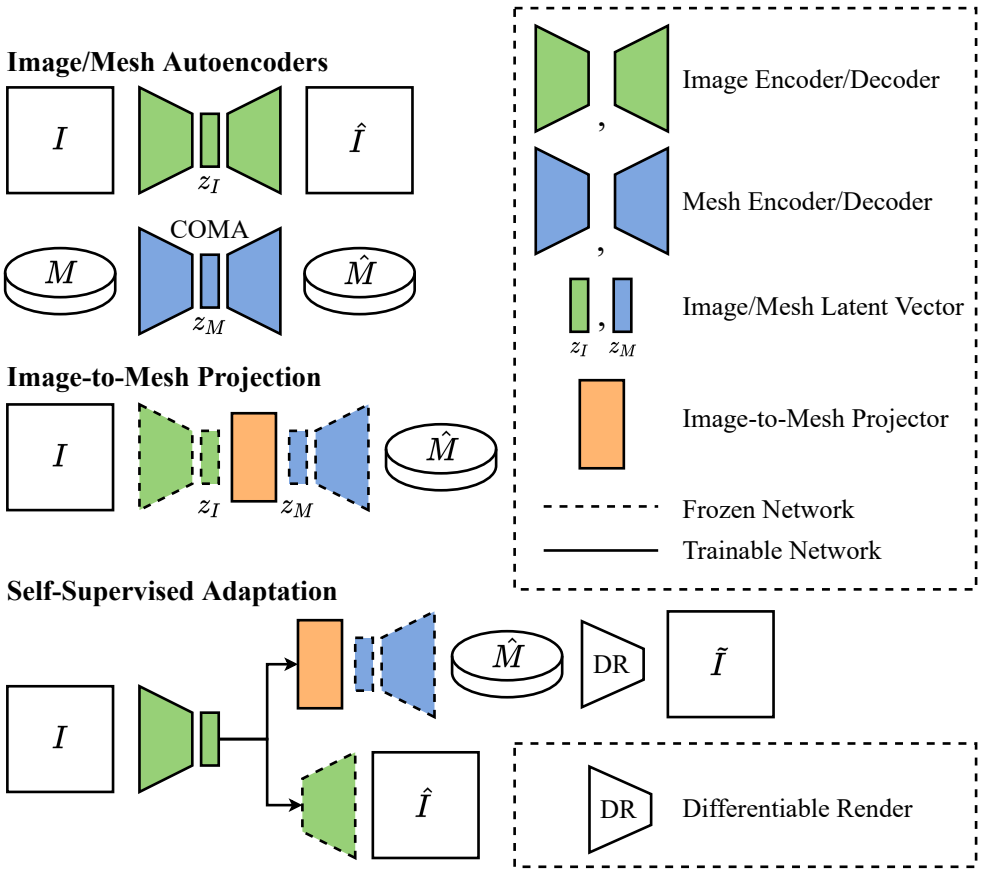


Figure 8.2: Training structure. The image-to-mesh projection network is optimized with pre-trained autoencoders. The self-supervised adaptation transfers the projection network to various domains with a differentiable render.

space, $\mathcal{V} \in \mathbb{R}^{n\times 3}$. The adjacency matrix $\mathcal{A} \in \{0,1\}^{n\times n}$ represents the edges. If vertices i and j are connected by an edge, $\mathcal{A}_{ij} = 1$, and $\mathcal{A}_{ij} = 0$ otherwise.

Supervised Image-to-Mesh Projection

Our goal is to map an input depth map I to a volumetric mesh $\mathcal{M}$. Although depth maps provide geometrical information for the contact surface, the projection from surface images to volumetric vertex positions is not injective and is hard to analyze. Specifically, different displacements can generate the same surface observation. Thus, in this chapter, we assume a fixed mesh tessellation (i.e., $\mathcal{A}$ fixed) to enforce the injective mapping and use a neural network to learn the underlying projection $\hat{\mathcal{M}} = f_\theta(I)$, with θ being the parameters of the network.

The image-to-mesh projection is learned with latent representations. Compared to previous work [79], the image observations have higher variance and more noise. This chapter introduces elaborate model designs, data augmentations, and self-supervised adaptations to resolve such difficulties.

Fig. 8.2 shows the training structure of the network. The image variational autoencoder (VAE) (in green) reconstructs depth maps I to $\hat{I}$ and is trained as a β-VAE:

$$\ell_I = \text{MSE}(I - \hat{I}) + \lambda_I \text{KL}(q(z_I|I) \parallel \mathcal{N}(0,1)) \tag{8.1}$$

where q is the image encoder, λ_I is the weight for the KL divergence term, and z_I is the latent vector.

We adopt the convolutional mesh autoencoders (COMA)[98] for the volumetric mesh VAE (shown in blue). COMA uses spectral graph convolutional networks [19] to extract features and a hierarchical pooling operation to reduce vertices. The network is trained with:

$$\ell_M = \text{MSE}(\mathcal{M} - \hat{\mathcal{M}}) + \lambda_M \text{KL}(h(z_M|I) \parallel \mathcal{N}(0,1)) \tag{8.2}$$

where h is the mesh encoder, λ_M is the KL loss weight, z_M is the latent vector, and the MSE is computed based on corresponding vertex positions $(\mathcal{V}, \hat{\mathcal{V}})$.

The latent projection model (shown in orange) is comprised of three fully connected layers. It is trained in a supervised manner with the encoder and decoder frozen. The details for the network, the latent dimensions, and weights are chosen via hyperparameter search, which is discussed in the following.

Self-Supervised Adaptation

When deploying the trained network to the real world, covariate shift problems may reduce the performance significantly [138]. Moreover, the real-world data only has depth maps $\{I_j\}$, and the ground-truth volumetric meshes are not available, making it hard to fine-tune the network in a supervised manner. Thus, we propose a self-supervised adaptation framework (Fig. 8.2) to resolve the covariate shift.

Specifically, the reconstructed volumetric mesh $\hat{\mathcal{M}}$ is rendered to the image $\tilde{I}$ using a differentiable renderer, which allows gradients to propagate backward. In parallel, we use the pre-trained image VAE to reconstruct the input depth map $\hat{I}$. The image VAE works as a noise filter as suggested in [60]. In practice, removing the image VAE can lead to poor adaptation results. The network is adapted using the mesh decoder with frozen weights to minimize the loss:

$$\ell_{adapt} = \text{MSE}(\tilde{I} - \hat{I}) \tag{8.3}$$

Datasets

Labeled synthetic data $\{(I_i, \mathcal{M}_i)\}$ and unlabeled real-world data $\{(I_j)\}$ are required to train the image-to-mesh projection and adapt the network among different domains.

