Joint Deformation and Contact Reasoning for Robotic Manipulation

by

Mark Van der Merwe

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2026

Doctoral Committee:

Associate Professor Dmitry Berenson, Co-Chair
Assistant Professor Nima Fazeli, Co-Chair
Assistant Professor Jeffrey Ichnowski
Associate Professor Ram Vasudevan

Mark Jacob Van der Merwe

markvdm@umich.edu

ORCID iD: 0000-0001-5744-3775

DEDICATION

To my parents, Jacobus and Monique Van der Merwe.

ACKNOWLEDGEMENTS

I owe thanks to many people for their support in completing this thesis. First and foremost, I would like to thank my advisors Nima Fazeli and Dmitry Berenson, for their guidance, patience, and support throughout my time at Michigan. I would also like to thank my committee Ram Vasudevan and Jeffrey Ichnowski for their feedback. I would like to thank Devesh Jha who was my mentor at Mitsubishi Electric Research Laboratories and helped guide work included in this thesis. Additionally, I would like to highlight Tucker Hermans, with whom I worked at the University of Utah during my undergraduate degree, where I first got the opportunity to work on robotics manipulation research.

Thank you also to all the labmates from the Autonomous Robot Manipulation Lab and the Manipulation and Machine Intelligence Lab for their support and collaboration, especially Miquel Oller, Andrea Sipos, and Youngsun Wi, with whom I started at Michigan and have had the pleasure to work alongside these last few years.

Finally, thank you to all my friends and family for their unending support and encouragement, without which this dissertation would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Central to the success of many dexterous manipulation tasks is *contact*. Humans can intuitively reason about and control contacts to cook, clean, push, build and complete many more manipulation tasks. Our goal in designing autonomous robotic manipulators is ultimately to design robotic systems capable of ubiquitous reasoning and control of contact. In this thesis, we focus on contact-rich manipulation on systems exhibiting *elastic deformation,* either in the manipulator or the objects being manipulated. This compliance yields exciting opportunities, such as robust compliant contact interfaces, gradual force buildup and dissipation, and the potential for observability of contact via the deformation. Even with compliance in the system, contacts are rarely directly observable, forcing reliance on indirect, local, and noisy sensing. Additionally, the compliance in conjunction with contact introduces high-dimensional states and complex dynamics.

In this thesis, we explore methodologies to provide robots a sense of contact. We investigate data-driven methodologies for overcoming the complex, partially observable nature of contact. In particular, we look to exploit the tight relationship between deformation and contact, as elucidated by multi-modal sensory feedback. With our sense of contact in hand, we then explore downstream reasoning and behaviors in a variety of manipulation settings, such as tool use, non-prehensile and prehensile manipulation, and multi-task visuomotor policies.

CHAPTER I

Introduction

The north star for robotic manipulation is the design of autonomous systems capable of achieving a wide range of complex dexterous tasks. For instance, in-home robots should be capable of washing dishes or cooking meals while industrial robots must be able to repeat assemblies with high precision, robustness, and efficiency while working safely around humans. At the center of many dexterous manipulation tasks is *contact*, and in particular the ability to *perceive and intentionally make contacts.* When cleaning a dish with a sponge, the contact the sponge makes with the plate ultimately determines whether the food is removed or remains. When using a screwdriver to insert a screw, where precise alignment and force transfer is needed to successful tighten, the contacts between our hands and the tool can inform where the tool is, while the contact between the tool and the screw determines if the tool successfully mates to the screw.

In this thesis, we focus on contact-rich manipulation on systems with *elastic deformation*, either in the tools used by the robot or in the robot/end-effector itself. Elastic deformation is a temporary, reversible change in the shape of an object upon application of a force, where the shape returns to its original form when the force is removed. The robotics community has seen a rise in interest for this compliance, as it contains several important and exciting opportunities. First, deformation allows the formation of stable contact interfaces, as the tool or robot can comply to the contacting surface. As such, elastically deformable tools are regularly found in human

environments, such as sponges, spatulas, and brushes. Second, deformation enables gradual force buildup and dissipation during contact events, resulting in more sensitive systems. This is exploited by soft robots that need to avoid high-forces, such as during human robot interactions. Finally, deformations enable observability of contact, as the contact necessarily causes deformation. This is exemplified by the recent explosion of compliant, visuo-tactile sensors to observe contacts.

Even with compliance present in the system, reasoning robustly about contact presents several challenges. *Extrinsic contacts*, contacts occurring between the tool and environment, are rarely directly observed, due to occlusions caused by the contacting surfaces. This forces reliance on indirect sensing, such as external visual or non-collocated tactile sensing. *Intrinsic contacts*, sensed contacts between our robot and the tool, are partially elucidated by the inclusion of tactile sensors at the grasp, but still yield only noisy and very local observations to the system at large. The introduction of compliance further complicates modeling, introducing high-dimensional states and complex dynamics.

This thesis introduces methodologies to give robots an explicit sense of contact. We investigate creating explicit representations of contact during complex, compliant manipulations, in order to directly decode *if, where,* and *how* contact is being made. To overcome the partially observable, complex, and high-dimensional nature of compliant contact, we utilize data-driven techniques to infer our representations from multi-modal sensory feedback, exploiting the tight relationship between deformation and contact. We then explore how these contact representations enable downstream reasoning and behaviors.

In Chapters III and IV, we focus on building representations for tasks where we want direct control over extrinsic contacts when using a deforming tool, such as during scraping, sweeping, and scrubbing tasks. We show how we can learn representations of the contact geometry from multi-modal sensing, and employ Model Predictive Control (MPC) to servo contact as desired.

In Chapters V and VI, we consider how contact representations can unlock downstream reasoning and behaviors for manipulating an object. In Chapter V, we investigate extrinsic dexterity, being able to deftly control an external object via a

deformable robot end-effector in a non-prehensile manner. We design a learnable contact representation to capture the complex compliant contact between tool and object, which enables us to exploit physical priors to reason about object-environment contacts. In Chapter VI, we shift to prehensile manipulation with compliant tactile sensors. We design contact representations at the tactile sensor to capture in-hand geometries and forces that unlock downstream model-based estimation of object poses and extrinsic contacts for the grasped object.

Finally, in Chapter VII, we consider large behavior models, visuomotor policies trained on multi-task behavior datasets. While these demonstrate impressive performance across a wide range of language-specified tasks, they have primarily focused on tasks driven by spatial and semantic reasoning. In our final chapter, we investigate how we can use in-hand tactile information from compliant visuo-tactile sensors to imbue large behavior models with an informative sense of contact.

In summary, this thesis makes the following contributions:

- We propose a line-based contact representation between a deforming tool and its environment and an architecture to identify and model the evolution of contact based on observed deformations in a noisy point cloud, as well as Force/Torque (F/T) sensing.

- We propose Neural Deforming Contact Fields (NDCF), a neural implicit representation of a deforming geometry and the contact patch it makes with its environment. This joint representation is learned from point cloud and F/T sensing.

- We propose a joint learning and model-based approach for recovering the contacts and forces between a deformable tool, a free-moving rigid body, and the environment. This method incorporates learned deformable modeling with quasi-static rigid body reasoning.

- We propose TacGraph, a factor graph based estimator that utilizes the deformation of a visuo-tactile sensor, along with the physical constraints of rigid contact, to estimate the in-hand pose and extrinsic contacts of a grasped object.

- We propose TaVLA, a method for bridging tactile reasoning and large-scale policy pre-training (i.e., Vision-Language-Action (VLA) models). Our proposed method directly relates force reasoning into the visual domain by annotating visual feedback with tactile-derived signals.

CHAPTER II

Related Work

In this chapter, we briefly review the literature relevant to this dissertation. We start with a discussion of existing contact representation and estimation techniques and touch on methods for modeling of deformable bodies. Finally, we provide brief overviews of relevant literature for the downstream tasks and systems we investigate in Chapters III, V, VI, and VII.

2.1 Contact Estimation and Representation

2.1.1 Extrinsic Contact Estimation

We define *extrinsic contact* as any contact occurring on a non-sensored surface, such that any relevant sensing is non-collocated and indirect. Examples include contact between a grasped or attached tool and the environment or on a non-sensored part of a robot or end-effector.

Many previous works have explored mechanisms for identifying extrinsic contact. Some early work focused primarily on detecting if *any* contact or collision occurs to a robotic arm [109]. By monitoring motor torques for external torques (not due to dynamics or commanded motions), contacts can be detected and used to quickly stop robots working in human environments. Moving from binary contact to a particular contact estimate can be challenging due to the lack of sensing on the robot surface and noisy torque estimates. Saund and Berenson [98] introduced the Contact Hypothesis Set (CHS), utilizing the set of robot surface points at the time of collision as

candidate contact points, as at least one must have caused the contact. This coarse representation can be useful for motion planning in uncertain environments [99], but is ill-suited to precise contact estimation and control.

Manuelli and Tedrake [77] utilized external torque measurements to localize a *contact point* on a robot by utilizing a particle filter to search the surface of the robot for a point and force consistent with the observed external torques. Sipos and Fazeli [105, 106] extend this to consider a grasped object, while jointly estimating the object pose with a complementary particle filter, utilizing F/T sensing to guide the search. Ma et al. [73], Kim and Rodriguez [58], and Kim et al. [57] utilize distributed tactile sensing to recover *contact point* and *contact line* estimates on a grasped object, using estimated-in hand motions and constrained optimization assuming sticking external contacts. Accurate estimation relies on actively taking exploratory actions [58, 57] or pairing with in-hand pose estimation [10].

More recent work has turned to data-driven methods of recovering extrinsic contact, seeking to directly output contact estimates. Several works have targeted the case where an object is grasped and contacting the environment. Kim et al. [55] learns to predict *image contact masks* directly from visual feedback. While richer than point or line representations, image space labels can lead to ambiguities during the image projection. Higuera et al. [41] and Ota et al. [89] predict extrinsic *contact patches* on object surfaces based on visuo-tactile feedback from the object grasp. Lee and Fazeli [63] estimates an extrinsic *contact patch* by first estimating in-hand pose, then predicting the patch.

Crucially, the majority of existing work has focused on extrinsic contact estimation for *rigid* bodies. In Chapters III, IV, and V, we propose methodologies for addressing *deformable* bodies, where we must consider the deformation of the surface when estimating contact.

2.1.2 Intrinsic Contact Estimation

In this thesis, we define *intrinsic contact* as any contact occurring on a sensored surface, where the sensing is collocated with the contact event. These tactile sensors

take several forms, often prioritizing different elements of the local contact. Small localized Force/Torque (F/T) sensors provide composite force information close to contact events [18]. However, the composite nature provides limited geometric and extended force information. Distributed sensors, such as ones utilizing gridded taxels to provide 3-axis forces at multiple locations, provide more spatial feedback while also capturing relative force changes across the surface [115]. Visual-tactile sensors utilize a deforming membrane viewed by a camera to provide even more dense tactile feedback. Deformations are captured via photometric stereo [136, 26, 113] as well as printed markers and patterns [2] viewed by the internally mounted cameras. While providing the most dense and rich feedback, the raw sensory feedback is simply images, requiring downstream processing to recover contact details. We next discuss methodologies for recovering contact geometries and patches as well as force information from visuo-tactile sensing.

While one can utilize photometric stereo to recover geometries [136, 113], data-driven techniques to directly predict contact patch geometries between the sensor and objects have become increasingly popular as they can capture unmodeled effects and noise in the images [104]. This is especially useful for contact estimation on highly compliant sensors, which can exhibit deformation at out of contact locations [28].

Force and torque can be recovered from visuo-tactile feedback via tracking of markers and modeling via linear [136] or Finite Element Method modeling [74]. Similar to geometric reason, these model-based estimators are increasingly replaced by data-driven models, which can be used to predict composite in-hand wrenches from a grasp [57].

We build on these intrinsic contact methodologies in Chapters VI and VII, where we propose new downstream algorithms utilizing geometric and force instrinsic contact representations for driving manipulation behaviors.

2.2 Deformable Object Modeling

Due to the prevalence of deformable objects in human environments, ranging from medical to service industries to manufacturing, there has been significant re-

search effort in the modeling of deforming bodies [132]. Several approaches, such as Mass-Spring Systems and Position-Based Dynamics, seek to approximate deforming bodies by treating the deforming volume as a set of points related via constitutive laws, such as spring-damper relationships [84]. However, these methods rely on non-physical parameters and struggle with realistic behaviors. The Finite Element Method (FEM) seeks to solve a more physically accurate continuum mechanics problem. FEM discretizes a continuum body into small elements and approximates the stress-strain relationship to capture material behaviors and internal stresses under external forces and contact [39]. The result is a more physically accurate model, at the cost of computational complexity. High computational cost can make online modeling difficult, thus some recent research has focused on utilizing data-driven techniques to capture deformable volumes [122] and dynamics [124, 70]. Masterjohn et al. [79] forgoes explicit modeling of deformations, utilizing soft-penetration based constraints to approximate force transfer during deformation. In Chapter III, we forgo any explicit modeling of deformation to simplify learning, while in Chapters IV and V we utilize data generated from FEM to train data-driven techniques bridging compliance modeling with raw sensory feedback.

Using deformation as an indicator for contact and force information has also been demonstrated in the space of continuum robots [31, 119]. This thesis builds on the same coupled deformation and contact relationship, but extends to volumetric tools and tactile sensing.

2.3 Contact Servoing

Contact Servoing is the task of explicitly tracking a set of desired contacts with a manipulation system. Several works have investigated tracking desired *intrinsic* contact locations and geometries on tactile sensors. Li et al. [68] use a large tactile pad and define contact configuration features of objects pressed against the sensor. They manually construct a feedback controller based on these features and use it to drive contacts to desired configurations. Sutanto et al. [111] use a smaller profile tactile sensor and learn the dynamics of a learned latent space. They then employ a

Model Predictive Control (MPC) scheme to drive contacts to desired configurations on the sensor.

Tracking desired extrinsic contact introduces further complexity, due to the indirect nature of feedback and control. Several works have investigated the problem of continuous force-control to keep a tool in contact with a surface at a desired force profile. This can be achieved via hybrid force and motion control, decoupling dimensions along which motion should occur from those along which force transfer should be retained [50]. Sakaino [97] instead uses imitation learning to learn a controller able to maintain contact between a mop and a tabletop.

In Chapter III, we build on existing literature by demonstrating extrinsic contact servoing for a *deforming* object. Unlike existing extrinsic contact methods that focus on maintaining contact, we further consider realizing desired contact geometries, exploiting the deformability of the tool.

2.4 Extrinsic Dexterity

Extrinsic dexterity involves controlling a freely moving object external to the robot via contacts the robot makes with the object. Enabling extrinsic dexterity has long been a subject of research, from pushing [134] and pivoting [46] primitives to the chaining of multiple manipulating primitives [44]. Existing work on extrinsic dexterity largely assumes rigid-to-rigid interactions [46, 102, 87] and/or assumes sensing at the contact interface [44, 110]. In Chapter V, we consider deformable-to-rigid contact, where the robot directly controls the deformable tool and indirectly manipulates the rigid body.

Several works forgo model-based reasoning in favor of learning how extrinsic bodies move from data. To predict the motion of a rigid body, existing work has investigated directly predicting SE3 transforms of objects [11, 12] or predicting point cloud or mesh vertex motion [101, 1]. Rigid body contact can yield discontinuities that are challenging for learned models, which generally favor smooth approximations. As such, Pfrommer et al. [92] propose to learn rigid body motion by parameterizing inter-body signed distances and contact Jacobians, enabling analytical physical

simulation that can reflect rigid contacts. Other work seeks to directly predict the motion of heterogenous materials by applying Graph Neural Networks [70]. In Chapter V, we fuse data-driven learning to capture object motion and deformable-rigid contact while building on quasi-static modeling techniques for handling the rigid object contacts with the environment [87].

2.5 Prehensile Manipulation

Prehensile manipulation is any manipulation that involves *grasping* an object. Unlike extrinsic dexterity tasks explored previously, grasping often enables more direct control over object motions, as well as opportunities for rich, intrinsic contact feedback via tactile sensors. As highlighted previous, several works have investigated intrinsic and extrinsic contact estimation during prehensile manipulation [55, 58, 57, 10, 41, 89, 105, 106]. In this section, we briefly review prehensile pose estimation, an important subtask for prehensile manipulation systems.

Several works investigate model-based pose estimation with combined visual and tactile feedback. Some works jointly reconstruct object geometries and estimate pose from visual and tactile geometric feedback [85, 110]. Zhong et al. [141] extends model-based pose estimation to include free-space non-penetration constraints as well as tactile point clouds, but does not consider contact consistency of any form.

Tac2Pose [4] learns to perform object pose estimation via an object-specific tactile model that returns pose distributions consistent with observed tactile feedback. This can then be extended to incorporate visual feedback to further resolve ambiguity [3]. Dikhale et al. [25] fuse tactile and visual feedback to learn to predict object poses directly, akin to purely visual based tracking models [116].

Due to the local nature of tactile sensing and occlusions during interactions, estimating in-hand object pose can yield ambiguous and uncertain results in some scenarios. To further constrain the search space, several methods investigate jointly estimating pose with extrinsic contact. Bronars et al. [10] extends Tac2Pose [4] to include additional geometric contact constraints with a known environment. SCOPE [105] and Multi-Scope [106] propose a model-based approach that utilizes F/T feedback on

the robot and environment to simultaneously estimate object and environment pose and extrinsic contacts, utilizing the physical constraints of contact. In Chapter VI, we propose a method for joint pose and contact estimation. We build on related work but do not rely on expensive object-specific training [10] or external wrench sensing [105].

2.6 Vision-Language-Action Models

While the previous sections reviewed specific down-stream tasks for which we investigate contact representation learning in this thesis, some recent robotics research work has turned from designing solutions for single tasks to focusing on broad, diverse, multi-task behaviors. In turn, methods have evolved from specific solutions which can exploit problem structures to methods emphasizing generality. However, the need for contact reasoning remains. In this final literature review, we briefly review recent works presenting Vision-Language-Action (VLA) models, as well as early efforts to embue these models with contact reasoning.

Recent efforts in the domain of robotic imitation learning have demonstrated impressive performance in utilizing transformer [140] and diffusion [17] architectures to learn reactive vision-driven policies, where the model is trained to recreate the human labeled behaviors. Utilizing large-scale demonstration datasets [54, 90], several works build Vision-Language-Action (VLA) models [35, 9] - language-specified, multi-task, visuomotor policies. To further push towards open-world generalization, modern VLA models often utilize a pre-trained Vision-Language Model (VLM) as the backbone [5, 6, 144, 49, 56, 90]. VLMs are trained on internet scale image-language corpora, and have demonstrated incredible generalization at text and image reasoning [131]. VLA models trained with VLM backbones can exploit the massive pre-training to retain language and visual comprehension.

Given the importance of contact and tactile feedback during manipulation, an important open research question is how best to incorporate this feedback into VLA models? Crucially, tactile datasets are still comparatively small [129, 33, 34] making large-scale pre-training for tactile data challenging [43]. Additionally, the variety

in raw sensory signals makes for difficult cross-embodiment reasoning [96], as even sensors in the same class can yield largely different data distributions.

Several recent works have proposed initial approaches for incorporating force and tactile sensing into VLAs. Hao et al. [40] trains a VLA from scratch with vision-based tactile data *replacing* vision to form a Tactile-Language-Action model while Zhang et al. [137] adds the vision back in to form a Vision-Tactile-Language-Action model. Both methods utilize the Qwen2-VL vision-language model [120] as the pre-trained backbone. Training from scratch however forgoes any policy pre-training available from existing VLA models and datasets, requiring large-scale tactile data collection. Several other methods look to start from an existing VLA and extend it to tactile inputs. Starting from pretrained VLAs such as π_0 [6] and Octo [35], new encoders are added to capture force [133], joint torque [138], taxel [48], and vision-based [16, 51] tactile feedback, fine-tuning on new demonstrations collected with the corresponding modalities available. To try to improve cross-modal reasoning, several methods further investigate adding auxiliary losses before or during imitation learning. Jones et al. [51] finetunes the Octo [35] VLA using both vision-based tactile and audio inputs, adding cross-modal contrastive loss as well as an auxiliary cross-modal language generation task. Cheng et al. [16] finetunes π_0 [6], adding cross-modal contrastive losses between the modalities. Zhang et al. [138] finetunes π_0 [6], adding a torque prediction auxiliary task. These methods all rely on architectural changes and result in data distribution shifts from the original policy pre-training which may inhibit effective policy transfer. In Chapter VII, we propose an alternative contact representation using *image annotations* to embue the VLA with relavant tactile and contact feedback, while limiting data distribution shift and requiring no architectural changes.

CHAPTER III

Learning the Dynamics of Compliant Tool-Environment Interaction for Visuo-Tactile Contact Servoing

This chapter addresses the task of extrinsic contact servoing between a deforming tool and its environment. We propose a summary contact representation consisting of a binary contact label, a contact line, and F/T feedback at the wrist. Utilizing point cloud feedback to observe the tool deformation, along with F/T sensing, we learn to predict the corresponding contact summary. A dynamics model over the contact representation is learned and the resulting dynamics are utilized via model predictive control to servo desired contacts. Our results indicate that accounting for deformation is crucial to precise extrinsic contact control.

3.1 Introduction

Many manipulation tasks require the robot to control the contact between a grasped tool and the environment. The ability to reason over and control this *extrinsic* contact is crucial to enabling helpful robots that can scrape a frying pan with a spatula, erase or wipe a surface [78], screw a bottle cap onto a bottle [65], perform peg-in-hole assemblies [64, 58], and perform many other tasks.

In this chapter, we seek to address the problem of controlling the extrinsic contact between a grasped *compliant* tool (e.g. a spatula) and the environment. In general,

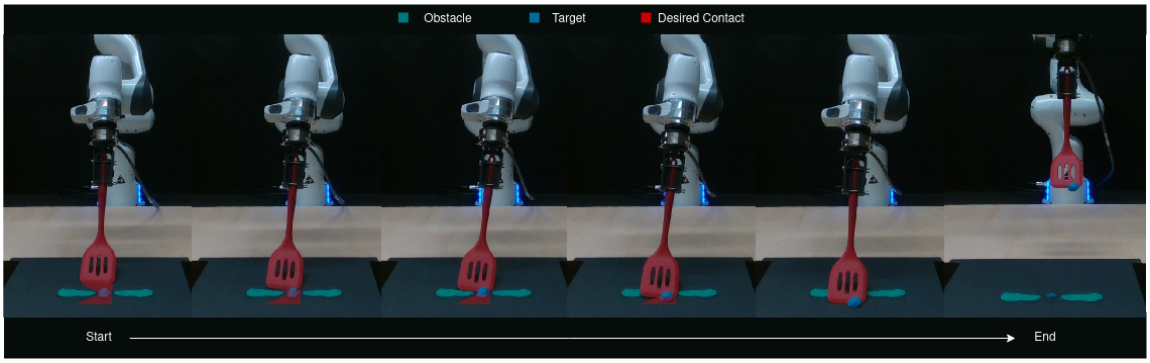


Figure 3.1: We present a method for *extrinsic contact servoing*, i.e., controlling contact between a compliant tool and the environment. Our method is able to complete the requested contact trajectory, avoiding contact with surface obstacles, and successfully scrape the target object. Note that to do this the spatula must be tilted so that only a corner of it is in contact.

the robot cannot expect to have the full geometry and physical properties (e.g., mass, friction, stiffness) of all the tools it must use or the geometries of the environments it must manipulate in. Instead, the robot must utilize multimodal sensory observations, such as pointclouds and tactile feedback, to act on the environment.

In recent years, learning-based methods have become increasingly popular to address the complexities of robotic manipulation, including for contact-rich tasks [61]. These methods can be loosely grouped into model-free methods, that directly learn a policy [64, 65, 66], and model-based methods, that learn system dynamics [128, 81, 38]. By focusing on modeling system dynamics, model-based methods can plan to reach new goals without retraining, and are often more data-efficient [38]. Therefore, we propose learning the dynamics of our system to solve the extrinsic contact servoing task.

It is not obvious which representation to use for these dynamics. Fully recovering tool and environment geometries from visual data [80, 19] and tactile feedback [121] has been widely explored, with recent extensions to compliant geometries [122]; however, even if the system can be fully identified, contact models to resolve interactions can have limited fidelity [30]. On the other hand, learned dynamics representations can be difficult to interpret and require demonstrations or observations from the

desired state to specify goals [128, 76]. Instead, we propose a novel *contact feature representation* for our learning method that focuses on tool-environment interaction and bypasses explicitly modeling the whole system. We represent the contact configuration as 1) a **binary contact mode** (indicating if the system is in contact); 2) a **contact geometry** (as a line in 3D space); and 3) an **end-effector wrench**.

We propose a learning architecture to model the dynamics of the proposed contact representation from raw sensory observations over candidate action trajectories. We propose structuring the model as a latent space dynamics model with a decoder that recovers the contact state. We also propose an action offset term in the dynamics that allows us to accurately propagate robot poses, despite controller errors (e.g. from robot impedance). To provide labels to our model, we collect self-supervised data on a 7DoF Franka Emika Panda, using sensor data to automatically label contact state.

We validate our proposed method by completing various desired contact trajectories on the real robot system. We first show that our method can track diverse desired contact trajectories in the absence of obstacles. Next, we demonstrate that we can utilize extrinsic contact servoing to scrape a target object from the table, while handling occlusions and avoiding contact with obstacles (Fig. 3.1).

3.2 Problem Formulation

We parameterize our contact feature as a binary contact indicator $c^b \in \{0, 1\}$, used to indicate whether the tool is in contact, a contact line $\boldsymbol{c}^l \in \mathbb{R}^{2\times 3}$ representing the contact geometry between the tool and the environment, and an end effector wrench $\boldsymbol{c}^w \in \mathbb{R}^6$. The geometry $\boldsymbol{c}^l$ is only active when the tool is in contact $c^b = 1$. The contact representation allows extrinsic contact goals to be expressed as *desired contact trajectories* $G = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_L]$, where each $\boldsymbol{g}_i \in \mathbb{R}^{2\times 3}$ is a desired contact line to reach. We assume that contact should be maintained throughout the task.

We formulate extrinsic contact servoing as a model predictive planning problem, given observations of the current state of the system $\boldsymbol{o}_0$. For a given horizon T, we select the next T desired contact lines $[\boldsymbol{g}_{i+1}, \ldots, \boldsymbol{g}_{i+T}] \subseteq G$ to be our current contact

goal sequence. The planning problem is:

$$\begin{aligned} \min_{\boldsymbol{a}_{0:T-1}} \; & \sum_{t=1}^{T} d(\boldsymbol{c}_t^l, \boldsymbol{g}_{i+t}) \\ \text{s.t. } & c_t^b = 1, \forall t \in [1, T] \\ & \{c_{0:T}^b, \boldsymbol{c}_{0:T}^l, \boldsymbol{c}_{0:T}^w\} = g(\boldsymbol{o}_0, \boldsymbol{a}_{0:T-1}) \end{aligned} \tag{3.1}$$

Here g is a model describing the *contact feature dynamics*. The binary constraint ensures that the tool remains in contact while the cost function d measures the distance between the two contact lines, as the average Euclidean distance between the line endpoints. Finally, if $d(\boldsymbol{c}_1^l, \boldsymbol{g}_{i+1}) < \epsilon$ we increment i, thus moving to the next sequence of desired contact lines for the next round of planning.

3.3 Method

3.3.1 Contact Feature Dynamics Model

To solve our constrained optimization Eq. 3.1, we require a model g which can map from raw observations $\boldsymbol{o}_0$ and a proposed action trajectory $\boldsymbol{a}_{0:T-1}$ to the resulting contact states $\{c_{0:T}^b, \boldsymbol{c}_{0:T}^l, \boldsymbol{c}_{0:T}^w\}$. We propose modeling the contact feature dynamics as a deep neural network. Our actions are changes in end effector pose.

We assume access to a pointcloud $\boldsymbol{v}_0$ and input wrench $\boldsymbol{h}_0$ measured at the robot's wrist as our observations, $\boldsymbol{o}_0 = (\boldsymbol{v}_0, \boldsymbol{h}_0)$. Note that end effector wrench is both an input to our method and part of the contact state; predicting future wrench aids the representation learning and provides expected wrenches for planning.

We perform all learning in the local end effector frame. We transform the pointcloud to the end effector frame ${}^{EE_0}\boldsymbol{v}_0$ and clip to a $0.5m^3$ bounding box region around the end effector that contains the contact event. We similarly predict our contact lines in the current end effector frame, ${}^{EE_t}\boldsymbol{c}_t^l, \forall t \in [1, T]$. Learning in the end effector frame provides invariance in the visual domain to translations and rotations of the end effector and removes distractors that do not contribute to the contact state, such as the rest of the robot arm or the scene background.

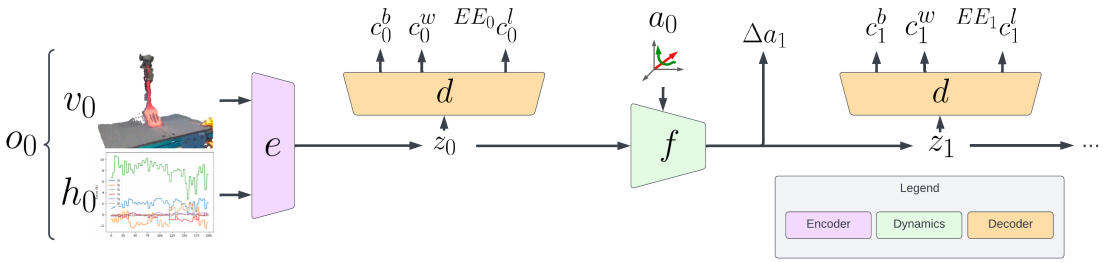


Figure 3.2: Our proposed contact feature dynamics model. Our architecture embeds raw observations into a latent space where dynamics can be unrolled. We then decode the contact state from the latent space. We also predict an action offset term in order to accurately predict future robot poses.

Our contact feature dynamics model (Figure 3.2) has three components: an encoder e which maps from raw observations to a learned latent space, a decoder d which maps from the latent space to the contact state, and a dynamics model f which captures dynamics in the latent space. We parameterize the models by a set of learned weights θ.

We start by embedding the current observations into the latent space with our encoder $\hat{\boldsymbol{z}}_0 = e(\boldsymbol{v}_0, \boldsymbol{h}_0)$. We unroll actions in the latent space as the contact state alone has insufficient contextual information (e.g. end-effector pose and local geometry information) to predict the next contact state. Because we predict the tth contact state in the current end effector frame EE_t, an important consideration when designing our dynamics model is being able to accurately recover this frame. Controller error, e.g., from the impedance of the robot, means the commanded action is not perfectly executed. To account for this, we predict an additional term from our dynamics model $\Delta\hat{\boldsymbol{a}}_{t+1}$, which is an $SE(3)$ transformation that predicts the offset between the commanded and realized next end effector pose. Thus, our dynamics model predicts, $\hat{\boldsymbol{z}}_{t+1}, \Delta\hat{\boldsymbol{a}}_{t+1} = f(\hat{\boldsymbol{z}}_t, \boldsymbol{a}_t)$. This allows us to construct the following recursive estimate of our end effector frame ${}^W\hat{T}_{EE_{t+1}} = {}^W\hat{T}_{EE_t} T(\boldsymbol{a}_t) T(\Delta\hat{\boldsymbol{a}}_{t+1})$, where ${}^W\hat{T}_{EE_t}$ is the $SE(3)$ transformation describing the pose of the end effector at time t. We know the initial transform ${}^W\hat{T}_{EE_0}$ from our robot proprioception, $T(\boldsymbol{a}_t)$ provides the transformation for the action command, and $T(\Delta\hat{\boldsymbol{a}}_{t+1})$ for the predicted

offset term. To enforce valid $SE(3)$ predictions we predict rotations in the axis-angle representation [11].

