Redefining the Perception-Action Interface: Visual Action Representations for Contact-Centric Manipulation

Thomas Weng

CMU-RI-TR-23-73

September 29, 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
David Held, *CMU*, Chair
Oliver Kroemer, *CMU*
Shubham Tulsiani, *CMU*
Alberto Rodriguez, *MIT*

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

To my grandfather Xiyu Weng 翁锡瑜,
for teaching me to dream, like him,
of inventions and possibilities.
Ah-gong, this work is yours.

Abstract

In robotics, understanding the link between perception and action is pivotal. Typically, perception systems process sensory data into state representations such as segmentations and bounding boxes, which a planner uses to plan actions. However, these state estimation approaches can fail in environments with partial observability, or in cases with challenging object properties like transparency and deformability. Alternatively, sensorimotor policies directly convert raw sensor input into actions, but they produce actions that are not grounded in contact, and perform poorly in unseen task configurations.

To address these shortcomings, we delve into visual action representations, a class of approaches in which the perception system conveys information to the planner about potential actions. Visual action representations do not require full state estimation, generalize well to unseen task configurations, and output object-centric actions, reasoning about where to make contact with an object, how to approach contact locations, and how to manipulate the object once contact is made. Reformulating the role of perception to include action reasoning simplifies downstream planning.

This thesis presents visual action representations for addressing visual and geometric challenges in manipulation. We devise a transfer learning method for grasping transparent and specular objects, and present Neural Grasp Distance Fields for 6-DOF grasping and motion planning. We then introduce algorithms for cloth manipulation, starting with adapting semantic segmentation for the task of grasping edges and corners of cloth. Next, we develop a tactile sensing-based closed-loop policy to manipulate stacked cloth layers. Finally, we present FabricFlowNet, a policy that learns optical flow-based correspondences for goal-conditioned, bimanual cloth folding.

Acknowledgments

First, I would like to express my deepest appreciation to my advisor, David Held. Dave has been a steadfast source of support and guidance throughout my PhD. Dave always made time to discuss research and provide feedback, reflecting his dedication to mentorship and teaching. His organized and methodical approach to research is rare and something I hope to emulate in my own work. I am extremely grateful for the opportunity to work with and learn from him.

I thank Oliver Kroemer, Shubham Tulsiani, and Alberto Rodriguez for serving on my thesis committee. Their feedback and guidance were invaluable in shaping my research and thesis. Collaborating with Oliver on multiple projects has been a great privilege. Shubham's ability to dive deep on technical questions is truly inspiring. I hold in high esteem Alberto's kind and thoughtful approach to advising. I also thank Christopher Atkeson and Mohit Sharma for serving on my qualifier committee.

I have a deep appreciation for Mustafa Mukadam and Franziska Meier for their mentorship and collaboration during my time as a visiting researcher at Meta AI. Mustafa is a gracious mentor with wide-ranging knowledge, and I learned so much under his guidance. Franziska's insightful feedback helped us frame our research contributions most effectively. I would also like to thank others that I had the pleasure of working with and learning from at Meta, including Taosha Fan, Austin Wang, Yixin Lin, Priyam Parashar, Tess Hellebrekers, Chris Paxton, Vikash Kumar, and others.

I thank the mentors who guided my path to the PhD, Henny Admoni, Maya Cakmak, and Reid Maker. When I was a Yale undergraduate, Henny took me under her wing and introduced me to robotics research, and has continued to mentor me on this path. Maya gave me the opportunity to work with her at UW, and I am deeply grateful for the time and energy she invested into my growth as a researcher. Reid was the best manager I could have had at Microsoft, and I try to emulate his principles of management and engineering in my own work. I also thank Brian Scassellati, Siddhartha Srinivasa, Stefanos Nikolaidis, Leah Perlmutter, Yasaman Sefidgar, Bryan Ford, Ennan Zhai, Anunay Kumar, and all others who influenced me at Yale, UW, and Microsoft.

To the members of R-PAD that had the pleasure of working with, thank you for your hard work and dedication to our research: Daniel Seita, Brian

Okorn, Yufei Wang, Aurora Qian, Sujay Bajracharya, Jenny Nan, Amith Pallankize, Mansi Agarwal, Sashank Tirumala, Sarthak Shetty, Khush Agrawal, Yimin Tang, Ji Liu, Rashmi Anil, and Patrick Liu. A warm thank you to other members of R-PAD for insightful research discussions and friendship: Xingyu Lin, Sid Ancha, Wenxuan Zhou, Ben Eisner, Jenny Wang, Edward Ahn, Chuer Pan, Carl Qi, Zixuan Huang, Fan Yang, Gautham Narasimhan, Olivia Xu, Harshit Sikchi, Qiao Gu, Harry Zhang, Zhanyi Sun, Bowen Jiang, Pranay Gupta, and others.

To the friends I made on the journey, thank you for all the laughs and camaraderie that made the path lighter: Cherie Ho, Suddhu Suresh, Ada Taylor, Raunaq Bhirangi, Ceci Morales, Ceci Padilla, Pragna Mannam, Leo Keselman, Abhijat Biswas, Kate Shih, and others; the Japan crew Mike Lee, Ravi Pandya, and Michelle Zhao; Gloomhaveners Brian Okorn, Ankit Bhatia, and Ben Newman; the *VR Cult* consisting of Xuning Yang, Alex Spitzer, Cherie, and Suddhu; Meta "Interns Internal"; Adelyn Yeoh and the climbers; and finally my brazilian jiu jitsu partners and teachers at StoutPGH.

I thank my partner Violet Wang for putting up with me when I'm being silly to make her laugh, for supporting me in stressful times, and simply for loving me as I am. To my sister Mary and brother-in-law Tim, thank you for being the best sister and brother I could have. Finally, I thank my parents Guoen and Shanna Weng, for their unwavering love and support throughout my life.

Funding

This work was supported by the US Air Force and DARPA (FA8750-18-C-0092), the Office of Naval Research (N00014-18-1-2775), the National Science Foundation (NSF) Smart and Autonomous Systems Program (IIS-1849154), an NSF CAREER Award (IIS-1849154), NSF Graduate Research Fellowship Award (DGE1745016, DGE2140739), an NSF grant (CMMI-2024794), a NASA Space Technology Research Fellowship (80NSSC17K0233), CMU GSA/Provost Conference Funding, and ShanghaiTech University. This work was also supported by the Meta AI Mentorship Program, LG Electronics, Sony Corporation, and the Efort Intelligent Equipment Company.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robots have the potential to transform society by automating physical, labor-intensive work. A longstanding vision for robotics has been to develop intelligent agents that can perform complex tasks in diverse environments. While today's robots are largely deployed in fully structured environments like factory floors to execute repetitive, pre-programmed movements, we would like the next generation of robots to operate in less structured environments, such as logistics hubs, medical facilities, and homes. Such robots could assist us in activities of daily living, for example preparing food, doing laundry, and cleaning, or automate commercial tasks like sorting, packing, and assembly. Robots with these capabilities could reshape the way we live, work, and interact with the world around us. However, a gap remains between existing approaches to robotics and the level of embodied intelligence required to achieve this vision.

A key limitation of existing robotic systems pertains to the design of the interface between perception and action. A traditional paradigm in robotics is to use a *state representation* as the perception-action interface. First, a perception module processes raw sensory information from the environment into a state representation. Examples of state representations include object poses [14, 65, 199], keypoints [50, 92, 121], bounding boxes [77, 149], and segmentations [9, 128]. The state representation is then input into a planning module, which outputs actions to achieve tasks.

While using state estimation as the interface between perception and action has been viable for some tasks, this interface is less effective in more challenging manipulation settings. In environments with partial observability or challenging object properties like transparency and deformability, recovering the full state may be difficult. For example, the deformability of cloth results in a much higher-dimensional configuration space compared to rigid objects, and in many of these configurations, the cloth occludes portions of itself. If our goal is to manipulate cloth, for example by smoothing or folding it, then the previously mentioned representations–object poses, segmentations, bounding boxes–do not capture the state information required to achieve the intended manipulations. Further, planning over state can be difficult in situations with large action spaces and complex dynamics, as is the case with cloth manipulation and other task domains.

A competing approach to the traditional state-based interface is to forego any intermediate representation, and instead process raw sensory inputs directly into predicted actions. Such *sensorimotor policies* are appealing as they can be trained end-to-end using a deep neural network and a task-relevant loss [52, 98]. However, such approaches have their own drawbacks; first, sensorimotor policies are particularly sensitive to out-of-distribution sensor inputs. For high-dimensional sensor inputs like visual data, slight variations in lighting, camera pose, background variation, and other factors can influence the predicted actions and negatively affect performance. Prior cloth manipulation work has found that such a visuomotor policy for cloth smoothing from RGB input generalizes poorly to new configurations [105]. Further, the actions output by sensorimotor policies are often delta end-effector or delta joint angle actions, which are low-level, robot-centric action spaces. Many of the actions in these robot-centric action spaces lead to no improvement in the object-centric and contact-centric tasks that we care about. Finally, sensorimotor policies lack explainability as they do not produce any intermediate representations; raw sensor data is processed directly into actions using a black-box neural network.

Due to the respective limitations of traditional state-based systems and sensorimotor policies, we consider alternatives. In this thesis, we focus on a class of approaches based on *visual action representations*. These approaches redefine the interface between perception and action, where the primary goal of perception is no longer to estimate state, but rather to output potential actions grounded in a

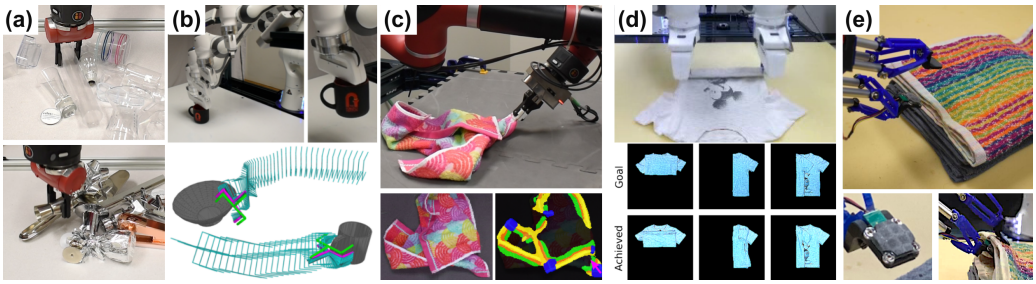


Figure 1.1: A summary of this thesis. Visual action representations can be used for object grasping: (a) grasping transparent and specular objects, and (b) joint 6-DOF grasp and motion optimization. Visual action representations are also presented for cloth manipulation: (c) segmenting and grasping cloth hems, and (d) bimanual cloth folding. (e) Tactile sensing is investigated as a complementary modality to vision for manipulating stacked layers of cloth.

vision-based representation, i.e. a visual action representation. Examples of visual action representations in prior literature include estimating interaction hotspots [135], detecting visual affordances [34], and reasoning about contact locations and end effector trajectories [194]. Visual action representations are suited to environments with partial observability and circumvent the need for full state estimation, as actions can be proposed just on the observable portions of the scene. Approaches using visual action representations generalize better to unseen inputs than sensorimotor policies, because perception and planning are still modularized. Further, due to being anchored to visual representations, the proposed actions are object- and contact-centric, as opposed to the robot-centric actions output by sensorimotor policies. By using object-centric actions, these approaches reason about where to make contact with objects, how to approach contact locations, and how to manipulate objects once contact is established. This new interface between perception and planning enables the perception system to assist the planner by reasoning about potential actions, which can then be used as a starting point for the planner to generate lower-level actions and joint angle trajectories.

In this thesis, we demonstrate several approaches involving visual action representations for manipulating rigid and deformable objects (see Fig 1.1). However, they all share the commonality that the visual output contains additional information about potential actions in the perception output, which is used with planning to improve manipulation in tasks with complex objects or other challenges, like joint

grasping and motion planning. By redefining the perception-action interface to use visual action representations, we enhance the capabilities of robot systems and enable more challenging manipulation tasks.

1.2 Thesis Organization

The chapters of this thesis can be categorized into two parts. The first part presents visual action representations for grasping. In these grasping problems, the objects are assumed to be rigid, and the chapters address challenges such as predicting grasps for objects with non-Lambertian reflectance (e.g. transparency and specularity) or predicting a continuous, implicit grasp set on objects with non-convex geometry.

The chapters in the second portion of the thesis focus on cloth manipulation. Cloth is deformable and has much greater degrees of freedom compared to rigid objects, which complicates both perception and interaction. These degrees of freedom enable a greater number of possible configurations, many of which have self-occlusions. The dynamics of cloth are also more complex than rigid objects, and cannot be represented by single rigid transformation. This thesis develops methods to semantically segment regions of fabric for grasping, grasp a desired number of layers of cloth, and perform bimanual cloth folding.

1.2.1 Visual Action Representations for Grasping

In Chapter 2, we address the task of grasping transparent and specular objects. State of the art approaches for planar grasping typically use depth sensing to infer the geometry of objects for grasp pose prediction. The depth image input is transformed using a convolutional network into a visual action representation, or a per-point "grasp quality" score of executing a grasp at that point. However, these networks fail for transparent and specular objects because depth sensors poorly reconstruct objects with non-Lambertian reflectance. To improve grasping performance for these objects, we leverage the insight that such objects are more visible in the RGB modality than in depth. We use cross-modal network distillation to transfer a trained depth-input grasp network to a new RGB-input network. In the framework of the thesis, we take a visual action representation network trained with depth and distill it to RGB to

improve grasp success for transparent and specular objects. This transfer learning approach requires only a pre-trained depth network and paired RGB-D images. Our grasping experiments demonstrate that our method outperforms depth-only baselines on transparent and reflective objects, while matching baseline performance for opaque objects. This work was previously published in Weng et al. [187].

In Chapter 3, we address the task of joint six degree-of-freedom (DOF) grasp prediction and motion planning. Whereas the previous chapter only considered where to make contact with objects by predicting where to grasp, the method proposed in this chapter reasons about both where to make contact and how to approach with a smooth, collision-free trajectory. The dominant paradigm for grasp and motion planning systems is to first predict candidate grasps without considering whether feasible reaching trajectories exist. Motion planning is then performed for each candidate until a feasible trajectory is found, resulting in brittle, multi-stage pipelines. We represent 6-DOF grasps as a "Neural Grasp Distance Field" (NGDF), a neural implicit function that estimates the pose distance between a 6-DOF end effector pose and the manifold of stable grasp poses for an object. Our distance-to-grasp representation is easily interpreted as a cost, where minimizing the distance to the learned manifold achieves a stable grasp pose. By incorporating NGDF into a trajectory optimizer, we optimize the distance-to-grasp cost along with other costs like smoothness and collision avoidance. During optimization, the grasp location smoothly varies in the continuous set of stable grasp poses, achieving both grasp and motion planning in a single optimization, as opposed to previous multi-stage approaches. The proposed NGDF is an implicit visual action representation, implicitly modeling the distribution of valid grasps on objects. incorporating this implicit neural field into a trajectory optimizer enables joint reasoning about where to make contact for grasping and how to approach desired contact locations. This work was previously published in Weng et al. [190].

1.2.2 Visual Action Representations for Cloth Manipulation

In Chapter 4, we address the task of grasping true edges and corners of cloth from images. For many cloth manipulation tasks, it is important to grasp specific regions of fabric to enable efficient downstream actions. Smoothing and folding cloth, for

example, is more effective manipulating a true cloth edge or "hem", as opposed to a wrinkle. However, it is difficult to identify hems from wrinkles in images of crumpled cloth using image processing techniques like edge detection. We train a segmentation network to predict cloth edges and corners from depth images, using a ground-truth labeling procedure that significantly reduces human time and effort. The output of the segmentation network is treated as a visual action representation, reasoning about where to make contact with the cloth and how to approach the contact location, as the orientation of the cloth edge or corner is important for the task. For each pixel in the segmentation, we estimate the orientation of the cloth hem to determine the best grasp point, taking uncertainty into account. We execute a grasp at the location with minimum orientation uncertainty and align the orientation of the grasp with the estimated orientation of the hem. Our method outperforms baselines and ablations, which either fail to distinguish between hems and wrinkles, or miss the grasp due to incorrect orientation estimation. Our method is invariant to visual textures as it only takes depth images as input, and it generalizes to unseen fabrics with varying material texture. This work was previously published in Qian et al. [145].

In Chapter 5, we propose a closed-loop tactile sensing policy to grasp a target number of stacked cloth layers. When manipulating cloth, grasping a specific number of cloth layers is desirable; for example, it is necessary to grasp two layers of cloth when folding a rectangular cloth in half twice, but when unfolding cloth, it is preferable to grasp just the top layer. We train a classifier on magnetometer-based tactile sensor data to predict the number of cloth layers that are grasped during a pinching action. The contact information provided by the classifier is incorporated into a closed-loop policy that adjusts the approach height until the desired number of layers is grasped. Results show that our tactile-based method outperforms image-based approaches, which are not robust to visual texture, viewpoint shift, or background distractors. Methods that use tactile sensing as an input modality can be a suitable alternative to vision-based approaches for similar lower-level, contact-rich subroutines, while remaining compatible with approaches from other thesis chapters based on visual action representations. This work was previously published in Tirumala et al. [173].