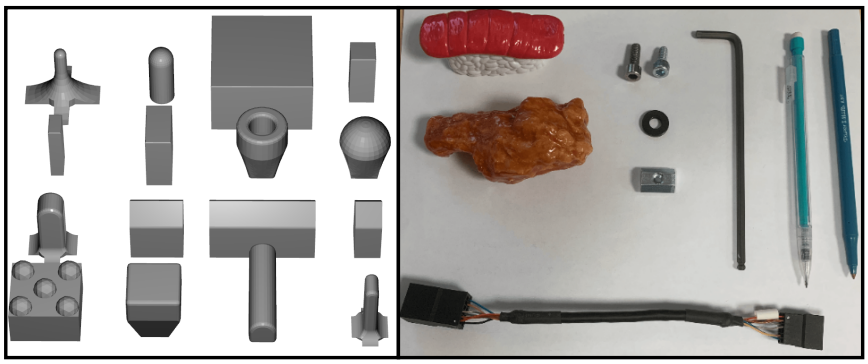


Figure 8.3: *Left*: Primitive indenters in simulation. *Right*: Novel contact objects in the real world.

Synthetic data. Labeled image-mesh pairs $\{(I_i, \mathcal{M}_i)\}$ for $i \in [1, ..., N]$ can be simulated using FEM and synthetic cameras. In this work, FEM is performed using the GPU-based Isaac Gym [69]. Isaac Gym models the dynamics of deformable bodies using linear-elastic models and assumes isotropic Coulomb contacts. The results of the simulation are optimized to match the real-world deformation.

A FEM model for the GelSlim is created with a similar procedure as [79]. The elastomer pad is modeled as a cylinder with a $1.75cm$ radius and $0.3cm$ height. The volumetric mesh has 5,415 nodes and 23,801 edges. A rigid backplate is added to imitate the structure of the physical GelSlim, Fig 8.1(b). To generate labeled data pairs, 16 primitive indenters (Fig. 8.3–Left) are utilized to interact with the elastomer at randomized positions and rotations. The primitive shapes contain a variety of complexity, texture, and geometry to reflect daily household objects.

The Isaac Gym simulator collects vertex positions $\mathcal{M}$ at each contact trajectory. The depth map I is then rendered based on the mesh $\mathcal{M}$ with a synthetic camera. This chapter uses an orthographic camera with a $\pm 1.75cm$ view range, which aligns with the specifications of the physical GelSlim. To optimize the FEM model in Isaac Gym, this chapter reuses the calibration data, contact images of a marble of known dimensions. From the depth map I, the contact position can be accurately estimated by finding the maximum displacement point. The contact trajectory can then be reproduced in the simulation, which yields a deformed mesh $\mathcal{M}$. Then, a depth image $\tilde{I}$ is rendered based on the simulated mesh $\mathcal{M}$. The elastic modulus E, Poisson's ratio ν, and surface friction μ are designated as free parameters in the simulator. A cross-entropy search strategy is used to find the best parameters:

$$E, \nu, \mu = \arg\min_{E,\nu,\mu} \left\| I - \tilde{I} \right\|$$

The optimal values for E, ν, μ are 145MPa, 0.32, and 0.94, respectively. Fig. 8.4 shows examples of synthetic data pairs with the calibrated FEM model.

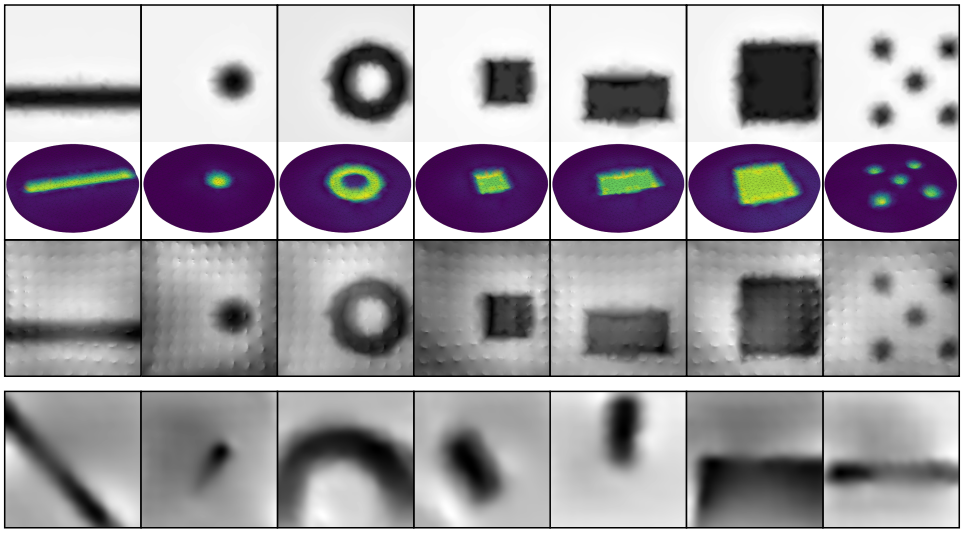


Figure 8.4: Data samples. *Top*: Raw synthetic depth observations, corresponding ground-truth meshes, and augmented synthetic depth observations. *Bottom*: Real-world depth observations for sample indenters.

Real-world data. Real-world datasets $\{I_j\}$ are obtained with physical GelSlim sensors and various indenters (Fig. 8.4). Primitive indenters are 3D printed, and interaction with the sensor is randomized. Besides primitive shapes, several household and industrial objects are used as a novel set (Fig. 8.3–Right). The novel set represents common objects that the GelSlim will work with. Moreover, we use two GelSlim sensors to collect real-world data.

Image augmentations. As shown in Fig. 8.4, the appearance of synthetic images is quite different from that of real-world depth maps. The depth reconstruction process for the physical GelSlim introduces significant noise into the image, enlarging the sim-to-real gap. To enhance the performance in the real world, this chapter injects Perlin noise and adds a real-world reference noise image into the synthetic images [138]. The Perlin noise provides a realistic gradient for the image and imitates the real-world camera noise. The reference image provides sensor-specific noise. Fig. 8.4 provides examples of the noised images.

In total, 1.28M unique labeled image-mesh pairs were obtained from the simulator, and 1,651 real-world images were obtained for 2 GelSlim sensors with 19 indenters.

8.3 Experiments

In this section, we present the network details, experiments for supervised image-to-mesh projection, self-supervised adaptation, and a comparative evaluation with a baseline.

Table 8.1: Experiments with synthetic data pairs. The root-mean-square error (RMSE, in cm) is measured between the ground-truth vertex positions $\mathcal{M}$ and predicted vertex positions $\hat{\mathcal{M}}$. The results with different dimensions of latent space.

λ_I \ λ_M	0	200	400	800
0	0.141	0.073	0.124	0.150
100	0.082	**0.012**	0.025	0.037
200	0.094	0.035	0.031	0.046

Table 8.2: Experiments with synthetic data pairs. The results with different loss weights.

λ_I \ λ_M	0	200	400	800
0	0.141	0.073	0.124	0.150
100	0.082	**0.012**	0.025	0.037
200	0.094	0.035	0.031	0.046

Network Details

As described above, we use an image VAE, a mesh VAE, and a latent projection module. In the image VAE, the encoder includes five downsampling layers with feature sizes 32, 64, 128, 256, 512 and two fully connected layers with 128 neurons each. In the volumetric mesh VAE, the encoder consists of four Chebyshev convolutional filters [19] with feature sizes 16, 16, 16, 32 and an output fully connected layer with 128 neurons. Each Chebyshev convolution is down-sampled by a factor of four. The image and mesh decoder are symmetric with the encoders. The latent projection module has three fully connected layers with 256, 512, and 256 neurons. All networks use the Adam optimizer with a learning rate of $1e-3$ and decay of 0.99.