Finally, we recover our contact state estimates with our decoder d given the latent state $\hat{\boldsymbol{z}}_t$: $\hat{c}_t^b, {}^{EE_t}\hat{\boldsymbol{c}}_t^l, \hat{\boldsymbol{c}}_t^w = d(\hat{\boldsymbol{z}}_t)$. We can then recover the predicted contact line in the world frame by composing with the estimate of the end effector frame transformation ${}^W\hat{T}_{EE_t}$. An overview of the model architectures is shown in Fig. 3.2 and full architecture details can be found in Appendix A.1.

3.3.1.1 Training Loss

We train our model on rollouts of the system, where a single example is a sequence $[\boldsymbol{v}_t, \boldsymbol{h}_t, \boldsymbol{a}_t, \Delta\boldsymbol{a}_t, \boldsymbol{c}_t^l, \boldsymbol{c}_t^w, c_t^b]_{t=0}^T$. We define the loss over the example as:

$$\begin{aligned}\mathcal{L}_\theta =&(\sum_{t=0}^{T} BCE(\hat{c}_t^b, c_t^b) + \alpha \cdot c_t^b \cdot MSE(\hat{\boldsymbol{c}}_t^l, \boldsymbol{c}_t^l) + \beta \cdot MSE(\hat{\boldsymbol{c}}_t^w, \boldsymbol{c}_t^w)) \\ &+ (\sum_{t=1}^{T} \rho \cdot MSE(\Delta\hat{\boldsymbol{a}}_t, \Delta\boldsymbol{a}_t) + \gamma \cdot MSE(\hat{\boldsymbol{z}}_t, e(\boldsymbol{v}_t, \boldsymbol{h}_t)))\end{aligned} \tag{3.2}$$

Here BCE is the Binary Cross Entropy classification loss and MSE is the Mean Square Error regression loss. α, β, ρ and γ are loss weighting terms. The first four loss terms are prediction losses over the contact mode, contact geometry, end effector wrench, and action offset transformation. The final loss term is a latent consistency loss, which encourages latent rollouts to match the latent state yielded by encoding future observations.

3.3.2 Extrinsic Contact Servoing Controller

We propose to solve our planning problem using Model Predictive Path Integral (MPPI), which has been shown to be effective for continuous control tasks where sampling is cheap and parallelizable (e.g., neural network representations) [125]. We convert our binary constraint to a penalty, penalizing a trajectory if it yields actions that lead out of contact. Pairing this with the contact line prediction loss yields the

following final cost function:

$$\sum_{t=1}^{T} d({}^{W}\hat{\boldsymbol{c}}_t^l, \boldsymbol{g}_{i+t}) + \phi \cdot \begin{cases} |\hat{c}_t^b - \psi| & \text{if } \hat{c}_t^b < \psi \\ 0 & \text{o.w.} \end{cases} \tag{3.3}$$

The constraint is violated if the likelihood of binary contact is below the classification threshold ψ, in which case we penalize by the distance to the threshold. ϕ weights the penalty against the contact line loss. With this cost function we apply MPPI to yield the next action and execute it on the robot.

3.3.3 Extrinsic Contact Dynamics Labeling

Our contact dynamics training loss in Eq. 3.2 requires ground truth contact state labels $(\boldsymbol{c}_t^l, c_t^b, \boldsymbol{c}_t^w)$ at time t. As accurate simulation of contactful interactions is challenging, we propose a method of data acquisition directly in the real world. To generate contact line labels, we use a high resolution, low frequency scanner, a Photoneo PhoXi 3D Scanner, to generate high quality scans of the contact interaction. Using these scans, we generate contact line labels by filtering points just above the table, clipped to the area around the end effector. We then cluster these points to remove noisy points on the tabletop and generate the contact line $\boldsymbol{c}_t^l$ by selecting the two furthest points in the cluster. See Appendix A.2 for examples of contact labels. We use a force torque sensor to identify the contact state wrench $\boldsymbol{c}_t^w$. We identify binary contact c_t^b automatically from whether a line was found in the pointcloud and/or based on force torque sensing.

3.4 Experiments

3.4.1 Experimental Setup

We test our method on a Franka Emika Panda with rigidly-mounted compliant tool at the end effector (Fig. 3.1). In Sections 3.4.3 and 3.4.4, we use a deformable spatula. In Section 3.4.5 we use a compliant brush. For our observations $\boldsymbol{o}_0$, we

use pointclouds $\boldsymbol{v}_0$ from an Intel Realsense D435 sensor[1] and mount an ATI Gamma Force/Torque sensor between the end effector and tool. We use the last four wrench values received after the previous action completed as the tactile input $\boldsymbol{h}_0$. To collect our datasets, we use a random action policy with a heuristic to encourage contact between the tool and the environment. No other objects are on the tabletop during data collection to allow proper data supervision, as detailed in Sec. 3.3.3.

3.4.2 Baseline

We compare our proposed method to modeling the contact dynamics as a rigid system. We assume the commanded actions are perfectly executed by the robot to recover the future poses of the end effector. We assume access to the tool geometry as a pointcloud in the end effector frame. This pointcloud is then transformed via the future poses of the end effector to recover where the tool would be, assuming rigid motions. We further assume that we know the table location and identify any points in the transformed point cloud that penetrate the table surface. If any exists, we set $c^b = 1$. We then choose the two furthest points in the intersecting set of points as the end points of the contact line $\boldsymbol{c}^l$. The baseline does not predict the wrench $\boldsymbol{c}^w$, but is enough to solve our planning problem in Eq. 3.1. This baseline makes three assumptions our method does not make: 1) it assumes access to a pointcloud of each tool, 2) it assumes knowledge of the current tool being used, 3) it assumes explicit knowledge of the environment.

3.4.3 Modeling Contact Feature Dynamics

We first investigate the ability of our model to capture the contact feature dynamics exhibited in our dataset. We train three variations on our model. First is the full model, as described in Sec. 3.3.1, hereafter called "Full Model." Second, to understand the importance of modeling the action offset of the robot, we ablate the offset action prediction, thus we propagate the end effector frame only with the

[1]We don't use the high-fidelity Photoneo scan as it is a very low-frequency scanner.

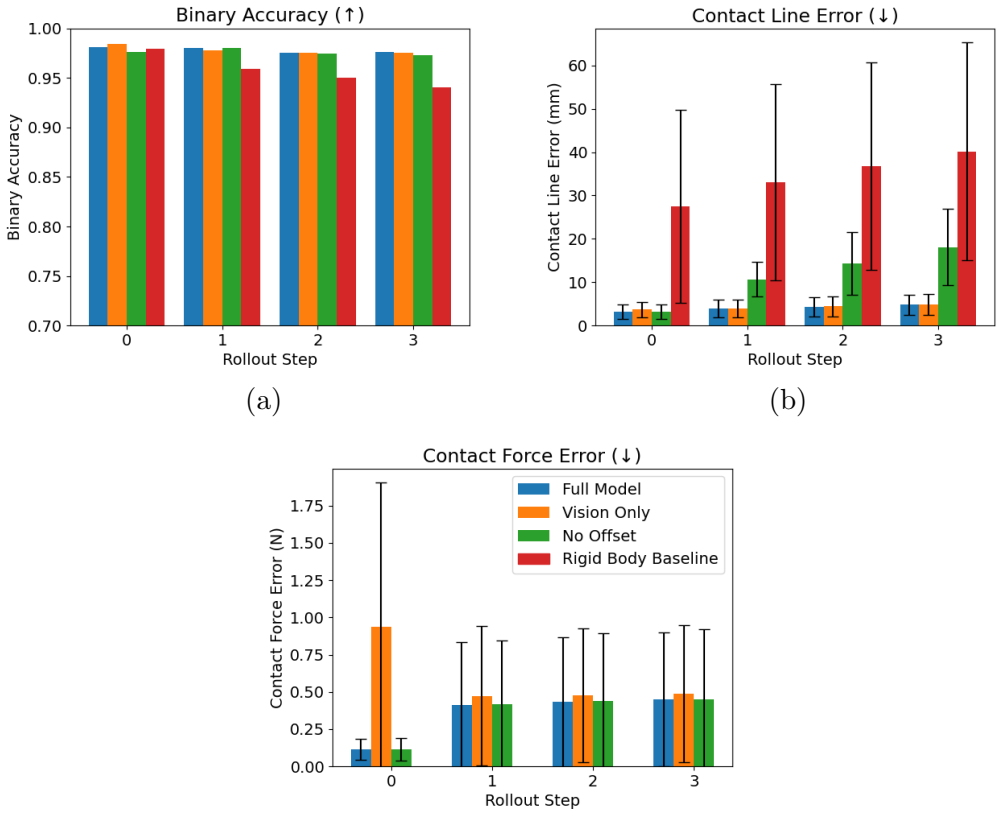


Figure 3.3: Contact Feature Dynamics Performance: our results show the importance of modeling the action offsets and the compliance of the tool; without both, contact line estimates drift. The Rigid Body Baseline does not predict contact wrenches, so is not shown in (c).

commanded action. We call this method "No Offset." Finally, we investigate our model trained only on visual input data, called "Vision-Only."

We train on a dataset collected from three spatulas (see "Training Tools" in Fig. 3.5), collecting 200 trajectories on each, for a total of 30000 transitions. We split the data 80/10/10 for train, validation, and test. We train with a rollout horizon of $T = 3$. All methods are trained with the Adam optimizer [59] until convergence on the validation set. We set $\alpha = 100.0, \beta = \gamma = \rho = 0.1$ in our loss term in Eq. 3.2 to balance the scale of the terms.

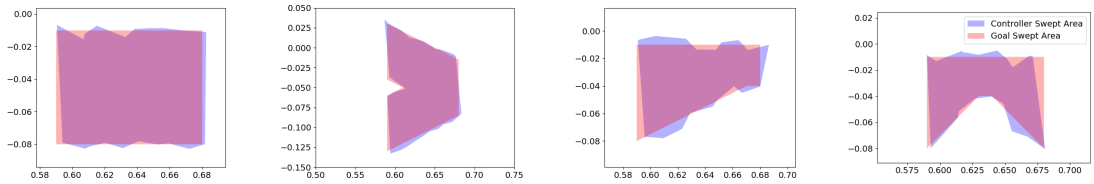
We compare the prediction performance of the models on the test split of the dataset in Fig. 3.3 (see Appendix A.3 for torque prediction results, whose results are very similar to force results). Our full model achieves high accuracy in predicting binary contact ($> 95\%$), 3-5mm contact line error, and less than 0.5N force error. Comparing "Full Model" to "No Offset" shows the importance of modeling action offsets, without which contact line error quickly grows. Comparing all learned models to the Rigid Body Baseline (Sec. 3.4.2) shows the importance of modeling the compliance of the tool. The "Vision-Only" model unsurprisingly struggles to recover contact forces. Appendix A.3 show additional results examining generalization of the model to the dynamics of an unseen spatula.

3.4.4 Extrinsic Contact Servoing - Spatula

Next, we investigate how our proposed controller performs following specified contact trajectories.

Obstacle-Free: We start by attempting to servo along four different contact trajectories in the obstacle-free environment. The desired contact trajectories are shown in Fig. 3.4, and explore translation of contact as well as cases where the robot must tilt the tool to achieve a contact smaller than the width of the tool. We use the same labeling technique introduced in Sec. 3.3.3 to get ground truth contact trajectories executed by the controller. We run the experiment once on a training spatula (left-most in Fig. 3.5) and once on an unseen spatula (right-most in Fig. 3.5).

We use the controller described in Sec. 3.3.2, with $\psi = 0.45, \phi = 0.05$ and compare



(a) Straight (b) V-Shape (c) Tilt (d) Tilt Back

Figure 3.4: Qualitative Scraped Areas: our "Full Model" controller (blue) is able to closely match desired contact trajectories (red).

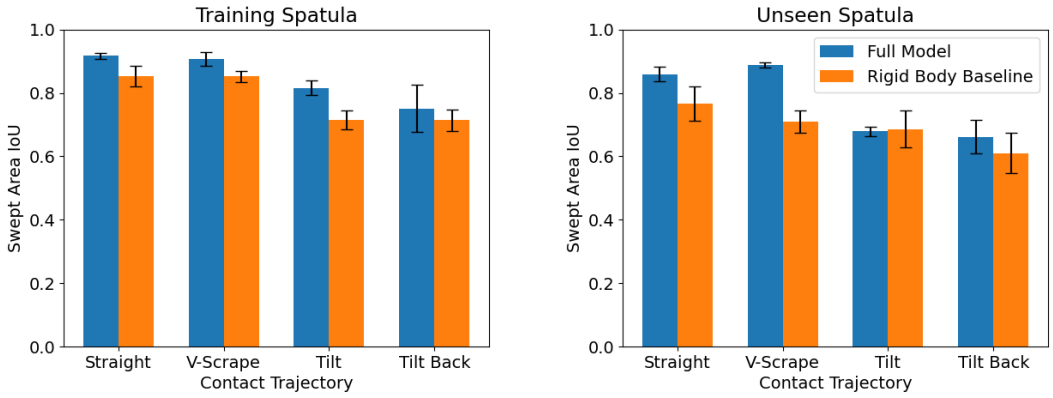
our "Full Model" learned dynamics (as trained in Sec. 3.4.3) vs the "Rigid Body Baseline" dynamics. To investigate the planning performance, we run the controller five times per trajectory and measure the Intersection over Union (IoU) of the desired and swept contact areas. We construct the goal contact area by sweeping the space between the specified goal contact lines and the realized controller swept area by assuming that the space between two consecutive contact states was swept out if the two states were both in contact.

We show qualitative examples of the "Full Model" controller realized scrapes on the training spatula compared to the goal scrapes in Fig. 3.4. We see that the controller is able to closely match the desired swept areas, including in the difficult tilting problems.

The IoU performance is shown for the training spatula (Fig. 3.6a) and unseen spatula (Fig. 3.6b). Our proposed method outperforms the baseline on nearly all cases, for both the training and unseen tool runs. Performance drops for all methods on the tilting problems, as it is more difficult to maintain dexterous contact on only a part of the tool.

With Obstacles: We next examine our method's robustness to visual occlusions and reaction forces arising from contact with a target object to be scraped. This task is common in construction, cooking, and cleaning. A deformable and slightly adhesive material (Playdough) is pressed onto the surface and a contact trajectory is specified through the object. In one case, the target object is alone on the tabletop (Fig. 3.7), and thus we specify a contact trajectory using the full width of the tool.

Figure 3.5: Train/Test tools used for experiments.



(a) (b)

Figure 3.6: Extrinsic contact servoing IoU performance on a training spatula (a) and unseen spatula (b). Our proposed method tracks the desired contacts with higher IoU, compared to a rigid body baseline method, even when running on an unseen tool.

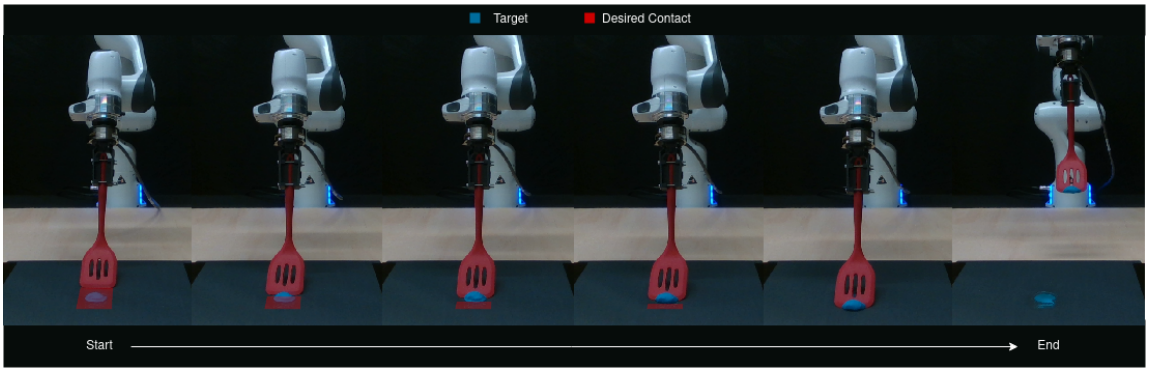
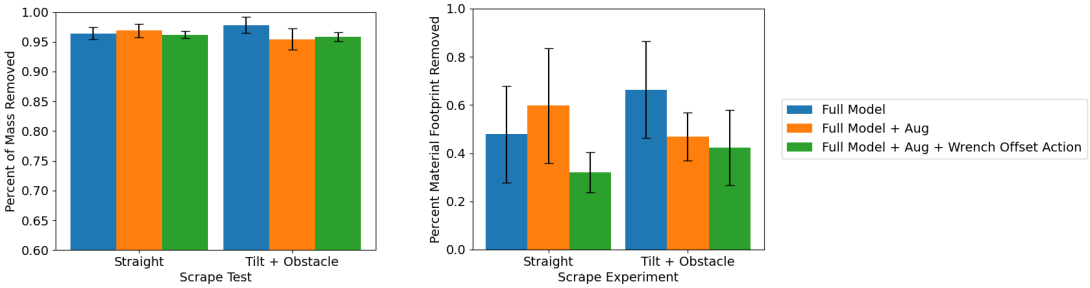


Figure 3.7: Example of extrinsic contact servoing execution for the "Straight" target obstacle scrape experiment. Our method is able to accurately servo along the desired contact trajectory and successfully scrape the target.

In the second scenario, obstacles are on the table near the target object (Fig. 3.1), thus we must specify a contact trajectory that avoids them. All experiments are performed on the leftmost spatula in Fig. 3.5. We use our "Full Model", and train on 22005 sequences collected only with the relevant tool.

Besides running our Full Model in these scenarios, we investigated enhancements of the method to aid its performance in the presence of visual occlusions and object reaction forces. First, we applied data augmentation to our training dataset, randomly generating ellipsoids in the pointcloud and using a hidden point removal algorithm [53] to provide corresponding occlusions in the original point cloud. See Appendix A.2 for examples of augmented inputs. We call this method "Full Model + Aug." Second, we investigate using the difference between the predicted and observed wrenches $\Delta \boldsymbol{w} = \hat{\boldsymbol{c}}_t^w - \boldsymbol{c}_t^w$ to derive an action offset to compensate for the extra wrench experienced by the robot. From Δw we derive an action that will counteract this wrench offset $\hat{\boldsymbol{a}}_t = \frac{\Delta \boldsymbol{w}}{\boldsymbol{k}_p}$. $\boldsymbol{k}_p$ is the pose gain of the impedance controller. The offset action is composed with the original action from the controller. We call this method "Full Model + Aug + Wrench Offset."

We use two metrics. First, we measure the approximate mass of the target object to be scraped before and after scraping and determine the percentage of mass successfully removed. Second, we compare the 2D footprint of the material before and



(a) Percent Mass Removed (b) Percent Footprint Removed

Figure 3.8: Target Scraping Results: Our "Full Model" and variations for addressing visual occlusions and reaction forces arising from contact with the target object perform comparably on both metrics over 5 trials on each experiment.

after scraping and report the percentage of the footprint successfully removed. The second is a more challenging metric, since even a slightly wrong scrape will leave residue. The quantitative results over 5 runs of each method in each experiment setup are shown in Fig. 3.8. Examples of scrape executions are shown in Fig. 3.1 and Fig. 3.7. See Appendix A.3 for more examples of scrape results.

We see that in each case, all methods were able to remove over 95% of material mass on average and about 40-60% of material's footprint from the tabletop. Surprisingly, we don't see a consistent improvement training our model with visual occlusions or adding action offsets. The Full Model's robustness to occlusions here could be due to the fact that we use a single tool in these experiments, and thus it may be sufficient in most cases to capture the location of the table with respect to the end effector in order to estimate the contact line. Even with visual occlusions near the tool contact, it is likely our method can still recover the relative pose of the tabletop from the surrounding points. The lack of clear improvement from the wrench offset action may be due to the fact that it is sufficient to be able to replan, as we do at every step with our MPPI controller.

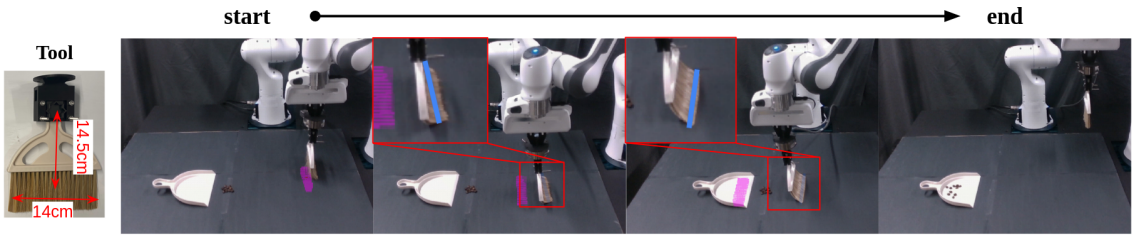


Figure 3.9: Example applying our contact servoing method to a compliant brush sweeping task. The tool is shown on the left. On the right we show a qualitative contact servoing rollout. The predicted contact goals from the contact planner [123] are shown in magenta, while the estimated contact features using our method are shown in blue. We see that our proposed method can address the deformation of the tool to successfully sweep the dust onto the dustpan.

3.4.5 Extrinsic Contact Servoing - Brush

Finally, we demonstrate application of our contact servoing formulation to a new tool - a compliant brush. We follow the same data collection and architecture as used for the preceding experiments. For this experiment, we use point clouds from the Photoneo as input, to retain accuracy at the cost of slower acquisition.

We integrate our contact servoing method into CALAMARI [123], a language-conditioned contact goal generation method. Instead of relying on hand-specified contact goals, CALAMARI learns to predict contact maps according to visual and language inputs. These contact maps can then be provided to our contact servoing controller to execute the task. The model is trained in simulation to generate contact goals for a variety of tasks.

We demonstrate the integrated system on the task of sweeping dust into a dustpan using the compliant brush. Failure to account for the deformations can lead to misaligned contacts and failing to properly remove the material. An example rollout using our contact servoing method is shown in Fig. 3.9. We find that our method is able to effectively handle the compliance of the brush to accurately track desired contacts generated by the contact planner. Across 10 trials, our method achieved a dust removal rate of 82%.

3.5 Discussion

Our approach simplifies contact rich interactions for compliant tool manipulation, by avoiding the necessity for full system identification while maintaining interpretability and accuracy by explicitly modeling the *contact state* of the system and how it evolves. In the future, we wish to investigate our method's applicability to other tasks where full state estimation is difficult, but the contact state is crucial, such as wiping with a cloth.

A common failure mode for our method is in controlling contact when the tool is tilted, where it is more likely for the method to yield actions that take the robot out of contact. This could, in part, be due to data imbalance. In future work, we are interested in utilizing online learning [38] or curiosity [94] to more effectively cover the space of contacts in our dataset.

There are cases where tool to environment contact is not represented well as a contact line. In Chapter. IV, we'll investigate how to extend our contact representation to handle more complex contact geometries.

Finally, our method relies upon supervision. For future contact-rich tasks of interest, the need for labels could become more costly. We hope to investigate how we can remove reliance upon supervision by exploring few/zero shot generalization [36, 139] and domain randomization techniques [82].

CHAPTER IV

Integrated Object Deformation and Contact Patch Estimation from Visuo-Tactile Feedback

In Chapter III, we utilized a summary contact representation, representing the contact geometry as a line. However, many compliant object contacts are not well represented by such a simple geometry. In this chapter, we propose a flexible extended contact representation. We jointly recover the deformed geometry and contact patch, as the deformation itself is necessarily caused by the contact. This is achieved via a novel neural implicit representation, enabling expressive representations of both deformed geometry and contact, conditioned on partial point clouds and F/T feedback. We demonstrate our representation on both simulated and real interactions. Our results indicate that jointly modeling deformation and contact yields precise recovery of complex contact geometries.

4.1 Introduction

The ability to manipulate elastically deformable objects (e.g., spatulas and sponges) is crucial for many contact-rich manipulation tasks. In order for robots to effectively use these compliant tools, they must reason over the coupled dynamics of object deformation and environmental contact to control the resulting contact interface between the two. However, there are two key challenges to address. First, the frictional interactions between these objects and their environment is governed by complex non-linear mechanics, making it challenging to model and control their be-

havior. Second, perception of these objects is challenging due to both self-occlusions and occlusions that occur at the contact location (e.g., when wiping a table with a sponge, the contact is occluded).

In this chapter, we propose Neural Deforming Contact Fields (NDCF) – a visuo-tactile neural implicit representation that models both the object and contact patch geometries as implicit fields (see Fig. 4.1). Specifically, our approach maps each point in 3D space to a signed-distance value and a probability of contact. By jointly reasoning over these coupled fields, our representation enables: i) modeling complex object and contact formation geometries; ii) enforcing physical priors such as ensuring that contacts lie on the surface of the object; and iii) estimating contact patches or deformed geometries given partial visuo-tactile observations. We further present a neural network implementation and learning algorithm for NDCF. Finally, we evaluate NDCF on simulated and real-world data and benchmark against explicit methods utilizing point cloud representations.

Our work builds on the recent advances in neural implicit and signed-distance field (SDF) representations of object geometry that have recently gained significant attention in computer vision [91, 107, 95] and robotics [27, 122, 124]. These representations offer several advantages over traditional geometric object and environment models. One of their key benefits is the ability to handle objects and environments with complex, non-linear shapes, such as deformable objects and cluttered scenes. Unlike traditional geometric models, which often rely on a fixed set of vertices and edges, neural implicit representations and SDFs encode the shape and behavior of objects as continuous functions. Recent work has utilized neural implicit representations to effectively model deformable object geometries [122, 124]. However, existing methods have largely ignored geometric representations of contact. Our approach brings contact geometry to center stage, considers object deformations, and allows for far greater flexibility than classical models such as a point contacts [77], line contact [58, 73], or planar patch contact [72, 134, 143] that are typically limited to rigid-body objects and have limited representation capacity.

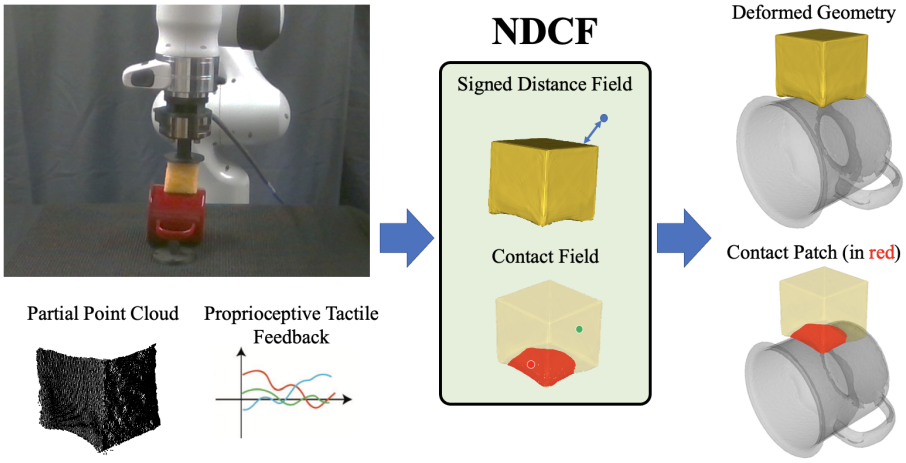


Figure 4.1: We present Neural Deforming Contact Fields (NDCFs), a method for recovering deformation and contact patch predictions from partial point clouds and tactile feedback (here, robot proprioceptive tactile feedback). We show that while NDCF is trained using simulated data, it is able to transfer to the real-world directly. Here, a robot with a deforming sponge presses into the YCB mug object. NDCF is able to faithfully recover the deformed tool geometry and contact patch between the deformed sponge and rigid mug (shown in red) given real-world sensing from this interaction.

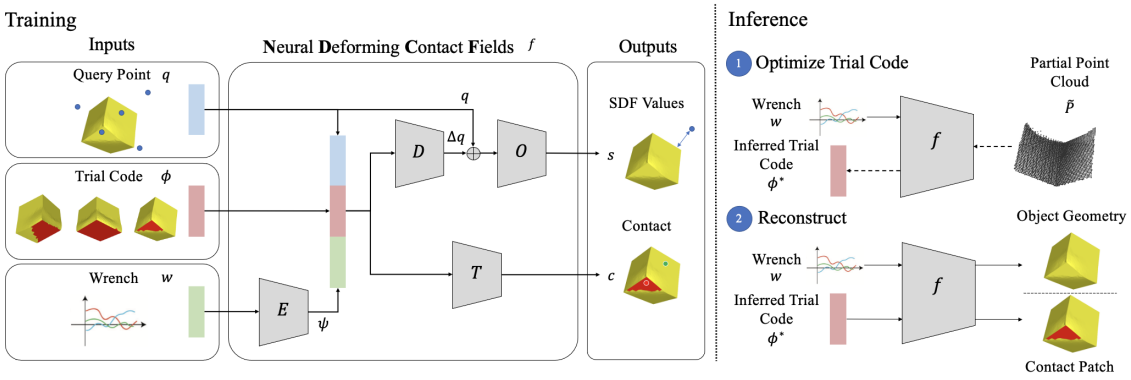


Figure 4.2: Method Overview: We propose **Neural Deforming Contact Fields**, to jointly reason over deformable object geometries and contact patches, conditioned on wrench feedback. We represent both geometry and contact as an implicit method, predicting an SDF s and a contact probability c for query points q. At inference time, we take in a partial point cloud and wrench value and recover a desired trial latent code, which can be used to reconstruct the full geometry and contact patch.

4.2 Problem Formulation

We propose Neural Deforming Contact Fields (NDCF), a learned multimodal implicit representation that simultaneously reasons over grasped object deformations, wrench feedback, and contact patches. The key insight of our method is to jointly predict the geometry of the object and the contacts on the object surface, rather than assume contacts are provided [122] or reasoning about contacts indirectly [124]. We choose to represent the contact patch implicitly as a neural field, which allows for contact patches of varying shapes and topologies to be represented with a single model. Additionally, by learning the deformed object geometry and contact patch jointly, we can enforce physical priors during training and inference, ensuring that contacts lie on the surface of the deformable object.

Our method is designed to incorporate feedback from common robotic sensors. As such, we assume access to a partial point cloud $\tilde{P} \in \mathbb{R}^{N\times 3}$ of the segmented object and wrench feedback at the robot wrist $\boldsymbol{w} \in \mathbb{R}^6$. Then, given a query point $\boldsymbol{q} \in \mathbb{R}^3$, we predict its SDF value $s \in \mathbb{R}$ and the likelihood that it is in contact $c \in [0, 1]$:

$$(s, c) = f(\tilde{P}, \boldsymbol{w}, \boldsymbol{q})$$

The deformable object surface is given by the zero-level set of the SDF value:

$$S = \{\boldsymbol{q} \mid (s = 0, c) = f(\tilde{P}, \boldsymbol{w}, \boldsymbol{q})\} \tag{4.1}$$

The zero level set can be recovered through Marching Cubes [71] or ray tracing methods and can easily be converted to a point cloud or a geometric mesh $\mathcal{M}$ [20].