In Chapter 6, we present a method for bimanual cloth folding. One limitation of grasping-focused work is that grasping is often a sub-task for a larger manipulation task like cloth folding: optimizing just the grasping sub-task may not improve

performance for the full task. To address this limitation, we present FabricFlowNet, a policy that predicts both where to grasp and how to manipulate for goal-conditioned, bimanual cloth folding. We adapted the concept of optical flow —a technique typically used for motion estimation between video frames—to the problem of correspondence estimation between images of observed cloth and desired cloth configurations: in other words, our flow predicts where each observed cloth point must move to achieve the desired configuration. We redefine this estimated flow as a visual action representation, where each flow vector represents a pick and place action. FabricFlowNet learns to select the best action for achieving the goal, reasoning about not just where to make contact with the cloth, but how to manipulate it once contact is established. Our approach easily transitions between one- and two-arm actions, to produce clean folds for diverse set of goals. FFN significantly outperforms model-free and model-based methods, and generalizes when trained on a single simulated cloth to real cloths such as towels and T-shirts. This work was previously published in Weng et al. [188].

Chapter 2

Multi-modal Transfer Learning for Grasping Transparent and Specular Objects

Abstract

State-of-the-art object grasping methods rely on depth sensing to plan robust grasps, but commercially available depth sensors fail to detect transparent and specular objects. To improve grasping performance on such objects, we introduce a method for learning a multi-modal perception model by bootstrapping from an existing uni-modal model. This transfer learning approach requires only a pre-existing uni-modal grasping model and paired multi-modal image data for training, foregoing the need for ground-truth grasp success labels nor real grasp attempts. Our experiments demonstrate that our approach is able to reliably grasp transparent and reflective objects. Video and supplementary material are available at https://sites.google.com/view/transparent-specular-grasping.

2.1 Introduction

Robotic grasping is a key prerequisite for a variety of tasks involving robot manipulation. Robust object grasping would enable a wide range of applications in both industrial and natural human environments. The challenge with grasping is that many factors influence the effectiveness of a grasp, such as gripper and object geometries, object mass distribution and friction, and environmental conditions like illumination.

Most state-of-the-art grasping methods rely on depth input from structured light or time-of-flight sensors to determine the best grasp for an object [57, 132, 153]. Under normal operation, such devices emit light patterns onto a scene and use a receiver to construct depth based on changes in the returned pattern. However, such depth sensors fail to detect objects that are transparent, specular, refractive, or have low surface albedo [71], causing depth-based grasp prediction methods to fail. These failures can take the form of both missing depth readings, as is the case with specular objects that deflect structured light patterns, and incorrect depth values, which occur when the emitted light passes through transparent objects (see Fig. 2.1).

Transparent and specular objects are common in a range of environments, such as in manufacturing facilities, retail spaces, and homes. Under certain lighting conditions and object properties, even seemingly opaque objects can exhibit sensor noise similar to transparency and specularity. The ubiquity of objects with these challenging properties requires us to design methods capable of bridging the sensory gap so that robots can robustly grasp a diverse set of objects.

Our contribution in this work is a method for learning to grasp transparent and specular objects that leverages existing depth-based models. Transparent and specular objects are more identifiable in RGB space, where transparencies and specularities produce changes in coloration, rather than the inaccurate or missing values that occur in depth space. Therefore, we make use of both color and depth modalities in our approach. We first train a color-based grasp prediction model from a depth-based one using *supervision transfer* [66], a technique for transferring a learned representation from one modality to another. This transfer technique only requires paired RGB-D images and an existing depth-based grasping method from which to transfer; our method does not require robot grasp attempts nor human annotations.

We conduct real robot grasping experiments on both isolated objects and clutter to

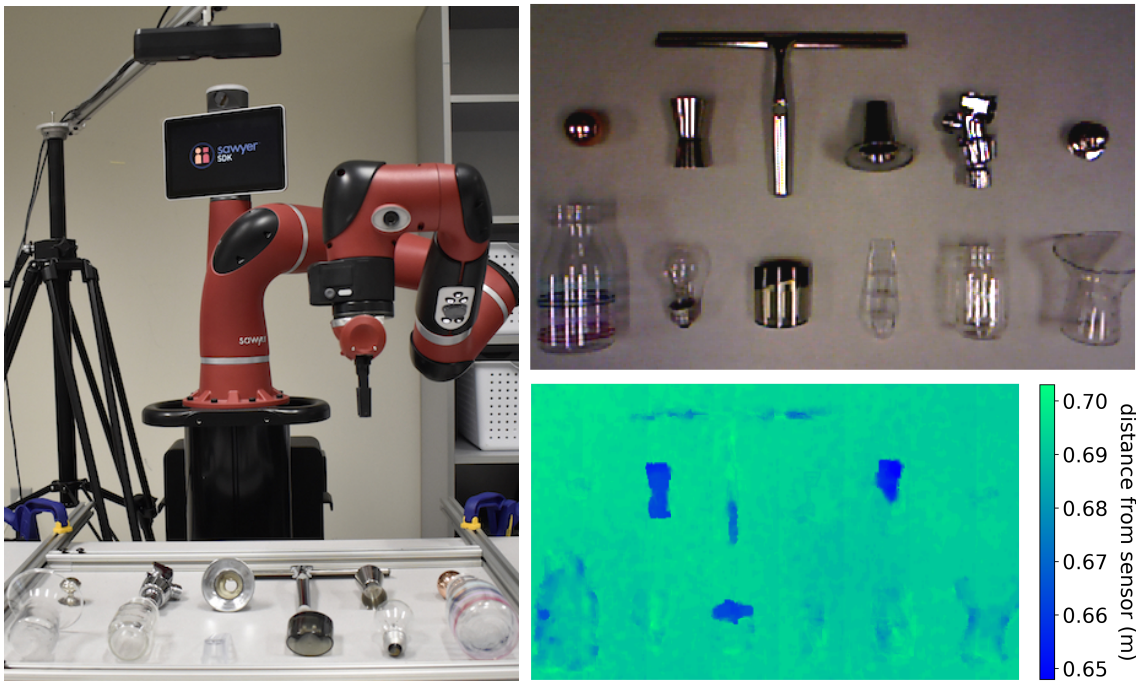


Figure 2.1: Transparent and specular objects provide poor depth readings with conventional depth sensors, posing a challenge for depth-based grasping techniques. (left) Robot workspace with fixed overhead sensor for grasping. (top right) Color image of scene from overhead sensor. (bottom right) Depth image of scene showing that most values in depth image are close to the table.

show that (1) the RGB-only network produces better grasp candidates for transparent and specular objects, compared to the depth-only network that it was trained from, and (2) the RGB-only network is complementary to the original depth model, such that combining the outputs of both models results in the best overall grasping performance on all three object types. We conduct additional experiments to demonstrate the robustness of our method against slight variations in illuminance, and we discuss failure cases as part of our analysis.

2.2 Related Work

2.2.1 Sensing Transparent and Specular Objects

Sensing transparent and specular objects is a well-studied challenge in the computer vision community. Ihrke *et al.* [71] provide a survey of recent approaches to transparent

and specular object reconstruction. Curless *et al.* [25] perform space-time analysis on structured light sensing to achieve better triangulation on transparent objects. Structured light sensing can also be paired with additional equipment like polarization lenses, light fields, or immersion in fluorescent or refractive liquids to detect transparent objects. While structured light sensing is the closest to commercial sensing, the survey also presents methods that improve on multi-view stereo matching to detect transparent and specular objects.

Light field photography for depth reconstruction is another direction for detecting specular and transparent objects [115, 191]. Light field photography has been used in robotics by Oberlin *et al.* [138] applied light field photography to robot manipulation tasks like grasping non-Lambertian objects under running water. However, this method requires capturing a dense set of images in a 3D volume over the scene of interest at both training and test time to construct suitable synthetic images for grasping. In comparison, our proposed method requires a single, static RGB-D sensor, resulting in faster and simpler training and deployment.

Commercial RGB-D sensors (*e.g.,* Intel RealSense, Microsoft Kinect, PrimeSense) use structured-light or time-of-flight techniques to estimate depth. These techniques fail on transparent and specular surfaces, either allowing light emitted by the sensor to pass through or scattering it by reflection. IR stereo and cross-modal stereo techniques have been used to improve depth reconstruction, but the reconstruction quality is still not comparable to that of Lambertian, or diffusely reflective, objects [1, 23, 117]. Lysenkov *et al.* [109, 110] painted over transparent objects to create a dataset of paired transparent and opaque objects, but this approach scales poorly for objects with arbitrary geometries and material properties. Our proposed method is able to use conventional RGB-D sensors without hardware and environmental modifications by combining depth and color information.

2.2.2 Grasp Synthesis

Grasp synthesis refers to the problem of finding a stable robotic grasp for a given object and is a longstanding research problem in robotics. Approaches to grasp synthesis can be classified into analytic and empirical methods; see Bohg *et al.* [9] for a survey. Analytic approaches use physics-based contact models to compute force closure on an

object, using the shape and estimated pose of the target object [125, 172, 186], but work poorly in the real world due to noisy sensing, simplified assumptions of contact physics, and difficulty in placing contact points accurately.

Empirical approaches, on the other hand, learn to predict the quality of grasp candidates from data on a diverse set of objects, images, and grasp attempts collected through human labeling [75, 97, 147, 154], self-supervision [99, 143], or simulated data [33, 57, 116, 131, 153]. Saxena *et al.* [154] trained a classifier on human-labeled RGB images to predict grasp points, triangulated the points on stereo RGB images, and demonstrated successful grasps on a limited set of household objects, including some transparent and specular objects. However, the predicted grasp points for transparent and specular objects were limited to grasps on points where stereo triangulation was successful. The Cornell Grasping Dataset [75], consisting of 1k RGB-D images of objects and human-labeled grasps parameterized as an oriented bounding box, has been used to train many deep learning-based grasping methods [97, 131, 147]. Self-supervised methods such as those by Pinto and Gupta [143] or Levine *et al.* [99] forego the need for human labels by training a robot to grasp directly from real grasp attempts, but these methods require tens of thousands of attempts to converge.

Recently, approaches trained on data gathered in simulation have demonstrated state-of-the-art performance. The Jacquard dataset by Amaury *et al.* [33] uses a grasp specification similar to the Cornell Grasping Dataset, contains simulated objects and grasp attempts, and has been successfully used for training by Morrison *et al.* 's GG-CNN [131]. Mahler *et al.* [116] developed GQCNN, which was trained on a dataset of simulated grasps generated using analytic model, representing a hybrid empirical and analytic approach.

As we will show, these depth-only grasping approaches fail on transparent and reflective objects. Note that GG-CNN could be modified to incorporate RGB images, which could potentially be used to grasp transparent and specular objects after training on simulated images (such as those in the Jacquard dataset [33]); however, such performance has not been demonstrated; this method has only been demonstrated for depth-based grasping of opaque objects. In this work, we build upon the fully convolutional version of GQCNN (FC-GQCNN) proposed by Satish *et al.* [153], but our method is agnostic to the specific network architecture used. Our method does not require any real-world grasps or labeled data but instead relies on supervision

transfer from a pre-trained depth network to obtain a multi-modal grasping method. The pre-trained depth network also may not require real-world grasps or human labels; for example, FC-GQCNN is trained entirely on simulated grasps.

2.2.3 Cross-modal Transfer Learning

Supervision transfer has been explored in the past for tasks such as image classification and object detection [59, 66, 100]. These approaches are typically used to transfer image-based networks trained on ImageNet [32] to depth-based or RGB-D based classification or detection networks. To our knowledge, such approaches have not been used previously in the context of multi-modal grasping. We show that such an approach can lead to greatly improved performance for grasping transparent and reflective objects, and can even improve performance on some opaque objects.

2.3 Approach

Here we describe our approach for supervision transfer, which enables us to transfer a grasping method trained in one modality $\mathcal{M}_d$ to also incorporate an additional modality $\mathcal{M}_s$ without needing any additional real grasp attempts, simulation, nor human-labeled data (other than the data used to train the initial uni-modal grasping method, which in our case is only simulated rendered depth data [153]).

2.3.1 Problem Statement

We assume that we initially have a grasping method that takes input from a given modality $\mathcal{M}_d$, such as depth. Specifically, we assume that we have a grasping method that, given a candidate grasp q and an image I_d of modality $\mathcal{M}_d$ (*e.g.,* a depth image), outputs a grasp score $G(q, I_d)$. We wish to transfer this scoring method to a new input modality $\mathcal{M}_s$ (*e.g.,* RGB). Ideally, this new modality $\mathcal{M}_s$ will allow our grasping method to succeed in grasping certain types of objects (e.g. transparent and specular) where the previous modality, $\mathcal{M}_d$, failed. In later sections, we will discuss combining these modalities to create more robust grasping methods.

We assume access to a dataset of image pairs (I_d, I_s), where each pair consists of one image from each modality. We assume that each pair of images was taken

at approximately the same time and thus represent images of the same scene under the two modalities $\mathcal{M}_d$ and $\mathcal{M}_s$. Paired images for RGB and depth modalities can be captured using commercially available RGB-D sensors (*e.g.,* Intel RealSense, Microsoft Kinect, PrimeSense).

Note that these paired images can be collected without needing to perform any grasp attempts or human labeling, making the collection of this dataset very efficient. Furthermore, because these paired images are collected in the real world, they contain all of the real-world noise and artifacts that one would encounter in a realistic setting, avoiding the need to create such artifacts in simulation.

2.3.2 Supervision Transfer for Multi-modal Perception

In attempting the modality transfer described above, we observe the following: different input modalities (*e.g.,* depth vs RGB) have complementary advantages. In other words, data that is difficult for computing successful grasps in one modality might not be as difficult for another modality, and vice versa. For example, transparent and reflective objects are extremely difficult for depth-based grasping methods, due to the resulting noise or missing data in the depth image. However, our experiments show that RGB-based grasping methods have a much higher success rate for these objects. On the other hand, highly textured objects may present difficulties for RGB grasping methods, but these textures do not manifest in depth-based methods.

Based on this observation, we first filter our dataset D into a new dataset D' for which we expect the grasping method of modality $\mathcal{M}_d$ to perform well. In other words, for images $I_d \in D'$, the grasp score $G(q, I_d)$ should have a high correlation with the success of an executed grasp. In our case, because I_d is a depth image, our filtered dataset D' contains only images of opaque objects, for which depth-based grasping methods typically perform well.

We then train a grasping method for modality $\mathcal{M}_s$ (*e.g.,* RGB) using supervision transfer [59, 66, 100] over dataset D'. For each paired image (I_d, I_s) in dataset D', we compute the grasping score $G(q, I_d)$ for the modality $\mathcal{M}_d$. Because of our filtering, this grasp score is likely to be accurate. We then train a method for computing the grasping score $G_\phi(q, I_s)$ of the second modality $\mathcal{M}_s$ using the grasp score from

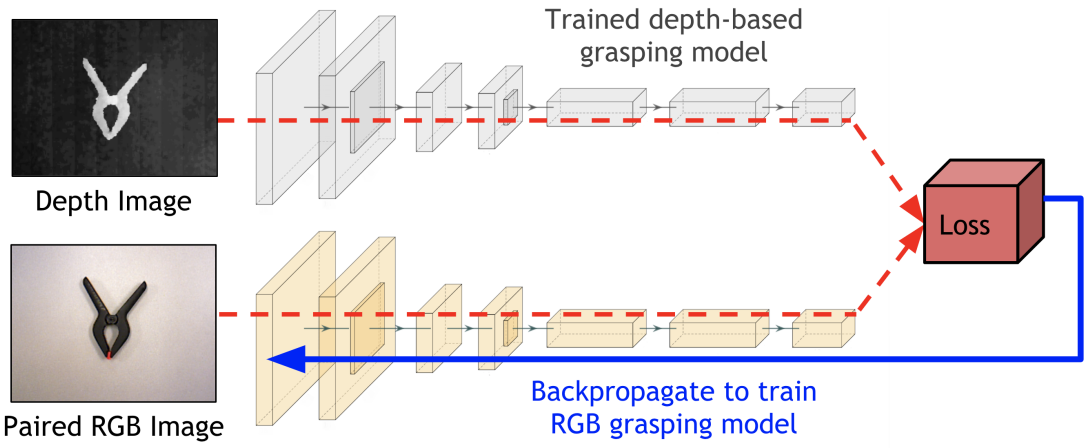


Figure 2.2: We train a grasp quality CNN that takes RGB input by supervising the loss of the network on the output of a trained depth model for paired, unlabeled RGB-D image data.

modality $\mathcal{M}_d$ as the grasp label; thus we define the loss to be

$$\mathcal{L}(\phi) = ||G(q, I_d) - G_\phi(q, I_s)||^2 \tag{2.1}$$

For paired images of dataset D', we train the grasping method on the new modality $\mathcal{M}_s$ (*e.g.,* RGB) to output the same grasping score as the score output of the previous grasping method on the original modality $\mathcal{M}_d$ (*e.g.,* depth). This procedure is shown in Figure 2.2.