Supervised Projection

Our proposed supervised image-to-mesh projection depends on several hyperparameters. In this section, we empirically estimate these. Furthermore, we pre-train the VAEs prior to training the image-to-mesh projection. We evaluate the pre-training by comparing with training the image-to-mesh projection directly from scratch.

The results reported here use a 80/20 split on the synthetic dataset for training and validation. Each model was trained for 300 epochs. We report the mean validation root-mean-square-error (RMSE) for the projected meshes.

Latent dimensions. We compare the image-to-mesh projection results for a 64, 128, and 256-dimensional latent space for each VAE, shown in Table 8.1. The 128-dimensional latent

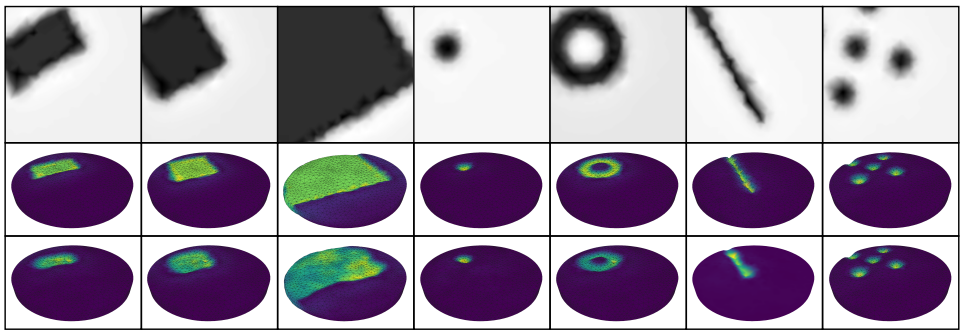


Figure 8.5: Image-to-mesh projection results with synthetic data. *First row*: Input depth observations. *Second row*: Corresponding ground-truth mesh. *Third row*: Reconstructed volumetric mesh with our approach.

space for both VAEs gives the best results.

Loss weights. We also compared the effectiveness of different values for λ_I, λ_M, from (8.1) and (8.2), shown in Table 8.2. We can see that the variational encoding, i.e., $\lambda_I > 0$, $\lambda_M > 0$, significantly improves the performance of latent projection, with the best performance for $\lambda_I = 100$, $\lambda_M = 200$. This suggests that the KL divergence term enforces a more meaningful latent distribution compared to a vanilla autoencoder. Fig. 8.5 shows a batch of projection results using the best-performing model.

Pre-training. Given the best-performing model, we investigate the usefulness of the VAE pre-training. We trained the image-to-mesh network from scratch with variational encoding. The training and validation errors were $0.009cm$ and $0.085cm$, respectively. This suggests that the network overfits without the pre-training, which aligns with the findings presented in [79].

Self-Supervised Adaptation

We propose a self-supervised adaptation method and synthetic data augmentations to resolve the covariate shift problem. This section provides results and ablation studies for the proposed method. We show that neither adaptation nor augmentation can achieve the objective alone, and the image VAE improves the adaptation results. Finally, we demonstrate that the proposed methods can adapt networks from simulation to reality, from primitive to novel contacts, and from one sensor to another.

The adaptation is performed with the real-world dataset $\{I_j\}$, without ground-truth mesh availability. To evaluate the performance of the adaptations, we use the RMSE between $\hat{I}$

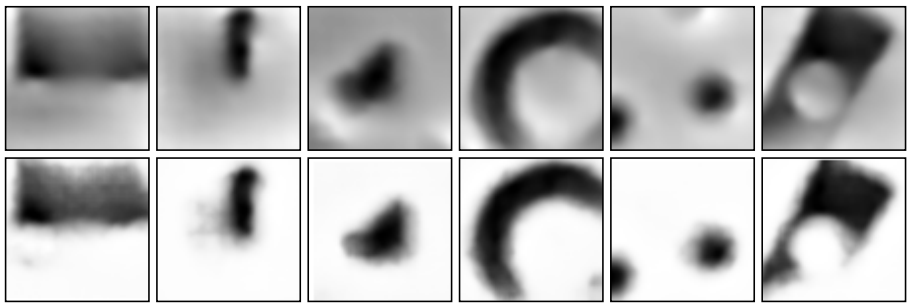


Figure 8.6: Reconstruction results for the image VAE with real-world images. *First row*: Real-world image observations. *Second row*: Reconstructed image with pre-trained VAE. The image VAE can effectively remove visual noises for both primitive and novel contacts.

Table 8.3: Experiments with real-world data. The root-mean-square error (RMSE) is measured between reconstructed images $\tilde{I}$ and rendered images $\hat{I}$. Ablation studies for adaptation, data augmentation, and VAE filtering.

	RMSE (cm)
Adapt + Aug.	**0.12**
No Aug. No Adapt	1.03
Only Aug.	0.57
Only Adapt	0.79
Adapt + Aug. w/o VAE	0.87

and $\tilde{I}$ as the evaluation metric, where $\hat{I}$ is the reconstructed input depth map via the pre-trained image VAE. As shown in Fig. 8.6, we can observe that the image VAE is robust in different domains and can effectively remove noise. Specifically, we tested the VAE on augmented synthetic images. Results show that the pre-trained image VAE can reconstruct the clean depth map with an RMSE of $0.07cm$.

Ablation studies. We compare the effects of the adaption model, synthetic data augmentations, and image VAE filtering. The results are listed in Table 8.3. As the table shows, data augmentation and self-supervised adaptation both contribute to resolving the sim-to-real gap. We observe that using only adaptation, or only augmentation, results in lower performance. The reason for higher performance when both are combined is two-fold. On one hand, the data augmentation enlarges the distribution of the synthetic dataset, which causes the real-world data to be within distribution (or close to). On the other hand, the adaptation model transfers the network from the simulated distribution to the real-world distribution, ensuring invariant feature encodings. Table 8.3 also shows that the VAE filter improves adaptation performance. It removes visual noises in real-world data and stabilizes

Table 8.4: Experiments with real-world data. Domain adaptation results.

Source → Target	RMSE before/after Adaptation (cm)
Sim-Prim. → *Real-Prim*	0.57 → 0.12
Sim-Prim → *Real-Prim-2*	0.77 → 0.20
Real-Prim → *Real-Prim-2*	0.35 → 0.16
Real-Prim → *Real-Novel*	0.64 → 0.41
Sim-Prim → *Real-Novel*	1.30 → 0.62

Networks were trained or tuned on source domains and then adapted to target domains. The RMSEs were measured before and after the adaptation.