The contact patch is given by the intersection of the object surface with points classified as in contact, where ϵ is the binary classification threshold:

$$C = S \cap \{\boldsymbol{q} \mid (s, c > \epsilon) = f(\tilde{P}, \boldsymbol{w}, \boldsymbol{q})\} \tag{4.2}$$

4.3 Method

4.3.1 Architecture

We propose a neural network architecture for learning NDCFs from data. The full architecture can be seen in Fig. 4.2.

1. Network Inputs: We train the network to reconstruct geometry and contact patches from static interactions between a deforming object and its environment. To this end, we adopt the encoder-less geometric reasoning popular for learning implicit geometries [83, 91, 122, 124]. By training without an encoder, we additionally decouple the input point cloud distribution from the network training. We introduce a trial code $\boldsymbol{\phi} \in \mathbb{R}^L$ that captures the current deformation and contact occurring in the scene. The latent space of $\boldsymbol{\phi}$ is learned simultaneous to the training of the network weights.

The wrench $\boldsymbol{w}$ is encoded through a neural network E into a latent vector $\boldsymbol{\psi} = E(\boldsymbol{w}) \in \mathbb{R}^L$ used to introduce force reasoning into the network. The latent vectors $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are jointly input to later parts of the network and trained together, allowing for joint reasoning over forces, deformations, and contact patches.

2. Signed Distance Field: We follow Wi et al. [122, 124] in representing deforming object SDFs by decomposing into a nominal SDF and deformation field, relating the deformed geometry to the nominal geometry. First, we represent the nominal SDF implicitly, with a neural network O, taking in a query point $\boldsymbol{q} \in \mathbb{R}^3$ and predicting a SDF $s \in \mathbb{R}$ as $s = O(\boldsymbol{q})$.

The second component is a predicted deformation field. This network predicts how each point in space can be deformed *back* to the nominal geometry, represented implicitly with the network D. As the deformation is dependent on the particular interaction, we provide this network with our latent vectors ($\boldsymbol{\phi}$, $\boldsymbol{\psi}$) to inform the deformation. D then maps from a query point $\boldsymbol{q} \in \mathbb{R}^3$ to a deformation $\Delta\boldsymbol{q} \in \mathbb{R}^3$ as $\Delta q = D(\boldsymbol{q}|\boldsymbol{\phi}, \boldsymbol{\psi})$. The final SDF prediction becomes:

$$s = O(\boldsymbol{q} + D(\boldsymbol{q}|\boldsymbol{\phi}, \boldsymbol{\psi}))$$

In this chapter, we learn an NDCF for a single object. In future work, we plan to extend this framework to learning across multiple classes of object by adding an additional object latent code [122, 83, 91].

3. Contact Field: We predict the likelihood of contact at every point in space. To make the prediction we use a neural network T. Similar to deformation, our contact is dependent on the particular interaction, so we provide this network with our latent vectors $(\boldsymbol{\phi}, \boldsymbol{\psi})$. T then maps from a query point $\boldsymbol{q} \in \mathbb{R}^3$ to a contact probability $c \in [0, 1]$:

$$c = T(\boldsymbol{q}|\boldsymbol{\phi}, \boldsymbol{\psi})$$

4.3.2 Training

Our training is composed of two steps: first, we pretrain the object module O such that the nominal geometry (without contact) is accurately represented by this module. Next, we train the entire architecture end-to-end to model deformations and contact patches.

1. Pretraining the Object Module: We pretrain our object module O to fit the SDF of our nominal object geometry. Given a dataset of sampled points around the nominal geometry, $\Omega = \{\boldsymbol{q}_i, s_i^*, \boldsymbol{n}_i^*\}_i$, we train with the following loss:

$$\mathcal{L}_{sdf} = \frac{1}{|\Omega|} \sum_{i=1}^{|\Omega|} |O(\boldsymbol{q}_i) - s_i^*| + \xi \frac{1}{|\Omega_S|} \sum_{i=1}^{|\Omega_S|} (1 - \langle \nabla O(\boldsymbol{q}_i), \boldsymbol{n}_i^* \rangle)$$

where Ω_S is the subset of sampled surface points and ξ weights the SDF and normal losses. The normal loss encourages the predicted and ground truth normals to align on the object surface. The predicted surface normal can be recovered by differentiating the SDF output with respect to the query point: $\nabla O(\boldsymbol{q}_i) = \partial O(\boldsymbol{q}_i)/\partial \boldsymbol{q}_i$. The gradient can be calculated using backpropagation. The weights of O are frozen after pretraining.

2. Loss Formulation: Our model is trained on a dataset of interactions provided

as:

$$\mathcal{D} = \{(\Omega_1, \boldsymbol{w}_1, \boldsymbol{\phi}_1), (\Omega_2, \boldsymbol{w}_2, \boldsymbol{\phi}_2), \ldots, (\Omega_N, \boldsymbol{w}_N, \boldsymbol{\phi}_N)\}$$

For each interaction we have a set of sampled points $\Omega_i = \{\boldsymbol{q}_j, s_j^*, \boldsymbol{n}_j^*, c_j^*\}_i$, as well as the generated trial code $\boldsymbol{\phi}_i$ and wrench for the example $\boldsymbol{w}_i$. For each example $(\Omega_i, \boldsymbol{w}_i, \boldsymbol{\phi}_i)$, we calculate the training loss given by:

$$\mathcal{L}_{train} = \mathcal{L}_{sdf} + \alpha\mathcal{L}_{embedding} + \beta\mathcal{L}_{deform} + \omega\mathcal{L}_{chamfer} + \gamma\mathcal{L}_{contact}$$

where $\mathcal{L}_{sdf}$ encourages the final predicted SDF values and normals to match the ground truth, and is defined as:

$$\begin{aligned}\mathcal{L}_{sdf} =& \frac{1}{|\Omega_i|}\sum_{j=1}^{|\Omega_i|} |O(\boldsymbol{q}_j + D(\boldsymbol{q}_j|\boldsymbol{\phi}_i, \boldsymbol{\psi}_i)) - s_j^*| \\ &+ \xi\frac{1}{|\Omega_{i,S}|}\sum_{j=1}^{|\Omega_{i,S}|}(1 - \langle\nabla O(\boldsymbol{q}_j + D(\boldsymbol{q}_j|\boldsymbol{\phi}_i)), \boldsymbol{\psi}_i), \boldsymbol{n}_j^*\rangle)\end{aligned}$$

where $\Omega_{i,S}$ are the points in Ω_i lying on the surface of the object, ξ weights the SDF and normal losses, and $\nabla O(\boldsymbol{q}_j + D(\boldsymbol{q}_j|\phi_i, E(\boldsymbol{w}_i)) = \frac{\partial O(\boldsymbol{q}_j + D(\boldsymbol{q}_j|\boldsymbol{\phi}_i, \boldsymbol{\psi}_i))}{\partial \boldsymbol{q}_j}$ is computed using backpropagation.

$\mathcal{L}_{embedding} = ||\boldsymbol{\phi}_i||_2^2$ is used to ensure the learned latent space is well-formed [91]. $\mathcal{L}_{deform} = \frac{1}{|\Omega_i|}\sum_{j=1}^{|\Omega_i|} ||D(\boldsymbol{q}_j|\boldsymbol{\phi}_i, \boldsymbol{\psi}_i)||_2^2$ is used to embed the prior that smaller deformations are preferred to complex large deformations.

The loss $\mathcal{L}_{chamfer}$ is a Chamfer distance used to encourage the predicted nominal surface, derived by adding the query points that lie on the surface to their predicted deformations, matches the ground truth nominal surface. Let

$$P = \{\boldsymbol{q}_j + D(\boldsymbol{q}_j|\boldsymbol{\phi}_i, \boldsymbol{\psi}_i)|s_j^* = 0\}$$

be the predicted nominal surface and P^* be the ground truth nominal surface point cloud. Then,

$$\mathcal{L}_{deform} = \text{CD}(P, P^*)$$

where CD is the Chamfer distance between two point clouds:

$$\text{CD}(P_1, P_2) = \frac{1}{|P_1|} \sum_{\boldsymbol{x} \in P_1} \min_{\boldsymbol{y} \in P_2} ||\boldsymbol{x} - \boldsymbol{y}||_2^2 + \frac{1}{|P_2|} \sum_{\boldsymbol{x} \in P_2} \min_{\boldsymbol{y} \in P_1} ||\boldsymbol{x} - \boldsymbol{y}||_2^2 \tag{4.3}$$

Finally, we supervise our contact field using $\mathcal{L}_{contact}$ defined as follows:

$$\mathcal{L}_{contact} = \frac{1}{|\Omega_{i,S}|} \sum_{j=1}^{|\Omega_{i,S}|} \text{BCE}(T(\boldsymbol{q}_j | \boldsymbol{\phi}_i, \boldsymbol{\psi}_i), c_j^*) \tag{4.4}$$

where BCE is the Binary Cross Entropy loss. Notably, we only evaluate binary classification on the *surface*, that is the set of points $\Omega_{i,S}$. This avoids having to learn a 2D contact surface directly, rather inheriting the geometric surface predicted by the SDF, while maintaining the flexibility of the implicit function in terms of shape and topology.

Training is achieved by solving the following optimization:

$$\boldsymbol{\theta}^*, \boldsymbol{\phi}_i^* = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\text{argmin}} \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}_{train}(\mathcal{D}_i) \tag{4.5}$$

As in other encoder-less methods, we simultaneously train our *trial codes* $\boldsymbol{\phi}_i$ alongside the weights of our network.

4.3.3 Inference

We assume access to a partial point cloud $\tilde{P}$ and wrench reading w. We then can perform inference to find the trial code ϕ^* for the example using the following optimization:

$$\phi^* = \min_{\phi} \frac{1}{|\tilde{P}|} \sum_{i=1}^{|\tilde{P}|} O(\tilde{P}_i + D(\tilde{P}_i | \boldsymbol{\phi}, E(\boldsymbol{w}))) + \eta ||\boldsymbol{\phi}||_2^2 \tag{4.6}$$

This optimization finds the latent vector $\boldsymbol{\phi}$ that places all the partial points on the surface of the generated geometry, additionally conditioned on the wrench $\boldsymbol{w}$, thus matching the generated NDCF to the observations. The second loss term is used to regularize the prediction [91] and prevents drifting away from well formed latent vectors. The geometry and contact patch can then be recovered from the partial point cloud and wrench measurements, as detailed in Sec. 4.2.

4.4 Implementation

4.4.1 NDCF Implementation

Our wrench encoder E is implemented as multi-layer perceptron (MLP) with a single hidden layer with size 16. Recent studies in neural implicit methods suggest the best method for conditioning outputs is using Hyper Networks, where conditioning vectors are used to predict the weights of an additional network, which takes in the query point and predicts the field value [122, 124, 24, 95]. As such we implement our deformation module D and contact module T as hypernetworks:

$$\Delta \boldsymbol{q} = D(\boldsymbol{q}|H_D(\boldsymbol{\phi}, \boldsymbol{\psi}))$$
$$c = T(\boldsymbol{q}|H_T(\boldsymbol{\phi}, \boldsymbol{\psi}))$$

Here H_D and H_T predict the weights of each module MLP. We add additional regularization terms to regularize the predicted weights.

$$\mathcal{L}_{hyper} = \zeta(\frac{1}{|H_D(\boldsymbol{\phi}, \boldsymbol{\psi})|}||H_D(\boldsymbol{\phi}, \boldsymbol{\psi})||_2^2 + \frac{1}{|H_T(\boldsymbol{\phi}, \boldsymbol{\psi})|}||H_T(\boldsymbol{\phi}, \boldsymbol{\psi})||_2^2)$$

ζ is a weighting on the loss. The predicted deformation MLP has a single hidden layer of size 256 while the predicted contact MLP has two hidden layers of size 256. The object nominal SDF module O is implemented as a MLP with 2 hidden layers of size 256. All models are implemented using Pytorch. We set latent space dimension of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ to be in $L = 16$.

4.4.2 Training and Inference Details

NDCF Pretraining is performed using the Adam Optimizer with learning rate $1e-5$. We set normal loss weighting term $\xi = 0.01$. The pretraining is run for 50000 epochs to effectively memorize the nominal object geometry.

NDCF Training is performed using the Adam Optimizer with learning rate $1e-4$. The latent codes ϕ_i are initialized from the zero mean Gaussian with standard deviation 0.1. From Sec. 4.3.2, we set $\alpha = 1e-3, \beta = 1.0, \omega = 0.01, \gamma = 0.1, \xi = 0.01$. From Sec. 4.4.1, we set $\zeta = 1e-6$. We train our method for 200 epochs.

NDCF Inference is performed with the Adam Optimizer with learning rate $2e-3$. Latent codes, as during training, are initialized from a zero mean Gaussian with standard deviation 0.1. In each experiment, we use a small validation set to select the choice of η in Eq. 4.6, the contact probability threshold ϵ, and the number of gradient descent iterations to perform. In particular, we perform a grid search and choose inference hyper-parameters with the best performance. We report the selected hyper-parameters for each experiment in Sec. 4.5.

4.4.3 Data Collection

4.4.3.1 Simulation

The training methodology described in Sec. 4.3.2 relies upon having strong supervision of sampled points and their corresponding SDF and contact patches. In this work, we utilize Isaac Gym [75], a GPU-based physics simulator, to collect simulated deformable-object interactions. Isaac Gym implements 3D Finite Element Method (FEM) on the GPU and has been experimentally validated for accuracy in rigid-deformable interactions [86].

In this work, our object is a 46mm cube sponge rigidly attached to a Franka Emika Panda end effector, with an ATI Gamma F/T sensor mounted between (see Fig. 4.3). We recreate the sponge in Isaac Gym, rigidly attached to the wrist mounting geometry used in the real world (see Fig. 4.4). Isaac Gym FEM assumes homogenous material properties and requires specification of the Poisson's ratio ν and Young's Modulus E. We set $\nu = 0.1$ (a low value) to reflect the fact that because the sponge

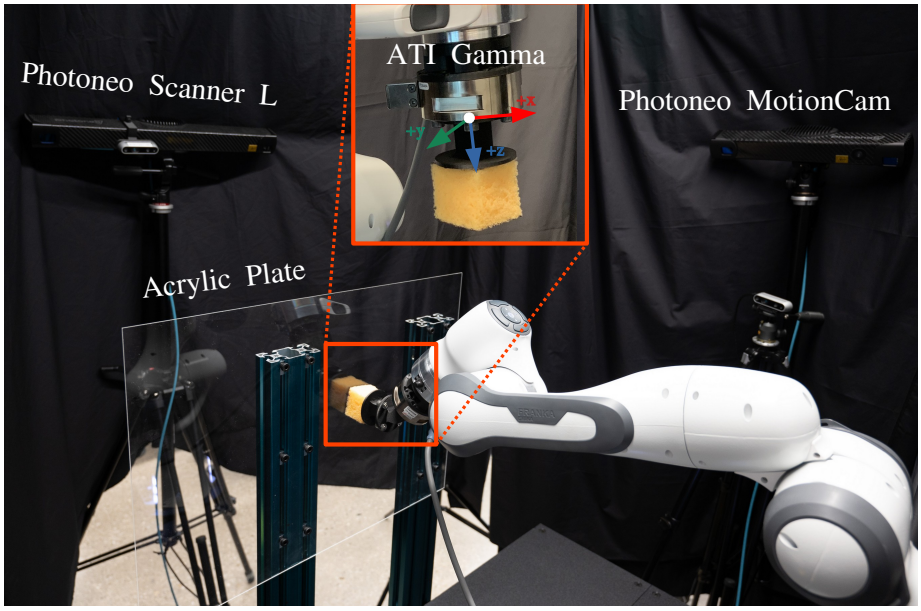


Figure 4.3: Real-world experiment setup with 1 Pandas arm, 2 depth cameras (Photoneo ScannerL and Photoneo MotionCam-3D color), 1 force-torque sensor (ATI Gamma), and 1 46mm sponge mounted on the force-torque sensor. ScannerL was used for labelling while MotionCam was used for the partial point cloud inputs.

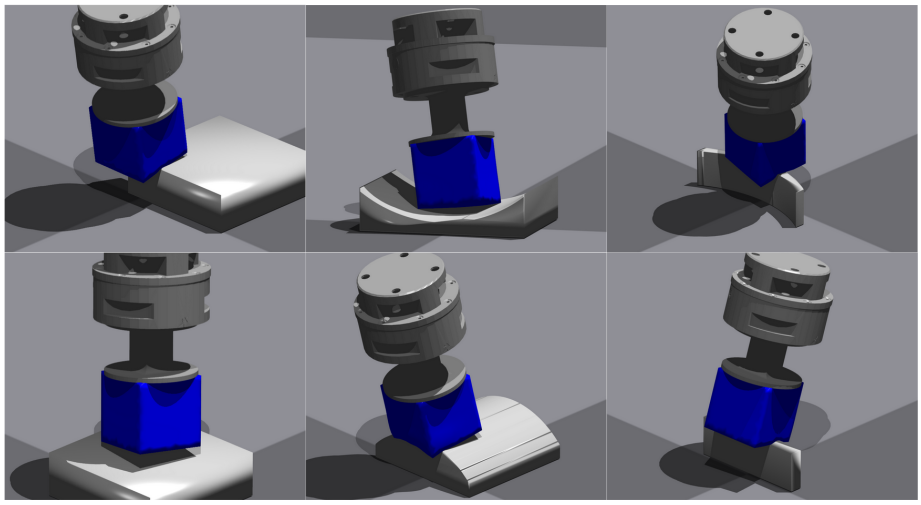


Figure 4.4: Visualization of generated environments and sampled interactions for our training dataset in Isaac Gym. Environments per column are **box**, **curves**, and **ridges** from left to right.

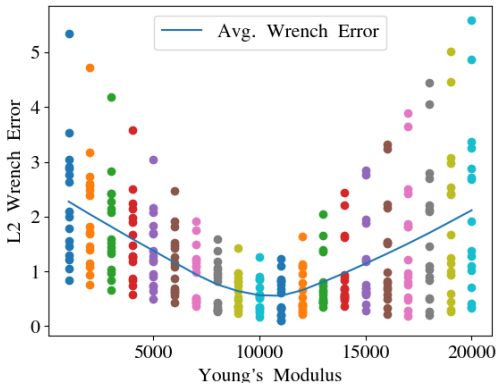


Figure 4.5: We perform system identification to match simulated material properties to our real object setup. 16 real setup presses are recreated in simulation and the simulated wrench is compared to that registered in the real world. Points indicate individual simulation trial comparisons to the real world data, colored for visual clarity by the used Young's Modulus. We found the optimal Young's Modulus to be $E = 1.1e4$ Pa.

is a porous material with air gaps, it has a significantly smaller transverse elongation due to compression than a solid object. To determine E we perform system identification. On the real system, we collect a series of 16 presses onto a flat surface and record the wrench response measured by the F/T sensor. We then recreate those presses in simulation, by moving the simulated wrist to the same pose, and measure the simulated response. We perform a line search over Young's Modulus values, using the euclidean distance between real and simulated wrenches to evaluate match. We find that $E = 1.1e4$ Pa best matched our robot setup in simulation (see Fig. 4.5).

We collect a dataset of simulated presses to train our NDCF. We use three scenarios: first is a box, second is a randomly generated curved surface, and third is a randomly generated ridge terrain (see Fig. 4.4). In each scenario, we sample random orientations for the wrist and randomize the starting position of the gripper to enable interactions near the edges of objects as well as on the surfaces, then lower straight down until a randomly sampled distance between 3mm and 1cm past the point of initial contact. The resulting data is saved and processed to form dataset

$\mathcal{D}$, including ground truth geometries, feedback wrenches, and contact labels.

In total, 3000 environments and interactions are sampled, 1000 per each environment type. For each interaction, we sample 40000 query points off surface and 20000 query points on the surface and generate ground truth SDF values, contact patches, and normal labels. We augment our dataset by rotating all inputs around the tool z-frame (frame shown in Fig. 4.3), as our system exhibits symmetry due to the shape of the tool.

To evaluate our method, we also collect an additional dataset of new interactions with full labels. We generate 300 new press interactions, 100 per environment, and prune examples where the resulting contact area was less than $1 \times 10^{-5} m^2$, for an evaluation set of 298 interactions. To provide partial geometric information, we place eight cameras spaced evenly around the wrist facing the tool to capture partial point clouds of the sponge in contact from multiple angles, which are used during inference, along with the simulated wrench feedback. Examples of partial views are shown in Fig. 4.6.

4.4.3.2 Physical Robot

We also demonstrate our method's effectiveness in the real-world using a physical robot. In order to provide useful evaluation of our method, we design a real-world test bed from which we can derive contact patch labels. Shown in Fig. 4.3, our real world setup places a clear acrylic plate in front of the Franka Emika Panda with the sponge mounted rigidly to the robot end effector. A Photoneo Phoxi 3D scanner (L) is placed opposite the acrylic. When the robot moves the sponge into contact with the acrylic, the Photoneo depth scan sees through the acrylic and locates the contacting face of the sponge. By calibrating the position of the acrylic, we can estimate the contact patch from the resulting depth scan by thresholding the points near the face of the acrylic. Examples of resulting labels can be found in Fig. 4.8. We use these as the ground truth contact patch to evaluate our methods on the real world data.

We use the wrench feedback from the mounted ATI gamma as the wrench input.

To get our partial point cloud, a Photoneo MotionCam-3D Color (M+) is mounted next to the robot viewing the contact interaction. The sponge is segmented from the resulting point cloud and provided as the partial geometry to each method.

We collect 48 different interactions with the acrylic sheet. We randomize the orientation of the sponge before pressing by sampling an offset angle in the range $[-0.3, 0.3]$ for rotations about the (x, y, z) axes relative to the nominal pose, which is the robot ATI Gamma frame (shown in Fig. 4.3) with the z axis pointing directly at the plate and the x axis pointing normal to the table underneath. The robot is moved to an offset position then pressed straight forward towards the acrylic 1cm past the point of initial contact.

4.5 Experiments

4.5.1 Baseline

We compare our implicit representation method to an explicit counterpart. Point clouds are flexible shape representations; however, they lack connectivity and are fundamentally discrete which limits their resolution and ability to capture high-frequency details. Here, we implement an explicit baseline representation that directly predicts both the geometry and contact patch as *point clouds*. We adopt the point cloud encoder-decoder architecture of [127] and condition the bottleneck feature on both the partial pointcloud $\tilde{P}$ and wrench $\boldsymbol{w}$. We implement two point cloud decoders, one to output deformed object geometries and the other to output the contact patch predictions.

$$P, C = f(\tilde{P}, \boldsymbol{w})$$

We use the following loss to train:

$$\mathcal{L}_{baseline} = \xi_1 \text{CD}(P, P^*) + \xi_2 \text{CD}(C, C^*) + \xi_3 \text{CD}_{uni}(C, P)$$

where P is the estimated object surface point cloud and P^* is the ground truth surface

point cloud. Similarly, C is the estimated contact patch and C^* is the ground truth contact patch. The final loss term incorporates the knowledge that we know that the contact patch should lie on the surface of the object. This can be encouraged in the learned model by a *uni-directional* Chamfer distance, which only calculates the distance from the contact patch C to the nearest points on the geometry P.

$$\text{CD}_{uni}(C, P) = \frac{1}{|C|} \sum_{\boldsymbol{x} \in C} \min_{\boldsymbol{y} \in P} ||\boldsymbol{x} - \boldsymbol{y}||_2^2$$

We used Adam Optimizer with learning rate 1e-4, set $\xi_1 = \xi_2 = \xi_3 = 1e4$, and ran 200 epochs for training on the same dataset used to train the NDCF. Upon reconstructing geometry from the point cloud P, we apply a meshing algorithm to convert the output to a mesh [21].

The use of an encoder-decoder structure means the baseline is sensitive to the distribution of the partial view $\tilde{P}$ seen for training examples, e.g., the placement of the cameras. As such, we perform augmentation during training, selecting subsets of the camera views so that the encoder is somewhat robust to the available partial view. We note our decoder-only method is invariant to the input distribution, since partial views are not used during training and rather are used in the loss during inference. As such, we do not need to perform any augmentation when training our method, in comparison to the baseline.

4.5.2 Metrics

1. Geometry: To evaluate the deformed object geometry, we use two metrics. The first metric is the Chamfer Distance (see Eq. 4.3) of the predicted surface point cloud to the ground truth surface point cloud. For NDCF, we generate the surface point cloud P by sampling from the generated object mesh. For the point cloud baseline, we directly compare the generated object surface point cloud to the ground truth. To make for fair comparison, we down-sample each resulting point cloud and ground truth point cloud to 10,000 points.

The second metric is a Volumetric IoU of the predicted mesh $\mathcal{M}$ and ground

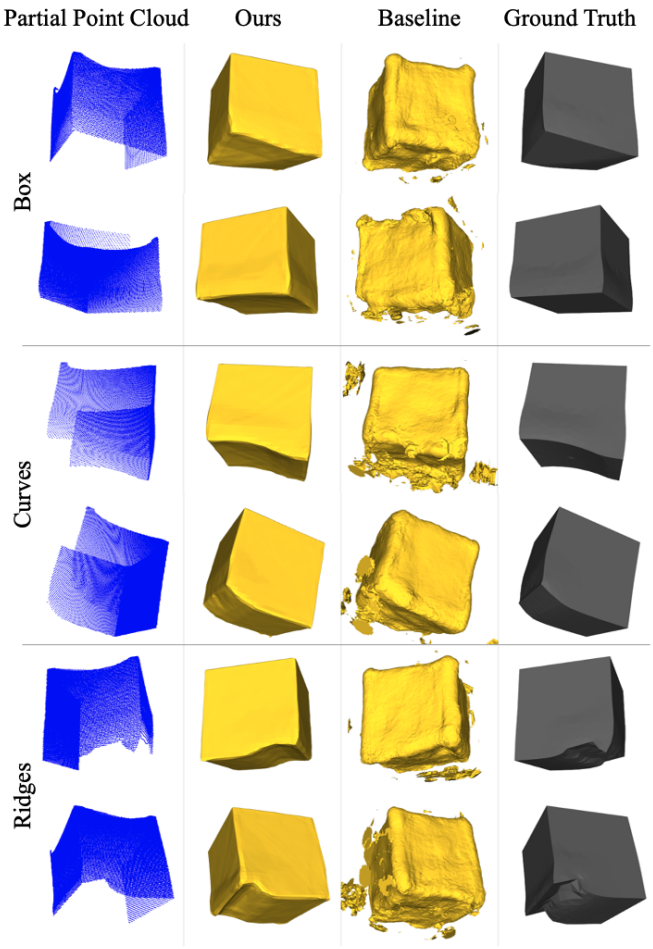


Figure 4.6: Selected simulation geometry results from each of our test environments (Box, Curves, and Ridges). Our proposed method NDCF shows crisp reconstructions with high retention of detail, while the baseline exhibits artifacts and lacks geometric details.

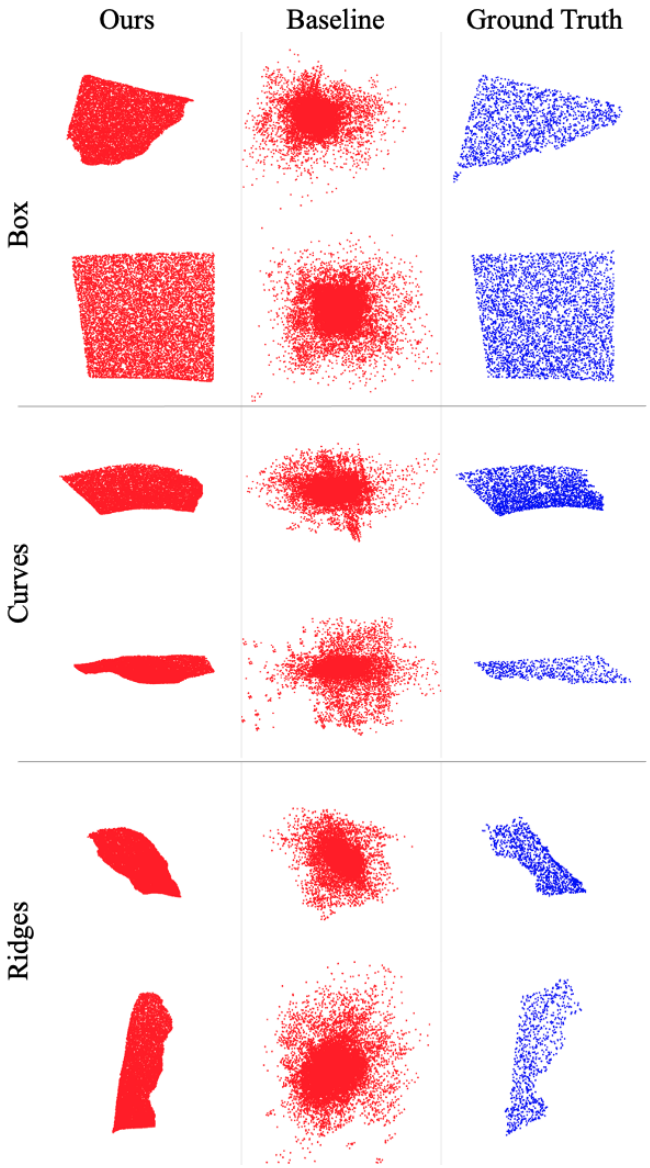


Figure 4.7: Simulation contact patch estimation results for examples from Fig. 4.6. Our proposed method NDCF captures the variety in contact shapes while the baseline estimates exhibit high variance with points predicted far from the contact patch.

Table 4.1: Model Performance on Simulated and Real-World Data

	Deformed Geometry (sim)		Contact Patch (sim)	Contact Patch (real-world)
	CD_{mm^2}	IoU	CD_{mm^2}	CD_{mm^2}
Ours	**0.910 (0.158)**	**0.987 (0.007)**	**22.840 (40.414)**	**39.502 (23.797)**
Baseline	5.746 (1.091)	0.817 (0.058)	36.537 (19.046)	56.853 (57.864)

truth mesh $\mathcal{M}^*$. The Volumetric IoU is defined as the quotient of the volume of the intersection of the meshes by the union of the meshes,

$$IoU(\mathcal{M}, \mathcal{M}^*) = \frac{|\mathcal{M} \cap \mathcal{M}^*|}{|\mathcal{M} \cup \mathcal{M}^*|}$$

We estimate the volumes by sampling 100,000 points near the object and determining which points lie inside the predicted and ground truth meshes [80].

2. Contact Patch: We evaluate the contact patch geometries using the Chamfer Distance (see Eq. 4.3). For NDCF, we use rejection sampling on the surface to recover the contact patch, evaluating the contact probability at each sample and accepting those above ϵ. For the point cloud baseline, we directly compare the generated contact patch point cloud to the ground truth. To make for fair comparison, we downsample the predicted and ground truth contact patch point clouds to 300 points. We found that the baseline did not spread the points evenly through the predicted patch, so we use a voxel-downsampling technique to better sample from the point cloud volume fairly.

4.5.3 Simulated Experiments

We first evaluate our method on 298 unseen simulated interactions, as detailed in Sec. 4.4.3.1. We use a validation set of 90 press interactions, 30 from each environment, to select inference hyper-parameters. We select $\eta = 0.001$, $\epsilon = 0.2$, and run inference for 100 gradient steps.