Because of the complementary nature of the two sensors, this grasping score function will often perform well on data that was originally filtered out of D and not included in D', even though $G_\phi(q, I_s)$ was only trained on data from D'. Specifically, we filter out transparent and reflective objects from D' because depth-based grasping methods perform poorly on these objects. Nonetheless, the image-based grasping method $G_\phi(q, I_s)$ still performs well on images of transparent and reflective objects, because the difference in appearance for these objects in the RGB modality is much smaller than the difference in appearance for these objects in the depth modality. Our experiments confirm this to be the case.

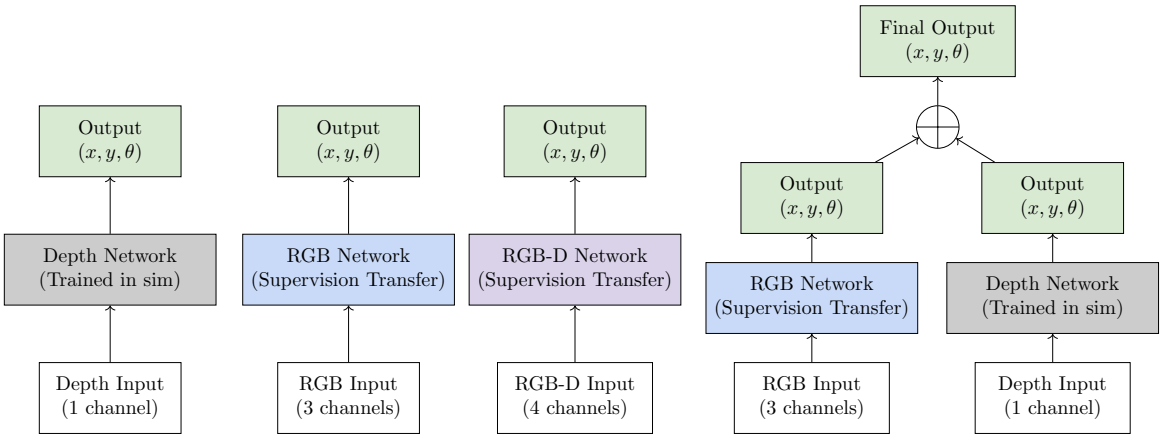
Further, because the modalities are complementary, we show that we can get the best performance by combining the grasping scores from the two modalities. Although

there are many potential ways to do this, we evaluate two possibilities. The "early fusion" approach for combining modalities is to transfer from a depth-based grasping network to a RGB-D grasping network ("RGBD-ST", see Fig. 2.3c). RGBD-ST takes as input both depth and RGB modalities concatenated together. For our second, "late-fusion" approach, we fuse the scores of each modality, averaging the outputs of the depth-based grasping network with a RGB-based grasping network trained using supervision transfer. We define the multi-modal grasping score as

$$G_\phi(q, I_d, I_s) = \frac{1}{2} \cdot (G(q, I_d) + G_\phi(q, I_s)) \tag{2.2}$$

This method is referred to below as "RGBD-M" (see Fig. 2.3d). Both of these approaches share the benefits that they represent multi-modal grasping methods that were trained from a depth-based grasping method only using paired RGB and depth images, without requiring real grasp attempts or human labels.

2.3.3 Implementation of Supervision Transfer



(a) FC-GQCNN (b) RGB-ST (c) RGBD-ST (d) RGBD-M

Figure 2.3: Diagrams of the four methods evaluated in this work. We compare against FC-GQCNN [153], which takes a depth image as input and outputs dense grasp scores over image coordinates x, y and rotation θ about the depth axis. RGB-ST and RGBD-ST are both trained using supervision transfer, but differ in the input they accept (3-channel RGB or 4-channel RGB-D input). RGBD-M takes the outputs of the RGB and Depth networks and averages them to produce the final output.

Our supervision transfer formulation is agnostic to the specific grasping method or representation we use for grasping in modality $\mathcal{M}_d$. For this work, we use the Fully Convolutional Grasp Quality CNNs (FC-GQCNN) representation as the pre-trained depth model from Satish *et al.* [153], although other depth-based grasping methods could equivalently be used.

FC-GQCNN learns a function $G(q_d, I_d)$ which predicts a grasp success rate for each grasp q_d based on a depth image I_d. In FC-GQCNN, grasps q_d are parameterized as $q_d = (x, y, \theta, z)$, where x and y are horizontal planar coordinates designating the desired grasp point of the gripper, z is the grasp depth relative to the camera, and θ is the clockwise rotation angle of the gripper about the vertical z axis. FC-GQCNN takes as input just a single depth image I_d and outputs a 4-dimensional tensor of grasping scores, producing one score per binned (x,y,z) position as well as binned orientation coordinates θ. FC-GQCNN is designed to be fully convolutional in order to output dense predictions $G(q_d, I_d)$ across the entire depth image. Our methods, shown in Figure 2.3, use a similarly dense (x, y) output and the same output angular encoding θ.

We wish to use the output of FC-GQCNN to train an image-based grasping method $G(q, I_s)$. Because the image modality does not have access to depth information, for image-based grasping we change the grasping parameterization to just $q = (x, y, \theta)$, without including a parameter for the grasp depth z. With this specification, each grasp starts at an approach height and moves down until it makes contact with either the table or an object before closing the gripper. Due to the difference in grasp representations, we modify our loss slightly, to be:

$$\mathcal{L}(\phi; q, I_d, I_s) = || \max_z G((q, z), I_d) - G_\phi(q, I_s)||^2 \tag{2.3}$$

where (q, z) is the concatenation of z to a grasp $q = (x, y, \theta)$ to form the new grasp representation (x, y, θ, z). In other words, to compute the target grasp score for some grasp $q = (x, y, \theta)$, we append various depths z to form a depth-based grasp parameterization (x, y, z, θ); for each of these grasp parameterizations we can compute the depth-based grasping score $G((q, z), I_d)$ using our depth-based grasping method (e.g. FC-GQCNN). We then compute the maximum grasp score over the values of z to obtain $\max_z G((q, z), I_d)$.

The network architecture that we use for image-based grasping is very similar to the architecture used in FC-GQCNN for depth-based grasping (see Appendix A). The only modification that we make is that we modify the first layer to accept a 3-channel RGB input rather than a 1-channel depth input. This is accomplished by adding an extra dimension to the first layer convolutional filters. In some of the experiments, we will alternatively use an RGB-D grasping network ("RGBD-ST"), in which case we modify the first layer to accept a 4-channel input, in a similar manner.

2.4 Experimental Setup

Following the reproducibility guidelines for grasping research as presented in [117], we describe our experimental setup and protocols below.

2.4.1 Physical Components

We use an ASUS Xtion Pro Live RGB-D sensor, fixed 0.7 m above and pointing down towards the workspace (see Fig. 2.4). Robot experiments were performed on a 7 DOF Rethink Robotics Sawyer robot equipped with an electric parallel jaw gripper, though our method can be applied to other robots and end-effectors. The robot's workspace is an approximately 0.65 m $\times$ 0.38 m area that is reachable by the robot with a vertical grasp. Aluminum extrusions enclose the workspace to prevent objects from rolling or sliding out of the space.

All experiments and network training were performed on an Ubuntu 16.04 machine with an NVIDIA GTX 1080 Ti GPU, a 2.1 GHz Intel Xeon CPU, and 32 GB RAM allocated per job. Grasp planning was implemented using off-the-shelf MoveIt! software.

2.4.2 Training the Network

We first collected a set of 100 opaque objects from home and office retail stores. Using the ASUS Xtion Pro Live RGB-D sensor fixed above the workspace, we captured 200 paired RGB-D training images and 50 paired validation images of the objects in varying amounts of clutter and with lighting conditions ranging from

standard office illuminance (approx. 500 lux) to dimmed illuminance (approx. 175 lux). We resized the images to account for differences between our sensor's intrinsic parameters and those of the pretrained FC-GQCNN model. To increase the amount of training data and improve domain robustness, we applied spatial augmentations (*e.g.,* random rotations and flips) and color-based augmentations (*e.g.,* hue, brightness, and contrast), generating approximately 20k paired training images. This image dataset is available at the URL in the abstract.

The network architecture was implemented in Python using Tensorflow and Keras. The RGB or RGB-D network's weights were randomly initialized, and the model was trained to convergence using an Adam optimizer with cross-entropy loss [78, 79]. We experimented with mean squared error loss, but it performed worse in initial experiments. The loss was supervised from the output of FC-GQCNN, taking the maximum over all values of z as discussed in Sec. 2.3.3. Hyperparameters are provided in Appendix C.

2.4.3 Test Objects

We collected objects distinct from the training objects to form three sets of 15 test objects each, one set per category (see Fig. 2.4). For the opaque object set, we primarily use YCB [15] objects that fit within the 5 cm stroke width of our gripper. We collected our own transparent and specular object sets due to the lack of existing benchmark sets for these categories.

Following typical procedures for grasping evaluations [116, 180], we remove bias related to object pose through the following procedure: objects are shaken in a box and then emptied onto the robot's workspace for each grasp attempt. This procedure is used for both isolated object grasping as well as for grasping in clutter.

2.5 Experimental Results

We design experiments to answer the following questions:

- To what extent can supervision transfer be used to grasp objects from new modalities (e.g. depth to RGB)?

Figure 2.4: Data collection setup and example images from the dataset of all three object types.

- To what extent can supervision transfer from depth to RGB be used to learn to grasp transparent and reflective objects?
- Do the depth and image modalities complement each other? That is, will combining both modalities outperform either modality alone?

Note that grasping performance is not directly comparable with previous work like FC-GQCNN [153] as we use a different robot, gripper, and depth sensor.

2.5.1 Multi-modal Perception

We evaluate whether multi-modal perception that combines depth and RGB data is better than uni-modal perception using either depth or RGB data alone. We refer to our method for Depth-to-RGB supervision transfer, described in Sections 2.3.2 and 2.3.3, as "RGB-ST" (see Fig. 2.3b).

We evaluate two approaches to multi-modal perception, both of which are described in Sections 2.3.2 and 2.3.3. The first "early-fusion" approach uses supervision transfer to directly train an RGB-D grasp prediction network from a depth-based network, called "RGBD-ST" (see Fig. 2.3c). The second "late-fusion" approach involves taking the mean of the outputs of an RGB-only network and a depth-based network. Specifically, we take the mean of the RGB-ST and FC-GQCNN grasping networks; we call this multi-modal method "RGBD-M" (see Fig. 2.3d).

The results are shown in Table 2.1. RGBD-ST and RGBD-M both significantly outperform depth-only grasping (FC-GQCNN) on transparent and specular objects, while maintaining comparable performance on opaque objects.

We also see that the multi-modal methods perform similarly to the RGB-based grasping method (RGB-ST) on opaque and transparent objects, but outperform this method on specular objects. These results support the notion that combining

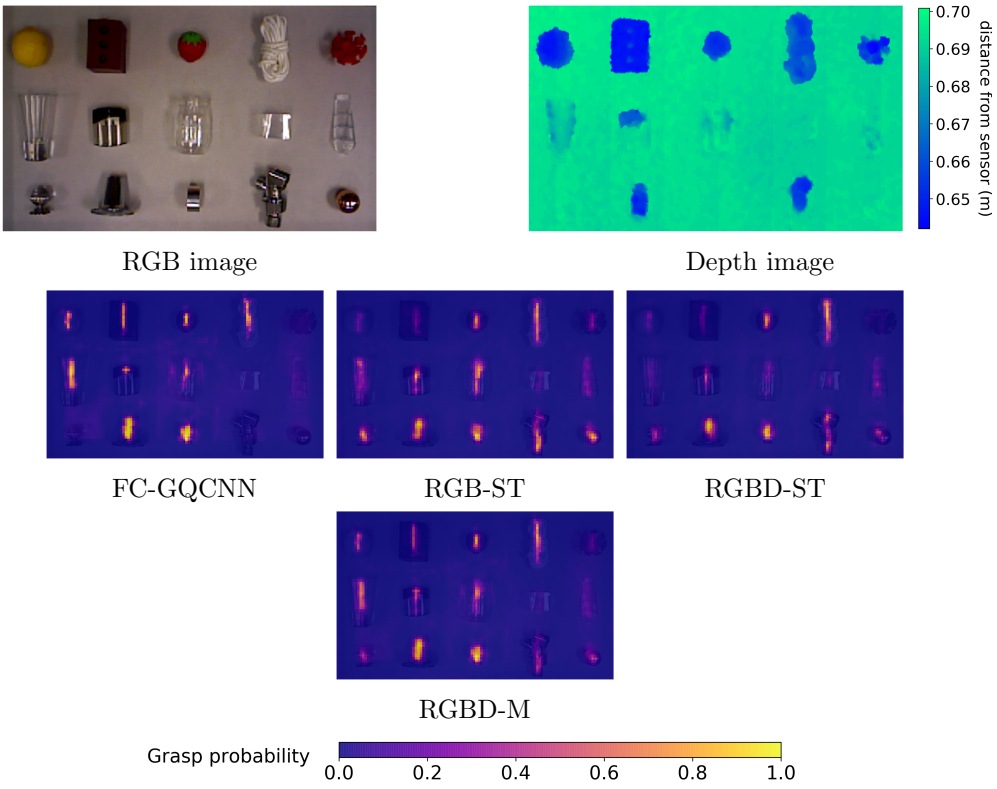


Figure 2.5: Probability heatmaps of grasping across methods for the max grasp score of a grasp with fingertips horizontal to the image, centered at the each pixel. Objects from each set are arrayed horizontally such that the top row is opaque objects, the next transparent, and the final one specular.

both RGB and depth modalities gives better grasping performance than using either modality alone.

2.5.2 Grasping in Clutter

We also evaluated our methods for grasping in clutter, as this is important for robots in various cluttered environments like homes and warehouses. The same test objects used in isolated object grasping were used for clutter experiments. Five trials of grasping in clutter were conducted for each object category. Following the procedure from Viereck *et al.* [180], a trial concluded after all objects were successfully grasped, 3 consecutive failed grasp attempts occurred, or all objects were outside the workspace.

To prevent a network from getting repeatedly stuck on attempting a bad but

Table 2.1: Isolated object grasping, averaged over five trials

Method	Opaque	Transparent	Specular
FC-GQCNN*	$\mathbf{0.92 \pm 0.06}$	0.40 ± 0.08	0.48 ± 0.17
RGB-ST†	0.89 ± 0.04	0.79 ± 0.09	0.71 ± 0.04
RGBD-ST†	0.91 ± 0.06	0.77 ± 0.08	$\mathbf{0.83 \pm 0.04}$
RGBD-M†	0.91 ± 0.14	$\mathbf{0.85 \pm 0.06}$	0.81 ± 0.07

*Trained on simulated grasps
†Trained on simulated grasps and opaque object images

highly rated grasp, we randomly sample a 0.2m square crop of the input image and select the grasp location within that region with the maximum predicted success probability. All methods including baselines performed similarly or worse without this sampling (see Appendix B). Crops whose grasp probabilities all fall below below a threshold are discarded and resampled to avoid attempting grasps based on noisy sensor readings.

Table 2.2: Grasping in clutter, averaged over five trials

Method	Opaque	Transparent	Specular
FC-GQCNN*	0.84 ± 0.06	0.23 ± 0.21	0.35 ± 0.16
RGB-ST†	0.77 ± 0.11	$\mathbf{0.67 \pm 0.10}$	$\mathbf{0.68 \pm 0.12}$
RGBD-ST†	0.86 ± 0.09	0.67 ± 0.27	0.35 ± 0.10
RGBD-M†	$\mathbf{0.97 \pm 0.15}$	0.51 ± 0.32	0.63 ± 0.12

*Trained on simulated grasps
†Trained on simulated grasps and opaque object images

The results are shown in Table 2.2. The results from grasping in clutter corroborate the result of isolated grasping. All methods perform well on opaque objects, although RGBD-M (averaging the output of depth-only grasping and RGB-only grasping networks) performs slightly better than the others. On non-opaque objects (e.g. transparent and specular), FC-GQCNN (e.g. depth-only grasping) performs poorly.

Table 2.2 shows that RGB-ST (RGB-only grasping) and RGBD-M (averaging the output of depth-only grasping and RGB-only grasping networks) perform well across all three object categories. We note that, despite averaging across five trials, the

results of grasping in clutter have relatively high variance and should be considered accordingly. Overall, our main conclusions are similar to that of isolated object grasping from Section 2.5.1: depth-only grasping performs poorly on transparent and specular objects; with supervision transfer, we can obtain a method that performs much better on grasping transparent and specular objects while maintaining similar performance on opaque objects. This method requires only paired RGB and depth images for training and does not require any real grasp attempts or human annotations, other than the simulated depth rendering data that was used to train the original FC-GQCNN [153] depth-based grasping method.