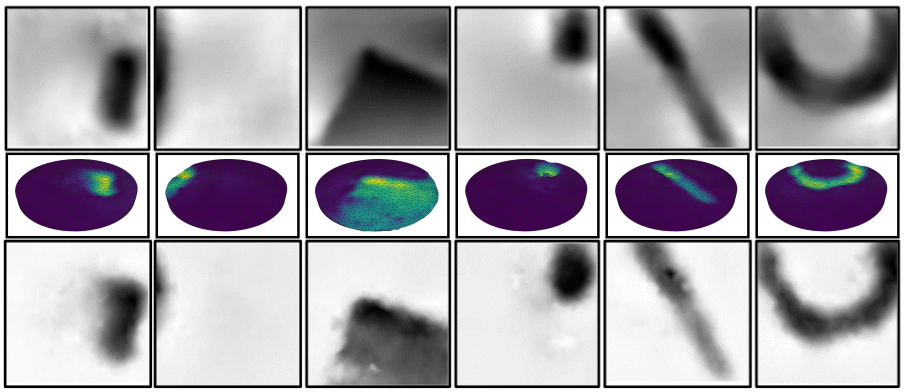


Figure 8.7: Experiments with real-world primitive contact objects. *First row*: Input depth observations. *Second row*: Reconstructed volumetric meshes. *Third row*: Rendered depth images from reconstructed meshes.

the adaptation process. A batch of qualitative reconstruction examples is shown in Fig 8.7.

Domain adaptations. Previous sections introduce various data domains, including simulated data with primitive contact objects (*Sim-Prim*), real-world data with primitive contact objects (*Real-Prim*), real-world data with novel contact objects (*Real-Novel*), and real-world primitive data with a second GelSlim sensor (*Real-Prim-2*).

While we showed the performance of the *Sim-Prim* → *Real-Prim* experiment above, Table 8.4 and Fig. 8.8 show the transfer results among other domains. The networks were first pre-trained or fine-tuned on source domains and then adapted to target domains. Experiments *Sim-Prim* → *Real-Prim*, *Sim-Prim* → *Real-Prim-2*, and *Real-Prim* → *Real-Prim-2* were executed with the same primitive shapes. The adaptation improves performance in all cases. For experiment *Real-Prim* → *Real-Novel*, the acquisition was done with the real sensor, but adaptation now is for primitive to novel shapes. From Table 8.4, we see that while

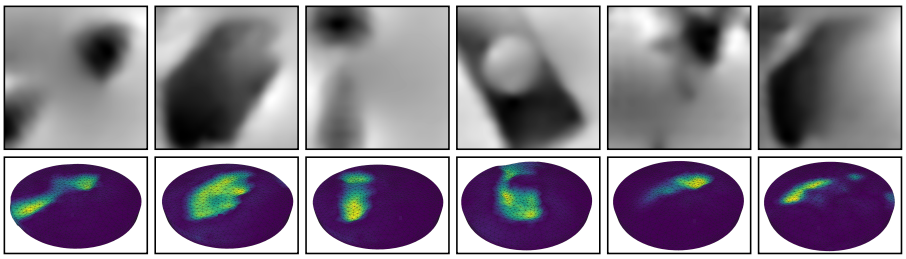


Figure 8.8: Experiments with real-world novel contact objects. *First row*: Input depth observations. *Second row*: Reconstructed volumetric mesh from the network.

the performance improves, improvement is less compared to the prior experiments. For the final experiment, *Sim-Prim* $\rightarrow$ *Real-Novel* transfer is both from sim-to-real, as well as from primitive to novel shapes, and thus is the hardest. Again, adaptation significantly improves performance, and predicted deformations were visually accurate (see Fig. 8.8). The results suggest that the proposed adaptation method can effectively improve the performance of the network under both visual noise and shape differences.

Overall performance for experiment *Sim-Prim* $\rightarrow$ *Real-Novel* is less compared to the other experiments. The covariate shifts for visual noise and shape differences are not correlated, and adaptation for each separately performs better compared to adaptation for both. Further optimizing performance for both in a self-supervised manner is a challenging topic for future work.

Baseline Comparisons

For regression from image observations to mesh deformations, two methods were evaluated: 1) our proposed method, denoted as *Volumetric Mesh*, and 2) a surface reconstruction baseline [65, 115], denoted as *Surface Mesh*. The latter uses tracking markers to determine the movement of the elastomer surface. Note that the *Surface Mesh* method does not estimate the volumetric mesh directly but rather gives a sparse surface deformation field for each contact.

Fig. 8.9 shows the reconstructed meshes with both methods. Interestingly, the computation takes 0.02 *sec.* for the *Volumetric Mesh* synthesis with our proposed approach versus 0.04 *sec.* for the *Surface Mesh* method (potentially due to the requirement of marker detection). Fig. 8.9 shows correspondences between the *Volumetric Mesh* and the *Surface Mesh* on the elastomer surface. In addition, we also conducted contact force estimations of the GelSlim based on the predicted meshes. An inverse FEM was used to compute the contact force with a linear-elastic model [115]. Compared to the *Surface Mesh* method, our method constructed more plausible and denser force distributions with the volumetric FEM mesh (Fig. 8.9). For example, predictions around contact edges were more realistic and had higher

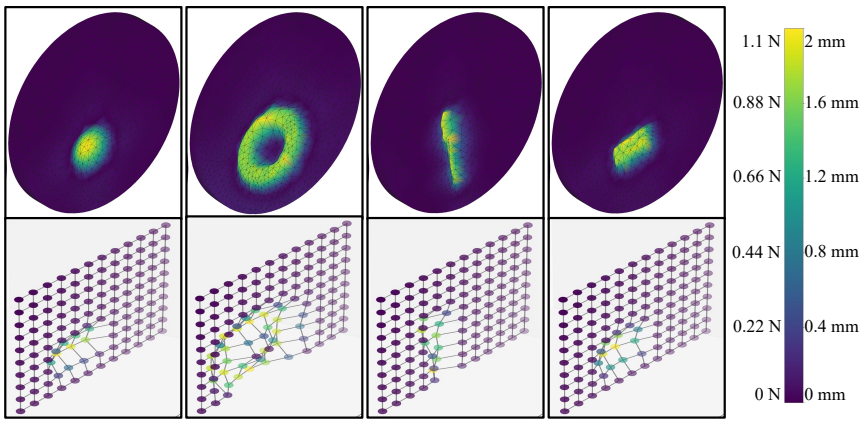


Figure 8.9: *First row*: Reconstructed meshes and estimated contact forces with the proposed approach, *Volumetric Mesh*. *Second row*: Comparison with baseline method, *Surface Mesh* [115].

resolution. Predicted force profiles were also smoother, which was due to the influence of internal vertices. We hypothesize that such denser force distributions obtained from our method may help improve policy learning for robotic manipulation tasks.

8.4 Chapter Summary

This chapter presents a framework to synthesize volumetric meshes of vision-based tactile sensors for novel contact interactions. Our work has several key contributions. First, we present a 3D FEM simulator for vision-based tactile sensors and a simulator calibration approach. Second, we generate a dataset for the GelSlim sensor with both simulated and real-world contacts using primitive and novel shapes. Third, we propose a label-free adaptation method and image augmentations for domain transfers; we show that this approach can effectively transfer networks to various visual and different shape scenarios. Lastly, our network efficiently reconstructs the volumetric mesh with depth images and precisely estimates the contact profiles of different shapes. Using these learned and adapted networks, our method can reconstruct the deformations of the elastomer for vision-based tactile sensors in various domains, as indicated by the quantitative and qualitative results.