Our method successfully predicted a contact patch in all but two cases, in which it failed to predict any points on the deformed surface with contact probability above

Table 4.2: Timing Performance on Simulated Data

	Inference Time (sim)	Meshing Time (sim)
	Seconds	Seconds
Ours	0.173 (0.006)	1.906 (0.378)
Baseline	**0.026 (0.001)**	–

$\epsilon = 0.2$. To avoid skewing baseline performance, we remove these two examples when calculating performance metrics for both methods.

Full performance results are shown in Table 4.1. We see that our method outperforms the proposed baseline on all three metrics, reflecting better prediction of both object geometry and contact patches. We also show the runtime of each method - while the inference procedure does take longer than a forward pass of a model, we still are able to run our method at roughly 0.5Hz, including the time to perform meshing via Marching Cubes. Further tuning of inference parameters could trade off performance and runtime for specific applications. Additional simulation performance analysis can be found in Appendix B.1.

In Fig. 4.6 and Fig. 4.7, we show the predicted geometry and contact patch for several examples from each environment. Our method, NDCF, shows high retention of geometric details, such as the curvature of the mesh in the first "Curves" example. The proposed baseline is much more noisy, with rounded edges and artifacts appearing off the surface from spuriously generated points. Our method also shows the ability to capture diverse contact shapes accurately, whereas the baseline method, while often clustered near the patch, is noisy, with contact points predicted at times far from the object surface. Our method, can directly take advantage of the geometric reasoning of NDCF, and as such the contact patches lie cleanly on the predicted surface of the object.

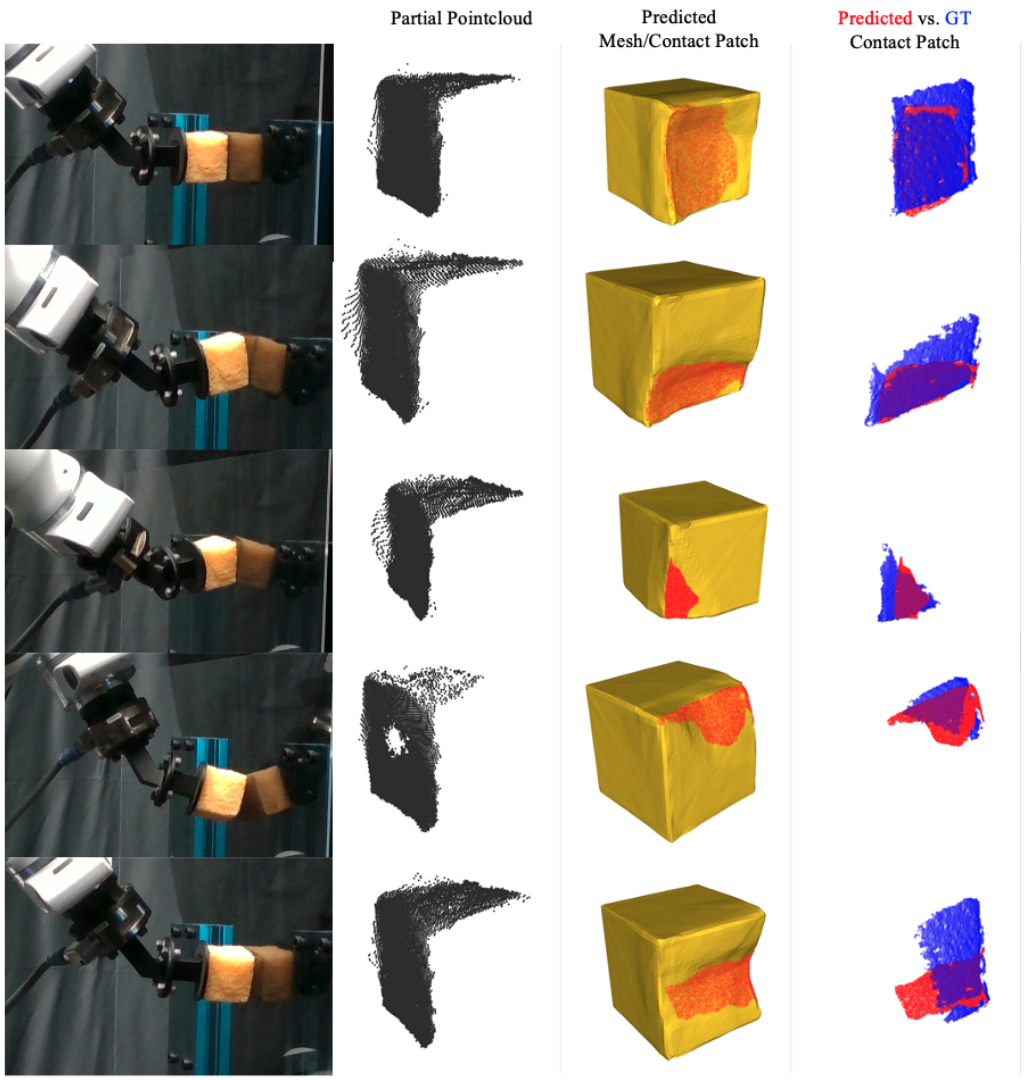


Figure 4.8: NDCF geometry and contact patch estimation on real world interactions with a flat surface. Our results show that NDCF trained on simulated data transferred to real world examples. Predicted meshes accurately predict where deformation occurs for each interaction and the contact patches are accurate to the measured ground truth patch. Note that the right three columns are rotated to view the contacting face of the object. The last row shows a failure case, where the contact occurs opposite the camera and thus is harder to detect.

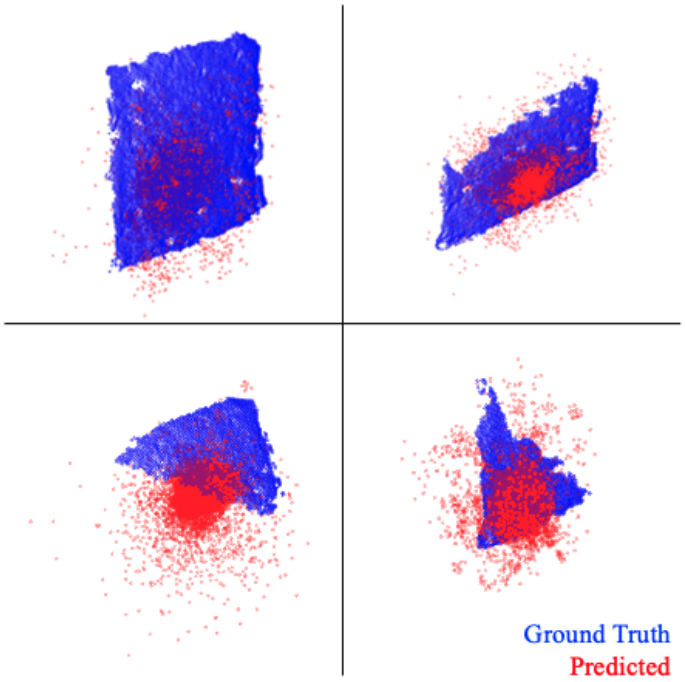


Figure 4.9: Clockwise from top-left: baseline contact predictions for the first four examples in Fig. 4.8. While the main cluster of points is located near the ground truth patch, the baseline predictions exhibit variance, with points predicted far from the patch and object surface.

4.5.4 Physical Robot Experiments

Here, we evaluate our methods ability to generalize from the simulated environment to the real world one. We use 48 physical robot interactions with ground truth contact patch labels, collected as described in Sec. 4.4.3.2. We use a set of 16 validation interactions, collected in the same manner, to choose inference hyper-parameters. We select $\eta = 0.1$, $\epsilon = 0.2$, and run inference for 100 gradient steps. A higher η is intuitive, since the real world sensing is more noisy than the simulated data, thus a term preventing the latent from drifting out of distribution prevents over-fitting to sensor noise.

Fig. 4.8 shows our methods performance on the real world data for several interactions with varying contact patch shapes. In each case, the predicted mesh appears to closely match the interaction and the contact patches are good estimates of the ground truth. Fig. 4.9 shows baseline predictions for the first four examples of Fig. 4.8. Similar to the simulation results, we found that the baseline often clustered points roughly near the patch but exhibited noise and often predicts points far from the contact.

Over the 48 test interactions, we evaluate the quality of the contact patch using the Chamfer Distance metric. As we do not have ground truth geometries, we forgo deformed geometry evaluation. As shown in Tab. 4.1, our method outperforms the baseline on patch CD.

4.5.5 YCB Interaction Experiments

As a final experiment, we qualitatively investigate our method's performance on objects in the real world. We place the mug and bowl object from the YCB dataset [14] in front of the robot. Poke interactions are prerecorded by a demonstrator, then repeated autonomously. We use the same robot and sensor setup described in Sec. 4.4.3.2, but now use partial point clouds combined from both the Photoneo MotionCam and Photoneo Scanner L as the partial visual data. We found using the inference hyper-parameters from Sec. 4.5.3 worked well for this data.

We show the qualitative results for three presses on each object in Fig. 4.1 and

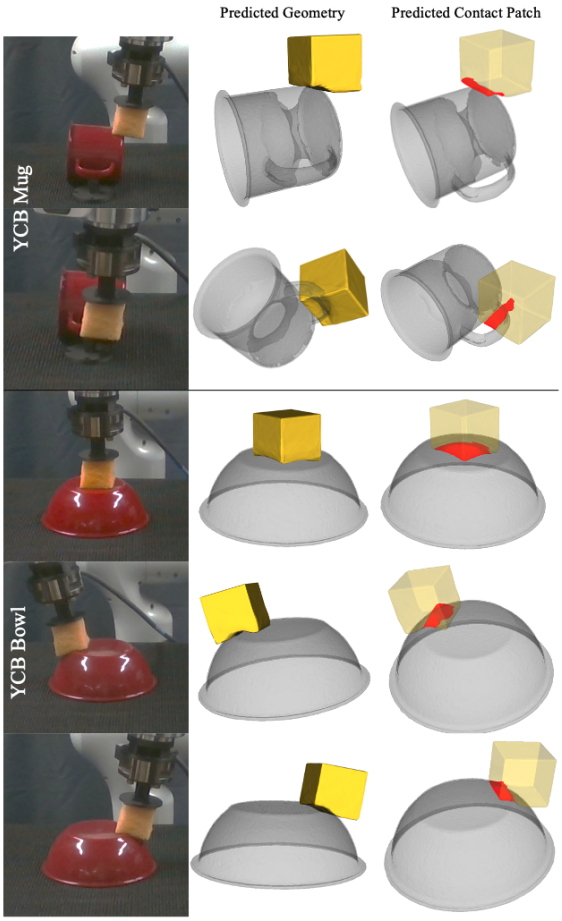


Figure 4.10: Qualitative results of applying NDCF to poke interactions with the YCB mug and bowl object. We find that for interactions with these objects, our method is able to recover feasible geometric completions and contact patches that are close to the ground truth object surface.

Fig. 4.10. The "ground truth" pose of the object used in visualization is approximated by performing ICP on a scan of the object from the Photoneo sensors with the robot moved clear. We see that NDCF is able to complete reasonable geometries and contact patches, including differentiating between flat contacts and contacts near the edge of objects, as well recognizing the "ridge" like contact of the mug handle.

To provide an approximate quantitative measure of real world performance, we measure the average squared distance from the predicted contact patch points to the surface of the object mesh (this can be seen as a uni-directional Chamfer Distance). Our method achieves 15.379 mm^2 for the Mug and 2.728 mm^2 for the Bowl.

4.6 Discussion

We present Neural Deforming Contact Fields (NDCFs), a representation that jointly reasons over extrinsic contact patches, deformable object geometry, and force transmitted via contact. Our method represents both the object geometry and contact patch using neural implicit fields, which makes the representation flexible enough to handle complex contact shapes and object deformations.

Our results indicate that we can utilize NDCFs to recover deformed object geometries and corresponding contact patches with high accuracy, and outperforms an explicit representation baseline method which uses point cloud representations. Additionally, we showed that we could transfer the representation to the real world without finetuning and demonstrated the ability to recover geometries and contact patches given real interaction data.

So far, our demonstration of NDCF has been limited to a single tool interacting statically with an environment. To handle multiple objects, we can condition our nominal object model using an object latent code to distinguish geometries, and introduce a materials latent code to address the case of identical geometry with varying material properties. For dynamic interactions, our representation can be augmented with dynamics in the latent space to predict future trial and wrench latent codes, conditioned on actions.

Another limitation of this current work is that it does not directly consider the

surrounding environment. Here, we primarily focused on demonstrating shared reasoning over geometries and contact, independent of the environment geometry. However, in certain cases (see last row Fig. 4.8) the partial geometry of the deformable object and the reaction wrench may not fully disambiguate the underlying deformations and contacts, as many varying contact configurations can yield similar observations.

CHAPTER V

Estimating Deformable-Rigid Contact Interactions for a Deformable Tool via Learning and Model-Based Optimization

This chapter addresses the case of a deformable object being utilized to manipulate a rigid object. This task necessitates understanding of force transfer and resulting motion of the manipulated object by the compliant tool. We propose a data-driven method that utilizes the observed deformation of the tool in the point cloud feedback to infer a) a summary contact representation capturing forces being applied, and b) the resulting motion of the block. This unlocks the ability to utilize model-based reasoning to solve for the extrinsic contacts and forces on the rigid object. We utilize our method to infer motion and forces in real interactions and to rotate an object while maintaining desired extrinsic forces. Our results show that our proposed hybrid learning and model-based approach outperforms purely data-driven techniques and enables direct control of extrinsic contact forces.

5.1 Introduction

In this chapter, we consider an elastically deformable tool manipulating a rigid object, supported by the environment. This scenario could arise when utilizing deforming tools such as spatula or sponges or manipulating objects with a soft robot [37] or end-effector [88]. Manipulation of such a system presents two important challenges:

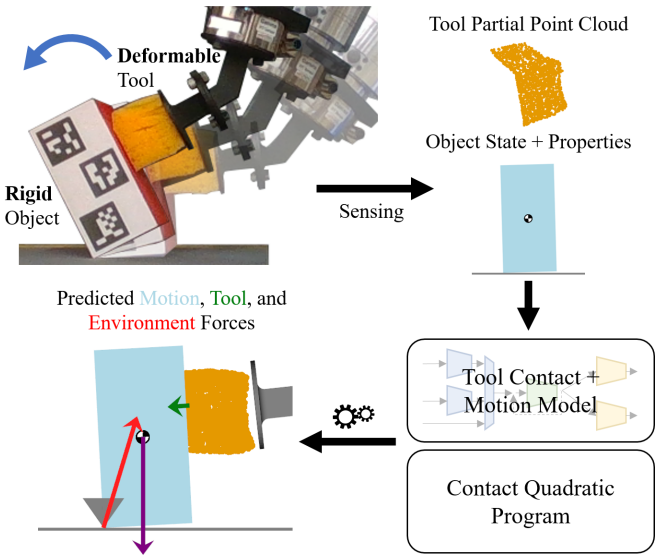


Figure 5.1: We present a method that estimates motions and forces during dexterous manipulation with a deformable tool. Our proposed method takes in object information (geometry, center of mass, mass, and friction) and sensing from the deforming tool (partial point cloud). It estimates the block motion, tool force the deforming tool enacts on the block, and the rigid contact forces between block and (known) environment, given the robot actions. On the bottom left, we show one-step predictions from our method for a real pivoting execution.

First, the deformation of the tool and motion of the rigid object are inherently linked - solving for one requires reasoning about both. However, modeling the deformation of the tool is challenging, due to the high-dimensional and non-linear dynamics [122]. Second, the system is only partially observable, and we must rely on sensing to intuit deformations and contact forces.

To address these challenges, we propose a hybrid learning and first principles method for modeling the motion, contacts, and forces on the tool and extrinsic object during manipulation. Accurate motion models are crucial for driving the object toward goal configurations while recovering forces can help enforce desired contact modes and prevent excessive force transfer.

Our method leverages learning to address the complexity of the paired object motion and deformable tool contacts. It then turns to first-principles and physical

priors to recover the extrinsic contacts and forces, enforcing quasi-static motion and Coulomb friction. This allows our method to overcome modeling challenges for the deformable object, while exploiting physical priors for modeling efficiency. We train our method on interactions between a tool and a variety of object geometries and physical properties using simulation [39]. We benchmark our method against baselines for modeling block motion and for recovering the environment-object forces. Finally, we deploy our model on a real robot, demonstrating sim-to-real transfer. In summary, our contributions are:

- a learning architecture for jointly resolving object motions and tool contacts and forces, given partial sensing and object information.
- a first-principles Contact Quadratic Program (CQP) that resolves environment contacts and forces, subject to quasi-static equilibrium and Coloumb friction.

5.2 Problem Formulation

Our goal is to reason over a) the motion of the extrinsic rigid object, and b) all the forces acting on it. We consider the case where the motion of the robot and rigid object are in the plane of gravity. We assume the following information from each part of the system:

1. **Object and Support Surface:** We assume knowledge of the mass m^e, the center of mass $\boldsymbol{p}^{e,CoM}$, the friction between object and rigid supporting surface μ^e, the geometry of the object, provided as a mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E})$, and the geometry of the support surface. Furthermore, we can track the motion of the rigid object to receive the current object pose $\boldsymbol{q}_t^e \in SE(2)$. By our assumed knowledge of the environment geometry, we additionally can use the known object pose and geometry to recover the current contact locations $\boldsymbol{p}_t^{c,1}, ..., \boldsymbol{p}_t^{c,K}$.
2. **Tool:** We assume access to a *segmented* partial pointcloud of the deformable tool $\boldsymbol{P}^{tool}$, provided by our sensors. Recent developments in category-agnostic

open world segmentation [60] make recovering segmentation masks possible on the real system.

Given these inputs and assumptions, and the actions to be taken $\boldsymbol{a}_{t:t+H}$ for some horizon H, where each action is a planar translation and rotation, we wish to recover the future poses of the object $\boldsymbol{q}^e_{t:t+H}$ and all forces acting on the object. We split our consideration of the forces into those from the deforming tool and those from the supporting surface. Contact interface representations have take many forms, including points [77], lines [58, 73] and patches [41]. Here, the most important feature of the contact interaction is not its particular geometry, but rather its *effect*, namely, the resulting force on the block. As such, we consider a simple contact representation: a summary contact point $\boldsymbol{p}^{tool}_t$ and force $\boldsymbol{f}^{tool}_t$, whose resulting wrench on the block is equivalent to the actual and potentially extended contact. As many points and forces can yield an equivalent wrench on the object, we use the centroid of the contact between the tool and object as $\boldsymbol{p}^{tool}_t$, which prevents ambiguity in the representation. Finally, we wish to also recover the set of contact points $\boldsymbol{p}^{c,1}_t, ..., \boldsymbol{p}^{c,K}_t$ and forces $\boldsymbol{f}^{c,1}_t, ..., \boldsymbol{f}^{c,K}_t$ between the rigid object and the environment. We assume access to a labeled dataset with the specified inputs and outputs listed above, which we utilize to train our method and the baselines.

5.3 Method

Our proposed method has two main components - a learned module for predicting a) the object motion and b) the contact location and forces between the deforming tool and object, and a model-based optimization problem for recovering the resulting frictional contacts with the environment. An overview of our approach can be found in Fig. 5.2.

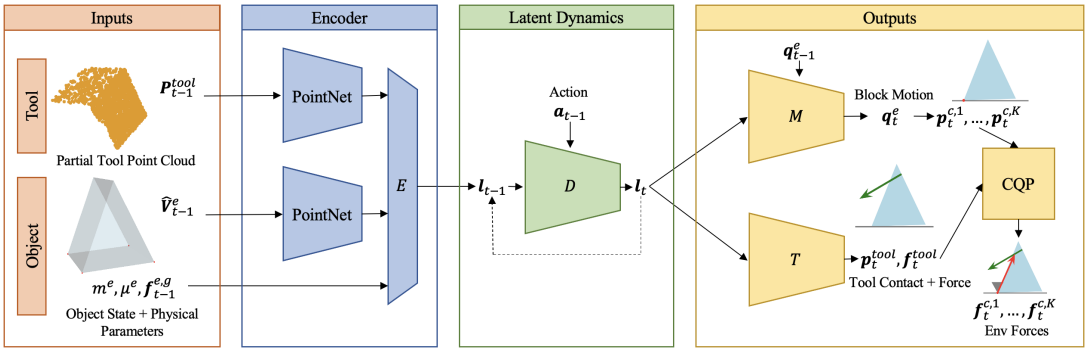


Figure 5.2: Overview of our proposed method. Our method takes in partial views of the tool via point clouds along with the current object state and parameters and encodes to a learned latent space. Our latent dynamics module then rolls out in the latent space given actions. Finally, our model regresses a) object motion and b) tool summary contact point and force. We use the object motion to determine environment contacts and solve a Contact Quadratic Program (CQP) of our design which resolves the environment contact forces subject to friction and quasi-static equilibrium.

5.3.1 Jointly Learning Object Motion and Deformable-Rigid Contact Interactions

The underlying mechanics of deforming objects is complex and can be expensive to compute [132, 39]. Simplifying models [79] can recover forces but lack fidelity to deforming surfaces. Learning-based methods have modeled geometries [122] and contact interfaces and crucially can recover information when only provided with partial information sensing, such as partial pointclouds, but have not recovered forces at the contacting surfaces. We adopt a learning-based approach to be able to recover the complex contacts and motions directly from partial sensing, and directly predict the resulting forces.

5.3.1.1 Architecture

An overview of our proposed architecture is shown in Fig. 5.2. Our architecture design encodes multimodal information from the object and the tool into a learned latent space. A latent dynamics backbone then propagates the latent information

provided with the actions. Finally, we have multiple output heads that decode the object motion and tool contact information from the latent. We detail the architecture below.

Network Inputs: In order to accurately reason about the interactions, we input multimodal information from the rigid object and the tool. We represent spatial quantities (poses, forces, points, etc.) in the robot end effector frame provided by $\boldsymbol{q}_t^r$. We first encode the partial pointcloud of the tool $\boldsymbol{P}_t^{tool}$ using a PointNet encoder [93]. We encode the geometry and current pose by extracting the mesh vertices V and transforming them to the tracked object location $\boldsymbol{q}_t^e$. We additionally use our knowledge of the environment to detect which vertices are contacting the surrounding environment and append these binary contact indicators to each vertex location. We use another PointNet encoder [93] to encode the resulting unstructured point set $\hat{\boldsymbol{V}}_t^e \in \mathbb{R}^{|V|\times 4}$. Note, that this also allows us to use a single encoder for objects with varying geometries and differing numbers of vertices. We fuse the resulting latent embeddings from the tool and rigid object, along with the object mass m^e, object friction μ^e, and the gravity vector $\boldsymbol{f}_t^{e,g}$ expressed in the robot frame. This allows the model to reason about gravitational effects while keeping all learning in the robot frame. These inputs are fused and passed through a final multi-layer perceptron (MLP) to yield the latent code $\boldsymbol{l}_t$. We call this encoder E.

Latent Dynamics: Given the input actions $\boldsymbol{a}_{t:t+T}$, we rollout the dynamics in the latent space [38] as the latent space can capture how the resulting deformations, motions, and forces evolve. Our latent dynamics model D is given the current latent state and action and predicts the future latent state.

$$\boldsymbol{l}_{t+1} = D(\boldsymbol{l}_t, \boldsymbol{a}_t) \tag{5.1}$$

This allows us to recursively predict successive states given action sequences.

Output Heads: The final learned components are two output networks which take the latent state at a given time step and estimate the motion. One MLP M is used to estimate the motion of the object, given the current latent state. We predict the

motion as a delta motion, utilizing the axis-angle representation [11, 12].[1]

$$\Delta \boldsymbol{q}_t^e = M(\boldsymbol{l}_t) \tag{5.2}$$

These delta poses can be combined with the tracked initial state $\boldsymbol{q}_t^e$ to predict the future rollout states of the rigid object.

A final MLP T is used to directly regress the tool contact point $\boldsymbol{p}_t^{tool}$ and force $\boldsymbol{f}_t^{tool}$.

$$\boldsymbol{p}_t^{tool}, \boldsymbol{f}_t^{tool} = T(\boldsymbol{l}_t) \tag{5.3}$$

5.3.1.2 Training Losses

We assume access to a labeled dataset $\mathcal{D}$ which contains example rollouts of the system with the necessary observations, actions, and labels provided. We train the model performance over a specified horizon H. We encode the starting observations using our encoder to yield our starting latent $\boldsymbol{l}_t = E(\boldsymbol{P}_t^{tool}, \hat{\boldsymbol{V}}_t^e, m^e, \mu^e, \boldsymbol{f}_t^{e,g})$ and recursively apply our dynamics D to yield the sequence of predicted latent vectors $\boldsymbol{l}_{t:t+T}$. We then apply our models M and T to recover our estimated object delta poses $\Delta \boldsymbol{q}_{t:t+H}^e$, tool contact points $\boldsymbol{p}_{t:t+H}^{tool}$, and tool contact forces $\boldsymbol{f}_{t:t+H}^{tool}$. We then apply the following supervised loss.

$$\mathcal{L}_{pred} = \sum_{i=t}^{t+H} ||\Delta \boldsymbol{q}_i^{e*} - \Delta \boldsymbol{q}_i^e||_2^2 + \alpha ||\boldsymbol{p}_i^{tool*} - \boldsymbol{p}_i^{tool}||_2^2 + \beta ||\boldsymbol{f}_i^{tool*} - \boldsymbol{f}_i^{tool}||_2^2 \tag{5.4}$$

Where ground truth values come from our dataset $\mathcal{D}$ and α and β are loss weighting terms.

To encourage accurate latent dynamics, we use future observations to encode ground truth latent rollouts. That is we derive $\boldsymbol{l}_{t:t+H}^*$ by applying E on the actual observations found when rolling out the system. We then encourage our recursively generated latents to match these values. We additionally regularize the scale of the predicted latent vectors.

[1] Our model can handle full SE(3) poses and actions but all examples considered are planar motions.

$$\mathcal{L}_{latent} = \sum_{i=t}^{t+H} \gamma ||\boldsymbol{l}_t^* - \boldsymbol{l}_t||_2^2 + \zeta ||\boldsymbol{l}_t||_2 \tag{5.5}$$

5.3.2 Model-Based Estimation of Frictional Object-Environment Contact

We next turn our attention to estimating the environment contacts. As outlined in Sec. 5.2, we can recover the set of K active contact points at a given time step $\boldsymbol{p}_t^{c,1}, ..., \boldsymbol{p}_t^{c,K}$ given our estimated object poses and the known object and environment geometries. The task is to find the contact forces at those points $\boldsymbol{f}_t^{c,1}, ..., \boldsymbol{f}_t^{c,K}$. To estimate these forces, we exploit our knowledge of Coloumb friction and quasi-static motion to formulate an optimization problem that recovers a set of valid contact forces.

Assuming the system is in quasi-static equilibrium, and given the contact force applied by the tool $\boldsymbol{f}_t^{tool}$, we know that the set of environment contact forces must be the solution to the following Quadratic Program (QP) optimization problem [87].

$$\begin{aligned} \min_{\boldsymbol{f}_t^{c,1},...,\boldsymbol{f}_t^{c,K}} \quad & \sum_{k=1}^{K} (\boldsymbol{f}_t^{c,k})^T U \boldsymbol{f}_t^{c,k} \\ \text{s.t. } & \boldsymbol{\tau}_t^g + J_t^{tool} \boldsymbol{f}_t^{tool} + \sum_{k=1}^{K} J_t^{c,k} \boldsymbol{f}_t^{c,k} = \mathbf{0} \\ & |\boldsymbol{f}_t^{c,k,x}| \leq \mu^e \boldsymbol{f}_t^{c,k,y} \quad k = 1, ..., K \\ & 0 \leq \boldsymbol{f}_t^{c,k,y} \quad k = 1, ..., K \end{aligned} \tag{5.6}$$

Here $\boldsymbol{\tau}_t^g$ are the gravitational terms, J_t^{tool} and $J_t^{c,k}$ are the contact Jacobians mapping contacts from the contact frame to generalized forces on the object, and we use x and y superscripts to indicate the tangential and normal force components in the contact frame, respectively. The first constraint expresses the quasi-static equilibrium of the rigid object. The second and third constraint enforces Coloumb friction, namely that forces must lie within the friction cone defined by μ^e and that

the forces can only apply pushing forces.

Were the system truly quasi-static and our tool force prediction perfectly accurate, we could utilize Eq. 5.6 to resolve our environment contact forces. In practice however, error in the model predictions, as well as our training data originating from a system that is not truly quasi-static can make Eq. 5.6 infeasible. To resolve this without sacrificing accuracy, we propose a relaxation of the QP.

$$\begin{aligned} \min_{\hat{\boldsymbol{f}}_t^{tool}, \boldsymbol{f}_t^{c,1}, ..., \boldsymbol{f}_t^{c,K}} \; & (\Delta \boldsymbol{f}_t^{tool})^T U (\Delta \boldsymbol{f}_t^{tool}) + \rho \sum_{k=1}^{K} (\boldsymbol{f}_t^{c,k})^T U \boldsymbol{f}_t^{c,k} \\ \text{s.t. } & \boldsymbol{\tau}_t^g + J_t^{tool} \hat{\boldsymbol{f}}_t^{tool} + \sum_{k=1}^{K} J_t^{c,k} \boldsymbol{f}_t^{c,k} = \mathbf{0} \\ & |\boldsymbol{f}_t^{c,k,x}| \leq \mu^e \boldsymbol{f}_t^{c,k,y} \quad k = 1, ..., K \\ & 0 \leq \boldsymbol{f}_t^{c,k,y} \quad k = 1, ..., K \end{aligned} \tag{5.7}$$

Here, $\Delta \boldsymbol{f}_t^{tool} = \boldsymbol{f}_t^{tool} - \hat{\boldsymbol{f}}_t^{tool}$ In our relaxation, we introduce a new decision variable $\hat{\boldsymbol{f}}_t^{tool}$ which we use to achieve force balance. We introduce a cost to minimize the difference between the new decision variable and our original estimated tool force $\boldsymbol{f}_t^{tool}$. We interpret the resulting $\hat{\boldsymbol{f}}_t^{tool}$ as the closest tool force to the estimated tool force, for which the system is in equilibrium. We utilize the resulting $\boldsymbol{f}_t^{c,1}, ..., \boldsymbol{f}_t^{c,K}$ as our estimate for the environment forces. We set ρ to a small value (1e-3) to prevent excessive drift from the estimated force while still realizing realistic contact forces.

5.4 Implementation

5.4.1 Data Collection

As described in Sec. 5.3.1.2, we train our learned model using supervised learning. This requires us to generate ground truth labels of the contact force. As such, we use simulation to generate example interactions and train our model. We use the Drake simulator [114], as it supports deformable-rigid contacts [39]. We attach a simple

Table 5.1: Object Parameter Variations

Parameter	Rectangle	Triangle	Pentagon
Geometry	$w \in [0.05, 0.1]$ $h \in [0.08, 0.15]$	$w \in [0.05, 0.1]$ $h \in [0.08, 0.15]$	$l \in [0.05, 0.1]$
Mass	$m \in [0.3, 0.43]$ kg		
Friction	$\mu^e \in \mathcal{B}([0.2, 0.8], \alpha = 2, \beta = 5)$		

deformable tool to the end of a gripper. We use a 46mm cube to match our real tool (Fig. 5.1) and utilize Drake's Finite Element Method simulation, setting Youngs Modulus to 1.1e4, Poisson's ratio to 0.1, and density to $30kg/m^3$. For the extrinsic object, we use three object primitives for which we generate variations: a rectangle, triangle and pentagon. All shapes are given a depth of 9cm. For each class, we perform geometric variations as well as varying the mass and friction parameters. The geometry and physical parameter variations used are shown in Table 5.1.