2.5.3 Lighting Variation Experiments

We note that domain shifts like lighting can be a problem for RGB methods, as mentioned in previous work [117]. To enable our method to be robust to lighting variations, our training images were collected with slight lighting variations, and we applied color-based augmentations like brightness and contrast.

We conducted experiments to evaluate the robustness of the trained networks to lighting variations. We varied the lighting by moving a floor lamp around the robot workspace as shown in Fig. 2.6a and performed the isolated object grasping experiments for RGBD-M. The additional lighting increased illumination to between 750 and 950 lux. With this variation in lighting, the RGBD-M network performed comparably, achieving grasp success rates of 0.81 ± 0.12 for transparent objects and 0.79 ± 0.09 for specular ones (compare with Table 2.1).

However, we found that the network performed poorly under more drastic lighting changes, in which we turned off the overhead lights and reduced the height of the floor light, dropping illumination to approx. 175 lux and causing long object shadows to appear. In this case, grasp performance dropped to 0.52 ± 0.18 on transparent objects and 0.60 ± 0.12 for specular ones. In such extreme lighting conditions, we observed the method predicting grasps on shadows for transparent objects (see Fig. 2.6b). Such drastic lighting would not normally occur in structured applications like bin-picking.

(a) Lighting setup.

(b) Extreme lighting.

Figure 2.6: (a) Setup for lighting variation experiments. Lighting is controlled using the overhead lights and floor lamp. (b) Failure case in the extreme lighting condition. The method predicts the best grasp to be on the object's shadow.

2.5.4 Failure Cases

In this section we discuss the most frequent and notable failure cases from our experiments. This section covers failures due to our approach, as well as external factors. Some examples of failure cases discussed in this section can be seen in Fig. 2.7 and the supplementary video.

Methods that used the depth modality like RGBD-ST and RGBD-M at times selected grasps that were highly rated by the depth network, but did not sufficiently account for transparencies or specularities (Fig. 2.7, top left). Both the color-based and depth-based networks at times failed to distinguish very transparent objects from the workspace surface, though this was rare and occurred far less frequently than with FC-GQCNN (Fig. 2.7, top right). Object mass distribution and deformability were not accounted for by our methods (Fig. 2.7, bottom row).

A failure case external to the methods evaluated involved our gripper hardware. Our parallel electric gripper has a relatively small stroke width, and is unable to execute pinch grasps with a 5cm opening width. This limitation causes grasps on thin parts of objects to fail, because the fingertips do not completely come together. While it is possible to adjust the fingertips to be closer together to enable pinch grasps, the opening width of the gripper would be reduced, which would prevent the gripper from being able to grasp large objects. This issue reduced performance across

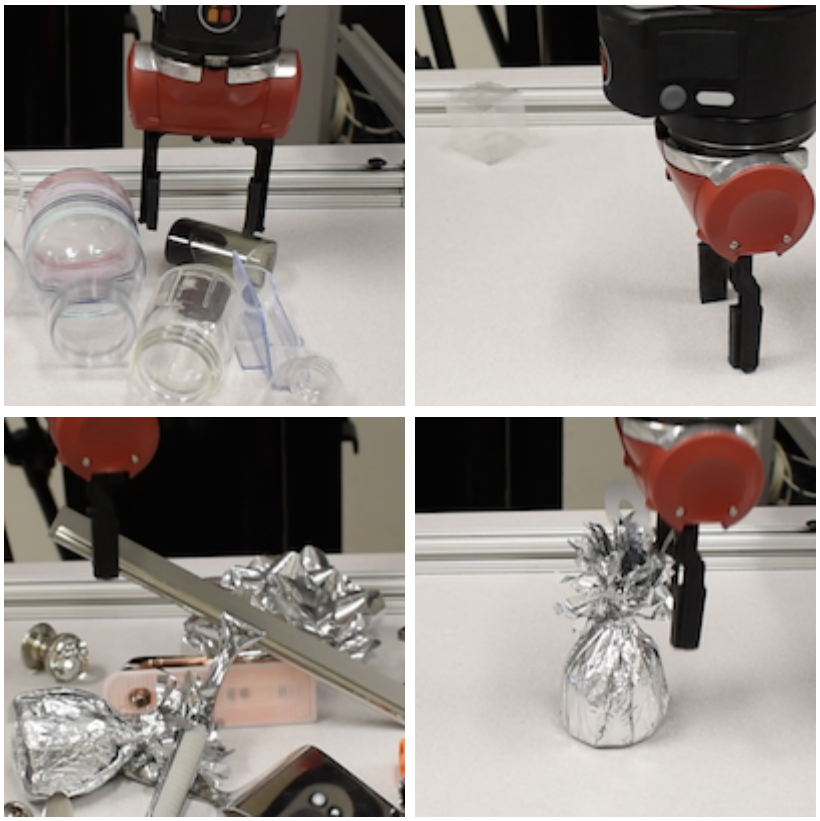


Figure 2.7: Examples of failure cases. (top left) Grasp does not account for transparent part of sharpener. (top right) Gripper fails to detect transparent plastic cube and grasps at table. (bottom left) Mass distribution of squeegee causes grasp to fail. (bottom right) Foil on top of balloon weight appears graspable but the gripper passes through.

all methods and would likely be mitigated by other grippers.

Since our paper focused on static grasping, our method fails to grasp objects that start rolling due to perturbation in clutter. Others have investigated ways to address this issue using closed-loop control techniques like visual servoing [132].

2.6 Conclusion

We present an approach for improving grasping on transparent and specular objects, for which existing depth-based grasping methods perform poorly. Our method transfers information learned by a depth-based grasping network to RGB or RGB-D networks, enabling multi-modal perception. Our method for supervision transfer

requires only real-world paired depth and RGB images, and does not require any human labeling nor real-world grasp attempts. We explore two avenues to multi-modal perception and demonstrate that making use of the RGB modality outperforms depth-only grasping in isolated object grasping as well as grasping in clutter. The method is extensible to other robots, environments, and end effectors. One potential direction for future work may be to adaptively weight predictions from different modalities instead of averaging them. Another is applying transfer learning techniques to other, less similar modalities like haptics and tactile feedback. Combining different sensor modalities might also be useful in determining the appropriate grasp height for each object.

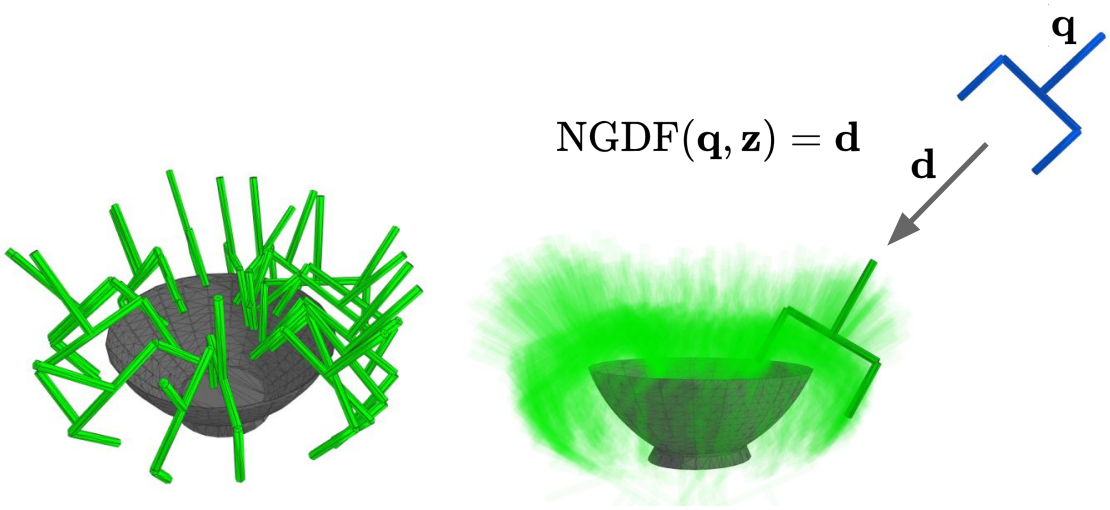
While we are able to get improved performance without using any real grasping data, we believe that real grasps can be used to further improve the performance of the network. We are also interested in extending this work to other types of grasping, such as 6-DOF, multi-fingered, or suction grasping.

Chapter 3

Neural Grasp Distance Fields for Robot Manipulation

Abstract

We formulate grasp learning as a neural field and present Neural Grasp Distance Fields (NGDF). Here, the input is a 6D pose of a robot end effector and output is a distance to a continuous manifold of valid grasps for an object. In contrast to current approaches that predict a set of discrete candidate grasps, the distance-based NGDF representation is easily interpreted as a cost, and minimizing this cost produces a successful grasp pose. This grasp distance cost can be incorporated directly into a trajectory optimizer for joint optimization with other costs such as trajectory smoothness and collision avoidance. During optimization, as the various costs are balanced and minimized, the grasp target is allowed to smoothly vary, as the learned grasp field is continuous. We evaluate NGDF on joint grasp and motion planning in simulation and the real world, outperforming baselines by 63% execution success while generalizing to unseen query poses and unseen object shapes. Project page: https://sites.google.com/view/neural-grasp-distance-fields.



(a) Discrete Grasp Set (b) Continuous Grasp Manifold with NGDF

Figure 3.1: (a) Existing grasp estimation methods produce discrete grasp sets which do not represent the true continuous manifold of possible grasps. (b) Our work, Neural Grasp Distance Fields (NGDF), learns a continuous grasp manifold: given a query pose $\mathbf{q}$ and an object shape embedding $\mathbf{z}$, NGDF outputs the distance $\mathbf{d}$ between $\mathbf{q}$ and the closest grasp. This distance can be leveraged as a cost for optimization, facilitating joint grasp and motion planning.

3.1 Introduction

We present ***N**eural **G**rasp **D**istance **F**ields* (NGDF), which model the continuous manifold of valid grasp poses as the level set of a neural implicit function. Given a 6D query pose, NGDF predicts the unsigned distance between the query and the closest valid grasp on the manifold (see Fig. 3.1).

Neural implicit fields have driven recent advancements in novel view synthesis [124] and 3D reconstruction [18, 19, 123, 140]. These approaches represent distributions as continuous functions that take a query as input and predict its relationship to the learned distribution. In 3D shape reconstruction, for instance, neural implicit fields are used to represent the surface of a shape: 3D points are used as queries, and the output is the distance to the surface, or occupancy at the query point. Unlike explicit methods, neural implicit fields can encode complex topological distributions and are not limited by resolution.

With NGDF, formulating grasp learning as a neural field allows us to interpret the implicit function as a cost such that a query pose can be optimized to result in a

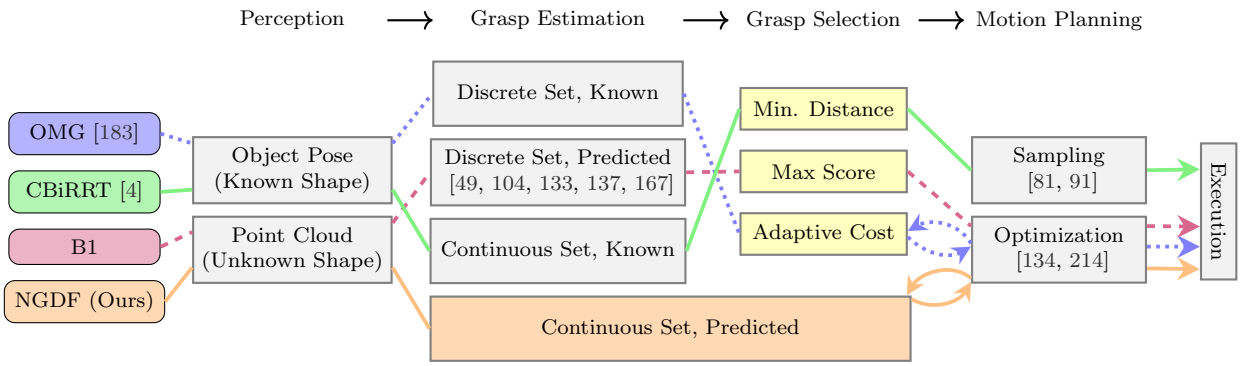


Figure 3.2: Columns illustrate design decisions within grasp and motion planning pipelines. The left-most column highlights representative pipelines like OMG-Planner [183], CBiRRT [4], and baseline B1 from Table 3.2 which uses a SOTA grasp estimator [167]. The respective design choices for these methods are traced through the columns. We identify learned continuous representations as an under-explored option for grasp estimation, and propose NGDF as a solution that does not require a heuristic grasp selection step since the grasp pose is jointly optimized with motion planning.

grasp pose. Prior grasp estimation methods largely output a discrete set of candidate grasps [49, 104, 133, 137, 167], from which one grasp must be selected to perform downstream planning. Instead, we incorporate the grasp distance cost directly into a gradient-based optimizer [214] to jointly optimize the grasp and reaching motion from an initial trajectory. During each optimization iteration, NGDF estimates the distance between the final gripper pose of the trajectory and the grasp level set. This "grasp distance" is minimized as a cost, along with other trajectory costs such as smoothness and collision avoidance. The gradient of the grasp cost for updating the trajectory is computed through fully differentiable operations. This optimization results in a smooth, collision-free trajectory that reaches a valid grasp pose.

In experiments, we find that NGDF learns the level set of valid grasp poses, outperforms baselines by 63% execution success on simulated reaching and grasping, and generalizes to unseen object shapes and poses in the real world. The key contributions of this chapter are:

- Neural Grasp Distance Fields (NGDF), a neural implicit function that predicts the distance between a query pose and the closest grasp, representing the manifold of grasps as a continuous level set.
- A gradient-based optimization algorithm that incorporates NGDF for joint reach and grasp planning.

3.2 Related Work

While grasping and motion planning are well-studied topics in robotics, prior works often propose different system designs with different assumptions, making comparison and contextualization difficult. We summarize the most important design decisions for 6-DOF grasp and motion planning and trace the decisions in representative methods (see Fig. 3.2).

3.2.1 6-DOF Grasp Estimation

6-DOF grasp estimation is a well-studied task [10, 89] that aims to predict successful grasps in SE(3) for target objects; we focus here on recent, data-driven methods. State-of-the-art methods take point clouds as input and output a discrete set of grasps, representing only a subset of the true continuous grasp set [49, 104, 133, 137, 167]. Outputting a finer discretization comes with a cost of a greater computational complexity for both grasp estimation as well as grasp selection: a final grasp must be chosen from the predicted set. Because these methods only predict discrete grasp sets, they necessitate a multi-stage approach, which can be brittle if any of the stages (grasp estimation, selection, or motion planning) fails. Our single-stage approach models grasps as the level set of a continuous implicit function to jointly optimize grasping and motion planning.

3.2.2 Joint Grasp Selection and Motion Planning

Following the multi-stage paradigm above, several works assume a grasp set is provided by an upstream method, and address the downstream task of planning a reaching trajectory. Berenson *et al.* [4] model grasp sets as a continuous range of poses called Task Space Regions, and use sampling-based planning to satisfy the constraint. GOMP [70], uses sequential quadratic programming on discrete grasp sets for fast bin picking. Goal-set CHOMP [41] incorporates hard constraints like goal sets into trajectory optimization. The methods above do not address the problem of switching between grasps during planning; OMG-Planner [183] therefore proposes online learning to estimate goal costs and switch to the minimum cost grasp at every optimization iteration. OMG-Planner used ground-truth grasp sets per object, though

their method can use estimated grasp sets as well. Our approach does not assume grasps are provided and does not require explicit grasp selection; instead, NGDF estimates and updates the grasp pose during trajectory optimization itself.

Other works propose closed-loop methods for 6-DOF grasping. Wang *et al.* [184] learn a latent space of trajectories for closed-loop grasping. Song *et al.* [164] learn a closed-loop policy from human demonstrations. Temporal GraspNet [205] updates a discrete grasp set over time by querying a grasp evaluator. In this work, we introduce a novel implicit representation for the grasp manifold. We focus on open-loop planning and leave closed-loop planning with NGDF as future work.

3.2.3 Implicit Neural Representations

Recent advances in vision and graphics research have used implicit neural representations to achieve impressive results on novel view synthesis [124] and 3D reconstruction [18, 19, 123, 140]. Karunratakul *et al.* [80] learn an implicit representation for human grasp poses. Inspired by these works, we learn an implicit neural function to predict distances between query gripper poses and grasp poses, and use this function to optimize grasp trajectories.