Chapter 9

Conclusions and Further Works

9.1 Conclusions

In general, the objective of this dissertation is to incorporate contact modeling in learning robot manipulations. Robotic grasp planning is first introduced in Chaters 2, 3, and 4. Various scenarios were then investigated in Chapters 5, 6, and 7 for contact-aware robotic manipulation. Eventually, contact sensing was discussed in Chapter 8.

Chapter 2 introduced the contrastive grasp proposal network (CGPN), a novel approach for generating robust 6-DoF grasps. By leveraging contrastive learning and variant depth image processing, CGPN effectively bridges the sim-to-real gap, achieving superior performance over existing grasping algorithms in real-world scenarios.

In Chapter 3, we tackled the issue of data sparsity in grasp planning with the Maximum Likelihood Grasp Sampling Loss (MLGSL). This approach significantly improved data efficiency, allowing FCN models to learn effective grasping strategies from minimal labels and achieve a high grasp success rate on household objects.

Chapter 4 focused on multi-fingered grasp pose detection in cluttered environments. The proposed MF-GPD algorithm, integrating cross-entropy sampling and local optimization, demonstrated a high success rate in locating collision-free grasps, marking a substantial advancement in multi-fingered hand manipulation.

The dissertation then expanded its scope in Chapter 5, exploring model-based learning from demonstrations for diverse robotic manipulation tasks. Here, we introduced a framework that combines differentiable rendering and simulations, enabling efficient learning of complex manipulation skills from human demonstrations.

Chapter 6 addressed contact-allowed robotic goal-reaching with operational and null space control. The proposed receding horizon trajectory planner and hybrid solver showcased efficient and safe manipulation strategies in varied collision conditions.

Chapter 7 focused on robot assembly planning. We introduced the Part Assembly Sequence Transformer (PAST), a multi-level framework for generating assembly plans that include part sequences, motions, and contact points. PAST infers feasible assembly sequences

based on target blueprints and part geometries, marking a substantial advancement in automated assembly. The development of a large-scale benchmark dataset for part assembly sequences further underscores the impact of this work.

In Chapter 8, we investigated contact sensing in manipulation, focusing on synthesizing volumetric meshes of vision-based tactile sensors for novel contact interactions. The contributions of this chapter are manifold: the development of a 3D FEM simulator for vision-based tactile sensors, the creation of a comprehensive dataset for the GelSlim sensor, and the proposal of a label-free adaptation method for domain transfers. Our method effectively reconstructs the deformations of the elastomer for vision-based tactile sensors, showcasing its efficacy through quantitative and qualitative results.

In conclusion, this dissertation presented a series of interconnected advancements in the field of robotic manipulation with contact modeling. Each chapter contributed unique methodologies and insights, cumulatively pushing the boundaries of what robotic systems can achieve in terms of dexterity, efficiency, and adaptability.

9.2 Further Works

In addition to the research presented in this dissertation, the application of contact mechanics and other physical principles offers promising avenues for enhancing robot learning in areas such as representation learning, the development of generalist robot policies, and skill transfer.

In the realm of robot representation learning, we postulate that focusing the network on human-environment interactions can yield more generalizable visual representations. This idea is crystallized in our work presented in [43], where we advocate for a human-oriented multi-task fine-tuning approach on pre-trained visual encoders. Each task in this approach represents a critical perceptual skill in everyday scenarios. We introduce the Task Fusion Decoder, a novel tool designed as a plug-and-play embedding translator. It leverages the interrelationships among these perceptual skills to guide the representation learning process, ultimately enhancing the encoding of structures crucial for robotic manipulation tasks. This methodology has demonstrated substantial improvements in the representation capabilities of three state-of-the-art visual encoders across a diverse array of robotic tasks and embodiments, both in simulations and real-world settings.

Moreover, the incorporation of physical principles in robot learning can be greatly facilitated by large-scale data. Real-world datasets inherently encode physical laws, and models trained on such datasets have shown remarkable efficiency in various applications. This leads to an intriguing question: can we establish a 'generalist' robot policy adaptable across diverse robots, tasks, and environments, akin to the consolidation of pre-trained models in NLP and Computer Vision? In [16], we explore this possibility by providing standardized datasets and models that encompass a wide range of robotic manipulation scenarios. Our dataset, assembled from 22 different robots across 21 institutions, showcases 527 skills across 160266 tasks. The resulting high-capacity model, RT-X, demonstrates the feasibility of posi-

tive transfer, enhancing multiple robots' capabilities by drawing on experiences from various platforms.

Finally, physical models can play a pivotal role in facilitating the transfer of robotic skills. In [131], we introduce DiffTransfer, a framework that leverages differentiable physics simulation for efficient skill transfer in robotics. DiffTransfer utilizes a novel path-planning method, incorporating Q-learning, to navigate through the task space, adapting actions from one sub-task to another. This method is grounded in the gradient information gleaned from differentiable physics simulations. Our implementation of DiffTransfer in simulation experiments, and its application to four challenging robotic manipulation transfer tasks, underscores its effectiveness and sets the stage for further explorations in skill transfer for intelligent robots.

Bibliography

[1] Rika Antonova et al. "Rethinking Optimization with Differentiable Simulation from a Global Perspective". In: *6th Annual Conference on Robot Learning*. 2022.

[2] P. Bao, L. Zhang, and X. Wu. "Canny Edge Detection Enhancement by Scale Multiplication". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27.09 (2005), pp. 1485–1490. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2005.173.

[3] Peter Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv* (2018).

[4] Maria Bauza, Oleguer Canal, and Alberto Rodriguez. "Tactile Mapping and Localization from High-Resolution Tactile Imprints". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019. DOI: 10.1109/ICRA.2019.8794298.

[5] Maria Bauza et al. "Tactile Object Pose Estimation from the First Touch with Geometric Contact Rendering". In: *ArXiv Preprint* (2020). eprint: 2012.05205.

[6] Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Zico Kolter. "Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction". In: *ICML*. 2020.

[7] Wenjing Bian et al. "Nope-nerf: Optimising neural radiance field with no pose prior". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4160–4169.

[8] Stephen Boyd and Ben Wegbreit. "Fast Computation of Optimal Contact Forces". In: *Robotics, IEEE Transactions on* 23 (Jan. 2008).

[9] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[10] Chun-Fu (Richard) Chen, Quanfu Fan, and Rameswar Panda. "CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification". In: *International Conference on Computer Vision (ICCV)*. 2021.

[11] Claire Chen et al. "TrajectoTree: Trajectory Optimization Meets Tree Search for Planning Multi-contact Dexterous Manipulation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8262–8268. DOI: 10.1109/IROS51168.2021.9636346.

[12] Ting Chen et al. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607. URL: http://proceedings.mlr.press/v119/chen20j.html.

[13] Xuefeng Chen, Xiabi Liu, and Yunde Jia. "Combining evolution strategy and gradient descent method for discriminative learning of Bayesian classifiers". In: *11th Annual Genetic and Evolutionary Computation Conference, GECCO-2009*. Jan. 2009, pp. 507–514. DOI: 10.1145/1569901.1569972.