We use a simple scripted policy to collect the data in simulation. We randomize the starting location of the wrist so that initial contact with the object is randomized along the contacting edges. We then perform a simple forward push or a pivot. In the push case, we command a translation along the supporting plane. In the pivot case, we press into the object, then pivot by following an arc around a selected pivot vertex on the rigid object. In both cases, we then add noise to the action to add more variation to the motions. We execute these policies and record the resulting object motion, contact forces from the environment and the tool, and partial point clouds using a simulated camera. We use this to construct our training dataset $\mathcal{D}$.

5.4.2 Model Details

We collect a training dataset by collecting 5000 push and 5000 pivot trajectories in our simulation with each geometry class we used. Each trajectory samples new geometric and physical properties and runs up to 40 time steps or until the object

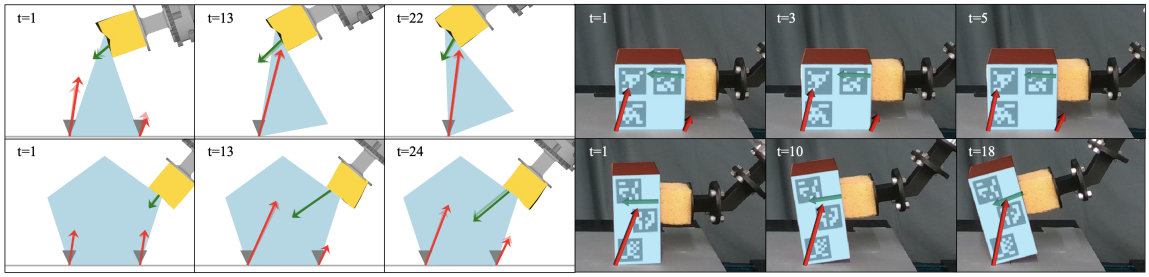


Figure 5.3: Qualitative Results. **Left:** We compare our methods one-step predictions (solid) to the ground truth (semi-transparent) object motion, tool force, and environment forces across several manipulation trajectories. The friction cones for each environment contact is shown in gray at the environment point of contact. Our method shows high-fidelity predictions for a) varying object geometries and physical parameters, and b) different interaction primitives (pivoting vs. pushing). **Right:** We show one-step predictions for real robot executions. The predictions match the observed motion and show qualitatively realistic forces (see Sec. 5.5.2 for quantitative results).

breaks contact with the tool. We filter the transitions to only include examples in contact. We additionally collect a validation dataset of 500 push and 500 pivot trajectories in the same manner. This yields a final training dataset of 262,780 train transitions and 66,199 validation transitions. During training, we use an action horizon of 4. We implement our network in PyTorch and solve our CQP using CvxPy. Our model is trained for 50 epochs and we use the validation dataset to select the best model. For all experiments, we set our constants from Sec. 5.3 as $T = 4, \alpha = 1.0, \beta = 1.0, \gamma = 0.1, \zeta = 1e-3$.

5.5 Experiments

5.5.1 Simulated Results

We start by examining our method's performance on a test dataset. We follow the same procedure outlined in Sec. 5.4 and collect 500 push and 500 pivot trajectories per object class. We examine our methods ability to predict tool contact information and object motion and forces.

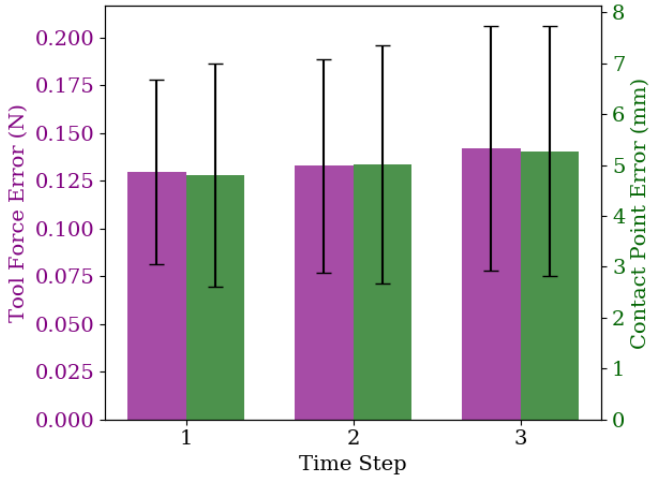
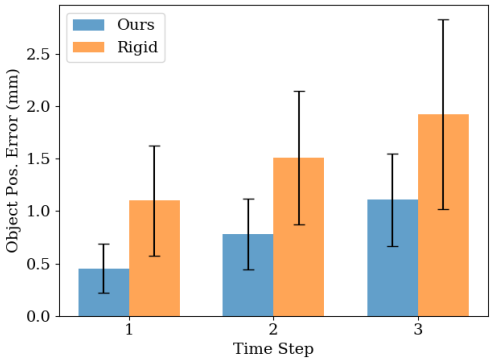


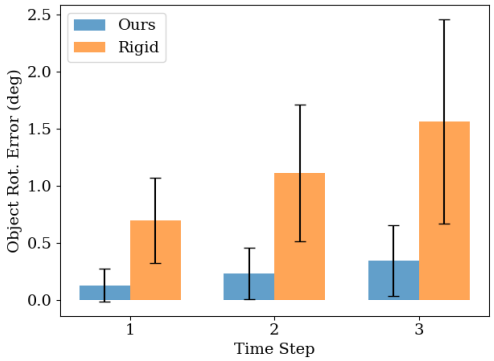
Figure 5.4: Tool Contact Force (purple) and Contact Point Location (green) error for our proposed model, plotted by prediction horizon time step. Error bars indicate one half standard deviation.

Table 5.2: Real World 1-Step Object and Force Error (↓)

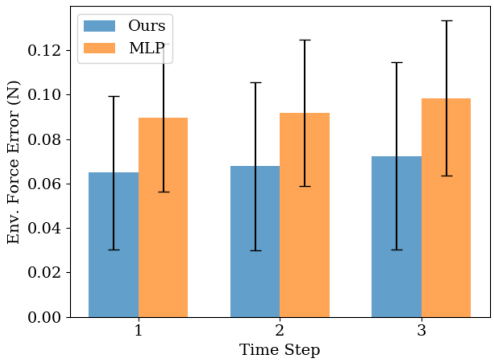
Metric		Pos. Error (mm)	Rot. Error (deg)	Force Error (N)
Blk. 1	Pivot	0.364 (0.128)	0.262 (0.125)	0.745 (0.292)
	Push	1.163 (0.420)	1.144 (0.613)	0.977 (0.445)
Blk. 2	Pivot	0.466 (0.196)	0.071 (0.054)	1.033 (0.382)
	Push	2.044 (0.211)	0.069 (0.032)	1.180 (0.160)



(a) Object Position Error (↓)



(b) Object Rotation Error (↓)



(c) Extrinsic Force Error (↓)

Figure 5.5: Baseline comparison of our method on test simulated interactions. For (a) and (b) we compare to a Rigid baseline that predicts block motion as if rigidly attached to the tool. For (c) we compare our model-based optimization for extrinsic contact recovery to directly predicting it from an MLP. Error bars indicate one half standard deviation.

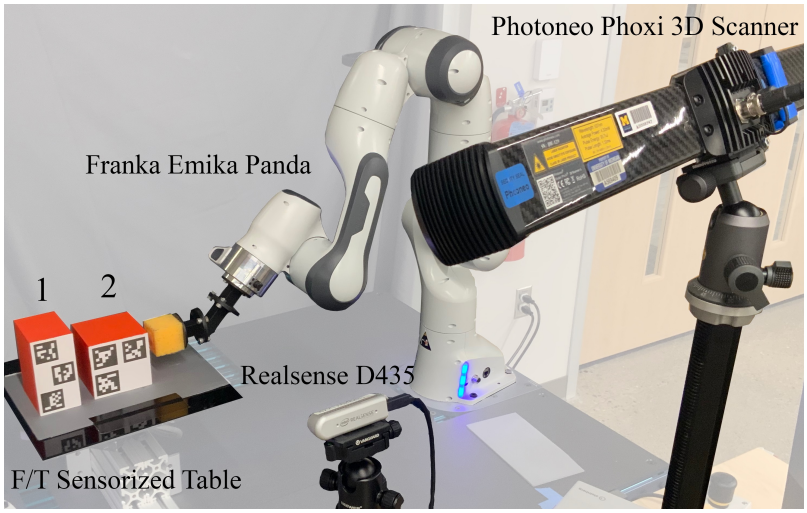


Figure 5.6: Our real robot setup, with the test objects. We use the Realsense D435 to track the block pose and the Phoxi 3D Scanner to recover partial pointclouds of the tool. We use a Force/Torque sensorized table to indirectly evaluate our force predictions.

5.5.1.1 Tool Contact Prediction

We report our method's mean performance over a prediction horizon of three steps for tool contact point and contact force accuracy across all our simulated test trajectories in Fig. 5.4. For all interactions and across our prediction horizon, our method showed the ability to accurately estimate future contact points between the deforming tool and extrinsic block, to within 6mm on average (against a tool of size 46mm), and showed the ability to estimate future contact forces to within 0.2N on average.

5.5.1.2 Object Motion Prediction

To benchmark our method's object motion performance, we introduce a *Rigid* baseline which assumes that the block moves "rigidly" with respect to the tool. This assumes that the deformation of the tool is negligible for the object motion and doesn't model collisions.

We report our method and the rigid method's object position and rotation accu-

racy in Figs. 5.5a and 5.5b. We show that across all interaction types, our method outperformed the *Rigid* baseline, with sub-millimeter position accuracy and less than 0.5 degrees of rotational error on average across the full prediction horizon.

5.5.1.3 Environment-Object Force Prediction

A key part of our proposed method is utilizing our physical priors and the quasi-static assumption to solve for the force between the environment and extrinsic object. We hypothesized that this allows for accurate modeling without requiring additional training effort. To benchmark this approach, we create a variation which replaces the CQP with another MLP following the form of our other network output heads. In particular, it takes in the latent state $\boldsymbol{l}_t$ and directly predicts the contact forces $\boldsymbol{f}_t^{c,1}, ..., \boldsymbol{f}_t^{c,K}$. In practice, the baseline predicts forces at set candidate points on the object and we use the predicted object motion to detect which points lie in contact.

We report our method and baseline method's environment-object force accuracy across all interaction types in Fig. 5.5c. We find that our proposed method outperforms the baseline, yielding the best average force prediction across all scenarios. Our method consistently performed to within 0.08N of error on average, across the prediction horizon. Our results indicate that our proposed CQP method for recovering environment forces outperforms direct learning.

Finally, Fig. 5.3 (Left) shows qualitative examples of our methods predictions on simulated data with different objects and actions (pivoting and pushing). The predicted object motion, tool contact and force, and environment forces are consistent with the ground truth, as indicated by the significant overlap with the ground truth (shown as semi-transparent).

5.5.2 Real Robot Results

We demonstrate our methods performance on a real robot system. We use a Franka Emika Panda robot and interact with two rectangular objects. Our setup and the test objects are shown in Fig. 5.6. We collect 5 push and 5 pivot trajectories with each object. We use a Realsense D435 and AprilTags to track the object state

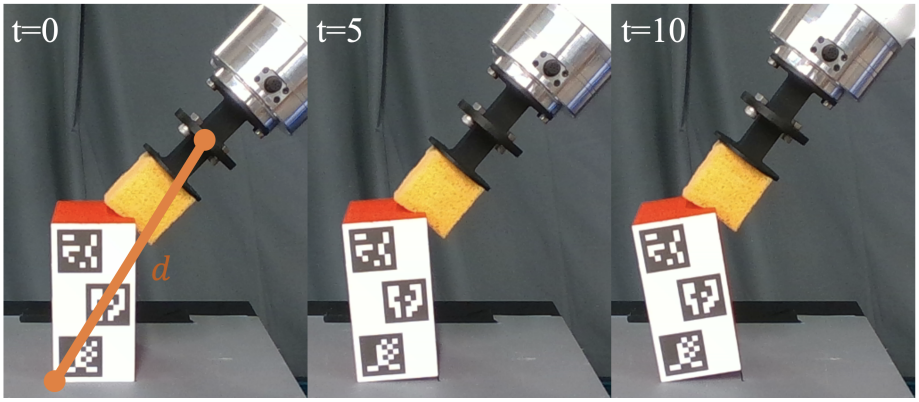


Figure 5.7: Our pivot force tracking task, with the decision variable d shown in leftmost frame.

and a Photoneo Phoxi 3D Scanner sensor and the Segment Anything Model [60] to get our segmented partial pointcloud of the tool. Finally, we mount an ATI Gamma Force/Torque sensor under the tabletop. This allows us to compare the cumulative force experienced by the table via the extrinsic contacts, which we compare to our predicted extrinsic contacts.

In Table 5.2, we show our one-step object motion prediction and extrinsic force prediction on the real data. Despite sensor noise and the sim-to-real gap, we are still able to predict object motion to within roughly 2mm and 2 degrees error. Our extrinsic contact prediction is within roughly 1N on average. In Fig. 5.3 (Right), we show full qualitative predictions of block motion and forces predictions. We see that our method yields qualitatively realistic force estimates given real sensor feedback on the physical system.

5.5.3 Force Tracking

Finally, we demonstrate a task utilizing our force predictions. Regulating force on the extrinsic object can be important in cases where the object or environment are fragile, such as when handling food, or when the robot or end effector can break or tear. Here, we demonstrate force tracking using our proposed extrinsic contact estimates. We use Block 1 from Sec. 5.5.2 and execute a pivoting motion, pivoting

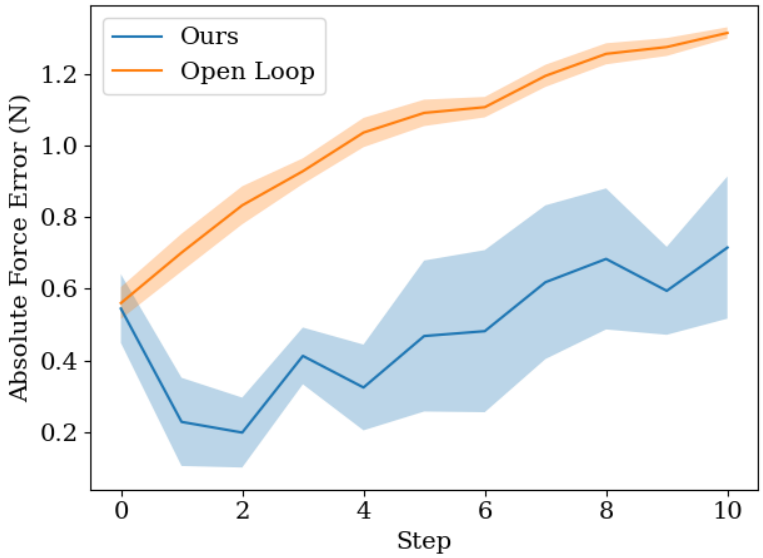


Figure 5.8: We plot, over five trials, the evolution of the absolute force magnitude error as a function of the trajectory step. Our method is able to better track the desired force, compared to a method without force feedback executing a similar pivot.

around the contact point furthest from the robot. After an initial press into the object, we have the robot follow an arc, such that the robot wrist is a specified distance d from the pivot point. We setup a simple greedy controller which samples various distances d for the next step, thereby allowing it to press harder or release as it pivots. We use our proposed model and predict the motion and forces for the sampled actions (with a single-action horizon). We then select the action with the resulting cumulative force magnitude closest to the desired force setting ν. We set the target force magnitude ν to be the gravitation force given the mass of the block plus 1.5N. We can then use the F/T sensor mounted on the environment to determine approximately how well we tracked the desired force. An example execution is shown in Fig. 5.7.

We ran five trials using this simple greedy controller to track the desired force profile. We then executed our baseline, which started at the same starting configuration, but kept the initial d fixed. We used our environment Force/Torque sensor to

measure the actual composite force transfer and plot how the errors evolved over the trials in Fig. 5.8. Our method is able to better track the desired force magnitude, while the open loop plan gradually drifts further from the desired force profile.

5.6 Discussion

We present a method capable of jointly recovering object motion along with tool and environment contact information during dexterous deformable manipulations. We found that our first-principles approach outperformed learning methods when recovering environment forces. In future work, we hope to leverage this work for the planning of dexterous deformable motion, where we aim to utilize force estimates to reason about contact modes and force targets.

A limitation of our method is the assumption of quasi-static motion. While this is a common assumption made during dexterous manipulation [46, 87], we may wish for more dynamic motions during certain tasks [118]. A potential direction to investigate is incorporating predicted forces into the underlying rigid body dynamics [13].

We have limited our investigation here to planar motions of the extrinsic object. While our method can be extended to full SE(3) motions, this requires extending the data collection in simulation to consider those motions and extending our CQP to consider 3D forces.

Additionally, our method assumes knowledge of the geometry of the extrinsic object and the environment, which may not be readily available. One potential remedy would be utilizing a shape reconstruction algorithm to estimate these geometries [91].

Finally, our method relies on supervised learning, requiring simulated data to train and successful transfer from simulation to the real robot. Using object motion consistency with predicted forces could be a way to directly train on real system rollouts [92].

CHAPTER VI

Simultaneous Extrinsic Contact and In-Hand Pose Estimation via Distributed Tactile Sensing

This chapter proposes a methodology for inferring the in-hand pose and extrinsic contact of a rigid object grasped via compliant tactile sensors. Our method utilizes the deformation of the sensors to infer the local contact geometry, in-hand displacement, and in-hand wrench via learned tactile models. We then incorporate this feedback into a factor graph, along with the physical constraints of contact, to efficiently resolve the global pose and extrinsic contacts. We demonstrate our method on a peg insertion task, across a variety of geometries. Our results demonstrate that jointly utilizing the geometric and wrench feedback enables precise contact estimation and can be used to resolve ambiguities present in the local observations.

6.1 Introduction

Prehensile manipulation tasks require precise reasoning over grasped object poses and contacts. Even small errors in object pose can prevent proper insertion of an object [135] or yield an undesired placement [67]. Effective tool use similarly relies on precise application of contacts and forces [45]. We require systems capable of simultaneously estimating where a grasped object is and how it is in contact with the environment.

Methods for object pose estimation largely rely on visual feedback [100, 116]. Prehensile manipulation, however, often suffers visual occlusions due to the grasp

and/or environment and sensor noise can corrupt estimation results. Distributed visuo-tactile sensors are a promising form of feedback for prehensile manipulation [2, 136] that can complement visual sensing [110]. These sensors utilize a compliant material that deforms on contact and a camera to observe these deformations. This provides local geometric information where the sensor contacts the object. While these geometric observations can resolve in-hand pose for small or highly featured objects [69], for many objects, the local nature of these observations still results in significant ambiguities [4] and must be paired with visual feedback for better convergence [3].

The compliance of the visuo-tactile sensor can also provide a signal of extrinsic contact, between the grasped object and the environment. How the sensor deforms when an extrinsic contact is made is indicative of the force experienced in the grasp [108], which in turn can be used as a signal to infer extrinsic contacts [57, 58]. Precisely identifying contact is still challenging, however, as multiple contact points can yield near-identical force profiles in the grasp.

In this work, we investigate how *jointly* estimating the object pose and extrinsic contacts can help resolve ambiguities and address noise. Intuitively, these two are tightly coupled, as the object pose determines which contacts are possible, and conversely, where contact is being made constrains object poses. We enforce non-penetration with the (assumed known) environment and ensure that the estimated extrinsic contact is kinematically feasible (i.e., lies on the surface of the object and environment) and yields forces consistent with tactile feedback. The result is a system that enforces consistency across multiple forms of feedback (geometry and forces) while enforcing physical realism (see Fig. 6.1). This reduces the space of acceptable solutions, resulting in accurate estimation.

Our contributions are as follows:

- A set of *object agnostic* models for extracting geometric and force signals for our estimator from tactile feedback.
- A factor graph method for simultaneous extrinsic contact and in-hand pose estimation, *TacGraph*, which jointly enforces geometric consistency, force bal-

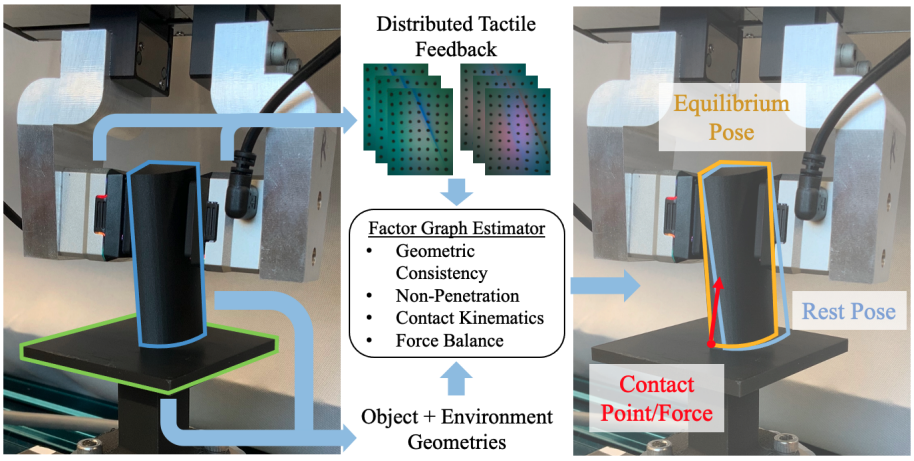


Figure 6.1: We propose *TacGraph*, an estimator that exploits geometric consistency, force balance, non-penetration, and contact kinematics to jointly estimate the object pose and extrinsic contacts.

ance, non-penetration, and contact kinematics.

- Demonstration of our method on a physical system and comparison to baselines.

6.2 Problem Formulation

Our goal is to estimate the pose of a grasped object and the extrinsic contact that the grasped object makes with a known environment. We make the following assumptions:

- The object is rigid and has a known geometry $\mathcal{M}_o$.
- The environment is rigid, has a known geometry $\mathcal{M}_e$, and is static.
- The grasp is elastic, meaning it complies due to external contact, but the object returns to the same location when the extrinsic contact is removed.
- We assume that the extrinsic contact can be described as a single summary contact point and force, and does not induce a torque.

We utilize feedback from a pair of Gelsight tactile sensors [69], located at the grasp, and robot proprioception. For some experiments, we additionally provide visual feedback from an external camera. Our inputs are:

- (Optional) $\boldsymbol{P}^V \in \mathbb{R}^{N_V \times 3}$ - initial partial point cloud of object from external camera.
- $\boldsymbol{g}_t \in SE(3)$ - gripper pose at time t.
- $\boldsymbol{I}_t^L, \boldsymbol{I}_t^R \in \mathbb{R}^{H \times W \times 3}$ - Gelsight tactile images from the left and right gripper fingers at time t.
- $\mathcal{M}_o, \mathcal{M}_e$ - triangle mesh geometry of the grasped object and environment.

Our goal is to estimate the object pose as well as the contact point and contact force. Our desired outputs are:

- $\boldsymbol{o}_t \in SE(3)$ - object pose at time t.
- $\boldsymbol{c}_t \in \mathbb{R}^3$ - contact point at time t.
- $\boldsymbol{f}_t \in \mathbb{R}^3$ - contact force at $\boldsymbol{c}_t$ at time t.

6.3 Method

We propose our method for simultaneously estimating in-hand object pose and extrinsic contacts from tactile feedback. First, we propose a set of object agnostic tactile models for extracting useful geometric and force signals to be used downstream. Second, we propose TacGraph, our factor-graph based estimator. An overview of our method is shown in Fig. 6.2.

6.3.1 Tactile Models

Distributed visuo-tactile sensors provide several forms of feedback that are valuable for contact-rich prehensile manipulation. First, we gain local geometric information from the grasp. Second, the deformation allowed by the sensor upon an external

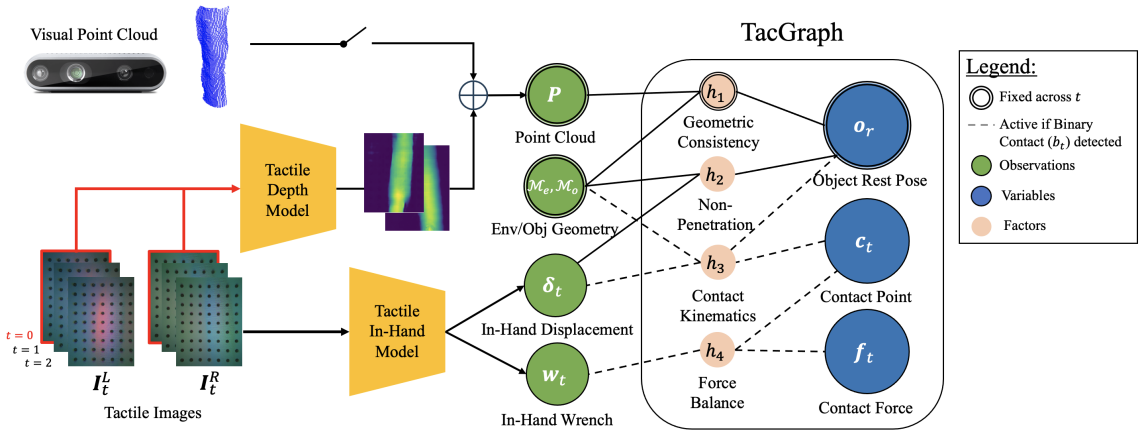


Figure 6.2: Overview of our proposed methodology. First, we propose a set of tactile models which process raw distributed tactile observations into geometric and force feedback terms. Second, these terms are utilized, along with known geometries and optionally with visual feedback, in a factor-graph based estimator, *TacGraph*, which estimates the object pose and extrinsic contacts. Observations, variables, and factors that are fixed (non time-varying) are double circled. Factors only active when contact is detected are connected with dashed lines.

contact provides information about a) the in-hand displacement of the object and b) the force applied on the object. We propose a set of *object-agnostic* models that extracts these feedback terms from the raw tactile images.

We use the initial tactile observations (i.e., at $t = 0$), to predict a tactile point cloud $\boldsymbol{P}^T \in \mathbb{R}^{N_T \times 3}$. We train a model that takes in a tactile image (either from the left or right sensor) and yields a depth image $D \in \mathbb{R}^{H \times W}$. We threshold the depth to determine which pixels are in contact and de-project the depths to yield our final point cloud $\boldsymbol{P}^T$.

As the grasped object makes contact with its environment, the compliance of the Gelsight sensor means the object moves in the grasp. In order to accurately recover the object pose, we must then reason about this displacement. We train an in-hand displacement model which takes in both tactile images and predicts $\boldsymbol{\delta}_t \in SE(3)$, the relative displacement of the object in-hand.

Finally, we wish to determine the in-hand wrench experienced at the grasp, which

is helpful for contact localization and for resolving the contact force [77, 105]. We train an additional model that takes in both tactile images and predicts $\boldsymbol{w}_t \in \mathbb{R}^6$, the wrench experience at the grasp. Additionally, we can utilize our predicted wrench to estimate whether the system is in contact at time t: $b_t = ||\boldsymbol{w}_t||_\Sigma > \epsilon$.

6.3.2 TacGraph

Our goal is to incorporate our tactile feedback terms along with the physical constraints of contact to determine the *maximum a posteriori* (MAP) estimate of our variables. As stated in Sec. 6.2, we aim to recover the object pose and extrinsic contact information at each time t.

We assume that the grasp is elastic, that is, the object moves in-hand due to sensor compliance upon the application of an external force, and returns to the same rest pose when the external force is removed. As described in Sec. 6.3.1, we estimate this in-hand displacement $\boldsymbol{\delta}_t$ from the tactile feedback. As such, we perform a change of variables and infer $\boldsymbol{o}_t$ by estimating a single *rest pose* $\boldsymbol{o}_r \in SE(3)$ and applying the predicted in-hand displacement at a given time t.

$$\boldsymbol{o}_t = \boldsymbol{g}_t{}^g \boldsymbol{\delta}_t{}^g \boldsymbol{o}_r \tag{6.1}$$

We use the prefix g to indicate that the displacement and rest pose are both expressed in the gripper pose frame, but drop the frames from here on.

A natural way to describe our MAP estimation problem is as a Factor Graph, a bi-partite graph of variables and the factors which describe the relationship between variables [23]. Assuming Gaussian noise models on the factors, the MAP estimation becomes a sum of nonlinear least-squares.

$$\boldsymbol{o}_r^*, \boldsymbol{c}_{1:T}^*, \boldsymbol{f}_{1:T}^* = \underset{\boldsymbol{o}_r, \boldsymbol{c}_{1:T}, \boldsymbol{f}_{1:T}}{\arg\min}\ H(\boldsymbol{o}_r, \boldsymbol{c}_{1:T}, \boldsymbol{f}_{1:T}) \tag{6.2}$$

$$H(\cdot) = ||h_1(\boldsymbol{o}_r)||^2_{\Sigma_1} + \sum_{t=1}^{T} \{||h_2(\boldsymbol{o}_r)||^2_{\Sigma_2} + \mathbf{1}[b_t](||h_3(\boldsymbol{o}_r, \boldsymbol{c}_t)||^2_{\Sigma_3} + ||h_4(\boldsymbol{c}_t, \boldsymbol{f}_t)||^2_{\Sigma_4})\} \tag{6.3}$$

The structure of the factor graph is shown in Fig. 6.2 in the box. Note, b_t is an observation (see Sec. 6.3.1), not a variable being solved for, and is used to enable contact-specific factors only when the object is actively in contact. Each factor h_i has an accompanying covariance matrix Σ_i which is empirically selected. The resulting MAP problem can be solved efficiently using the iSAM2 solver [52].

6.3.3 Factors

6.3.3.1 Geometric Consistency

Our first factor ensures the estimated rest pose $\boldsymbol{o}_r$ is consistent with the observed object point cloud at time $t = 0$. This point cloud $\boldsymbol{P}$ includes the tactile point cloud $\boldsymbol{P}^T$ and, if available, a visual point cloud $\boldsymbol{P}^V$. As we assume that the grasp is elastic, it is sufficient to ensure the rest pose is geometrically consistent, without enforcing at each time step.

We define our factor error function as the signed-distance value of each point of the observed point cloud to the surface of the object:

$$h_1(\boldsymbol{o}_r; \boldsymbol{P}, \mathcal{M}_o) = \boldsymbol{S} \tag{6.4}$$

Here, $\boldsymbol{S}$ is a vector containing the signed distance value of each point to the surface.

$$\boldsymbol{S}_i = SDF(\boldsymbol{o}_r^{-1} \boldsymbol{P}_i^T | \mathcal{M}_o) \tag{6.5}$$

SDF is the signed distance value computed using the geometry $\mathcal{M}_o$. We transform each point into the object frame using the estimated $\boldsymbol{o}_r$. Both the SDF and the SDF gradients can be efficiently computed using a triangle mesh geometry [141].

6.3.3.2 Non-Penetration

We ensure the estimated pose at each time-step does not yield penetration with the environment geometry. We apply Eq. 6.1 to get the estimated object pose, considering the observed gripper pose and in-hand displacement, as well as the current

estimate of the rest pose.