The robotics community has also explored neural implicit functions for a variety of manipulation tasks [42, 51, 69, 103, 162, 192, 213]. GIGA [76] proposed using neural implicit functions to model both 3D shape and grasp quality. However, GIGA predicts a single grasp parameterization per 3D location, and requires a sampling procedure to select the final pose from the implicit set. Our approach predicts grasp distance, allowing multiple grasp orientations per 3D location, and uses optimization to minimize grasp distance and achieve the grasp pose.

Concurrent works have proposed continuous representations for dexterous hands [198] and multiple grippers [84]. Urain *et al.* [179] represents grasps as diffusion fields, framing joint grasp and motion planning as an inverse diffusion process. In this chapter, we use an implicit function to represent grasp distance, and use gradient-based trajectory optimization for joint grasp and motion planning.

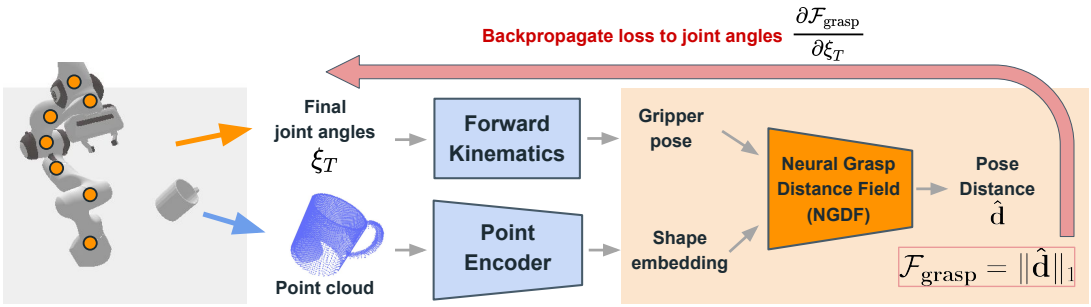


Figure 3.3: We use NGDF as a goal cost function on the final state of a trajectory during gradient-based optimization. Given the current robot joint configuration and a point cloud of an object or scene, the current gripper pose and a shape embedding are computed as inputs for NGDF. Then, NGDF predicts the distance of the current gripper pose to the closest grasp (Sec. 3.4.1). The predicted distance is used as the cost and the gradient with respect to the joint configuration is computed with backpropagation. This cost (with gradient) is used with other costs like smoothness and collision avoidance to update the trajectory (Sec. 3.4.2).

3.3 Background

Neural Implicit Functions. Neural implicit functions (NIFs) are neural networks that take a query $\mathbf{q} \in \mathbb{R}^d$ and optionally a context embedding $\mathbf{z} \in \mathcal{Z}$ to output a scalar value that represents a relationship to an underlying distribution: $f(\mathbf{q}, \mathbf{z}) : \mathbb{R}^d \times \mathcal{Z} \mapsto \mathbb{R}$. In the domain of 3D shape reconstruction, the context $\mathbf{z}$ is a latent shape embedding, the query $\mathbf{q}$ is a 3D point, and the scalar output is either distance to the closest surface [19, 140], or occupancy [18, 123]. The shape surface is represented by the zero level set in distance-based methods, or the decision boundary in occupancy-based methods. Unlike explicit functions, NIFs are not limited by resolution as they predict a value at any query point, and also better represent underlying distributions that are disjoint [51]. Our approach leverages both properties in learning a manifold of grasps.

Gradient-based Trajectory Optimization. A mapping from time t to robot joint configuration $\mathbf{p}$ is defined as a trajectory $\xi : [0, T] \rightarrow \mathbf{p}$. Trajectory optimization aims to find the optimal trajectory given an objective functional $\mathcal{U}$:

$$\xi^* = \arg\min_\xi \mathcal{U}[\xi], \text{ s.t. } \xi(0) = \mathbf{p}_s, \ \xi(T) = \mathbf{p}_g \tag{3.1}$$

for a given start $\mathbf{p}_s$ and goal $\mathbf{p}_g$ configuration. In manipulation, the objective $\mathcal{U}$

contains cost terms for smoothness and collision avoidance. CHOMP [214] solves for ξ^* with functional gradient descent:

$$\xi_{t+1} = \xi_t - \eta A^{-1} \bar{\nabla} \mathcal{U}(\xi_t) \tag{3.2}$$

where A is an acceleration metric that helps propagate updates over the entire trajectory.

3.4 Method

In this work, we represent a set of poses $\mathcal{M} \subset \mathbf{SE}(3)$ as the level set of a neural implicit function. This implicit function takes a query pose $\mathbf{q}$ as input and estimates its distance to the learned level set. Sec. 3.4.1 describes how Neural Grasp Distance Fields (NGDF) leverage this insight to learn the level set of valid *grasp* poses. Sec. 3.4.2 explains how to incorporate NGDF into a trajectory optimization framework to jointly reason over smooth and collision-free reaching trajectories that end at a valid grasp pose. Fig. 3.3 provides an overview of our method.

3.4.1 Neural Grasp Distance Fields

Given a query pose $\mathbf{q} \in \mathbf{SE}(3)$ and a shape embedding $\mathbf{z} \in \mathcal{Z}$, NGDF defines an implicit function: $\text{NGDF}(\mathbf{q}, \mathbf{z}) = \mathbf{d}$, where $\mathbf{d}$ is the distance from $\mathbf{q}$ to the closest valid grasp $\mathbf{g} \in \mathcal{M} \subset \mathbf{SE}(3)$ for an object in a scene. Valid grasps are poses where a gripper can stably grasp an object by closing its fingers. For the distance metric $\mathbf{d}$ we combine translation and orientation distances into a single "control points" metric [133]:

$$d_i = \|\mathcal{T}(\mathbf{q}; \mathbf{c}_i) - \mathcal{T}(\mathbf{g}; \mathbf{c}_i)\|_1 \ , \ i = 0, \ldots, N \tag{3.3}$$

where $\mathcal{T}(\cdot; \mathbf{c}_i)$ is the transformation of a predefined set of points $\{\mathbf{c}_i\}$ on the gripper. Since $\mathbf{q}$ and $\mathbf{g}$ belong to $\mathbf{SE}(3)$, the distance could be defined based on the manifold geodesic distance between those poses, however we find that the control points based distance metric balances the translation and rotation costs better in practice. NGDF estimates the distance for each control point $\mathbf{c}_{0\ldots N}$ separately: $\mathbf{d}(\mathbf{q}, \mathbf{g}) = \Big[d_0, \ldots, d_N\Big]^T$. During training, the estimated distances $\hat{\mathbf{d}}$ are supervised with L1 loss: $\mathcal{L} = \|\hat{\mathbf{d}} - \mathbf{d}\|_1$.

3.4.2 Optimization of Grasping Trajectories using NGDF

For a given query pose, NGDF outputs the distance to the closest grasp pose. We now show how to enable joint optimization for reaching and grasping with NGDF. We incorporate NGDF as a goal cost estimator within a gradient-based trajectory optimizer that already has cost terms for smoothness and collision avoidance.

In this work, we combine NGDF with CHOMP [214] (described in Sec. 3.3), though NGDF can be used in any gradient-based trajectory optimization algorithm. Since CHOMP specifies a fixed goal $\mathbf{p}_g$, we modify CHOMP to include $\mathbf{p}_g$ as a variable in the optimization following Dragan *et al.* [41]. We then add our grasp cost $\mathcal{F}_{\text{grasp}}$ as the variable goal cost to the objective functional $\mathcal{U}$:

$$\mathcal{U}[\xi] = \lambda_1 \boldsymbol{\mathcal{F}}_{\mathbf{grasp}}[\boldsymbol{\xi}] + \lambda_2 \mathcal{F}_{\text{smooth}}[\xi] + \lambda_3 \mathcal{F}_{\text{obs}}[\xi] \tag{3.4}$$

where λ_i are cost weights.

Grasp Distance as a Goal Cost. We now define $\mathcal{F}_{\text{grasp}}$ and derive its functional gradient $\bar{\nabla}\mathcal{F}_{\text{grasp}}$ for gradient-based optimization. For a trajectory (during any iteration of optimization), we calculate the gripper pose from the final joint configuration using forward kinematics: $\mathbf{q}_T = \text{FK}(\xi_{\text{T}})$. We then use NGDF to estimate the distance of this gripper pose to a valid grasp: $\text{NGDF}(\mathbf{q}_T, \mathbf{z}) = \hat{\mathbf{d}}$. The norm of this distance becomes our grasp cost: $\mathcal{F}_{\text{grasp}}[\xi] = \|\hat{\mathbf{d}}\|_1$. We can compute the gradient of the grasp cost with respect to the joint configuration ξ_T through backpropagation:

$$\frac{\partial \mathcal{F}_{\text{grasp}}}{\partial \xi_T} = \frac{\partial \mathcal{F}_{\text{grasp}}}{\partial \mathbf{q}_T} \frac{\partial \mathbf{q}_T}{\partial \text{FK}} \frac{\partial \text{FK}}{\partial \xi_T} \tag{3.5}$$

Since the grasp cost only applies to the final configuration in a trajectory, the functional gradient $\bar{\nabla}\mathcal{F}_{\text{grasp}}$ contains all zeros except for the last row: $\bar{\nabla}\mathcal{F}_{\text{grasp}} = [\mathbf{0}, \mathbf{0}, \ldots, \frac{\partial \mathcal{F}_{\text{grasp}}}{\partial \xi_T}]^T$.

Joint Optimization of Trajectory Costs. Similar to the objective functional (Eq. 3.4), the objective functional gradient $\bar{\nabla}\mathcal{U}$ is a weighted sum of gradients: $\bar{\nabla}\mathcal{U}[\xi] = \lambda_1 \bar{\nabla}\mathcal{F}_{\text{grasp}} + \lambda_2 \bar{\nabla}\mathcal{F}_{\text{smooth}} + \lambda_3 \bar{\nabla}\mathcal{F}_{\text{obs}}$. At every optimization iteration, we compute the costs and functional gradients as described above, then update the trajectory according to the A-metric update rule (Eq. 3.2). Since our objective cost

has terms for minimizing distance to a valid grasp, maintaining smoothness, and avoiding collisions, our algorithm jointly optimizes all three to produce reaching and grasping trajectories.

3.4.3 Implementation Details

Dataset. Training NGDF requires a dataset of point clouds, valid grasp poses, and query poses. We use the ACRONYM [44] dataset, which contains object meshes and successful grasp poses collected in NVIDIA FleX [113]. For grasp poses, our evaluations in Sec. 3.5 are run in PyBullet [24], so we relabel the successful grasp poses based on their success in PyBullet with the same linear and rotational shaking parameters used in ACRONYM. In addition, we filter the positive grasp set to only include grasps where the normals at the mesh and finger contact points are opposed to each other (-0.98 cosine similarity). Our results in Sec. 3.5.2 show that this filtering improves grasp performance. To collect query poses for the dataset, we sample 1 million random $\mathbf{SE}(3)$ poses within a 0.5 m radius of the object mesh centroid. While it is possible that some of the sampled poses could be positive grasps, we assume they are few in number and do not run additional grasp evaluation to filter them. For each sampled pose, we use distance to the closest grasp in the valid grasp set (see Sec. 3.4.1) as our supervision.

Architecture. An input point cloud is converted into the shape embedding $\mathbf{z}$ using a VN-OccNet [31] encoder pre-trained on 3D reconstruction [162]. The input to NGDF is a concatenation of this shape embedding $\mathbf{z}$ with the input query $\mathbf{q}$'s position and quaternion. The NGDF network is based on DeepSDF [140] and consists of 8 MLP layers, 512 units each, and ReLU activations on the hidden layers. A softplus activation on the output layer ensures positive outputs.

Training Procedure. We freeze the weights of the pre-trained point encoder during training and only train the NGDF network. Each training sample consists of a partial point cloud, a query pose, and the closest valid grasp. Similar to NDF [162], the partial point cloud is merged together from 4 camera views and downsampled to 1500 points using farthest point sampling. Random rotation augmentations are applied to each sample with 70% probability. Finding the ground truth closest grasp pose is computationally expensive and requires multiple simulated grasp attempts per

query pose. Therefore, our supervision is pseudo-ground truth, as the closest grasp pose comes from a large but discrete set of grasps [44]. We find that this discrete grasp set is dense enough to train NGDF, while still representing unseen valid grasp poses at or near the zero level set (Sec. 3.5.1).

Trajectory Optimization. CHOMP [214] uses a fixed or decaying step size for functional gradient updates, which is sufficient for trajectories with fixed start and goal joint configurations. However, with our modification of CHOMP in Sec. 3.4.2 to allow a variable goal configuration, we found that such simple step size strategies resulted in poor convergence. We address this issue by using Adam [86] to adaptively update the step size ("CHOMP-Adam"). We use differentiable SE(3) operations [142] and a differentiable robot model [168] to backpropagate gradients from the output of NGDF to the robot joint configuration (Eq. 3.5).

3.5 Experiments

We first evaluate how well NGDFs represent valid grasp manifolds as their zero level sets (Sec. 3.5.1). Then we perform a full system evaluation with NGDFs on a "reaching and grasping" task (Sec. 3.5.2), where an NGDF is used within a gradient-based trajectory optimizer as a goal cost function. We evaluate generalization on grasping intra-category unseen objects (Sec. 3.5.3), and demonstrate grasping on a real robot system (Sec. 3.5.4).

3.5.1 NGDF Level Set Evaluation

First, we investigate whether the learned level set of an NGDF represents successful grasps. Our evaluation procedure considers driving an initial query pose to the learned level set. We use the distance output from NGDF as a loss, and update the query pose with Adam [86] using backpropagated gradients. Note that this evaluation optimizes just the gripper pose; full-arm trajectory optimization is considered in the next subsection. We evaluate NGDF on three objects: Bottle, Bowl, and Mug. For this evaluation, we train a single NGDF model for each object, and evaluate models trained with and without the dataset filtering procedure described in Sec. 3.4.3. We run the optimization for 3k steps with a learning rate of 1e-4. Since we represent

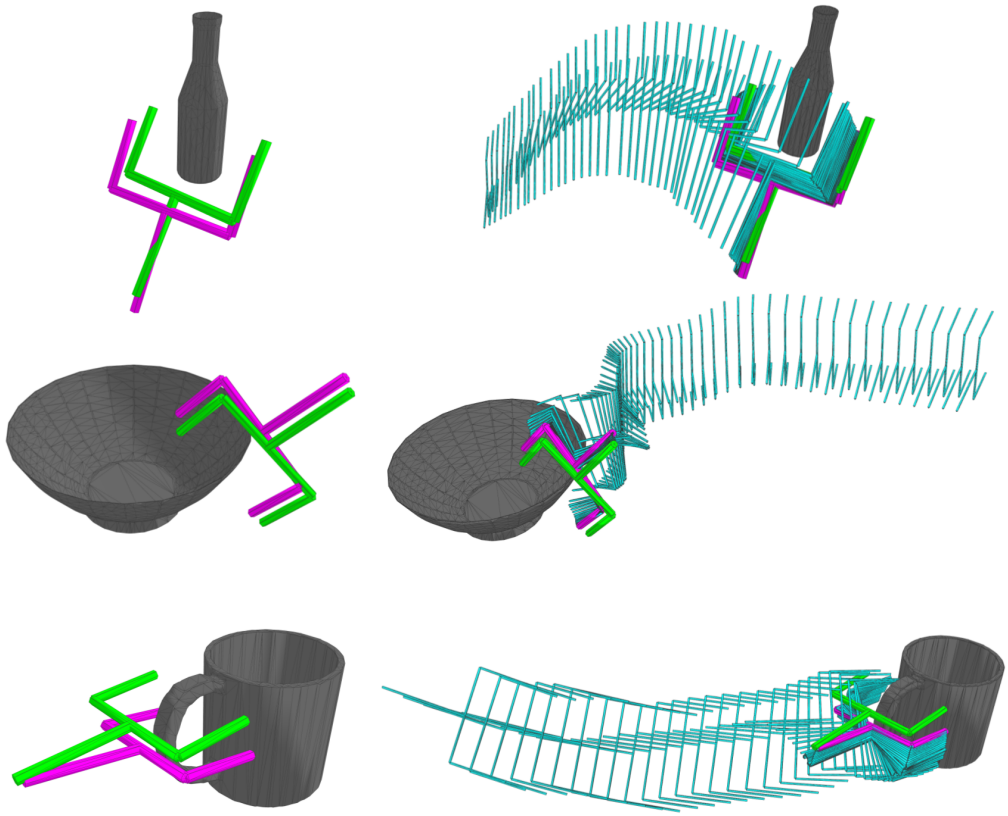


Figure 3.4: Grasp Level Set Evaluation. Left: Final predicted pose (magenta) and its closest grasp pose (green) in the training dataset. Right: Gripper path (teal) as it is optimized from initial to final pose. Object meshes shown for visual clarity; our method takes point clouds as input.

poses as positions and quaternions, we normalize the quaternion after each gradient update to ensure valid rotations.