[14] Yun-Chun Chen et al. "Neural Shape Mating: Self-Supervised Object Assembly with Adversarial Shape Priors". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[15] Cheng Chi et al. "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2023.

[16] Embodiment Collaboration et al. "Open X-Embodiment: Robotic Learning Datasets and RT-X Models". In: *arXiv Preprint*. 2023. eprint: 2310.08864 (cs.RO).

[17] Erwin Coumans and Yunfei Bai. *Pybullet, a python module for physics simulation in robotics, games and machine learning*. 2017.

[18] Sudeep Dasari, Abhinav Gupta, and Vikash Kumar. "Learning Dexterous Manipulation from Exemplar Object Trajectories and Pre-Grasps". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 3889–3896.

[19] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems*. 2016.

[20] Amaury Depierre, Emmanuel Dellandréa, and Liming Chen. "Jacquard: A Large Scale Dataset for Robotic Grasp Detection". In: *2018 IROS*. 2018, pp. 3511–3516. DOI: 10.1109/IROS.2018.8593950.

[21] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[22] Mehmet R. Dogar and Siddhartha S. Srinivasa. "Push-grasping with dexterous hands: Mechanics and a method". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 2123–2130. DOI: 10.1109/IROS.2010.5652970.

[23] Siyuan Dong et al. "Tactile-RL for Insertion: Generalization to Objects of Unknown Geometry". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[24] Elliott Donlon et al. "GelSlim: A High-Resolution, Compact, Robust, and Calibrated Tactile-sensing Finger". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[25] Hugh Durrant-Whyte, Nicholas Roy, and Pieter Abbeel. "Motion Planning under Uncertainty in Highly Deformable Environments". In: *Robotics: Science and Systems VII*. 2012, pp. 241–248.

[26] Yongxiang Fan, Xinghao Zhu, and Masayoshi Tomizuka. "Optimization Model for Planning Precision Grasps with Multi-Fingered Hands". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1548–1554. DOI: 10.1109/IROS40897.2019.8967560.

[27] Yongxiang Fan et al. "Grasp planning for customized grippers by iterative surface fitting". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2018, pp. 28–34.

[28] Yongxiang Fan et al. "Real-time finger gaits planning for dexterous manipulation". In: *IFAC* 50.1 (2017), pp. 12765–12772.

[29] Yongxiang Fan et al. "Real-time grasp planning for multi-fingered hands by finger splitting". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4045–4052.

[30] Yongxiang Fan et al. "Real-time robust finger gaits planning under object shape and dynamics uncertainties". In: *2017 IROS*. IEEE. 2017, pp. 1267–1273.

[31] C. Daniel Freeman et al. *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation*. Version 0.0.15. 2021. URL: http://github.com/google/brax.

[32] Seyed Kamyar Seyed Ghasemipour et al. "Blocks Assemble! Learning to Assemble with Large-Scale Structured Reinforcement Learning". In: *International Conference on Machine Learning*. 2022.

[33] Raghav Goyal et al. "The "something something" video database for learning and evaluating visual common sense". In: *CoRR* abs/1706.04261 (2017). URL: https://20bn.com/datasets/something-something.

[34] Marcus Gualtieri and Robert Platt. "Learning 6-DoF Grasping and Pick-Place Using Attention Focus". In: *Conference on Robot Learning*. Vol. 87. 2018, pp. 477–486.

[35] Marcus Gualtieri and Robert Platt. "Learning Manipulation Skills via Hierarchical Spatial Attention". In: *IEEE Transactions on Robotics* 36.4 (), pp. 1067–1078. ISSN: 1941-0468. DOI: 10.1109/tro.2020.2974093.

[36] Nikolaus Hansen. "The CMA Evolution Strategy: A Tutorial". In: *arXiv Preprint*. 2016. DOI: 10.48550/ARXIV.1604.00772.

[37] Peter Hart, Nils Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/tssc.1968.300136. URL: https://doi.org/10.1109/tssc.1968.300136.

[38] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[39] Kaiming He et al. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[40] Francois R. Hogan et al. "Tactile Dexterity: Manipulation Primitives with Tactile Feedback". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020. DOI: 10.1109/ICRA40945.2020.9196976.

[41] Neville Hogan. "Impedance Control: An Approach to Manipulation: Part I—Theory". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (Mar. 1985), pp. 1–7. ISSN: 0022-0434. DOI: 10.1115/1.3140702.

[42] Jialei Huang et al. "Generative 3D Part Assembly via Dynamic Graph Learning". In: *Advances in Neural Information Processing Systems*. 2020.

[43] Mingxiao Huo et al. "Human-oriented Representation Learning for Robotic Manipulation". In: *arXiv preprint arXiv:2310.03023* (2023).

[44] Shan Jiang et al. "Single-Grasp Detection Based on Rotational Region CNN". In: *Advances in Computational Intelligence Systems*. Cham: Springer International Publishing, 2020, pp. 131–141. ISBN: 978-3-030-29933-0.

[45] Shiyu Jin et al. "Contact pose identification for peg-in-hole assembly under uncertainties". In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 48–53.

[46] Micah K. Johnson and Edward H. Adelson. "Retrographic sensing for the measurement of surface texture and shape". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009. DOI: 10.1109/CVPR.2009.5206534.

[47] Mikael Jorda, Elena Galbally Herrero, and Oussama Khatib. "Contact-Driven Posture Behavior for Safe and Interactive Robot Operation". In: *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9243–9249. DOI: 10.1109/ICRA.2019.8793691.

[48] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *arXiv Preprint*. 2011. DOI: 10.48550/ARXIV.1105.1186.

[49] Alexander Kirillov et al. "Segment Anything". In: *arXiv:2304.02643* (2023).

[50] J. J. Kuffner and S. M. LaValle. "RRT-connect: An efficient approach to single-query path planning". In: *2000 ICRA*. Vol. 2. 2000, 995–1001 vol.2. DOI: 10.1109/ROBOT.2000.844730.

[51] Mike Lambeta et al. "Digit: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 3838–3845.

[52] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *The International Journal of Robotics Research* 34.4-5 (), pp. 705–724. DOI: 10.1177/0278364914549607.

[53] Nathan F. Lepora et al. "From Pixels to Percepts: Highly Robust Edge Perception and Contour Following Using Deep Learning and an Optical Biomimetic Tactile Sensor". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2101–2107. DOI: 10.1109/LRA.2019.2899192.

[54] Yichen Li et al. "Learning 3D Part Assembly from a Single Image". In: *European conference on computer vision* (2020).

[55] Yulong Li, Andy Zeng, and Shuran Song. "Rearrangement Planning for General Part Assembly". In: *Conference on Robot Learning*. PMLR. 2023.

[56] Yunzhu Li et al. "Visual Grounding of Learned Physical Models". In: *ICML*. 2020.

[57] Hongzhuo Liang et al. "PointNetGPD: Detecting Grasp Configurations from Point Sets". In: *2019 International Conference on Robotics and Automation (ICRA)* (2019). DOI: 10.1109/icra.2019.8794435. URL: http://dx.doi.org/10.1109/ICRA.2019.8794435.