Given the known geometries $\mathcal{M}_e, \mathcal{M}_o$ and the object pose, we then compute a penetration check. In practice, we approximate our object geometry for this factor as a point cloud $\boldsymbol{P}^{obj} \in \mathbb{R}^{N_P \times 3}$ by sampling points on the surface of the geometry $\mathcal{M}_o$. Our factor error function then returns for each point the distance to the surface of the environment, if the point is inside of the environment geometry, thus indicating penetration.

$$h_2(\boldsymbol{o}_r | \mathcal{M}_o, \mathcal{M}_e, \boldsymbol{g}_t, \boldsymbol{\delta}_t) = \boldsymbol{S} \tag{6.6}$$

Once again $\boldsymbol{S}$ is a vector which contains the SDF terms when a point is in penetration.

$$\boldsymbol{S}_i = \min(0, SDF(\boldsymbol{g}_t \boldsymbol{\delta}_t \boldsymbol{o}_r \boldsymbol{P}_i^{obj} | \mathcal{M}_e)) \tag{6.7}$$

Each point is transformed into the world frame before evaluating the SDF against the environment geometry.

6.3.3.3 Contact Kinematics

When we detect that contact has been made b_t, we add additional variables and factors to the graph at that time step to estimate the contact location and force. First, we address the kinematic constraints on the contact. Namely, the contact point $\boldsymbol{c}_t$ should lie on the surface of *both* the environment and object geometries. This amounts to the following error function:

$$h_3(\boldsymbol{o}_r, \boldsymbol{c}_t; \mathcal{M}_e, \mathcal{M}_o) = \begin{bmatrix} SDF(\boldsymbol{c}_t | \mathcal{M}_e) \\ SDF((\boldsymbol{g}_t \boldsymbol{\delta}_t \boldsymbol{o}_r)^{-1} \boldsymbol{c}_t | \mathcal{M}_o) \end{bmatrix} \tag{6.8}$$

We again apply Eq. 6.1 to derive the object pose and invert to map the point to the object frame.

6.3.3.4 Force Balance

In Sec. 6.3.1, we showed how we can recover the wrench experienced at the grasp $\boldsymbol{w}_t$ from our tactile sensors. This in-hand wrench is the result of the application of

an external contact force. As such, we add a factor which ensures that the in-hand wrench implied by the contact matches our $\boldsymbol{w}_t$ observation.

To estimate the in-hand wrench applied by a contact force $\boldsymbol{f}_t$ applied at $\boldsymbol{c}_t$, we apply the contact Jacobian to map the force from the contact point to the gripper frame.

$$\hat{\boldsymbol{w}}_t = J(\boldsymbol{g}_t^{-1}\boldsymbol{c}_t)\ \boldsymbol{f}_t \tag{6.9}$$

We can then define our factor error function as the difference between the observed and predicted in-hand wrench.

$$h_4(\boldsymbol{c}_t, \boldsymbol{f}_t; \boldsymbol{g}_t) = \hat{\boldsymbol{w}}_t - \boldsymbol{w}_t \tag{6.10}$$

6.3.4 Inference

Solving for the MAP estimate with our proposed TacGraph enables us to find the most likely rest pose $\boldsymbol{o}_r$ and contact points $\boldsymbol{c}_{1:T}$ and forces $\boldsymbol{f}_{1:T}$. We then apply Eq. 6.1 to resolve our object poses at each timestep $\boldsymbol{o}_{1:T}$.

The non-linear nature of our system means that the minimization performed by iSAM2 in Eq. 6.2 is subject to local minimums. As such, we initialize a set of particles to represent possible rest poses $\{\boldsymbol{o}_r^1, \boldsymbol{o}_r^2, \ldots, \boldsymbol{o}_r^K\}$. We then perform our inference procedure on each rest pose particle separately, resulting in K solutions. We utilize our factor graph cost in Eq. 6.3 to score each solution particle, and select our final solution accordingly:

$$\boldsymbol{o}_r^*, \boldsymbol{c}_{1:T}^*, \boldsymbol{f}_{1:T}^* = \arg\min_k H(\boldsymbol{o}_r^k, \boldsymbol{c}_{1:T}^k, \boldsymbol{f}_{1:T}^k) \tag{6.11}$$

We utilize Iterative Closest Point (ICP) to match our initial rest pose particles to the available point cloud $\boldsymbol{P}$. We apply a sampling heuristic based on likely grasp directions [4] to ensure that the rest pose particles cover the space of possible initializations well.

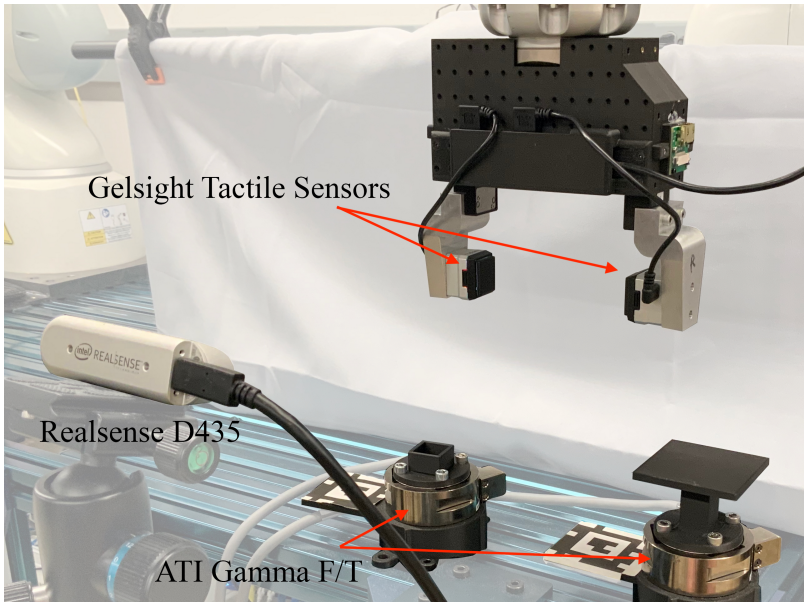


Figure 6.3: Our experimental setup. On the left ATI Gamma is an example object fixture. On the right ATI Gamma is the sensorized press surface we use for data collection/experiments.

6.4 Implementation

6.4.1 Experimental Setup

Our experimental setup is shown in Fig. 6.3. We use a WSG-50 parallel jaw gripper attached to a KUKA LBR iiwa Med R820 robot. We attach a GelSight Mini sensor to each finger. We have an external Intel Realsense D435 sensor for visual feedback. We utilize Segment Anything Model for visual point cloud segmentation [60]. Two ATI Gamma Force/Torque sensors are mounted in the scene and used to attach fixtures and environment geometries. We utilize these extrinsic F/T sensors *only* for training our tactile models and evaluation; our method does not utilize these sensors' feedback. Our tactile models are implemented using Pytorch and our factor graph is implemented using GTSAM [22].

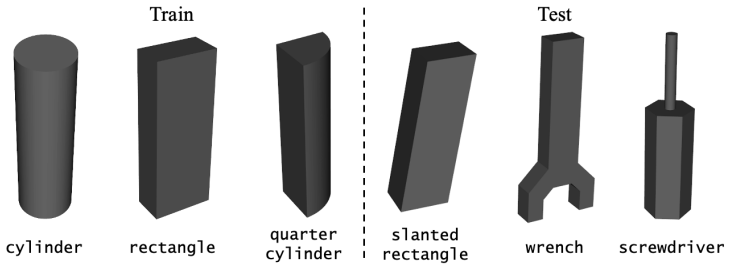


Figure 6.4: Train/Test objects used in our experiments.

6.4.2 Tactile Model Training

We train our tactile models described in Sec. 6.3.1 utilizing datasets collected on our physical system, using three different train objects (see Fig. 6.4). All models are convolutional models trained via supervised learning.

6.4.2.1 Tactile Depth

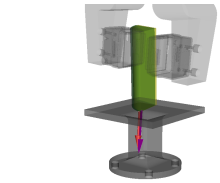
To generate tactile images with ground truth depth labels, we rigidly attach fixtures of known geometry to our environment and apply random grasps to the fixture, adjusting the position and orientation of the grasp. Then using the known fixture geometry, we render corresponding depth images.

6.4.2.2 In-Hand Displacement

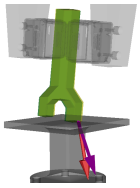
To generate in-hand displacement labels, we rigidly attach fixtures of known geometry to our environment. Following Kim and Rodriguez [58], we randomly grasp the fixture, then apply small delta motions to the end effector. These delta motions become the in-hand displacement labels $\boldsymbol{\delta}_t$ associated with the tactile images.

6.4.2.3 In-Hand Wrench

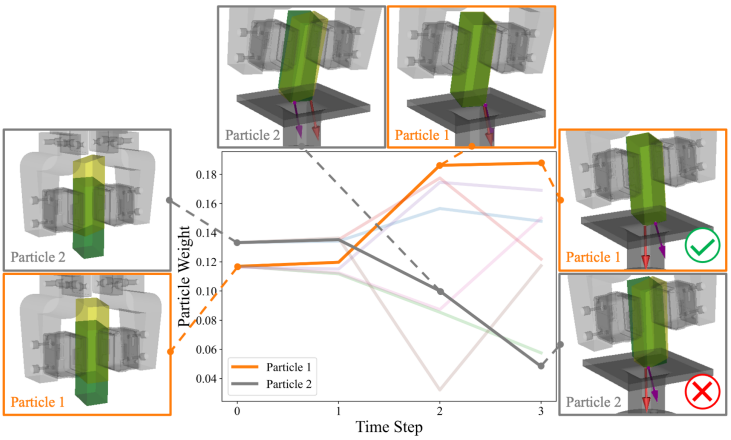
To generate in-hand wrench labels, we utilize the tabletop fixture shown in Fig. 6.3 and grasp objects, with randomized grasps, from a fixture before performing a random poke into the tabletop. We then use the environment mounted F/T sensor to



(a) `quarter cylinder`



(b) `wrench`



(c) Particle Weight Progression

Figure 6.5: We show comparison of predicted and ground truth object pose, and predicted and ground truth extrinsic contact. (a-b) final qualitative TacGraph estimates for two different objects. (c) progression of particle weights within TacGraph for a tactile-only inference. Two highlighted particles indicate how initial orientations can be filtered based on the contacts made to correctly select particle solutions. Figure best viewed in color.

get the wrench caused by the poke $\boldsymbol{w}_t^e$. We transform this wrench into the grasp frame, recovered from the robot proprioception, to label the wrench.

6.5 Experiments

6.5.1 Baselines

- **ICP:** Iterative Closest Point enforces geometric consistency in the object pose. To estimate a contact point and force, we use our tactile models to detect when contact occurs as well as the force. When in contact, we select the point on the object closest to the environment as our contact point, assigning the predicted force from the tactile model.

- **CHSEL [141]:** CHSEL adds free-space reasoning as well as a Quality Diversity

(QD) to gradient-based pose registration. CHSEL thus considers geometric consistency and non-penetration via free points. At each timestep, we sample points from the surface of the environment, transform into the grasp frame, and add as free points in the CHSEL optimization, to ensure the object does not collide. We follow the same procedure as ICP to label extrinsic contact. We run 2 CHSEL iterations per update.

- **SCOPE [106]:** SCOPE utilizes non-penetration, contact kinematics, and force balance for pose and contact estimation via a dual particle filter method. Object pose particles are scored based on non-penetration and their contacts, determined by a Contact Particle Filter (CPF) [77] associated for each object pose particle to estimate the most likely contacts. We modify SCOPE to fit our setup by removing environment pose filtering and replace the F/T sensing with our learned tactile model. Note, this method does not consider geometric consistency. We run two versions of SCOPE. SCOPE (v1) uses 8 pose and 30 contact particles, with 3 iterations every update. SCOPE (v2) uses 64 pose and 200 contact particles, with 20 iterations every update.

All methods are initialized with the same procedure as TacGraph for fair comparison.

6.5.2 Pose and Contact Estimation

We test pose and contact estimation performance on the objects shown in Fig. 6.4. For the environment geometry, we use the geometry shown on the right ATI Gamma F/T sensor in Fig. 6.3. Each object is grasped from a fixture, randomizing the in-hand grasp pose. We perform a fixed set of three angled pokes into the environment. We get sensor feedback before poking and once during each poke, when contact is made. We run inference for each method under two conditions: Vision + Tactile and Tactile. We apply each method iteratively on each subsequent timestep of data.

	Methods	Train			Test		
		cylinder	rectangle	quarter cylinder	slanted rectangle	wrench	screwdriver
Vision + Tactile	ICP	3.08 (0.65)	3.68 (0.73)	3.09 (1.08)	4.86 (4.81)	2.19 (0.76)	3.33 (1.25)
	CHSEL	1.04 (0.36)	1.97 (0.39)	1.23 (0.51)	12.28 (5.11)	1.66 (0.88)	2.44 (0.83)
	SCOPE (v1)	5.96 (2.83)	7.11 (2.70)	16.50 (2.69)	15.56 (4.36)	5.86 (1.68)	7.34 (2.53)
	SCOPE (v2)	4.77 (2.50)	2.59 (1.22)	14.04 (5.55)	13.28 (4.79)	3.45 (1.57)	5.63 (3.11)
	TacGraph	0.68 (0.23)	1.48 (0.14)	1.34 (0.34)	1.05 (0.32)	0.92 (0.28)	1.62 (0.22)
Tactile	ICP	17.80 (13.86)	17.15 (10.66)	23.70 (7.58)	20.10 (6.94)	16.77 (6.03)	12.32 (7.87)
	CHSEL	30.33 (12.01)	28.55 (17.20)	28.40 (6.86)	20.71 (7.09)	13.61 (8.16)	29.42 (17.20)
	SCOPE (v1)	9.01 (6.49)	12.26 (12.38)	16.13 (10.33)	18.44 (5.08)	9.89 (7.39)	8.32 (3.85)
	SCOPE (v2)	4.16 (1.00)	4.37 (2.78)	10.74 (5.37)	12.90 (4.94)	4.75 (3.77)	11.03 (5.31)
	TacGraph	2.96 (2.54)	1.29 (2.24)	8.45 (5.70)	7.58 (6.18)	0.78 (0.40)	1.54 (1.29)

Table 6.1: Quantitative results for pose estimation. Mean and std. dev. of Averaged 3D Distance (ADD) in mm reported; best for each method in Vision+Tactile and Tactile regime highlighted.

6.5.2.1 Metrics

We determine the ground truth pose of the object using the fixture (tolerance of ∼1mm). We use the Average 3D Distance metric [4] using 1000 points as our object pose metric.

For the extrinsic contact, we utilize the F/T sensor mounted under the environment. We compute the ground truth contact by sampling a set of candidate contact points on the environment surface $\boldsymbol{C} \in \mathbb{R}^{L\times 3}$. We then identify the ground truth by solving the following minimization:

$$\boldsymbol{c}_t^* = \arg\min_l ||\boldsymbol{\tau}_t^{FT} - \boldsymbol{C}_l \times \boldsymbol{f}_t^{FT}||_2 \tag{6.12}$$

$\boldsymbol{f}_t^{FT}$ and $\boldsymbol{\tau}_t^{FT}$ are the force and torque at the sensor. The ground truth force is then $\boldsymbol{f}_t^* = \boldsymbol{f}_t^{FT}$. We compute the euclidean errors on the contact point and force estimate.

6.5.2.2 Results

We report our pose estimation results in Table 6.1. When vision is available, ICP can perform well, but its performance is subject to sensor noise. CHSEL can

	Methods	Train			Test			Overall
		cylinder	rectangle	quarter cylinder	slanted rectangle	wrench	screwdriver	
Vision + Tactile	ICP	3.43 (0.80)	4.77 (2.10)	4.33 (2.48)	5.30 (5.68)	2.61 (1.34)	4.38 (1.17)	4.14 (2.92)
	CHSEL	1.84 (1.02)	3.15 (2.72)	2.79 (2.62)	12.59 (9.44)	2.44 (1.41)	3.15 (0.55)	4.33 (5.62)
	SCOPE (v1)	5.43 (2.33)	9.59 (8.45)	6.33 (3.09)	10.97 (4.51)	4.84 (3.53)	36.51 (14.33)	12.28 (13.28)
	SCOPE (v2)	4.64 (2.48)	8.71 (8.52)	5.33 (3.38)	9.48 (5.35)	3.75 (1.87)	27.10 (17.32)	9.83 (11.59)
	TacGraph	2.78 (1.31)	4.79 (4.63)	1.93 (1.06)	7.03 (6.11)	2.29 (1.13)	3.29 (1.20)	3.68 (3.72)
Tactile	ICP	18.20 (13.49)	17.38 (10.98)	18.65 (8.92)	20.71 (8.19)	17.06 (5.54)	13.20 (7.70)	17.53 (9.75)
	CHSEL	25.16 (14.40)	32.66 (14.68)	22.54 (9.46)	21.15 (10.17)	13.54 (8.08)	31.41 (18.30)	24.41 (14.52)
	SCOPE (v1)	8.19 (8.31)	15.49 (12.55)	9.25 (12.25)	16.36 (8.89)	11.15 (8.39)	36.74 (13.21)	16.20 (14.49)
	SCOPE (v2)	4.73 (2.25)	11.16 (10.65)	3.72 (2.59)	9.34 (5.32)	6.98 (5.35)	26.49 (13.63)	10.40 (10.93)
	TacGraph	3.29 (1.52)	2.60 (2.02)	3.82 (2.55)	7.63 (7.03)	2.34 (0.85)	2.64 (1.69)	3.72 (3.78)

Table 6.2: Quantitative results for contact point estimation. Mean and std. dev. of distance to G.T. contact point in mm reported; best for each method in Vision+Tactile and Tactile regime highlighted.

Methods	Vision + Tactile	Tactile
ICP	0.68 (0.39)	0.68 (0.39)
CHSEL	0.68 (0.39)	0.68 (0.39)
SCOPE (v1)	0.61 (0.37)	0.63 (0.44)
SCOPE (v2)	0.62 (0.37)	0.61 (0.41)
TacGraph	0.61 (0.36)	0.61 (0.36)

Table 6.3: Summary Quantitative results for contact force estimation. Mean and std. dev. of difference to G.T. contact force in Newtons reported; best for each method in Vision+Tactile and Tactile regime highlighted.

Object	Tactile-Only			
	ICP	CHSEL	SCOPE	TacGraph
cylinder	4 / 10	4 / 10	1 / 10	7 / 10
rectangle	2 / 10	6 / 10	1 / 10	6 / 10
quarter cylinder	0 / 10	1 / 10	0 / 10	1 / 10
wrench	5 / 10	3 / 10	1 / 10	9 / 10
Overall	11 / 40	14 / 40	3 / 40	23 / 40

Table 6.4: Tactile-Only Peg Insertion Results (Success / Attempt)

rectify some noise via its non-penetration handling. Neither method works well on tactile-only, as they cannot exploit contact to improve. SCOPE outperforms ICP and CHSEL on tactile only, as it can exploit the contact information, but lack of geometric feedback and noise in the filtering reduce precision. We find that TacGraph consistently outperforms the baselines across nearly all cases, exploiting contact to inform the object pose.

Table 6.2 and Table 6.3 show the contact estimation performance. When Vision and Tactile are available, we find that geometric based methods perform comparably to ours. However, when vision is removed the contact estimate quickly drifts due to the pose error. In contrast, our method retains similar overall contact estimate performance. Since all methods utilize the learned tactile force model, force estimates differ only slightly - we can see that TacGraph can improve force estimates, due to improved contact location and torque feedback at the grasp.

Fig. 6.5 shows qualitative predictions by TacGraph. In Fig. 6.5c we show the progression of K particles by their weight (i.e., $e^{-H(\cdot)}$). We see that TacGraph uses subsequent contacts to correctly identify the correct local solution, rejecting initial solutions that may be consistent with only the local geometric information.

6.5.3 Peg Insertion

We evaluate how our proposed methodology aids in a prehensile task requiring precise in-hand pose estimation. We perform a peg insertion task with several objects from Fig. 6.4. Each fixture for insertion is designed per the object geometry with 3 mm clearance. We perform the same inference procedure outlined in Sec. 6.5.2, fixing the grasps across run for fair comparison. We then take the final rest pose estimate $\boldsymbol{o}_r^*$ and perform an open-loop insertion. We use a F/T sensor mounted under the insertion fixture to stop and release if a force greater than 5 N is registered. If no force threshold is met, the object is released 5 mm above the floor of the insertion fixture. We run our experiment in the challenging tactile-only setting.

Results

The peg insertion success rates are reported across 10 presses per object for four of our objects for a total of 40 trials in Tab. 6.4. Our method is able to, from tactile alone, achieve precise open-loop peg insertion, outperforming the baselines across the objects. The `quarter cylinder` object we found had a noisy depth prediction from our model, as the gelsight is not as sensitive in the normal direction of the sensor, which caused a low success rate. We found ICP and CHSEL could align the overall orientation of the object well in some cases, and hence can succeed despite inaccurate poses, while SCOPE struggled to achieve accurate orientations leading to many failures.

6.6 Discussion

In this chapter, we proposed TacGraph, a factor-graph based method that can utilize the physical constraints of contact, along with sensory feedback, to resolve object pose and contact simultaneously. While our results demonstrate the value of uniting physical constraints and sensory feedback, the work has several limitations.

TacGraph is still inherently a local method, which means we are sensitive to initialization. A potential extension to explore is enforcing diversity across parti-

cles [141]. In the experiments, our contact interactions here were manually selected - in future, we would like to utilize our estimator online during contact-rich tasks, as well as exploring action selection to explicitly drive down uncertainty in poses [142].

Our method has several limiting assumptions: that we know object and environment geometries, that no relative slip occurs, and that contact can be described by a single point. Exploring how we can utilize reconstructed object models could relax our geometry assumption [4, 110]. Utilizing our proposed TacGraph estimator in a control loop could allow corrective actions to avoid high force/torque interactions, thereby avoiding or limiting slip. Additionally, if one could detect a slip event [117], the estimator could be re-initialized. Finally, extending to handle multiple contact points or extended contact geometries would help extend this work to a more rich set of interactions.

CHAPTER VII

TaVLA: Tactile-Annotated Vision Language Action Models

In the preceding chapters, we have designed methodologies to infer and utilize contact representations in relatively narrow scenarios, such as working with a certain class of tools or grasping and interacting with certain objects. However, recent trends in robot learning have emphasized generality, investigating how we can construct and learn foundation models that can be utilized to address many scenarios. One instance of this has been the advent of the Vision-Language-Action model, in which large-scale behavior datasets are used to train language conditioned multi-task policies. In this chapter, we investigate how we can bring a sense of contact to these large behavior models. We propose a simple, flexible contact representation derived from shear fields in compliant tactile sensors annotated into the visual feedback utilized by large scale behavior models. Our results indicate that this general, computationally efficient representation is beneficial for closing the gap between existing visuomotor policy methods and our desired tactile-informed behaviors.

7.1 Introduction

Effective robotic manipulation systems ultimately require robotic policies that can react to feedback during interaction to guide and correct actions. Recent efforts in the domain of robotic imitation learning have demonstrated impressive performance in utilizing transformer [140] and diffusion [17] architectures to learn reactive

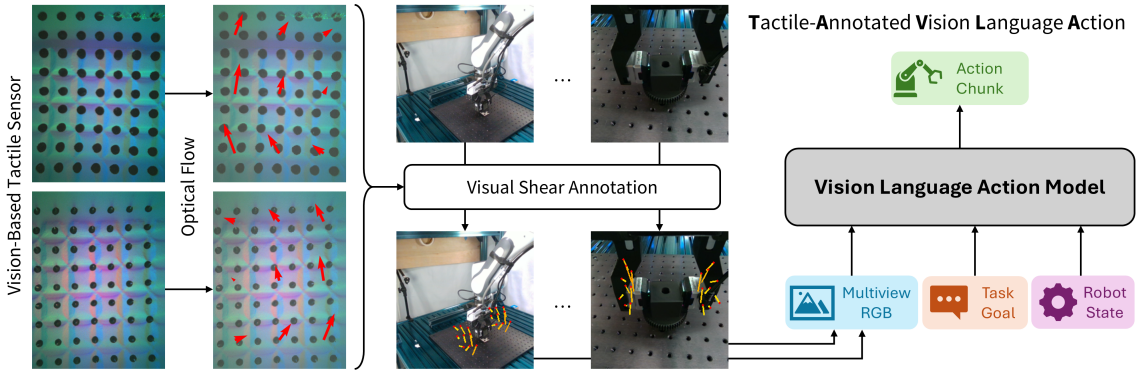


Figure 7.1: We propose Tactile-Annotated Vision Language Action (TaVLA) models, for bridging large-scale pre-trained behaviors with tactile feedback. We extract shear fields from our tactile image and directly annotate the shear into the image during contact-rich tasks, such as this hole-in-peg insertion. The multi-view images are then provided along with the language goal and robot state to fine-tune pre-trained VLA models for tactile-aware behaviors.

vision-driven policies. By utilizing large-scale robot demonstration datasets [54, 90], these policies have been scaled to build Vision-Language-Action (VLA) models [35, 9]—language-specified, multi-task, visuomotor policies. To further push towards open-world generalization, modern VLA models often utilize a pre-trained Vision-Language Model (VLM) as the backbone [5, 6, 144, 49, 56, 90]. VLA models have been shown to outperform task-specific models [54] and can exploit the semantic reasoning of VLM backbones to drive open-world behaviors [144].

While these large-scale models have demonstrated impressive range, the axes of generalization so far have largely focused on spatial, visual, and semantic variations. Undoubtedly these are crucial elements to policy success, however, many robotics tasks additionally require *physical* reasoning. For example, tool usage requires the careful application of forces from a tool to the environment [45, 88] while dynamic manipulation tasks such as in-hand reorientation require reasoning about object mass and friction information [118].

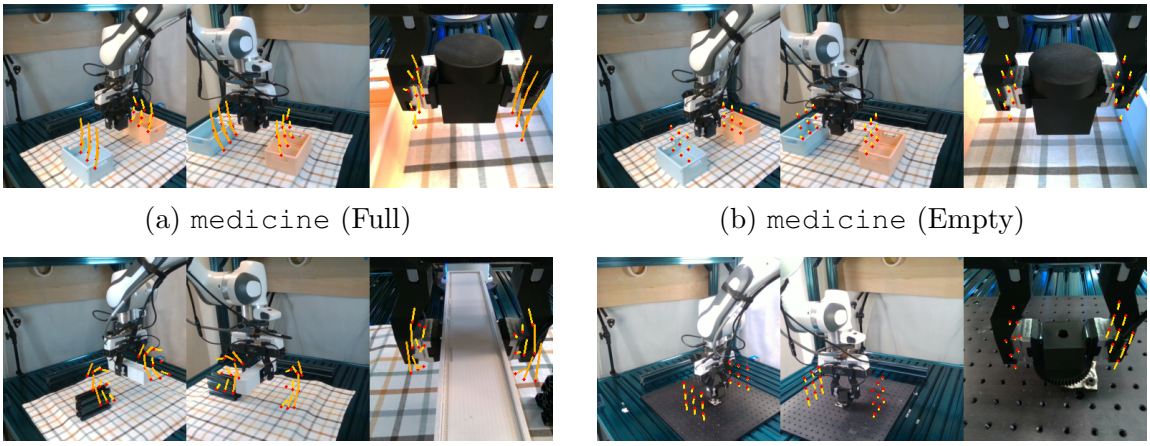
To realize feedback of forceful and physical properties, visuo-motor policies have been adapted to utilize *tactile feedback*, either in isolation [8, 126] or in combination with visual feedback [47, 15]. Tactile sensing can elucidate underlying forces during

interactions, enabling physical feedback that can drive policy behavior. This naturally invites the research question: *can we imbue Vision-Language-Action models with tactile feedback?*

Integrating tactile feedback into VLAs introduces several challenges. First, unlike visual feedback, tactile sensors output a wide range of raw sensory signals that differ significantly in semantics. Force/Torque sensors, for instance, output 6D wrenches [18] while visuo-tactile sensors such as the GelSight output RGB images [136]. Tactile datasets are often limited to single sensor classes and dwarf in comparison to vision and language datasets [32, 129]. This makes large-scale pre-training of tactile data challenging. Second, the majority of existing large-scale demonstration datasets do not contain any tactile feedback [54, 90]. Thus incorporating tactile feedback implies a data distribution shift from the large-scale pre-training that has driven VLA success.

In this work, we propose Tactile-Annotated Vision Language Action models (TaVLA), a lightweight visual augmentation technique to bridge tactile sensing and visual feedback for visuomotor policies. Inspired by the ability of VLMs to reason over image-level annotations [103], we propose to incorporate tactile feedback as annotations on the visual feedback provided to the policy. In particular, we utilize the GelSight visuo-tactile sensor [136]. We extract shear images from the sensors to capture informative forceful signals and draw down-sampled shear vectors spatially in the multi-view images provided to the policy. The result is a simple, spatially grounded, interpretable representation of tactile sensing. Our representation requires no pre-training and incurs negligible compute overhead, while taking a step towards closing the sensing gap in VLA models.

We demonstrate our proposed TaVLA approach on three real world tasks, targeting physical reasoning. We show in comparison to fine-tuning with tactile as a new modality or forgoing tactile entirely, that TaVLA is effective at incorporating tactile sensing into pre-trained VLA models.



(a) `medicine` (Full) (b) `medicine` (Empty)

(c) `balance` (d) `gear`

Figure 7.2: Example shear-based visual annotations of multi-view RGB for our three tasks. We see the shear allows visual differentiation between the full vs. empty medicine bottle (figs. 7.2a vs. 7.2b), elucidates the center of mass of an object to be balanced (fig. 7.2c), and shows the misalignment of the gear when placing on the peg (fig. 7.2d).

7.2 Problem Formulation

We propose our method, Tactile-Annotated Vision Language Action models (TaVLA), for integrating tactile feedback into pre-trained VLA visuomotor policies. In particular, we are interested in incorporating tactile signals from high-dimensional, visuo-tactile sensors [136]. These sensors provide rich tactile signals, indicating local geometric [74] and force feedback [113] at contact. However, existing VLA models are trained without visuo-tactile inputs [5, 56]. Therefore, naively injecting the visuo-tactile feedback into the model introduces significant distribution shift. Inspired by the emergent ability of VLM models to reason over visual annotations [103], we propose to incorporate tactile information as annotations onto the visual feedback of the policy. This requires no architectural changes while still providing a rich interpretable cue to the VLA model. Fine-tuning is then applied on the annotated feedback, enabling the VLA model to leverage the enhanced feedback.