The quantitative results on grasp level set optimization are shown in Table 3.1. We use two metrics for this evaluation. The "Train Set Error" metric is the minimum control points distance (Eq. 3.3) between the optimized gripper pose and the closest grasp pose in the discrete training set. Since NGDF should learn a continuous level set and interpolate between grasps in the training set, we expect NGDF not to achieve zero error on this metric, but it provides a good surrogate for comparing models. The "Grasp Success" metric measures the grasp quality of the optimized gripper poses. For each pose, we load the target object in PyBullet [24] and attempt a grasp at the specified pose. The robot gripper is always initialized to the same position; the object

Table 3.1: NGDF Grasp Level Set Results

	Train Set Error (m) ↓	Grasp Success ↑
Bottle-NoFilter	$\mathbf{0.023 \pm 0.01}$	0.480
Bottle	0.029 ± 0.01	**0.880**
Bowl-NoFilter	0.036 ± 0.02	0.540
Bowl	$\mathbf{0.033 \pm 0.01}$	**0.760**
Mug-NoFilter	0.038 ± 0.01	0.680
Mug	$\mathbf{0.035 \pm 0.01}$	**0.860**

Results are averaged over 50 unseen query poses per object, sampled from within a 0.5 m radius of the object centroid.

Table 3.2: Reaching and Grasping Results

Method	Perception	Grasp Estimation	Grasp Selection	Goal	Execution Success ↑
O1 *(Oracle)*	*Known Object Pose*	*Known Discrete Grasps*	*Min. Distance*	*Fixed*	0.96
OMG [183] *(Oracle)*	*Known Object Pose*	*Known Discrete Grasps*	*Adaptive Cost*	*Variable*	**0.99**
B1	Unknown Object Pose	Predicted Discrete Grasps [167]	Max Score	Fixed	0.37
B2	Unknown Object Pose	Predicted Discrete Grasps [167]	Min. Distance	Fixed	0.39
B3	Unknown Object Pose	Predicted Discrete Grasps [167]	Min. Distance	Variable	0.38
B4	Unknown Object Pose	Predicted Discrete Grasps [167]	Adaptive Cost	Variable	0.31
NGDF (Ours)	Unknown Object Pose	Predicted Continuous Grasps	*N/A*	Variable	**0.61**

Middle columns correspond to design decisions found in Fig. 3.2; color-coded methods also correspond to those shown in the same figure.

is transformed relative to the gripper. Linear and rotational shaking are applied after gripping the object [44], and the grasp is successful if the object is still gripped after the shaking.

Our results show that while NGDFs trained on filtered and unfiltered data have similar Train Set Error, the Grasp Success for filtered data models is much higher. These results also indicate that NGDFs have learned continuous level sets, since the mean distance predicted by NGDF after optimization is less than 1e-5, much lower than the minimum distance to the training set of grasps. Fig. 3.4 shows examples of the optimization path and achieved gripper pose.

3.5.2 Simulated Reaching and Grasping Evaluation

Next, we evaluate our method on a full reaching and grasping task, which requires planning a smooth, collision-free grasping trajectory for the full robot arm starting

from an initial robot joint configuration. This evaluates the full pipeline as opposed to just the stand-alone gripper pose in the previous subsection. The task is considered successful if the robot executes the trajectory, closes its fingers to grasp the object, and lifts the object without losing it. We place Bottle, Bowl, and Mug objects in simulation in 30 random orientations each (see Appendix Fig. B.2 in [189], left-most column), thus 90 trials in total. Our results indicate that even in a seemingly simple setting, randomly oriented objects present an overall challenging benchmark.

For this evaluation, we train a separate NGDF (similar to NeRF approaches [69, 124]) for each object, though our method can be extended to generalize across objects like other shape-conditioned implicit approaches [140]. We also evaluate intra-category (known class, unseen shape) generalization in the next subsection. We run 500 iterations of CHOMP-Adam (see Sec. 3.4.3) with a learning rate of 3e-3. The grasp cost is weighted heavily relative to the collision and smoothness costs. The trajectory is initialized using inverse kinematics so the gripper pose of the final joint configuration is within 0.3 m of the center of the object point cloud; the rest of the initial trajectory is interpolated between the start and end joint configurations.

The results are shown in Table 3.2. We compare against oracle methods that provide upper-bound task performance, and against baselines that predict discrete grasps. Oracle methods assume perfect object pose estimation and known discrete grasp set. All discrete grasp methods run inverse kinematics over all discrete grasp goals and discard infeasible grasps. For planning, methods use goal-set CHOMP [41] or CHOMP [214], depending on whether the goal is fixed or can vary. "O1" selects the goal with minimum distance to the initial joint configuration, and keeps it fixed throughout planning. "OMG" [183] adaptively learns a cost for each grasp and selects the grasp with minimum cost at every optimization iteration (Variable Goal).

The baselines that predict discrete grasps use Contact-Graspnet [167] as the grasp estimator. We use weights (provided by the authors) that are trained on millions of grasps and shapes. "B1" selects the grasp goal with the maximum score estimated by Contact-GraspNet and keeps it fixed during planning. "B2" selects the grasp goal with minimum distance to the initial joints and keeps it fixed during planning. "B3" allows varying grasps during planning using the minimum distance metric. "B4" uses the same adaptive cost from OMG [167] to select grasp goals during planning.

Our results show that while oracle methods perform well, methods that don't

assume known object pose and use predicted grasps have much lower Execution Success. Of the predicted grasp methods, NGDF performs best. Surprisingly, the B3 and B4 variable goal variants do not outperform fixed goal variants B1 and B2. Failure cases for all methods are largely due to collisions between the gripper fingers and the object, which are a relatively small obstacle cost and may be difficult for the planner to balance with the other costs. Appendix Fig. B.2 in [189] contains qualitative NGDF results, and App. B.1 contains additional ablation experiments.

3.5.3 Intra-Category Generalization

To evaluate whether our method can generalize to shapes in the same object category, we train an NGDF model on 7 shapes in the "Bottle" category from ACRONYM [44]. Training samples are generated from the meshes using the same data collection procedure described in Sec. 3.4.3. We evaluate performance on a held-out Bottle instance, the same instance used in the previous evaluations. The intra-category model achieves 0.63 execution success on 30 Bottle trials for the reaching and grasping evaluation, which is comparable with the single-object NGDF results from Table 3.2, demonstrating intra-category generalization without loss of performance.

3.5.4 Real Robot Reaching and Grasping Evaluation

Finally, we test our method's reaching and grasping performance on a real robot system. At the start of each trial, an object is placed in a random stable pose. A partial point cloud of the scene is obtained from four Azure Kinect depth sensors (Fig. 3.5b), similar to NDF [162]. The object point cloud is segmented via plane fitting, then passed as input to NGDF models from Sec. 3.5.1, which are trained on one instance per category in simulation. The cloud is also converted to a signed distance field to enable computing collision costs with CHOMP [214]. The optimized trajectory is executed on a Franka Panda robot with impedance control, and the trial is considered successful if the object is grasped and lifted without being dropped (Fig. 3.5c). 9 test objects were evaluated, 3 from each shape category (Fig. 3.5d). 3 grasp attempts were performed per object for a total of 27 trials. See App. B.3 in [189] for additional details.

Our overall grasp success rate was 81%, with success per category being 7/9

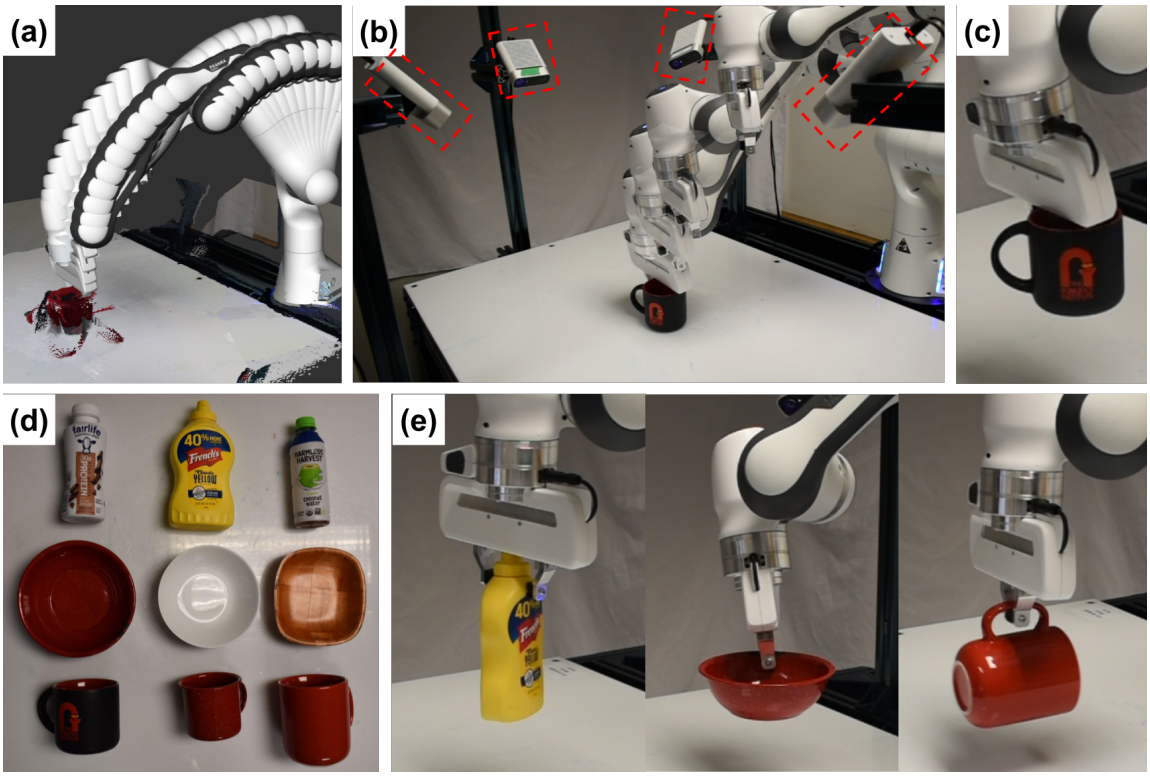


Figure 3.5: Real System Evaluation. (a) Visualizing the plan and imperfect object point cloud; (b) executing the plan on hardware (cameras highlighted with red boxes); (c) lifting the object. (d) The nine objects used for testing. (e) Additional successful grasps.

Bottles, 9/9 Bowls, and 6/9 Mugs. Our system successfully grasped every object, despite many of them being outside of its training distribution in terms of size and shape. The method also demonstrated robustness to noisy perception and execution with impedance control. Failure cases were due to slight collisions between the fingers and objects, similar to what we observed in simulation.

3.6 Discussion

Neural implicit functions have been widely explored for 3D vision tasks such as shape reconstruction. NGDF extends this concept to grasp estimation, using 6D poses as queries on grasp manifolds. Our work differs from existing work on 3D reconstruction, not only due to the higher dimensionality of our problem, but also because of the challenge in acquiring ground truth labels. The ground truth grasp distance between an arbitrary query pose and the corresponding closest grasp is

expensive to compute. Instead, we train on large-scale discrete grasp sets [44] as near-ground truth supervision. Our experiments in Sec. 3.5.1 show that NGDF is able to learn the continuous grasp manifold as the level set of the neural field from this discrete supervision.

NGDF decouples the problem of learning a grasp manifold representation from the problem of finding a good grasp pose. For the latter, we formulate the distance output of NGDF as a cost to be minimized. For the full robot motion planning regime, we jointly optimize the grasp cost with smoothness and collision costs. We outperform baselines in Sec. 3.5.2 that represent what a practitioner would implement for a reaching and grasping task. While the performance of oracle methods indicate room for improvement, our results show that joint optimization with NGDF is a promising direction for manipulation. We also demonstrate scalability with intra-category generalization results in Sec. 3.5.3, and deploy our method on real hardware in Sec. 3.5.4.

In terms of limitations, NGDF is trained on a gripper-specific dataset; NGDF for other grippers may require different datasets. The method also depends on upstream object segmentation. Further, the cost weights are fixed during optimization in the reach and grasp planning task; learning to adjust the weights each iteration could improve performance.

3.7 Conclusion

We propose Neural Grasp Distance Fields (NGDF), which represent the continuous manifold of grasps as the zero-level set of a neural field. We formulate the estimated distance as a cost for a gradient-based trajectory optimizer to jointly optimize with other trajectory costs such as smoothness and collision avoidance to perform reach and grasp planning. Our results show that NGDF outperforms existing methods, while generalizing to unseen poses and unseen objects.

Chapter 4

Cloth Region Segmentation for Robust Grasp Selection

Abstract

Cloth detection and manipulation is a common task in domestic and industrial settings, yet such tasks remain a challenge for robots due to cloth deformability. Furthermore, in many cloth-related tasks like laundry folding and bed making, it is crucial to manipulate specific regions like edges and corners, as opposed to folds. In this work, we focus on the problem of segmenting and grasping these key regions. Our approach trains a network to segment the edges and corners of a cloth from a depth image, distinguishing such regions from wrinkles or folds. We also provide a novel algorithm for estimating the grasp location, direction, and directional uncertainty from the segmentation. We demonstrate our method on a real robot system and show that it outperforms baseline methods on grasping success. Video and other supplementary materials are available at: https://sites.google.com/view/cloth-segmentation.

4.1 Introduction

Manipulating and interacting with cloth is a key part of daily life, yet cloth manipulation by robots remains a challenging problem. Cloth is difficult to perceive and manipulate because its deformable nature breaks the rigid-body assumptions of many algorithms. For example, most pose estimation algorithms assume that objects can only transform in 6 degrees of freedom (translation and rotation). However, cloth can deform at any location and thus has nearly an infinite number of degrees of freedom.

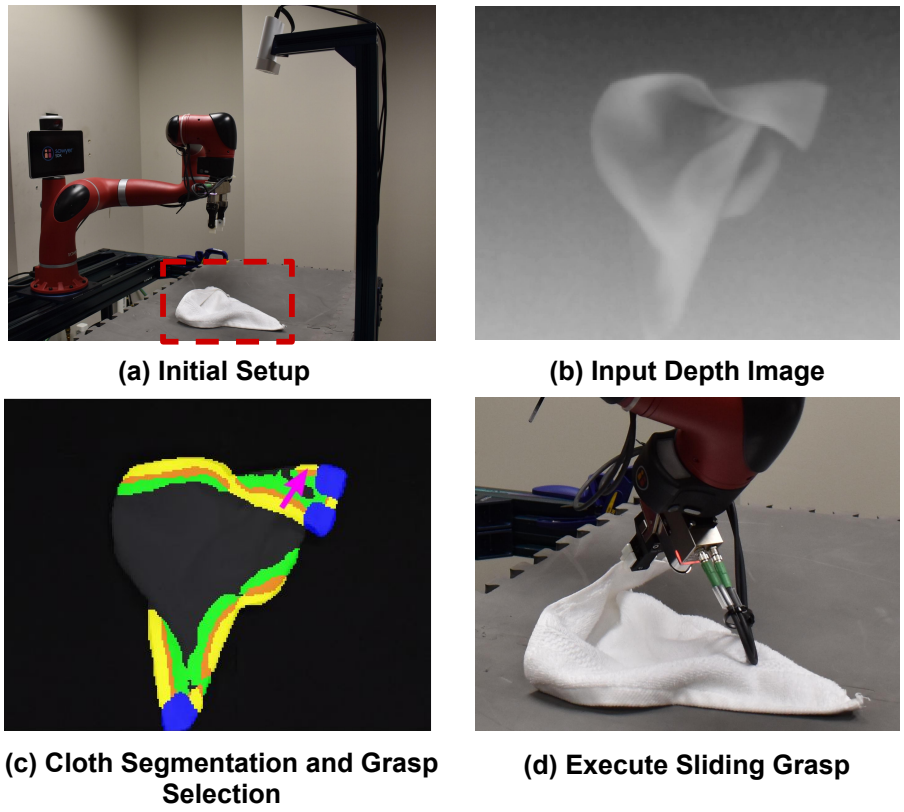


Figure 4.1: Grasping using cloth region segmentation: Robot with depth sensor (a) captures depth image of test cloth (b). Depth image is segmented into outer edges (yellow), inner edges (green) and corners (blue) using our cloth region segmentation network (c). Ambiguous regions are colored in orange. Our method selects a grasp location and direction, shown as a magenta arrow. The robot executes a sliding grasp and successfully grips the cloth by its edge.

In cloth-based tasks like laundry folding and textile manufacturing, it is important to detect and grasp specific regions of cloth, e.g. corners and edges, for downstream

manipulation like folding or smoothing. These edges and corners are distinct from wrinkles and folds, which are less useful for downstream tasks.