[58] Min Liu et al. "Deep Differentiable Grasp Planner for High-DOF Grippers". In: *2020 Robotics: Science and Systems (RSS)* (2020).

[59] Ruoshi Liu et al. "Zero-1-to-3: Zero-shot One Image to 3D Object". In: *arXiv:2303.11328* (2023).

[60] Romain Lopez et al. "Information Constraints on Auto-Encoding Variational Bayes". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 6117–6128.

[61] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 163–169. ISBN: 0897912276. DOI: 10.1145/37401.37422. URL: https://doi.org/10.1145/37401.37422.

[62] Qingkai Lu et al. "Multi-Fingered Grasp Planning via Inference in Deep Neural Networks". In: *IEEE robotics automation magazine* (2020).

[63] Qingkai Lu et al. "Planning Multi-Fingered Grasps as Probabilistic Inference in a Learned Deep Network". In: *Robotics Research* (2020), pp. 455–472.

[64] Daolin Ma, Siyuan Dong, and Alberto Rodriguez. "Extrinsic Contact Sensing with Relative-Motion Tracking from Distributed Tactile Measurements". In: *ArXiv Preprint* (2021). eprint: 2103.08108.

[65] Daolin Ma et al. "Dense Tactile Force Estimation using GelSlim and inverse FEM". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019.

[66] Jianqi Ma et al. "Arbitrary-Oriented Scene Text Detection via Rotation Proposals". In: *IEEE Transactions on Multimedia* 20.11 (2018), pp. 3111–3122. ISSN: 1941-0077. DOI: 10.1109/tmm.2018.2818020. URL: http://dx.doi.org/10.1109/TMM.2018.2818020.

[67] J. Mahler et al. "Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5620–5627. DOI: 10.1109/ICRA.2018.8460887.

[68] Jeffrey Mahler et al. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *Robotics: Science and Systems (RSS)*. 2017.

[69] Viktor Makoviychuk et al. "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning". In: *arXiv preprint arXiv:2108.10470* (2021).

[70] Roberto Martín-Martín et al. "Variable Impedance Control in End-Effector Space. An Action Space for Reinforcement Learning in Contact Rich Tasks". In: *International Conference of Intelligent Robots and Systems (IROS)*. 2019.

[71] Carolyn Matl, Josephine Koe, and Ruzena Bajcsy. "StRETcH: a Soft to Resistive Elastic Tactile Hand". In: *arXiv preprint arXiv:2105.08154* (2021).

[72] Benjamin W. McInroe et al. "Towards a Soft Fingertip with Integrated Sensing and Actuation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018. DOI: 10.1109/IROS.2018.8594032.

[73] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach". In: *Robotics: Science and Systems (RSS)* (2018).

[74] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Learning robust, real-time, reactive robotic grasping". In: *The International Journal of Robotics Research* 39.2-3 (). DOI: 10.1177/0278364919859066.

[75] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 2901–2910. DOI: 10.1109/ICCV.2019.00299. URL: https://doi.org/10.1109/ICCV.2019.00299.

[76] A. Murali et al. "6-DOF Grasping for Target-driven Object Manipulation in Clutter". In: *International Conference on Robotics and Automation (ICRA)*. 2020.

[77] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[78] Yashraj Narang et al. "Factory: Fast contact for robotic assembly". In: *Robotics: Science and Systems*. 2022.

[79] Yashraj Narang et al. "Sim-to-Real for Robotic Tactile Sensing via Physics-Based Simulation and Learned Latent Projections". In: *Proceeding of the 2021 International Conference on Robotics and Automation (ICRA)* (2021).

[80] Yashraj S. Narang et al. "Interpreting and Predicting Tactile Signals via a Physics-Based and Data-Driven Framework". In: *ArXiv Preprint* (2020). eprint: 2006.03777.

[81] Maria Augusta Neto et al. "Finite Element Method for 3D Solids". In: *Engineering Computation of Structures: The Finite Element Method*. Cham: Springer International Publishing, 2015, pp. 233–263. ISBN: 978-3-319-17710-6. DOI: 10.1007/978-3-319-17710-6_7.

[82] Peiyuan Ni et al. "PointNet++ Grasping: Learning An End-to-end Spatial Grasp Generation Algorithm from Sparse Point Clouds". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3619–3625. DOI: 10.1109/ICRA40945.2020.9196740.

[83] nimble. *Nimble Physics Documentation*. https://nimblephysics.org/docs.

[84] Yuki Onishi and et al. "An automated fruit harvesting robot by using deep learning". In: *Robomech*. 2019. DOI: https://doi.org/10.1186/s40648-019-0141-2.

[85] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: 1807.03748 [cs.LG].

[86] Takayuki Osa et al. "An algorithmic perspective on imitation learning". In: *Foundations and Trends® in Robotics* 7.1-2 (2018), pp. 1–179.

[87] Akhil Padmanabha et al. "Omnitact: A multi-directional high-resolution touch sensor". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020.

[88] J. Pan, S. Chitta, and D. Manocha. "FCL: A general purpose library for collision and proximity queries". In: *2012 ICRA*. 2012, pp. 3859–3866. DOI: 10.1109/ICRA.2012.6225337.

[89] Tao Pang et al. *Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models*. 2022. DOI: 10.48550/ARXIV.2206.10787.

[90] Dongwon Park and Se Young Chun. "Classification based Grasp Detection using Spatial Transformer Network". In: *CoRR* abs/1803.01356 (2018). arXiv: 1803.01356. URL: http://arxiv.org/abs/1803.01356.

[91] Andreas ten Pas et al. "Grasp Pose Detection in Point Clouds". In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1455–1473. DOI: 10.1177/0278364917735594. URL: https://doi.org/10.1177/0278364917735594.

[92] Vladimír Petrík et al. "Learning object manipulation skills via approximate state estimation from real videos". In: *Conference on Robot Learning*. PMLR. 2021, pp. 296–312.

[93] Tobias Pfaff et al. "Learning Mesh-Based Simulation with Graph Networks". In: *International Conference on Learning Representations*. 2021.

[94] Calder Phillips-Grafflin and Dmitry Berenson. "A representation of deformable objects for motion planning with no physical simulation". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 98–105. DOI: 10.1109/ICRA.2014.6906595.

[95] Charles Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16. URL: https://doi.org/10.1109/CVPR.2017.16.

[96] Charles Ruizhongtai Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 2017, pp. 5099–5108. URL: https://proceedings.neurips.cc/paper/2017/hash/d8bf84be3800d12f74d8b05e9b89836f-Abstract.html.

[97] Haozhi Qi et al. "In-Hand Object Rotation via Rapid Motor Adaptation". In: *Conference on Robot Learning (CoRL)*. 2022.

[98] Anurag Ranjan et al. "Generating 3D faces using Convolutional Mesh Autoencoders". In: *European Conference on Computer Vision (ECCV)*. 2018.