VLA models are trained to take multi-view RGB images $I_t^1, \ldots, I_t^K \in \mathbb{R}^{H_C \times W_C \times 3}$,

proprioception $\theta_t \in \mathbb{R}^N$, and a language task description $l \in \Sigma^*$ and produce an action chunk $a_{t:t+H} \in \mathbb{R}^{H \times M}$.

$$\text{VLA} : \{I_t^1, \ldots, I_t^K\} \times \theta_t \times l \to a_{t:t+H} \tag{7.1}$$

Visuo-tactile sensors provide high-dimensional tactile feedback signals, in the form of RGB images $I_t^L, I_t^R \in \mathbb{R}^{H_T \times W_T \times 3}$. We assume a parallel-jaw gripper setup, where each finger is sensorized. Ultimately, our goal is to augment our VLA policy mapping to incorporate the tactile signals.

$$\text{TVLA} : \{I_t^1, \ldots, I_t^K\} \times \{I_t^L, I_t^R\} \times \theta_t \times l \to a_{t:t+H} \tag{7.2}$$

Rather than learn the TVLA mapping directly, we propose to utilize image annotation to provide a spatially grounded, interpretable tactile signal *in the original observation space of the VLA*. We design an annotation function that takes a visual feedback image and produces an augmented version capturing the tactile signal.

$$\hat{I}_t^i = f(I_t^i, I_t^L, I_t^R, \theta_t, M^i), \forall i \in [1 \ldots K] \tag{7.3}$$

Here M^i is the projection matrix of the camera, capturing the intrinsics and extrinsics of the ith camera.

This allows us to reframe our tactile-informed policy while retaining the input structure of the original VLA, simply replacing the original multi-view visual feedback with their annotated counterparts.

$$\text{TaVLA} : \{\hat{I}_t^1, \ldots, \hat{I}_t^K\} \times \theta_t \times l \to a_{t:t+H} \tag{7.4}$$

Beyond the new tactile modality, the only additional assumption required for our method is that the cameras are calibrated such that the projection matrices are known.

7.3 Method

7.3.1 Visual Annotation

We now describe our visual annotation function f. Visuo-tactile sensors utilize a compliant gel membrane which deforms during forceful interactions. A camera is mounted behind the membrane and observes the deformations, aided often by markers printed on the gel [136, 113]. This results in raw sensory signal of RGB images from the grasp I_t^L and I_t^R. Instead of directly training policies and estimation models on the raw tactile images, recent research has identified optical flow as a useful intermediate representation [7]. Here we propose visual annotation of shear fields as a mechanism to provide tactile feedback to VLA policies.

7.3.1.1 Tactile Shear Field Computation

At each timestep, we compute the shear field using the Farneback algorithm [29]. This measures the optical flow of the markers in the images from a given base image I_r.

$$S_t = \text{Flow}(I_r, I_t) \tag{7.5}$$

The resulting 2-dimensional shear field $S_t \in \mathbb{R}^{H_T \times W_T \times 2}$ captures per-pixel deformation caused by forces acting on the sensor during the action.

Deformations can be caused both due to internal forces from the grasp as well as external forces (e.g., gravity or extrinsic contacts). We consider the case where the latter is more informative for the task, and reset our base image per sensor I_r whenever the grasp state changes (i.e., from grasped to released or vice versa). This allows us to isolate external force signals from the deformations caused by the grasp.

7.3.1.2 Visual Augmentation with Shear Fields

Our goal is to provide a clear, interpretable, and spatially grounded signal of the tactile shear via our annotation. We discuss our annotation procedure for a single shear field. As our shear fields are high-dimensional, we first apply mean pooling to reduce the size to lower-dimensional fields $S_t \in \mathbb{R}^{H_S \times W_S \times 2}$. From proprioception,

we recover the location of the corresponding sensor fingertip $T_t \in SE(3)$. We then define a grid of 3D points in the frame of the fingertip of matching size $H_S \times W_S$ to our shear field, lying in the plane of the sensor surface, $P_t^s \in \mathbb{R}^{H_S \times W_S \times 3}$. We apply the shear vectors in the sensor plane to determine corresponding end points in 3D for our grid $P_t^e \in \mathbb{R}^{H_S \times W_S \times 3}$.

$$P_t^e = P_t^s + \alpha \bar{S}_t \tag{7.6}$$

Here $\bar{S}_t$ are the homogeneous shear vectors with zero in the third dimension. α is a heuristic scaling term. The result is a 3D spatially grounded realization of the shear field.

For each camera in our scene, we can then use our camera projection matrix M^i to project the shear start and end points into the pixel space. For each point $p \in P_t^s \cup P_t^e$, we first transform into the world frame using the fingertip pose then project into the camera frame utilizing the projection matrix.

$$\bar{u} = M^i T_t \bar{p} \tag{7.7}$$

We can then annotate our shear in pixel space as a line from each grid start and end point. We also apply color grading on the lines to convey magnitude.

We repeat this procedure for the left and right tactile shear fields for each camera view accordingly, resulting in two grids of vectors annotated into each multi-view image. This forms our annotated images $\{\hat{I}_t^1, \ldots, \hat{I}_t^K\}$ that we use to finetune the pre-trained VLA model. The overall TaVLA pipeline is shown in Fig. 7.1.

Example annotated images can be found in Fig. 7.2. We see that the resulting shear annotation can elucidate task relevant features such as mass, center of mass, or misalignments during insertion.

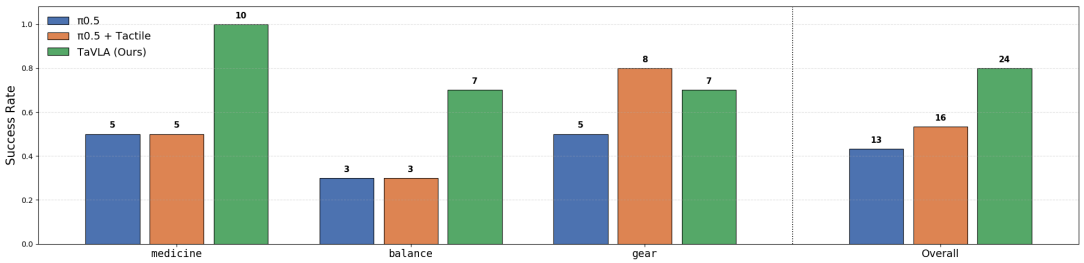


Figure 7.3: Quantitative performance across our tasks. Each method was run 10 times per task.

7.4 Experiments

7.4.1 Robot Setup

We conduct our experiments on a 7-DoF Franka Emika Panda robot. We attach GelSight sensors [136] to the default Franka gripper. We utilize three RealSense cameras: two RealSense D435 cameras on the left and right hand side of the robot providing over-the-shoulder views and a wrist-mounted RealSense D405 providing a local view. We calibrate the external cameras to the robot and use the robot proprioception to track the wrist camera pose. Example views from each camera can be seen in Fig. 7.2.

7.4.2 Tasks

We setup three tasks to test the ability of the fine-tuned VLA model to reason about physical forces during interaction.

1. `medicine` - the robot is presented with an opaque "medicine" bottle and two containers. The task is to place the bottle in the orange container if it is full or in the blue container if it is empty. This task involves reasoning about the mass of the object. During our experiment, the bottle weighs 38g when empty and 422g when full.

2. `balance` - the robot is tasked with balancing one object on another in the scene. The object to be balanced has a fixed mass, but is designed so that the

center of mass can be adjusted. The robot must reason about the center of mass to stably place the object without toppling. We run the task shifting the center of mass between three locations: centered and at each end of the object.

3. `gear` - the robot starts grasping a gear object from the IndustReal benchmark [112] and must place the gear onto a peg mounted on the tabletop. This requires reasoning online about extrinsic forces during the interaction to align the gear and the peg.

We collect demonstration data by teleoperating the robot via the Oculus Quest 2 VR device, recording observations and action labels at roughly 10Hz. We collect 100 demonstrations per task, varying spatial arrangements and lighting throughout, for a total of 71k data samples.

7.4.3 Implementation Details

We implement our method utilizing the $\pi_{0.5}$ VLA model [5]. We finetune the model on our dataset to learn a single multi-task policy. We input the shear-annotated over-the-shoulder and wrist images, along with the robot joint state (7-dim) and the task language label (see Fig. 7.1). Actions are predicted as delta joint angles along with a binary gripper open/close command (8-dim). We use an action horizon $H = 10$. We train with AdamW for 20k train steps with a batch size of 32.

To scale the shear annotations into the image in a reliable and interpretable way, we compute the 95th percentile of the shear vector magnitudes for each task during training and scale the vectors accordingly. We also clip vectors if they exceed twice the magnitude of the 95th percentile, to avoid spurious shears causing unexpected visuals.

7.4.4 Baselines

In our experiments, we want to understand a) the utility of tactile feedback during contact-rich task execution and b) the effectiveness of our proposed visual annotation method TaVLA for providing tactile feedback. We compare to two baseline methods:

- $\pi_{0.5}$: we finetune the base VLA model *only* on the visual feedback.
- $\pi_{0.5}$ + Tactile: we include the *raw* tactile images as auxiliary visual inputs to the VLA model, treating them as additional "viewpoints."

We use the same finetuning procedure and dataset used by TaVLA.

7.4.5 Results

Fig. 7.3 reports task success rates, out of 10 trials per task, across our three real-world manipulation tasks. Our results indicate that TaVLA performs best overall, with 24 of 30 successes. Both TaVLA and $\pi_{0.5}$ + Tactile outperformed $\pi_{0.5}$ finetuned with no tactile feedback, highlighting the importance of tactile for these tasks. However, we see that TaVLA is the only method that performs consistently across the tasks.

On the `medicine` and `balance` tasks, we see both $\pi_{0.5}$ and $\pi_{0.5}$ + Tactile perform equivalent to noise. The likelihood of the medicine bottle being full or empty is 50% and the center of mass of the object to be balanced takes one of three configurations. This indicates that neither method was able to consistently identify the relevant physical properties sufficiently to guide behavior. In contrast, TaVLa succeeds on every trial (10/10) of the `medicine` task and the majority (7/10) of the `balance` task trials.

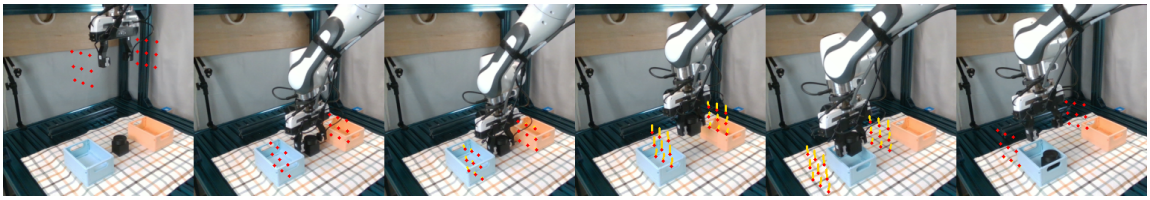
For the gear task, $\pi_{0.5}$ + Tactile slightly outperforms TaVLA, while both outperform $\pi_{0.5}$. We found that $\pi_{0.5}$ frequently jammed, often making contact and coarse alignment with the peg, but unable to complete the insertion. In contrast, both tactile driven methods exhibited more corrective behaviors.

Finally, in Fig. 7.4, we show qualitative rollouts of TaVLA on each task, showing one of the annotated over-the-shoulder views. Figs. 7.4a and 7.4b show how the annotations reveal whether the bottle is full or empty to guide the robot to correctly place in the corresponding bin. Figs. 7.4c and 7.4d show how the policy can adjust its placement strategy according to the implied center of mass from the annotations. Finally, Fig. 7.4e shows adjustments to continuously changing shears to correct for misalignment during the gear on peg task.

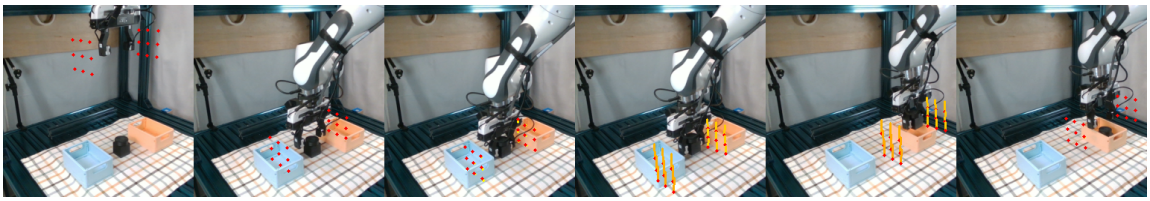
7.5 Discussion

We present Tactile-Annotated Vision Language Action (TaVLA) models for bridging pre-trained VLA models with tactile sensing. We propose a lightweight, shear-based annotation scheme that allows us to embed rich tactile feedback directly into pixel space. This reduces distribution shift and requires no architectural changes while incurring minimal computational overhead. In our experiments, we show the value of TaVLA for guiding manipulation based on forceful feedback, outperforming naively supplying raw tactile to the VLA or forgoing tactile entirely.

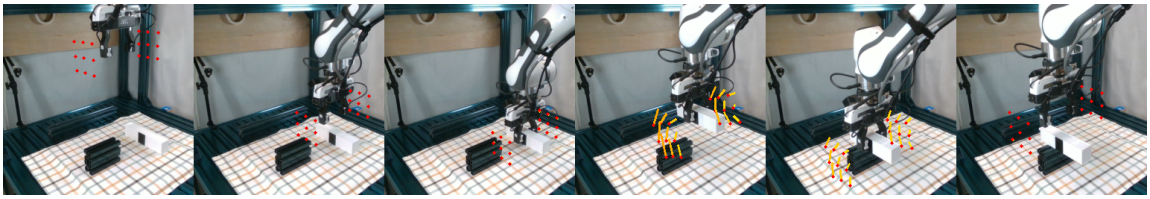
We highlight several limitations of our method. First, our conversion to shear and subsequent downsampling removes a significant amount of signal from the tactile feedback. While this aids in efficient training, we are potentially removing valuable feedback. For example, grasp geometry is completely removed, as we focus here on external forces (e.g., gravity and extrinsic contacts). Over the long term, we expect that large-scale training of tactile reasoning in tandem with behaviors will outperform hand-designed features as presented here. We believe this work highlights the need for these methods. Second, in visually dense scenarios, our annotation may unintentionally occlude important visual signals to the policy. This could be addressed by providing additional annotation-free images, at the cost of some redundancy. Similarly, the introduction of many tactile sensors, e.g., via a multi-fingered hand, may yield cluttered annotations which may detract from interpretability.



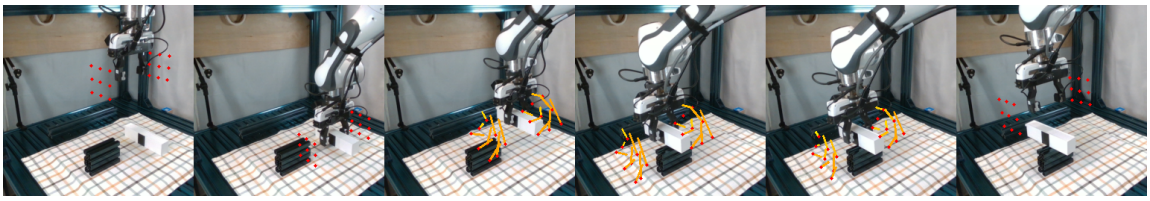
(a) `medicine` (empty)



(b) `medicine` (full)



(c) `balance` (CoM "far")



(d) `balance` (CoM "near")

(e) `gear`

Figure 7.4: Sample successful rollouts of our proposed method, TaVLA, across our three test tasks. We see that the finetuned VLA can utilize the shear annotations to effectively react to tactile feedback.

CHAPTER VIII

Conclusion

This thesis introduces several methodologies for providing our robots an explicit sense of *contact*. Central to our work is the inclusion of *elastic deformability* into our systems, via robot manipulators and tools. We propose methods that exploit the coupling of deformation and contact to reason and control contacts during manipulation, directly from raw sensory feedback.

In Chapters III and IV, we focus on representing and controlling contact between a deforming tool and the environment. We propose representations of contact to capture if, where, and how contact is being made along with data-driven model architectures and training paradigms to recover these features.

In Chapters V, VI, and VII, we shift our focus to *utilizing* contact in downstream reasoning and behaviors. In Chapter V and VI, we investigate how contact representations allow us to incorporate physical structure into our algorithms to address the challenges of reasoning over non-prehensile and prehensile manipulations. Finally, in Chapter VII, we shift to multi-task policy methods, and show that our choice of tactile representation can help enable contact-rich and physically grounded reasoning.

8.1 Future Work

8.1.1 Scaling Contact Representations for Planning and Control

Across this thesis, our representations of contact have primarily been applied in narrow scenarios. For example, our contact representation in Chap. IV was limited to a single tool and environments similar to those during training, while our tactile representations in Chap. VI were limited to objects of similar geometries. Scaling up our learned representations to handle a larger variety of scenarios is crucial to unlocking their benefits. The resulting learning problem becomes more difficult however, both from a data and methodological perspective. Scaling of contact representations may require additional efforts in simulation to bridge the realism gap while enabling large scale data collection. Similarly, scaling models to capture all these variations may require more advanced architectures and training methods to effectively capture the space of contacts.

Additionally, further study of integrating contact representations into downstream planning and control methods is needed to elucidate in what situations and how to incorporate structure generally. It is worth noting that recent successes in machine learning largely forgo structure and rely instead on scaling data [131]. The role which structure should play in general manipulation systems is still an open research problem. In this thesis, we have shown the value of structure in both dynamics-based planning (Chaps. III and V), state estimation (Chap. VI) and visuomotor control policy (Chap. VII) methods. Related research projects have similarly shown the value of contact representations for policy methods [42] and for incorporating physical structure into learned dynamics [92]. Tasks that require a high level of precision and have increased uncertainty seem likely to benefit most from structure. Understanding the relationship between structure and purely data-driven behaviors is crucial future work.

8.1.2 Capturing Uncertainty

This thesis primarily focuses on providing point estimates of our contact representations and downstream state estimations. However, in many cases, the partially observable nature of the manipulation problem means there is inherent uncertainties present, especially when reasoning about contacts which are heavily occluded. In Chap. VI, we utilize several particle initializations during our pose and contact estimation, but did not investigate in detail the ability to fully capture the distribution. A future direction for research is to investigate how to capture uncertainty for poses, contacts, and forces. This can enable for uncertainty-aware planning to disambiguate crucial information to drive behaviors [142].

8.1.3 Tactile Sensing

Throughout this thesis, we incorporate both sparse (force/torque) and dense (vision-based) tactile sensing. Force/torque signals can provide useful summary force information but can obfuscate details by nature of the sparsity. Meanwhile dense vision-based sensors provide rich geometric and force information, but current realizations are brittle during manipulations (e.g., tearing of gels) and require translating from raw sensory feedback to the relevant geometric or force signals. An interesting future direction is exploring alternative tactile sensing and how they can be utilized to construct contact-aware manipulation behaviors. Taxel-based sensors [115] are an interesting intermediate that provides 3-axis force signals at several taxel locations. This signal is more dense then force/torque sensing yet has a directly interpretable output space.

Another interesting direction of future work is scaling tactile contact representations. Analagous to the role of depth [130] and segmentation [60] for visual data, we may ask ourselves what the important primitives are for tactile information. In Chapters VI and VII, we found that geometry and force information can provide useful signals for manipulation. Scaling learning around these entities may provide reusable *foundation models* for tactile that can push forward tactile reasoning and behaviors.

8.1.4 Multi-fingered Hands

A final direction of future work is extending our methods beyond attached tools or parallel jaw grasps to multi-fingered robotic end effectors. Multi-fingered hands enable for more complex manipulations, such as in-hand reorientation [110] and complex contact-rich sliding and tool use [62], which can significantly extend the utility of robotic manipulation systems.

Multi-fingered hands come with several challenges and opportunities. The hands contain more degrees of freedom which results in more challenging, high-dimensional systems. Additionally, the number and complexity of available contact interactions increases. The increased complexity does, however, enable more dexterous contacts and provides an avenue for more rich geometric and forceful reasoning for estimation and control. For instance, in the case of pose estimation in Chapter VI, the additional non-penetration constraints from the robot fingers would significantly narrow the search space. Extension of contact representations and downstream reasoning pipelines to multi-fingered hands is an important direction of future work for creating more dexterous and performant manipulation systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1157–1167. PMLR, 14–18 Dec 2023.

[2] Alex Alspach, Kunimatsu Hashimoto, Naveen Kuppuswamy, and Russ Tedrake. Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 597–604, 2019.

[3] Maria Bauza, Antonia Bronars, Yifan Hou, Ian Taylor, Nikhil Chavan-Dafle, and Alberto Rodriguez. Simple, a visuotactile method learned in simulation to precisely pick, localize, regrasp, and place objects. *Science Robotics*, 9(91):eadi8808, 2024.

[4] Maria Bauza, Antonia Bronars, and Alberto Rodriguez. Tac2pose: Tactile object pose estimation from the first touch. *The International Journal of Robotics Research*, 42(13):1185–1209, 2023.

[5] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. pi0.5: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky.

pi0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.

[7] William van den Bogert, Madhavan Iyengar, and Nima Fazeli. Built different: Tactile perception to overcome cross-embodiment capability differences in collaborative manipulation. *arXiv preprint arXiv:2409.14896*, 2024.

[8] William van den Bogert, Gregory Linkowski, and Nima Fazeli. Gromp: Grasped object manifold projection for multimodal imitation learning of manipulation. *arXiv preprint arXiv:2512.03347*, 2025.

[9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[10] Antonia Bronars, Sangwoon Kim, Parag Patre, and Alberto Rodriguez. Texterity: Tactile extrinsic dexterity. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7976–7983. IEEE, 2024.

[11] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180, 2017.

[12] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3346, 2018.

[13] Roberto Calandra, Serena Ivaldi, Marc Peter Deisenroth, Elmar Rueckert, and Jan Peters. Learning inverse dynamics models with contacts. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3186–3191, 2015.

[14] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[15] Claire Chen, Zhongchun Yu, Hojung Choi, Mark Cutkosky, and Jeannette Bohg. Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation. *IEEE Robotics and Automation Letters*, 2025.

[16] Zhengxue Cheng, Yiqian Zhang, Wenkang Zhang, Haoyu Li, Keyu Wang, Li Song, and Hengdi Zhang. Omnivtla: Vision-tactile-language-action model with semantic-aligned tactile sensing. *arXiv preprint arXiv:2508.08706*, 2025.

[17] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[18] Hojung Choi, Jun En Low, Tae Myung Huh, Gabriela A Uribe, Seongheon Hong, Kenneth AW Hoffman, Julia Di, Tony G Chen, Andrew A Stanley, and Mark R Cutkosky. Coinft: A coin-sized, capacitive 6-axis force torque sensor for robotic applications. *arXiv preprint arXiv:2503.19225*, 2025.

[19] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[20] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, Guido Ranzuglia, et al. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy, 2008.

[21] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[22] Frank Dellaert and GTSAM Contributors. borglab/gtsam, May 2022.

[23] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends® in Robotics*, 6(1-2):1–139, 2017.

[24] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021.

[25] Snehal Dikhale, Karankumar Patel, Daksh Dhingra, Itoshi Naramura, Akinobu Hayashi, Soshi Iba, and Nawid Jamali. VisuoTactile 6D Pose Estimation of an In-Hand Object Using Vision and Tactile Sensor Data. *IEEE Robotics and Automation Letters*, 7(2):2148–2155, 2022.

[26] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1927–1934, 2018.

[27] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Conference on Robot Learning*, pages 1755–1768. PMLR, 2023.

[28] Jose A Eyzaguirre, Miquel Oller, and Nima Fazeli. Tactile neural de-rendering. *arXiv preprint arXiv:2409.13923*, 2024.

[29] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[30] Nima Fazeli, Samuel Zapolsky, Evan Drumwright, and Alberto Rodriguez. Fundamental limitations in performance and interpretability of common planar rigid-body contact models. In *Robotics Research*, pages 555–571. Springer, 2020.

[31] James M Ferguson, D Caleb Rucker, and Robert J Webster. Unified shape and external load state estimation for continuum robots. *IEEE Transactions on Robotics*, 40:1813–1827, 2024.

[32] Letian Fu, Gaurav Datta, Huang Huang, William Chung-Ho Panitch, Jaimyn Drake, Joseph Ortiz, Mustafa Mukadam, Mike Lambeta, Roberto Calandra, and Ken Goldberg. A touch, vision, and language dataset for multimodal alignment. In *International Conference on Machine Learning*, pages 14080–14101. PMLR, 2024.

[33] Ruohan Gao, Yiming Dou, Hao Li, Tanmay Agarwal, Jeannette Bohg, Yunzhu Li, Li Fei-Fei, and Jiajun Wu. The objectfolder benchmark: Multisensory learning with neural and real objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17276–17286, June 2023.

[34] Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10598–10608, June 2022.

[35] Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, 2024.

[36] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4367–4375, 2018.

[37] Aimee Goncalves, Naveen Kuppuswamy, Andrew Beaulieu, Avinash Uttamchandani, Katherine M. Tsui, and Alex Alspach. Punyo-1: Soft tactile-sensing upper-body robot for large object manipulation and physical human interaction. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 844–851, 2022.

[38] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 09–15 Jun 2019.

[39] Xuchen Han, Joseph Masterjohn, and Alejandro Castro. A convex formulation of frictional contact between rigid and deformable bodies. *IEEE Robotics and Automation Letters*, 8(10):6219–6226, 2023.

[40] Peng Hao, Chaofan Zhang, Dingzhe Li, Xiaoge Cao, Xiaoshuai Hao, Shaowei Cui, and Shuo Wang. Tla: Tactile-language-action model for contact-rich manipulation. *arXiv preprint arXiv:2503.08548*, 2025.

[41] Carolina Higuera, Siyuan Dong, Byron Boots, and Mustafa Mukadam. Neural contact fields: Tracking extrinsic contact with tactile sensing. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12576–12582, 2023.

[42] Carolina Higuera, Joseph Ortiz, Haozhi Qi, Luis Pineda, Byron Boots, and Mustafa Mukadam. Perceiving extrinsic contacts from touch improves learning insertion policies. *arXiv preprint arXiv:2309.16652*, 2023.

[43] Carolina Higuera, Akash Sharma, Chaithanya Krishna Bodduluri, Taosha Fan, Patrick Lancaster, Mrinal Kalakrishnan, Michael Kaess, Byron Boots, Mike Lambeta, Tingfan Wu, et al. Sparsh: Self-supervised touch representations for vision-based tactile sensing. In *Conference on Robot Learning*, pages 885–915. PMLR, 2025.

[44] Francois R. Hogan, Jose Ballester, Siyuan Dong, and Alberto Rodriguez. Tactile dexterity: Manipulation primitives with tactile feedback. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8863–8869, 2020.

[45] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Force-and-motion constrained planning for tool use. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7409–7416, 2019.

[46] Yifan Hou, Zhenzhong Jia, and Matthew T. Mason. Fast planning for 3d any-pose-reorienting using pivoting. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1631–1638, 2018.

[47] Yifan Hou, Zeyi Liu, Cheng Chi, Eric Cousineau, Naveen Kuppuswamy, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4829–4836. IEEE, 2025.

[48] Jialei Huang, Shuo Wang, Fanqi Lin, Yihang Hu, Chuan Wen, and Yang Gao. Tactile-vla: unlocking vision-language-action model's physical knowledge for tactile generalization. *arXiv preprint arXiv:2507.09160*, 2025.

[49] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[50] Maged Iskandar, Christian Ott, Alin Albu-Schäffer, Bruno Siciliano, and Alexander Dietrich. Hybrid force-impedance control for fast end-effector motions. *IEEE Robotics and Automation Letters*, 8(7):3931–3938, 2023.

[51] Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[52] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[53] Sagi Katz, Ayellet Tal, and Ronen Basri. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*, pages 24–es. ACM, 2007.

[54] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024.

[55] Leon Kim, Yunshuang Li, Michael Posa, and Dinesh Jayaraman. Im2Contact: Vision-Based Contact Localization Without Touch or Force Sensing. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1533–1546. PMLR, 06–09 Nov 2023.

[56] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[57] Sangwoon Kim, Devesh K. Jha, Diego Romeres, Parag Patre, and Alberto Rodriguez. Simultaneous Tactile Estimation and Control of Extrinsic Contact. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12563–12569, 2023.

[58] Sangwoon Kim and Alberto Rodriguez. Active extrinsic contact sensing: Application to general peg-in-hole insertion. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10241–10247, 2022.

[59] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*

2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[60] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[61] Oliver Kroemer, Scott Niekum, and George Dimitri Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30), 2021.

[62] Abhinav Kumar, Thomas Power, Fan Yang, Sergio Aguilera Marinovic, Soshi Iba, Rana Soltani Zarrin, and Dmitry Berenson. Diffusion-informed probabilistic contact search for multi-finger manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14277–14283, 2025.

[63] Jayjun Lee and Nima Fazeli. Vitascope: Visuo-tactile implicit representation for in-hand pose and extrinsic contact estimation. In *Proceedings of Robotics: Science and Systems*, 2025.

[64] Michelle A. Lee, Yuke Zhu, Krishnan Srinivasan, Parth Shah, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8943–8950, 2019.

[65] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[66] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[67] Linfeng Li, Gang Yang, Lin Shao, and David Hsu. Stable object placement under geometric uncertainty via differentiable contact dynamics. *arXiv preprint arXiv:2409.17725*, 2024.

[68] Qiang Li, Carsten Schürmann, Robert Haschke, and Helge J Ritter. A control framework for tactile servoing. In *Robotics: Science and systems*. Citeseer, 2013.

[69] Rui Li, Robert Platt, Wenzhen Yuan, Andreas ten Pas, Nathan Roscup, Mandayam A. Srinivasan, and Edward Adelson. Localization and manipulation of small parts using gelsight tactile sensing. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3988–3993, 2014.

[70] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019.

[71] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[72] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The international journal of robotics research*, 15(6):533–556, 1996.

[73] Daolin Ma, Siyuan Dong, and Alberto Rodriguez. Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11262–11268. IEEE, 2021.

[74] Daolin Ma, Elliott Donlon, Siyuan Dong, and Alberto Rodriguez. Dense tactile force estimation using gelslim and inverse fem. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5418–5424. IEEE, 2019.

[75] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[76] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 693–710. PMLR, 16–18 Nov 2021.

[77] Lucas Manuelli and Russ Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069, 2016.

[78] Roberto Martín-Martín, Michelle A Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1010–1017. IEEE, 2019.

[79] Joseph Masterjohn, Damrong Guoy, John Shepherd, and Alejandro Castro. Velocity level approximation of pressure field contact patches. *IEEE Robotics and Automation Letters*, 7(4):11593–11600, 2022.

[80] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[81] P Mitrano, D McConachie, and D Berenson. Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170, 2021.

[82] Peter Mitrano and Dmitry Berenson. Data augmentation for manipulation. *Robotics Science and Systems (RSS)*, 2022.

[83] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021.

[84] Matthias Müller, Jos Stam, Doug James, and Nils Thürey. Real time physics: class notes. In *ACM siggraph 2008 classes*, pages 1–90. ACM, 2008.

[85] Prajval Kumar Murali, Bernd Porr, and Mohsen Kaboli. Shared visuo-tactile interactive perception for robust object pose estimation. *The International Journal of Robotics Research*, 44(7):1186–1216, 2025.

[86] Yashraj Narang, Balakumar Sundaralingam, Miles Macklin, Arsalan Mousavian, and Dieter Fox. Sim-to-real for robotic tactile sensing via physics-based simulation and learned latent projections. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6444–6451. IEEE, 2021.

[87] Miquel Oller, Dmitry Berenson, and Nima Fazeli. Tactile-Driven Non-Prehensile Object Manipulation via Extrinsic Contact Mode Control. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.

[88] Miquel Oller, Mireia Planas i Lisbona, Dmitry Berenson, and Nima Fazeli. Manipulation via membranes: High-resolution and highly deformable tactile sensing and control. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1850–1859. PMLR, 14–18 Dec 2023.