In order to grasp the cloth along an edge or corner, we must not only detect the cloth edges and corners but also estimate the appropriate grasping direction. Given a grasp position, the grasp direction specifies the approach vector the gripper follows towards this point. Although estimating the grasping direction would be relatively simple if the cloth were lying flat on the table, it is much more challenging in crumpled configurations. Much work has been done for perception and manipulation of cloth in both randomized and predefined cloth configurations, yet cloth-related tasks like laundry folding and assisted dressing remain challenging due to the inherent complexity of cloth dynamics.

In this thesis chapter, we present an approach for segmenting these key regions of cloth, even in highly crumpled configurations. To achieve this, we train a neural network to predict cloth edges and corners from a depth image. We also train the network to predict the inner edges, the region interior to the cloth's true edges, for grasp direction estimation. The network is trained on a dataset of RGB-D images extracted from 8 minutes of video of a human manipulating the cloth. The ground-truth for the network is provided by color-labeling the cloth (see Fig. 4.1), forgoing the need for expensive human annotations.

The segmentation output of our network allows us to quickly and robustly estimate the appropriate position and grasp direction from a crumpled cloth. It also allows us to estimate the grasp directional uncertainty for every edge/corner pixel. This estimation is important for grasping the cloth, as mis-estimating the grasp direction and approaching at an angle not orthogonal to the cloth edge is more likely to fail. Using a dense estimate of grasp directional uncertainty, we can choose the grasp point most likely to succeed.

We implement our method on a real robot system and evaluate its performance on grasp success metrics against a number of baselines; this evaluation demonstrates the strength of our system in estimating cloth edge and corner positions, grasp direction, and grasp uncertainty.

Our contributions include:

- A method to segment regions of cloth critical for downstream manipulation tasks.

- An algorithm to determine a robust grasp configuration accounting for uncertainty about the cloth direction.
- An evaluation of our method against baselines on a real robot system for grasping edges and corners of cloth in crumpled configurations.

4.2 Related Work

4.2.1 Cloth Perception

Robotic cloth manipulation is a well-studied domain with a variety of unsolved tasks, including laundry folding [5, 119], laundry unfolding or smoothing [38, 61, 176, 177, 193, 201], bed making [94, 156], and grasping [28, 130, 195].

Many of these approaches use traditional computer vision algorithms to detect cloth regions for various downstream tasks: [193] chooses candidate grasp points by using Harris corner detection and discontinuity checks on the depth image for peak ridges and peak corners. [176] uses a pre-task manipulation, lifting the towel into the air and shaking it to remove wrinkles before returning it to the table. Canny edge detection is then used to compute contours for interior and exterior corner classification. [119] performs background subtraction and uses stereo images to select a centered point in a pile of towels. They grasp the towel from a central point and and rotate it to obtain a sequence of images. Towel corners are fit to these images using RANSAC. These perception algorithms usually require significant pre-manipulations to get a more structured configuration of the cloth, thus they are more time consuming than many learning-based methods. Furthermore, without these pre-manipulations, these methods are likely to fail under difficult initial configurations, such as highly crumpled cloth. We will show in Sec. 4.4.2 that our method is much more robust to these crumpled cloth configurations compared to these traditional methods.

Another group of methods apply learning-based algorithms for image feature extraction. [28] uses the YOLO detection network to detect the thickest folded edge and grasp a folded towel from a stack. [201] uses an autoencoder network to predict the real edges of towels. This is similar to our approach; however, their method trains a network to output latent features and performs nearest-neighbor classification on input features to predict good grasp points, whereas our network *directly* outputs

segmentation masks of grasp regions and also determines good *grasp directions*. Their method also operates on RGB images and requires a human-annotated dataset of corners, whereas our method takes depth images as input to be invariant to changes in visual texture, and does not require human labeling.

The most similar method to ours is [156] which learns to identify a corner of a bed sheet by painting the corner red. Our method expands upon this work by estimating a *dense segmentation* of multiple real edges, inner edges, and corners, as opposed to regressing to a *single* corner position. Furthermore, our method outputs dense grasp direction proposals as well as their corresponding uncertainty estimates. As we will show in Sec. 4.4.2, the grasp direction proposals and uncertainty estimates are crucial for our performance on our grasping evaluation. Specifically, these two outputs enable us to handle challenging crumpled cloth configurations.

4.2.2 Cloth Grasping

Although the focus of our work is on perception rather than grasping, we review prior work on cloth grasping strategies. A simple top-down or angled grasp is commonly used once a grasp point has been selected [156, 193]. A top-down grasp followed by a 6DOF grasping on detected corners of the the hanging cloth has also been studied [119].

Other prior works learn a policy for grasping. [130] learns parameters for motion and grasp primitives to grasp a folded towel. [28] uses Q-learning to train a policy for grasping a folded towel from a stack. [195] uses Soft-Actor-Critic to train a policy for rope and cloth manipulation.

In our work, we identify the real corners and edges of the cloth and select a robust grasping point. Then we execute a hand-designed sliding grasp policy on the selected grasping point in order to pick up the cloth by a single edge or corner.

4.3 Approach

4.3.1 Problem Statement

In cloth manipulation tasks such as laundry folding, it is important that the robot be able to identify and grasp key regions of the cloth. These regions typically include the "real edges" or corners of a cloth. By "real edges," we mean the edges of the cloth in the unfolded configuration, as opposed to any folds or creases that may appear as edges in a particular configuration. If the robot grasps a cloth fold or crease and attempts to use such a grasp to neatly fold the cloth, the result likely will not end up as expected. Thus, failing to grasp the cloth along the real edges could lead to failures for many downstream tasks.

As we will show, traditional computer vision algorithms fail to distinguish the difference between a real cloth edges and apparent edges created by creases or folds. In addition, the robot must also determine the appropriate grasping direction along the cloth edge, which is non-trivial if the cloth is in a crumpled configuration; we will show that simple heuristics frequently fail at this task. In this section, we provide a method that identifies edges and corners of a cloth, predicts grasp directions, and estimates the uncertainty of these directions. These predictions will then be used to quickly and reliably grasp the cloth along its edges and corners, even from crumpled configurations.

4.3.2 Method Overview

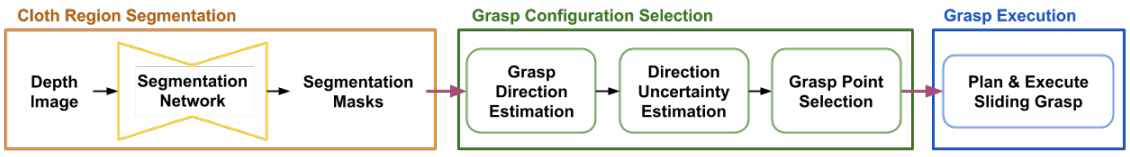


Figure 4.2: Pipeline for our method. Cloth region segmentation takes a depth image and outputs segmentation masks for cloth edges and corners. Grasp selection uses the masks to compute a grasp point and direction in the camera frame. Grasp execution transforms the grasp configuration into the robot frame and executes the grasp.

Fig. 4.2 provides the overall pipeline of our method. First, our segmentation

network takes in a depth image and predicts the outer edges, inner edges and corners. Based on the segmentation, we estimate the grasp direction by computing a correspondence between outer edge and inner edge points. Next, we compute the grasp direction estimation and select a grasp point based on our uncertainty estimate $\mathbf{U}(\mathbf{p})$ for an outer edge point $\mathbf{p}$. Finally, we estimate the 6D robot pre-grasp pose based on the grasp point selected and execute our sliding grasp policy. These components are explained in greater detail in the following sections.

4.3.3 Cloth Region Segmentation

We frame the problem of identifying important regions cloth as semantic segmentation. We train a neural network which receives as input a depth image of the scene containing the cloth. The network predicts semantic labels for each pixel, giving the probability that the pixel contains a cloth outer edge, inner edge, corner, or none of these. We can then threshold this probability to obtain a semantic segmentation mask for the cloth edge and corner locations. Fig. 4.1c shows an example output of our network.

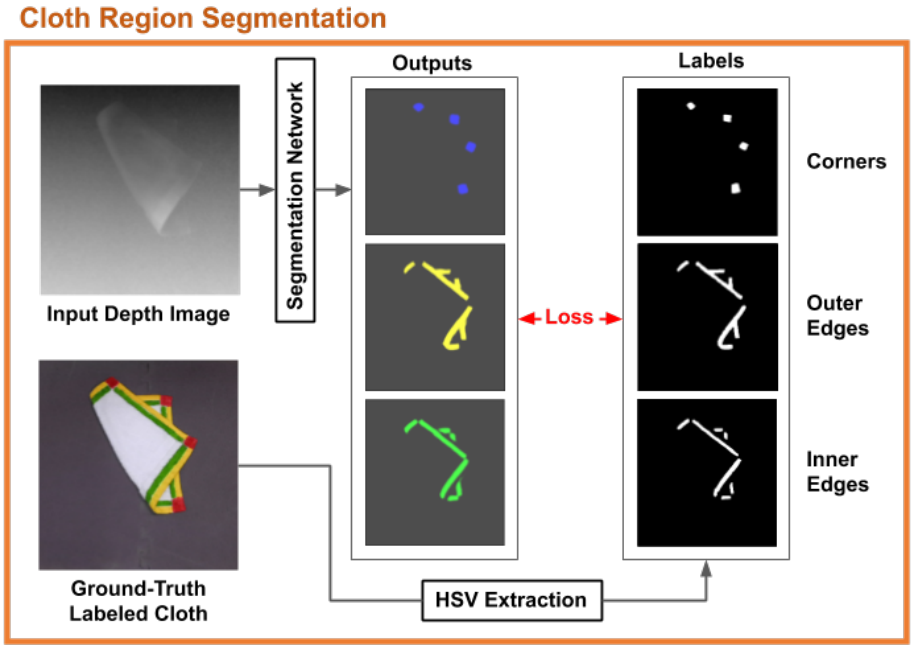


Figure 4.3: Training the segmentation network. The network receives a depth image as input. A paired RGB image supervises the network through the color labels of the cloth. Different colors are used to label the corners, outer edges, and inner edges. The ground-truth color for corner labels was changed from red to blue in the outputs to be color-blind friendly.

To train such a network, we need ground-truth labels for the cloth edges and corners. Unfortunately, these are difficult to obtain in images with crumpled cloth, as this would require a large amount of human annotation effort. Instead, we adopt an approach similar to that of [156], in which they mark a single corner of a cloth with a red marker, and train a network to regress to the single corner location. In our case, we mark all edges and corners with different colors of paint and set up the problem as semantic segmentation, to estimate the position of all cloth edges and corners in the image (other differences from [156] are explained in Sec. 4.2).

As we will show, these labels will allow our network to differentiate between real edges or corners of the cloth from cloth folds, which may appear similar to edges in an image. Fig. 4.3 is a visualization of our training method.

To train the segmentation network parameters θ using these labels, we define the loss $\mathcal{L}$ to be the mean of the pixel-wise binary cross-entropy loss ℓ_k for each class $k \in K$:

$$\mathcal{L}(\theta) = \frac{1}{K} \sum_{k}^{K} \ell_k \tag{4.1a}$$

$$\ell_k = -\sum_{i \in I} w_k (y_i \log \hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \tag{4.1b}$$

where i is a pixel in the input depth image I, w_k is a per-class weight to handle the imbalanced distribution between positive and negative labels, y_i is the binary pixel label, and $\hat{y}_i$ is the network prediction for pixel i.

4.3.4 Grasp Configuration Selection

Grasp Direction Estimation

Once the edges and corners are estimated, the next step is to determine the appropriate grasp direction. To achieve this, we augment the above pipeline by also predicting the cloth "inner edges." We define the cloth outer edge as the region within 1.5 cm of the cloth edge, the cloth corners as the region within 3×3 cm of the corner, and the inner edge as a 1.5 cm region interior to the cloth outer edge. The inner edge labels are shown in green in Fig 4.3. As before, we obtain cloth inner edge ground-truth labels using another color to paint the inner edge of a cloth, and we train a neural

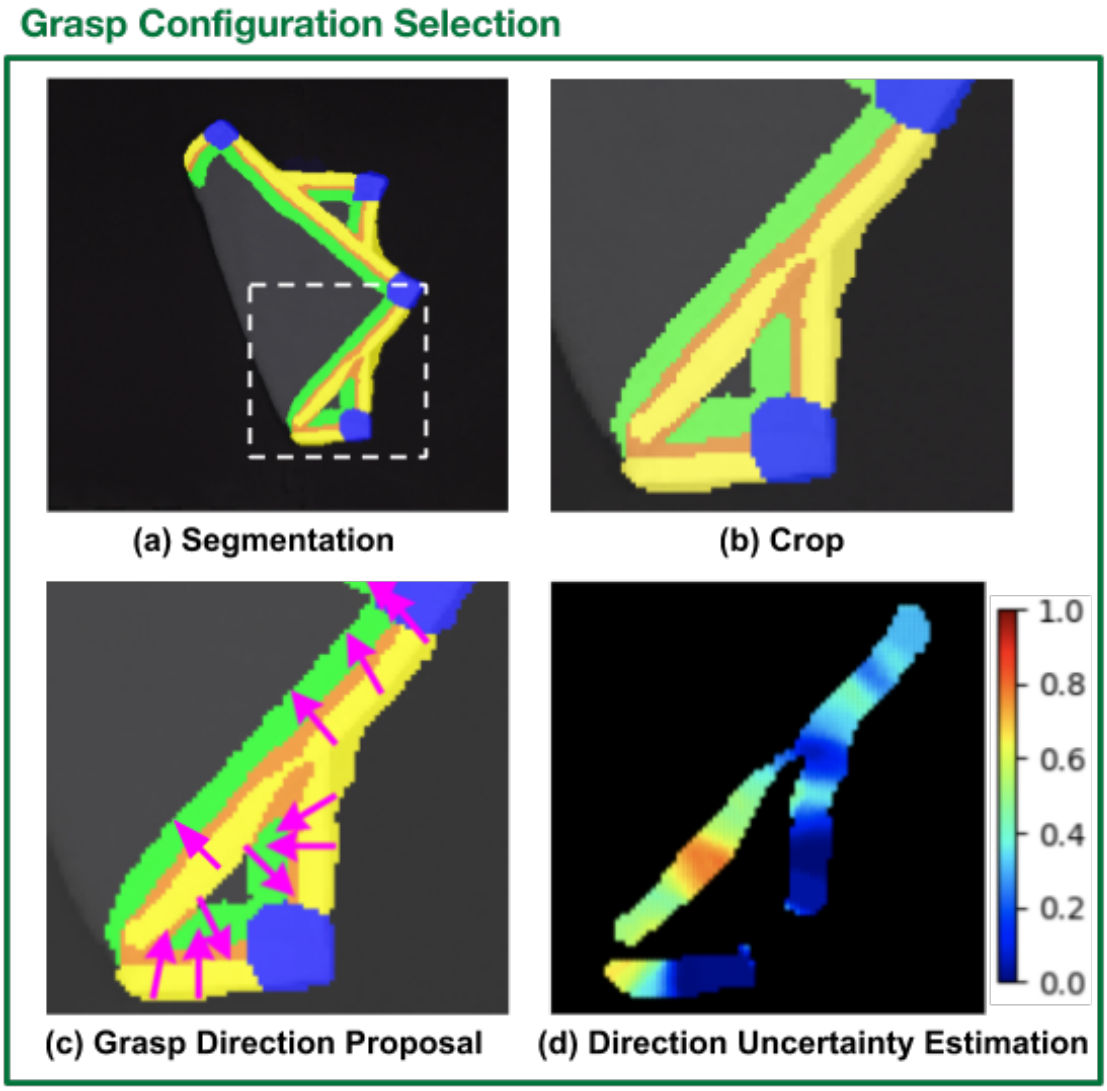


Figure 4.4: Illustration of grasp configuration selection. Corners are labeled in blue, outer edges in yellow, inner edges in green. Overlapping outer edge and inner edge segmentations are in orange; After obtaining the cloth region segmentation, (b) shows the cropped section in (a); (c) shows a subsample of grasp direction proposals for each outer edge points; (d) shows the grasp directional uncertainty for each outer edge points.

network to predict the cloth inner edge from a depth image.

Given the predicted segmentation for these cloth regions, we now select a grasp point and direction. We want to select the direction that allows our sliding grasp policy to most easily grasp the cloth. A sliding grasp that starts with the gripper

oriented towards a cloth edge as in Fig. 4.5 will intercept the edge upon translation. However, a grasp oriented parallel to the edge or approaching from the reverse direction will not intercept the edge and will fail to grasp. Grasp direction is similarly important for corners, as sliding grasps that approach the corner head-on or aligned with the edge of the cloth are more likely to succeed than other orientations.