[99] Joseph Redmon and Anelia Angelova. "Real-time grasp detection using convolutional neural networks". In: *2015 ICRA*. 2015, pp. 1316–1322. DOI: 10.1109/ICRA.2015.7139361.

[100] Maximo Roa and Raul Suarez. "Grasp Quality Measures: Review and Performance". In: *Autonomous Robots* 38 (2014), pp. 65–88. DOI: 10.1007/s10514-014-9402-3.

[101] S. Rodriguez, Jyh-Ming Lien, and N.M. Amato. "Planning motion in completely deformable environments". In: *2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 2006, pp. 2466–2471. DOI: 10.1109/ROBOT.2006.1642072.

[102] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 627–635. URL: https://proceedings.mlr.press/v15/ross11a.html.

[103] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[104] Hamid Sadeghian et al. "Multi-priority control in redundant robotic systems". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 3752–3757. DOI: 10.1109/IROS.2011.6094609.

[105] Alvaro Sanchez-Gonzalez et al. "Graph networks as learnable physics engines for inference and control". In: *ArXiv Preprint* (2018). eprint: 1806.01242.

[106] Alvaro Sanchez-Gonzalez et al. "Learning to Simulate Complex Physics with Graph Networks". In: *International Conference on Machine Learning*. 2020.

[107] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. "On-Policy Dataset Synthesis for Learning Robot Grasping Policies Using Fully Convolutional Deep Networks". In: *IEEE Robotics and Automation Letters* (2019).

[108] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[109] N. Shafii, S. H. Kasaei, and L. S. Lopes. "Learning to grasp familiar objects using object view recognition and template matching". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 2895–2900. DOI: 10.1109/IROS.2016.7759448.

[110] Lin Shao et al. "UniGrasp: Learning a Unified Model to Grasp With Multifingered Robotic Hands". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2286–2293. ISSN: 2377-3774. DOI: 10.1109/lra.2020.2969946. URL: http://dx.doi.org/10.1109/LRA.2020.2969946.

[111] Yu She et al. "Cable Manipulation with a Tactile-Reactive Gripper". In: *Robotics: Science and Systems (RSS)*. 2020.

[112] Lingfeng Sun et al. "Efficient Multi-Task and Transfer Reinforcement Learning With Parameter-Compositional Framework". In: *IEEE Robotics and Automation Letters* 8.8 (2023), pp. 4569–4576.

[113] Lingfeng Sun et al. "PaCo: Parameter-Compositional Multi-task Reinforcement Learning". In: *NeurIPS*. 2022.

[114] Priya Sundaresan et al. *Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data*. 2020. arXiv: 2003.01835 [cs.RO].

[115] Ian Taylor, Siyuan Dong, and Alberto Rodriguez. "GelSlim3.0: High-Resolution Measurement of Shape, Force and Slip in a Compact Tactile-Sensing Finger". In: *ArXiv Preprint* abs/2103.12269 (2021).

[116] Erik G. Thompson. *Introduction to the Finite Element Method: Theory, Programming and Applications*. Wiley Text Books, 2004. ISBN: 0471267538.

[117] Yunsheng Tian et al. "Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly". In: *ACM Trans. Graph.* 41.6 (2022).

[118] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017.

[119] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. "Differentiable Signed Distance Function Rendering". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 2022), 125:1–125:18. DOI: 10.1145/3528223.3530139.

[120] Video for paper: 6-DoF Contrastive Grasp Proposal Network. https://www.youtube.com/watch?v=rXldZyf6Kks.

[121] Video for paper: 6-DoF Contrastive Grasp Proposal Network. https://www.youtube.com/watch?v=XNYkWSHkAaU.

[122] Video for paper: Learn to Grasp with Less Supervision using MLGSL. https://www.youtube.com/watch?v=vHTMWdj4n7o.

[123] Chen Wang et al. "SwingBot: Learning Physical Features from In-hand Tactile Exploration for Dynamic Swing-up Manipulation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. DOI: 10.1109/IROS45743.2020.9341006.

[124] Yue Wang et al. "Dynamic Graph CNN for Learning on Point Clouds". In: *ACM Transactions on Graphics (TOG)* (2019).

[125] Website for paper: Diff-LfD. https://sites.google.com/view/diff-lfd.

[126] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach. Learn.* 8.3–4 (May 1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696.

[127] Karl D. D. Willis et al. "Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences". In: *ACM Transactions on Graphics* 40.4 (2021).

[128] Karl D.D. Willis et al. "JoinABLe: Learning Bottom-Up Assembly of Parametric CAD Joints". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 15849–15860.

[129] W. Wohlkinger et al. "3DNet: Large-scale object class recognition from CAD models". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 5384–5391.

[130] Sanghyun Woo et al. "CBAM: Convolutional Block Attention Module". In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[131] Yuqi Xiang et al. *Diff-Transfer: Model-based Robotic Manipulation Skill Transfer via Differentiable Physics Simulation*. 2023. arXiv: 2310.04930 [cs.RO].

[132] Mingxin Yu et al. "RoboAssembly: Learning Generalizable Furniture Assembly Policy in a Novel Multi-robot Contact-rich Simulation Environment". In: *International Conference on Robotics and Automation (ICRA)*. 2022.

[133] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. "GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force". In: *Sensors* 17.12 (2017). ISSN: 1424-8220.

[134] Andy Zeng and et al. "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching". In: *2018 ICRA*. 2018, pp. 3750–3757. DOI: 10.1109/ICRA.2018.8461044.

[135] Andy Zeng et al. "Learning Synergies between Pushing and Grasping with Self-supervised Deep Reinforcement Learning". In: *IROS*. 2018.

[136] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.

[137] Xiang Zhu, Shucheng Kang, and Jianyu Chen. "A Contact-Safe Reinforcement Learning Framework for Contact-Rich Robot Manipulation". In: *arXiv Preprint*. 2022. DOI: 10.48550/ARXIV.2207.13438.

[138] Xinghao Zhu et al. "6-DoF Contrastive Grasp Proposal Network". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 6371–6377. DOI: 10.1109/ICRA48506.2021.9561954.

[139] Xinghao Zhu et al. "Allowing Safe Contact in Robotic Goal-Reaching: Planning and Tracking in Operational and Null Spaces". In: IEEE International Conference on Robotics and Automation (ICRA), 2023.

[140] Xinghao Zhu et al. "Diff-LfD: Contact-aware Model-based Learning from Visual Demonstration for Robotic Manipulation via Differentiable Physics-based Simulation and Rendering". In: *Conference on Robot Learning*. PMLR. 2023.

[141] Xinghao Zhu et al. "Fanuc Manipulation: A Dataset for Learning-based Manipulation with FANUC Mate 200iD Robot". In: 2023.

[142] Xinghao Zhu et al. "Learn to grasp with less supervision: A data-efficient maximum likelihood grasp sampling loss". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 721–727.

[143] Xinghao Zhu et al. "Learning to Synthesize Volumetric Meshes from Vision-based Tactile Imprints". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 4833–4839. DOI: 10.1109/ICRA46639.2022.9812092.

[144] Xinghao Zhu et al. "Multi-level Reasoning for Robotic Assembly: From Sequence Inference to Contact Selection". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*.