[89] Kei Ota, Devesh K. Jha, Krishna Murthy Jatavallabhula, Asako Kanezaki, and Joshua B. Tenenbaum. Tactile estimation of extrinsic contact patch for stable placement. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13876–13882, 2024.

[90] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[91] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[92] Samuel Pfrommer, Mathew Halm, and Michael Posa. Contactnets: Learning discontinuous contact dynamics with smooth, implicit representations. In *Conference on Robot Learning*, pages 2279–2291. PMLR, 2021.

[93] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[94] Sai Rajeswar, Cyril Ibrahim, Nitin Surya, Florian Golemo, David Vazquez, Aaron Courville, and Pedro O Pinheiro. Haptics-based curiosity for sparse-reward tasks. In *Conference on Robot Learning*, pages 395–405. PMLR, 2022.

[95] Daniel Rebain, Mark J. Matthews, Kwang Moo Yi, Gopal Sharma, Dmitry Lagun, and Andrea Tagliasacchi. Attention beats concatenation for conditioning neural fields. *Transactions on Machine Learning Research*, 2023.

[96] Samanta Rodriguez, Yiming Dou, Miquel Oller, Andrew Owens, and Nima Fazeli. Touch2touch: Cross-modal tactile generation for object manipulation. *arXiv preprint arXiv:2409.08269*, 2024.

[97] Sho Sakaino. Bilateral control-based imitation learning for velocity-controlled robot. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, pages 1–6. IEEE, 2021.

[98] Brad Saund and Dmitry Berenson. Motion planning for manipulators in unknown environments with contact sensing uncertainty. In *International Symposium on Experimental Robotics*, pages 461–474. Springer, 2018.

[99] Brad Saund, Sanjiban Choudhury, Siddhartha Srinivasa, and Dmitry Berenson. The blindfolded robot: A bayesian approach to planning with contact feedback. In *The International Symposium of Robotics Research*, pages 443–459. Springer, 2019.

[100] Tanner Schmidt, Richard A Newcombe, and Dieter Fox. DART: Dense Articulated Real-Time Tracking. In *Robotics: Science and systems*, pages 1–9. Berkeley, CA, 2014.

[101] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1038–1049. PMLR, 14–18 Dec 2023.

[102] Yuki Shirai, Devesh K. Jha, Arvind U. Raghunathan, and Dennis Hong. Tactile tool manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12597–12603, 2023.

[103] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11987–11997, October 2023.

[104] Andrea Sipos, William van den Bogert, and Nima Fazeli. Gelslim 4.0: Focusing on touch and reproducibility. *arXiv preprint arXiv:2409.19770*, 2024.

[105] Andrea Sipos and Nima Fazeli. Simultaneous contact location and object pose estimation using proprioception and tactile feedback. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3233–3240. IEEE, 2022.

[106] Andrea Sipos and Nima Fazeli. MultiSCOPE: Disambiguating In-Hand Object Poses with Proprioception and Tactile Feedback. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[107] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[108] HJ Terry Suh, Naveen Kuppuswamy, Tao Pang, Paul Mitiguy, Alex Alspach, and Russ Tedrake. Seed: Series elastic end effectors in 6d for visuotactile tool use. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4684–4691. IEEE, 2022.

[109] K. Suita, Y. Yamada, N. Tsuchida, K. Imai, H. Ikeda, and N. Sugimoto. A failure-to-safety "kyozon" system with simple contact detection and stop capabilities for safe human-autonomous robot coexistence. In *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 3089–3096 vol.3, 1995.

[110] Sudharshan Suresh, Haozhi Qi, Tingfan Wu, Taosha Fan, Luis Pineda, Mike Lambeta, Jitendra Malik, Mrinal Kalakrishnan, Roberto Calandra, Michael Kaess, et al. Neuralfeels with neural fields: Visuotactile perception for in-hand manipulation. *Science Robotics*, 9(96):eadl0628, 2024.

[111] Giovanni Sutanto, Nathan Ratliff, Balakumar Sundaralingam, Yevgen Chebotar, Zhe Su, Ankur Handa, and Dieter Fox. Learning latent space dynamics for tactile servoing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3622–3628. IEEE, 2019.

[112] Bingjie Tang, Michael A Lin, Iretiayo Akinola, Ankur Handa, Gaurav S Sukhatme, Fabio Ramos, Dieter Fox, and Yashraj Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. In *Proceedings of Robotics: Science and Systems*, Daegu, Korea, July 2023.

[113] Ian H. Taylor, Siyuan Dong, and Alberto Rodriguez. Gelslim 3.0: High-resolution measurement of shape, force and slip in a compact tactile-sensing

finger. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10781–10787, 2022.

[114] Russ Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019.

[115] Tito Pradhono Tomo, Alexander Schmitz, Wai Keat Wong, Harris Kristanto, Sophon Somlor, Jinsun Hwang, Lorenzo Jamone, and Shigeki Sugano. Covering a robot fingertip with uskin: A soft electronic skin with distributed 3-axis force sensitive elements for robot hands. *IEEE Robotics and Automation Letters*, 3(1):124–131, 2018.

[116] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 306–316. PMLR, 29–31 Oct 2018.

[117] Filipe Veiga, Jan Peters, and Tucker Hermans. Grip stabilization of novel objects using slip prediction. *IEEE transactions on haptics*, 11(4):531–542, 2018.

[118] Chen Wang, Shaoxiong Wang, Branden Romero, Filipe Veiga, and Edward Adelson. 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)*, pages 5633–5640, 2020.

[119] Peiyi Wang, Zhexin Xie, Wenci Xin, Zhiqiang Tang, Xinhua Yang, Muralidharan Mohanakrishnan, Sheng Guo, and Cecilia Laschi. Sensing expectation enables simultaneous proprioception and contact detection in an intelligent soft continuum robot. *Nature Communications*, 15(1):9978, 2024.

[120] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[121] David Watkins-Valls, Jacob Varley, and Peter Allen. Multi-modal geometric learning for grasping and manipulation. In *2019 International conference on robotics and automation (ICRA)*, pages 7339–7345. IEEE, 2019.

[122] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590, 2022.

[123] Youngsun Wi, Mark Van der Merwe, Pete Florence, Andy Zeng, and Nima Fazeli. Calamari: Contact-aware and language conditioned spatial action mapping for contact-rich manipulation. In *Conference on Robot Learning*, pages 2753–2771. PMLR, 2023.

[124] Youngsun Wi, Andy Zeng, Pete Florence, and Nima Fazeli. Virdo++: Real-world, visuo-tactile dynamics and perception of deformable objects. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1806–1816. PMLR, 14–18 Dec 2023.

[125] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

[126] Yansong Wu, Zongxie Chen, Fan Wu, Lingyun Chen, Liding Zhang, Zhenshan Bing, Abdalla Swikir, Sami Haddadin, and Alois Knoll. Tacdiffusion: Force-domain diffusion policy for precise tactile manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11831–11837. IEEE, 2025.

[127] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX*, pages 365–381. Springer, 2020.

[128] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning*, pages 564–574. PMLR, 2021.

[129] Fengyu Yang, Chenyang Ma, Jiacheng Zhang, Jing Zhu, Wenzhen Yuan, and Andrew Owens. Touch and go: Learning from human-collected vision and touch. *Advances in Neural Information Processing Systems*, 35:8081–8103, 2022.

[130] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10371–10381, June 2024.

[131] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 2023.

[132] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.

[133] Jiawen Yu, Hairuo Liu, Qiaojun Yu, Jieji Ren, Ce Hao, Haitong Ding, Guangyu Huang, Guofan Huang, Yan Song, Panpan Cai, et al. Forcevla: Enhancing vla models with a force-aware moe for contact-rich manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[134] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, 2016.

[135] Kuan-Ting Yu and Alberto Rodriguez. Realtime state estimation with tactile and visual sensing for inserting a suction-held object. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1628–1635, 2018.

[136] Wenzhen Yuan, Siyuan Dong, and Edward H Adelson. Gelsight: High-resolution robot tactile sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[137] Chaofan Zhang, Peng Hao, Xiaoge Cao, Xiaoshuai Hao, Shaowei Cui, and Shuo Wang. Vtla: Vision-tactile-language-action model with preference learning for insertion manipulation. *arXiv preprint arXiv:2505.09577*, 2025.

[138] Zongzheng Zhang, Haobo Xu, Zhuo Yang, Chenghao Yue, Zehao Lin, Huan-ang Gao, Ziwei Wang, and Hao Zhao. Elucidating the design space of torque-aware vision-language-action models. In *9th Annual Conference on Robot Learning*, 2025.

[139] Fang Zhao, Jian Zhao, Shuicheng Yan, and Jiashi Feng. Dynamic conditional networks for few-shot learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–35, 2018.

[140] Tony Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems XIX*, 2023.

[141] Sheng Zhong, Dmitry Berenson, and Nima Fazeli. CHSEL: Producing Diverse Plausible Pose Estimates from Contact and Free Space Data. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[142] Sheng Zhong, Nima Fazeli, and Dmitry Berenson. Rumi: Rummaging using mutual information. *IEEE Transactions on Robotics*, 41:5431–5450, 2025.

[143] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016.

[144] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

APPENDICES

APPENDIX A

Appendix for Chapter 3: Learning the Dynamics of Compliant Tool-Environment Interaction for Visuo-Tactile Contact Servoing

A.1 Network Architecture Details

Here, we describe in detail the network architecture of our contact feature dynamics model introduced in Sec. 3.3.1. Our network has three main components, an encoder e, dynamics module f, and decoder d:

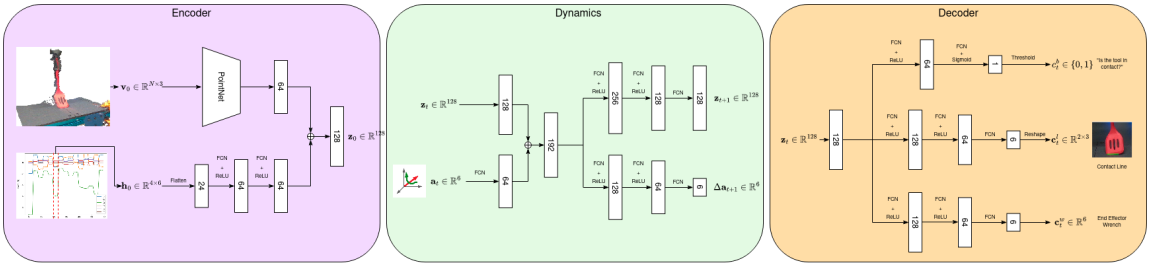


Figure A.1: Architecture Details for the Components of the Contact Feature Dynamics Model.

1. **Encoder:** The encoder takes in a pointcloud $\boldsymbol{v}_0 \in \mathbb{R}^{P\times 3}$ and four most recent wrench values from the force/torque sensor $\boldsymbol{h}_0 \in \mathbb{R}^{4\times 6}$. The pointcloud is encoded using a PointNet encoder [93], designed to specifically handle unstructured pointclouds. We encode the wrench inputs by flattening to a vector length 24 and passing through a Multi-Layer Perceptron (MLP) of three layers with hidden sizes 64 and 64, with ReLU non-linearities. The output of both the visual and wrench encodings are latent vectors of size 64 that are concatenated to yield the final latent state code $\boldsymbol{z}_t \in \mathbb{R}^{128}$. The "Vision-Only" model does not have the MLP for tactile, and instead outputs $\boldsymbol{z}_t \in \mathbb{R}^{128}$ directly from the PointNet encoder.

2. **Dynamics:** The dynamics module has three MLP modules. First is a single layer network to increase the dimensionality of the action $\boldsymbol{a}_t$ from 6 to a vector $\boldsymbol{z}_t^a \in \mathbb{R}^{64}$. This vector is concatenated with the latent vector $\boldsymbol{z}_t$ to yield the combined state action latent vector, $\boldsymbol{z}_t^q \in \mathbb{R}^{192}$. The second MLP module takes in $\boldsymbol{z}_t^q$ and passes through three layers, with hidden sizes 256 and 128 and ReLU non-linearities, yielding the next latent state code $\boldsymbol{z}_{t+1} \in \mathbb{R}^{128}$. The third MLP module takes in $\boldsymbol{z}_t^q$ and passes through three layers, with hidden sizes 128 and 64 and ReLU non-linearities, yielding the action offset $\Delta\boldsymbol{a}_{t+1} \in \mathbb{R}^6$. The first three terms of $\Delta\boldsymbol{a}_{t+1}$ is the translation and the second three terms are the axis-angle rotation for the delta action. The "No Offset" model does not contain this last MLP module, as it does not predict the offset action.

3. **Decoder:** The decoder module has three MLP modules that predict each component of the contact feature. Each network takes in the current latent state code $\boldsymbol{z}_t \in \mathbb{R}^{128}$. The first MLP is a classification head, predicting the binary contact state. The network has a single hidden layer of size 64, with a ReLU non-linearity. The final prediction is passed through a Sigmoid to recover the likelihood of $c_t^b = 1$, i.e., likelihood the system is in contact. The second MLP regresses the contact lines. The network has 3 layers, with hidden sizes 128 and 64 and ReLU non-linearities. The output of the network is a vector $\boldsymbol{c}^l \in \mathbb{R}^6$ which is reshaped to be $\boldsymbol{c}^l \in \mathbb{R}^{2\times 3}$, interpreted to be the two

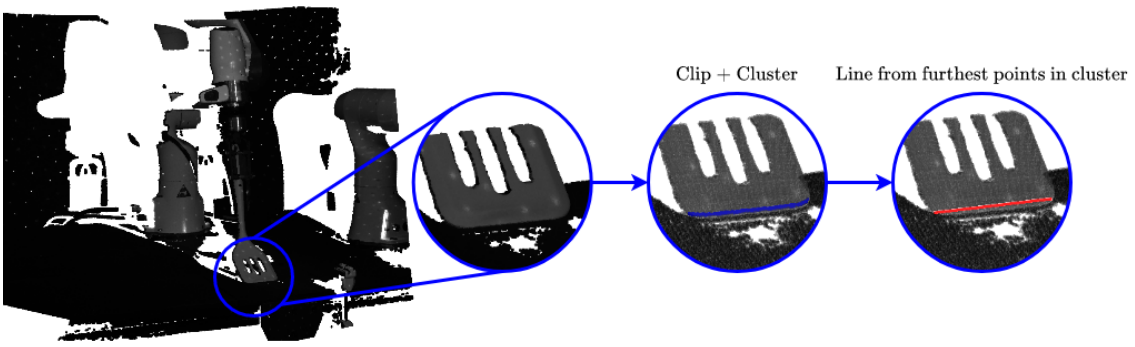


Figure A.2: Examples of labeling contact lines automatically from high fidelity point clouds.

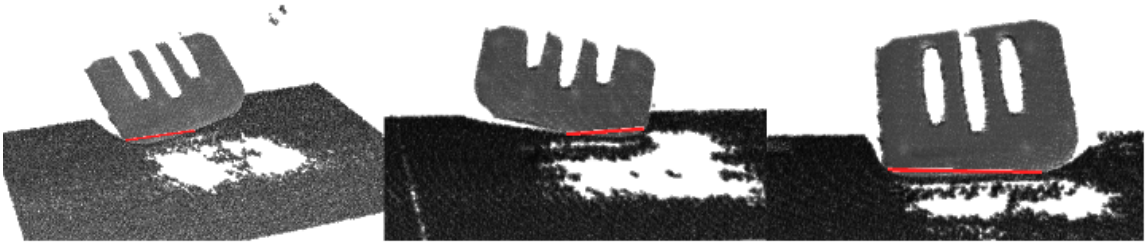


Figure A.3: Labeled contact lines from various tool-environment contact scenarios. Our labeling procedure automatically derives accurate contact lines.

endpoints of the contact line in 3D space. The final MLP regresses the end-effector wrench. The network has 3 layers with hidden size 128 and 64 and ReLU non-linearities. The final prediction is a wrench $\boldsymbol{c}_t^w \in \mathbb{R}^6$.

The detailed module architectures are shown in Fig. A.1.

A.2 Data Collection

A.2.1 Extrinsic Contact Dynamics Labeling Examples

Here we show qualitative examples of labeled contact lines, collected on a real world system as described in Sec. 3.3.3. Fig. A.2 visualizes our procedure of using a Photoneo PhoXi 3D Scanner to recover contact lines from high fidelity pointclouds.

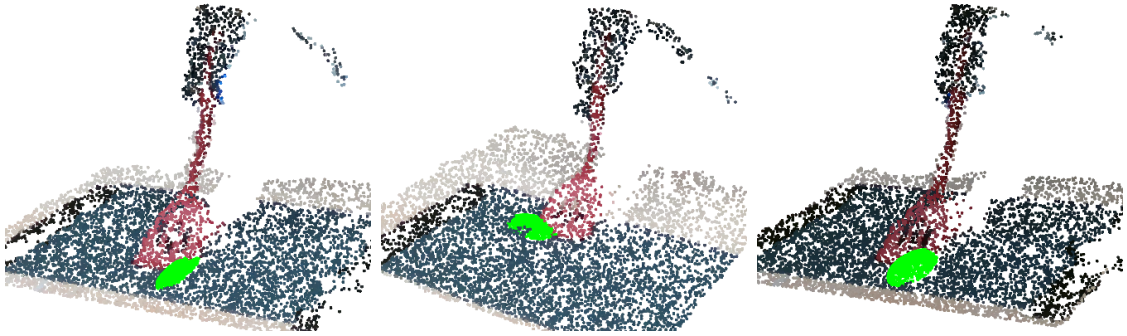


Figure A.4: Examples of input pointclouds augmented with randomly sampled ellipsoids (in green) to encourage robustness to visual occlusions. Note: color is not input to our model.

Fig. A.3 shows several examples of labels from different tool-environment scenarios. We see our labeling procedure can automatically recover accurate contact lines.

A.2.2 Data Augmentation

Here we show examples of augmented pointclouds from our dataset, as described for the "With Obstacles" case in Sec. 3.4.4. Fig. A.4 shows examples of pointclouds with randomly generated ellipsoids inserted to provide occlusions during training. Note, that while the ellipsoids here are colored green, we only input the point positions into the network.

A.3 Additional Results

A.3.1 Modeling Contact Feature Dynamics

A.3.1.1 Torque Prediction Results

In Fig. A.5 we show the performance of all models discussed in Sec. 3.4.3 on predicting end effector *torque*. Similar to force prediction quality, "Full Model" outperformed "Vision-Only" due to having access to wrench inputs. This makes predicting 0th step wrench equivalent to reconstruction, and helps the model predict future torques accurately. "Vision-Only" performs better at predicting future wrench

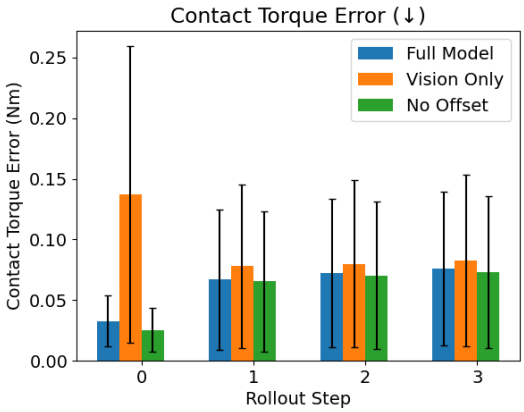


Figure A.5: Model performance in future end effector torque prediction. Similar to the case of force prediction, we are able to accurately model future torques. Learning without wrench inputs struggles to accurately predict future torques.

values - we think this could be because the *action* is highly discriminative with regards to the resulting wrench.

A.3.1.2 Performance on Unseen Tool Data

We also test our model performance when running our model on data from a tool unseen during training (right-most tool in Fig. 3.5). The prediction performance is shown in Fig. A.6. Overall, the results indicate that our proposed method generalizes well to the unseen tool, with high accuracy on binary contact, roughly 1cm error on the contact line, and similar errors on force and torque as those found for the training tools. Additionally, our method outperformed the rigid body baseline (Sec. 3.4.2) on the contact line error and performed comparably on binary accuracy. Note that the baseline here is given access to the unseen spatula geometry still, and thus has more information than our method.

A.3.2 Obstacle Scraping Results

Here we show qualitative results of our target object footprint metric. Fig. A.7 shows the masks generated before and after several scraping examples that show how

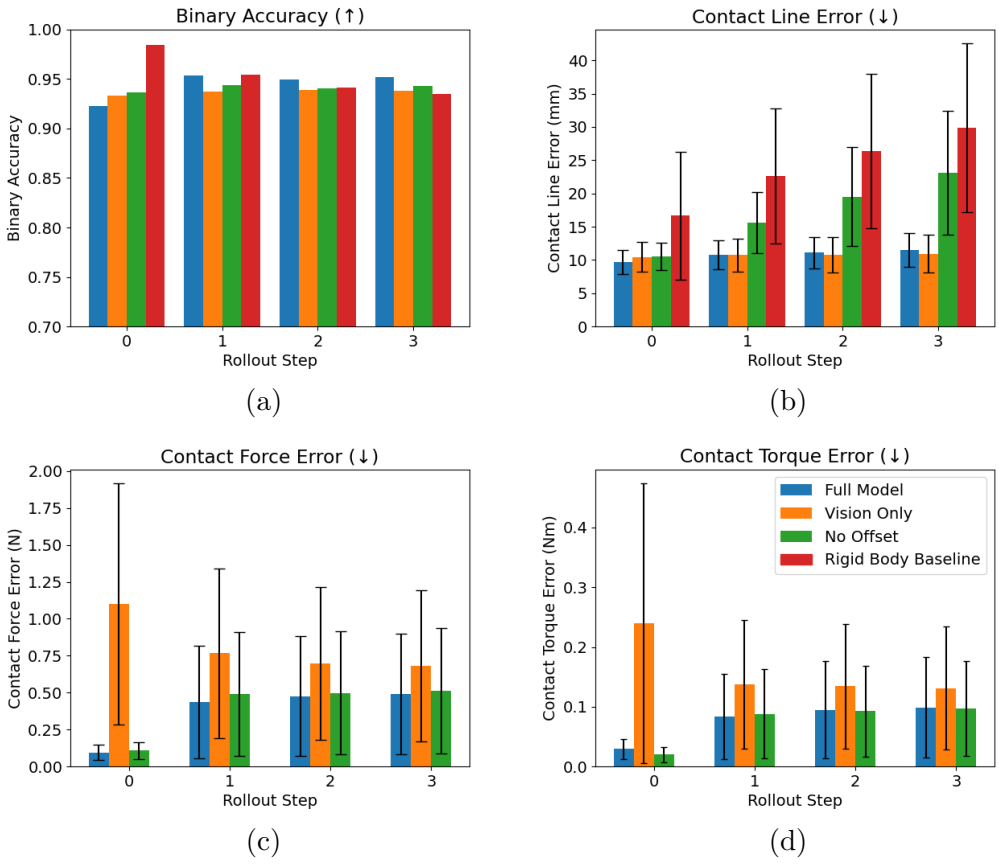


Figure A.6: Contact Feature Dynamics Performance on Unseen Tool

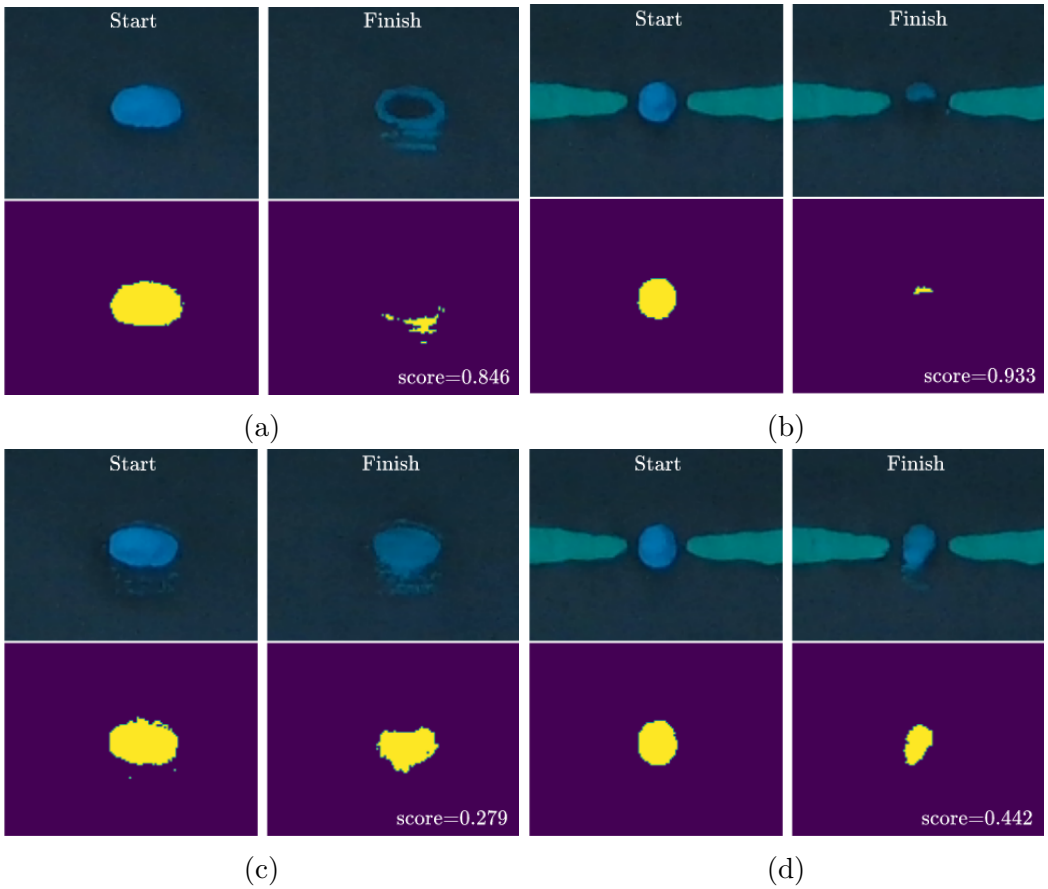


Figure A.7: Qualitative Results showing pre and post-scrape footprint masks, used to generate Percent Footprint Removed metric. The corresponding score is shown for each run.

we estimate footprint removed. These also convey the difficulty of this metric; even small errors in the contact line can leave residue which is picked up by our masking procedure, lowering the score.

We segment the starting object footprint using a binary segmentation method, tuned by hand to accurately capture the starting footprint. For the finished object footprint, we compare each pixel in the color space to the rough nominal color of the target object. We then apply a threshold to choose which points we consider to still contain the object. We manually select a threshold such that a very small amount of remaining residue is not captured, while thicker remaining areas are penalized.

We balance this metric with the percent mass removed metric. The mass metric is easier to perform well on, as the residue left on the surface often has far less mass than the overall obstacle object to be removed. Between the two metrics, we believe our results show early indication that our method of modeling tool-environment interactions allows us to solve interesting contact servoing tasks, even in the presence of visual occlusions and novel reaction forces.

APPENDIX B

Appendix for Chapter 4: Integrated Object Deformation and Contact Patch Estimation from Visuo-Tactile Feedback

B.1 Contact Patch Performance Details

We provide further details on the simulated contact patch Chamfer Distance (CD) performance. In Fig. B.1 we show the spread of the contact patch CD as a boxplot. Our method shows better median and quartile performance than the baseline. Our method also outperforms on average, but we see that our average is pulled up by outliers. This also explains the higher standard deviation in Tab. 4.1.

In Fig. B.2, we show our method and the baseline on our method's outlier example from Fig. B.1. This is a difficult example due to its small deformation and contact, as well as the split contact, which can alias other contact configurations. We see that our method predicts a patch that could have a similar wrench feedback to the composite wrench of the actual interaction. The baseline prediction has a cluster of points that is similarly located, but the baseline's tendency to predict a noisy spread of points means the evaluation is much more kind to the baseline, with a CD of $103.5mm^2$, than to our method, with a CD of $435.9mm^2$, despite the fact that neither method

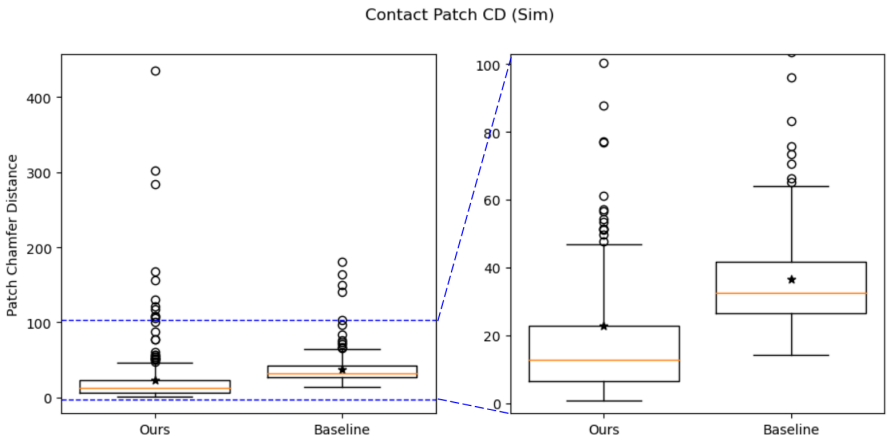


Figure B.1: Boxplot of Simulated Test Contact Patch CD. Stars indicate the mean of each method. The right pane shows a zoomed view to highlight performance details. Our method outperforms the baseline on mean, median, and quartile performance, but does have outliers with high error.

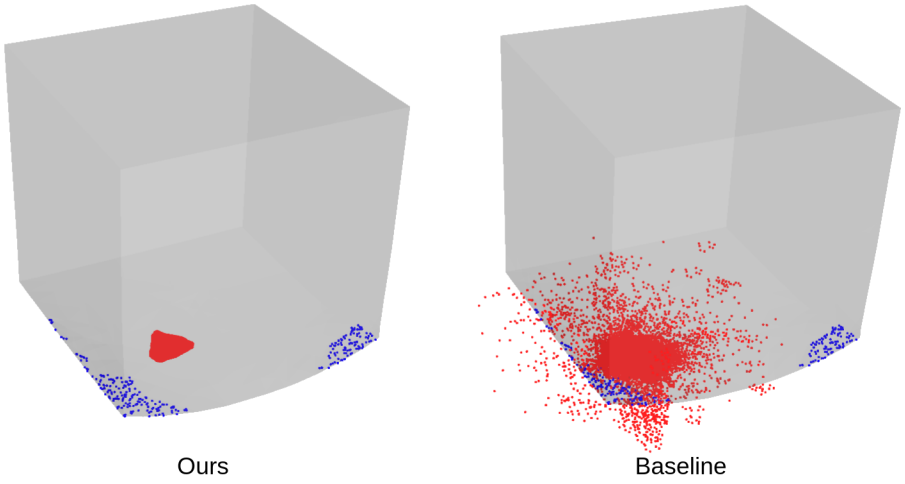


Figure B.2: Contact Patch predictions (in red) vs. ground truth (in blue) for our method and the baseline on our method's outlier result in the simulation dataset. We see that our method predicts a patch that could have similar composite wrench feedback and similar geometry.

predicted accurately on this example (note, as explained in Sec. 4.5, each point cloud is sampled to 300 points before evaluating CD). This result helps contextualize why our method has higher error on a small number of outlier examples.

We found that for most examples, however, as shown qualitatively in Fig. 4.7 and in the distribution of CD values in Fig. B.1, our method performs favorably to the baseline.