The following is our procedure for computing the appropriate grasp direction. We first threshold the output of the network described in Sec. 4.3.3 to obtain a set of points estimated to belong to the outer edge $\mathbf{E_O}$ and a set of points that belong to the inner edge $\mathbf{E_I}$. Then, for each outer edge point $\mathbf{p} = [p_x, p_y] \in \mathbf{E_O}$, we find the closest inner edge point $\mathbf{q}^* = [q_x, q_y]$. More formally, we define $\mathbf{q}^*$ to be

$$\mathbf{q}^* = \underset{\mathbf{q} \in \mathbf{E_I}}{\operatorname{argmin}} \|\mathbf{p} - \mathbf{q}\|_2 \tag{4.2}$$

With the correspondence between $\mathbf{p}$ and $\mathbf{q}^*$, we further define the grasp direction at point $\mathbf{p}$ to be the direction along the vector from $\mathbf{p}$ to $\mathbf{q}^*$. Fig. 4.4c shows a subset of those grasp directions. The vector from $\mathbf{p}$ to $\mathbf{q}^*$ often defines an appropriate grasp direction at point $\mathbf{p}$. This direction can be used by the robot to grasp the cloth.

Directional Uncertainty Estimation

Fig. 4.4c also shows a few cases where, due to the complex folds of the cloth, the vector from $\mathbf{p}$ to $\mathbf{q}^*$ does not indicate an appropriate grasp direction. Thus, for robust grasping, we also compute a measure of the uncertainty in this grasp direction.

We define the uncertainty of the grasp direction for a single point $\mathbf{p}$ to be the variance of the grasp directions predicted by its neighbours. To compute this variance, let $\mathbf{N_k}(\mathbf{p})$ be the set of k closest pixel points in $\mathbf{E_O}$ of $\mathbf{p}$ in Euclidean distance; let α be the angle between $\overrightarrow{\mathbf{pq}^*}$ and a unit vector along the horizontal x axis. Formally we can define the cosine and sine of the grasp direction at $\mathbf{p}$ as

$$f_{cos}(\mathbf{p}) = cos(\alpha) = \frac{q_x - p_x}{\|\mathbf{q}^* - \mathbf{p}\|_2} \tag{4.3}$$

$$f_{sin}(\mathbf{p}) = sin(\alpha) = \frac{q_y - p_y}{\|\mathbf{q}^* - \mathbf{p}\|_2} \tag{4.4}$$

We can then define observation vectors $\mathbf{x_0}(\mathbf{p})$ and $\mathbf{x_1}(\mathbf{p})$ to contain the cosine and

sine of the grasp direction of all points in $\mathbf{N_k(p)}$:

$$\mathbf{x_0(p)} = \left\{ f_{cos}(n) \mid n \in \mathbf{N_k(p)} \right\} \tag{4.5}$$

$$\mathbf{x_1(p)} = \left\{ f_{sin}(n) \mid n \in \mathbf{N_k(p)} \right\} \tag{4.6}$$

Next, we define the sample covariance matrix $\mathbf{K(p)}$ in the usual manner from the observations $\mathbf{x_0(p)}$ and $\mathbf{x_1(p)}$

$$\mathbf{K}_{ij}(p) = \frac{1}{N-1} \sum_{k=1}^{N} \left(x_{ik}(p) - \bar{x}_i(p) \right) \left(x_{jk}(p) - \bar{x}_j(p) \right) \tag{4.7}$$

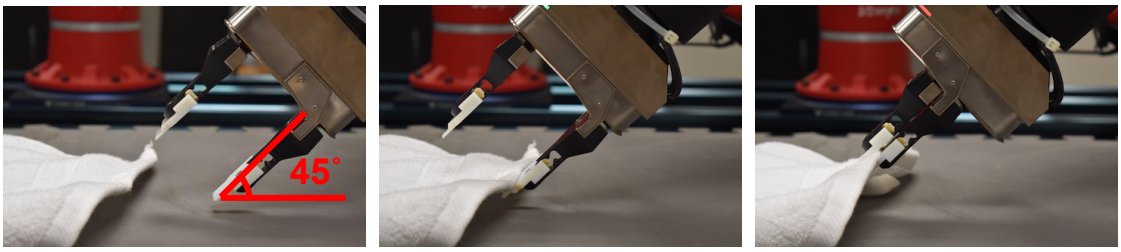
where $x_{ij}(p)$ is the jth element of $\mathbf{x_i(p)}$, and $\bar{x}_i(p)$ is the mean of $\mathbf{x_i(p)}$.

Finally, we define the uncertainty of our grasp direction prediction to be the sum of the variances of the individual dimensions, or the trace of $\mathbf{K}$: $Tr(\mathbf{K(p)}) = Var(\mathbf{x_0(p)}) + Var(\mathbf{x_1(p)})$, where $Var(\mathbf{x_i(p)})$ is the variance of $\mathbf{x_i(p)}$. Since the trace of a matrix is equal to the sum of its eigenvalues, this means that $Tr(\mathbf{K})$ measures the summation of the uncertainty in the principal directions for the covariance matrix $\mathbf{K}$. The trace therefore captures the uncertainty of the grasp direction while being invariant to axis transformations. Fig. 4.4d shows an example of our uncertainty estimate.

Grasp Point Selection

Finally, we describe our method for grasp point selection, which considers the outer edge predictions of Sec. 4.3.3 and the directional uncertainty estimates of Sec. 4.3.4. For each outer edge point $\mathbf{p} \in \mathbf{E_O}$, we compute an uncertainty estimate $\mathbf{U(p)} = Tr(\mathbf{K(p)})$ as described above. Finally, for grasp point selection, we pick the outer edge point $\mathbf{p}$ that has the lowest uncertainty:

$$\mathbf{p} = \underset{\mathbf{p} \in \mathbf{E_O}}{\operatorname{argmin}} \ \mathbf{U(p)} \tag{4.8}$$



(a) Pre-slide pose. (b) Post-slide pose. (c) Pinch grasp.

Figure 4.5: Sequence of poses for the sliding grasp policy. The sliding action is a translation from the pre-slide to post-slide pose. The slide intercepts the target grasp point on the cloth.

4.3.5 Grasp Execution

Once a grasp configuration with point and direction is chosen, we execute a hand-designed grasping policy to slide one of the gripper's fingertips under the cloth for a pinch grasp. We use this sliding grasp policy instead of a simpler top-down grasping routine, because top-down pinch grasps on edges and corners that are folded over (and hence overlap parts of the cloth) often result in grasping multiple layers of the cloth. A tilted sliding grasp can separate one layer of cloth from another.

The configuration ($\mathbf{p}$, α) specifies the grasp point on the cloth and the direction for the sliding grasp. This configuration is specified in image coordinates; to transform it into the world frame, we perform a 2D-to-3D projection using known camera intrinsics and extrinsics. This provides an intermediate 6D pre-grasp pose $\tilde{\mathbf{g}}$ consisting of the 3D position of the target cloth point (corresponding to $\mathbf{p}$ in 2D), and the 3D orientation of the end-effector (corresponding to α in 2D). The intermediate pre-grasp pose $\tilde{\mathbf{g}}$ is oriented top-down and rotated about the z-axis in the world frame. We apply a final transformation that tilts the grasp pose about the horizontal x-axis by 45-degrees to obtain a new pre-grasp pose $\mathbf{g}$. This pose allows one of the fingertips to get under the cloth during the slide action. This transformation also includes a z-offset to account for the z-height of the gripper tip lowering due to the rotation. Finally, we compute offsets to $\mathbf{g}$ in the xy plane parallel to the workspace to get pre-slide and post-slide poses. As shown in Fig. 4.5, the sliding grasp policy moves to the pre-slide pose, translates to the post-slide pose, then pinches to grasp the cloth.

4.3.6 Implementation Details

Network Implementation Details

To train our segmentation network, we collected a dataset of paired RGB-D images. The images were extracted from RGB-D video of a human manipulating a cloth with regions of interest labeled using acrylic paint. The cloth was square, 12 inches each side, and painted with red 3×3 cm corners, yellow 1.5 cm thick outer edges, and green 1.5 cm thick inner edges. See Fig. 4.3 for an image of the labeled cloth.

The human manipulated this semantically labeled cloth in the robot's workspace by folding it, dropping it, bunching it up, etc. We collected 8 minutes of video for a total of about 6700 RGB-D images. These images were split into 4:1:1 train, validation, and test sets.

Our segmentation network is based on U-Net [148]. We augmented the data during training with random image flips and rotations to improve robustness. Additional details on training and the network architecture are provided in the appendix. All training was performed on an Ubuntu 16.04 machine with an NVIDIA GTX 1080 Ti GPU, a 2.1 GHz Intel Xeon CPU, and 32 GB RAM.

Physical Implementation Details

All experiments were performed on a 7 DOF Rethink Robotics Sawyer Robot with a Weiss WSG-32 parallel-jaw gripper. The robot's workspace was a 0.6×0.6 m area. A Microsoft Azure Kinect sensor was mounted 0.7 m above the workspace to provide RGB-D images. Our test cloth is a white, unlabeled cloth with the same dimensions as the labeled one used for training. See Fig. 4.1a for the complete workspace setup. The default fingertips of the Weiss gripper were too thick to get under the cloth during the sliding maneuver, so we 3D-printed and attached thinner fingertips (see Fig. 4.5).

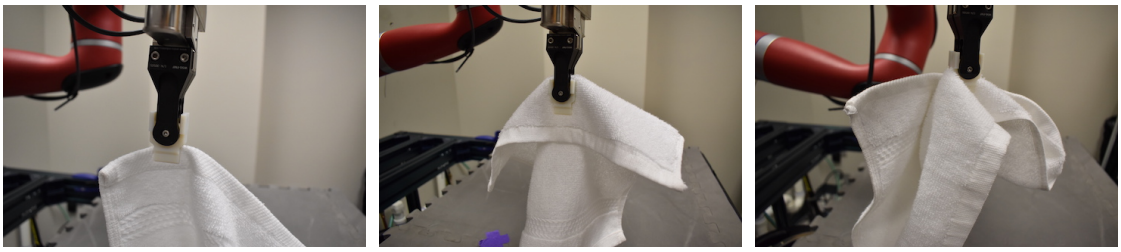
4.4 Experiments

Our experiments are designed to answer the following questions:

- How does our learned method for finding cloth edges and corners compare to non-learned baselines?
- How does our method for estimating the grasp direction compare to non-learned baselines?
- Do we obtain more robust grasps using our method for estimating grasp directional uncertainty?

4.4.1 Experimental Design

We designed two experiments to evaluate our method against various baselines. The first experiment evaluated performance for grasping cloth edges (as opposed to creases or folds), and the second evaluated grasping cloth corners. In both experiments, each grasping trial starts with a randomly crumpled cloth in the center of the robot's workspace. To enable reproducibility of our results, we used the following protocol in all of our experiments to generate the initial cloth configuration for each trial: at the beginning of each trial, a human grasps the square cloth at the midpoint of an edge. They then hold the cloth at a height such that the lowest point of the cloth is 0.1 m from the workspace surface. Finally, they let go of the cloth from this height to obtain a randomly crumpled cloth pose. This initialization procedure is based on the protocol from [55], adapted for our cloth grasping task.



(a) No fold. (b) Single fold. (c) Multiple folds.

Figure 4.6: Examples of cloth grasps. Folds longer than 2cm from edge to fold are considered grasp failures; of these three, only (a) is considered a success.

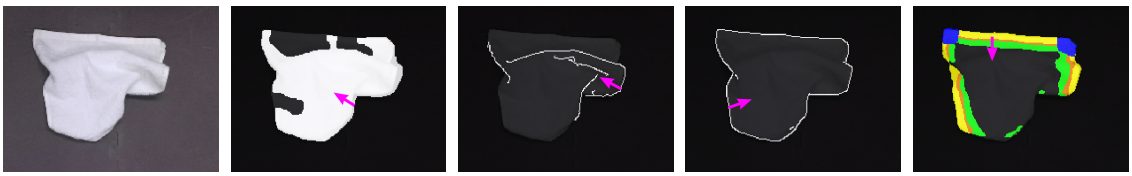
We define success metrics for grasping the cloth at edges and corners. A grasp is considered a success if it pinches a cloth edge or corner and lifts it 30 cm above the workspace. The flexible and deformable nature of cloth can cause pinch grasps

on edges and corners to fold over some of the material. Fig. 4.6 shows examples of grasps with flat and folded cloth. For grasping edges, we consider a grasp with cloth folded over to be a success if the fold is less than or equal to 2 cm at its maximum length. For grasping corners, we use a threshold of 5 cm from the corner to the fold. These thresholds apply when there is a single cloth fold pinched; if multiple folds are held within the pinch grasp, the grasp is considered a failure.

4.4.2 Experimental Results

We evaluate whether our learned method performs better than baselines for identifying cloth edges and corners (as opposed to wrinkles and folds). Our method consists of the cloth region segmentation network, grasp direction estimation, grasp directional uncertainty estimation, and grasp selection, as described in Sec. 4.3.

Grasping Cloth Edges



(a) Cloth Pose (for reference). (b) Segment-Edge. (c) Canny-Depth [16]. (d) Canny-Color [16]. (e) Our Method.

Figure 4.7: Segmentation and selected grasp point for edge grasping methods. (b)-(e) visualize the output of each method on top of the reference image (a). Note that the color image is only provided as input to Canny-Color (d); all other methods take the corresponding depth image as input. As shown in (e), our method correctly identifies most of the apparent edges of the cloth as folds, whereas the other methods fail to make this distinction.

For the task of identifying cloth edges, we evaluate against three baselines:

- "Segment-Edge" segments the cloth from the table using RANSAC plane fitting. A grasp point is randomly selected from the edge pixels of the segmentation. The grasp direction is determined by the direction of the depth gradient at the selected grasp point.

- "Canny-Depth" applies Canny edge detection [16] to the depth image. The grasp point is sampled uniformly from the set of edge points above an intensity threshold. The grasp direction is determined by the depth gradient direction, as in the above.
- "Canny-Color" is the same as Canny-Depth, except it applies Canny edge detection to the gray-scaled color image. The grasp direction is determined by the color gradient direction instead of depth.

See Fig. 4.7 for visualizations of these methods.

The results are shown in Table 4.1. We performed 3 trials with 10 grasps each to estimate a mean and variance for each method. Our method significantly outperforms the baselines in terms of grasp success. The network is largely able to correctly distinguish between edges and folds, determine an appropriate grasp configuration direction, and execute a successful grasp. Averaging over the trials, there were an average of 2.7 failures out of 10 grasps due to misdetection, meaning that the grasp point selected was not a real edge. There was an average of 0.3 failures out of 10 grasps due to failed grasping. See Sec. 4.4.2 for more details on failure cases.

The baselines perform poorly largely due to an inability to distinguish between real cloth edges versus folds. Canny-Depth relies on the intensity of depth gradients to find cloth edges, but depth gradients occur for both cloth edges and large folds. Segment-Edge fails due to noisy segmentation; because the cloth is thin, parts of the cloth can fall within the inlier threshold of the RANSAC table segmentation, despite careful parameter tuning. Still, even with a clean segmentation, grasping at an edge point on the segmentation mask often results in grasping a cloth fold for our highly crumpled cloth configurations. Canny-Color uses color gradients to find edges. It is less affected by noise compared to the depth-based baselines, as the white cloth stands out from the darker background of the table, resulting in strong edges. However, this method is still unable to discriminate between real cloth edges from folds, resulting in failure in a majority of grasp attempts.

Our network is able to perform better than all of these baselines by using a learned segmentation. The successful grasps are also of higher quality, meaning that the grasps are more often flat with no folding of the cloth, and the edge is horizontal to the gripper tip. In terms of execution time, the perception component of our

method runs in approximately 0.25s, with the segmentation network contributing approximately 0.14s to that total. Grasp execution is a larger bottleneck and requires approximately 15s for all methods.

Table 4.1: Grasping Cloth Edges

Method	Grasp Success
Canny-Depth	0.20 ± 0.00
Segment-Edge	0.30 ± 0.00
Canny-Color	0.33 ± 0.12
Our Method	$\mathbf{0.70 \pm 0.20}$

3 trials per method, 10 grasp attempts per trial

Grasping Cloth Corners

We also evaluated our method on grasping corners. Our method remains the same, except that corners are used for grasp point selection instead of edges. The corners still use correspondence with inner edges to determine grasp direction, as well as our method for estimating grasp directional uncertainty described in Sec. 4.3.

For this task, we evaluated against the following baselines:

- "Harris-Depth" applies Harris corner detection [63] to the depth image. The maximum intensity value is selected as the grasp point. The depth gradient direction at the grasp point is used to determine the grasping direction, as in the edge grasping experiments.
- "Harris-Color" takes a grayscaled RGB image as input and uses color gradients to determine the grasping direction, but is otherwise the same as the above.

The results are shown in Table 4.2. Our method outperforms the baselines on corner grasping, being able to more reliably detect corners in any cloth configuration. Averaging over the trials, there were an average of 3 failures out of 10 grasps due to misdetection. There were an average of 1.3 failures out of 10 grasps due to grasping error. Our method performs worse on corners than on edges. Fewer regions of the image are corners compared to edges, so false positives are more problematic. Sec. 4.4.2 for details on failure cases.

The baselines perform poorly for largely the same reason of misdetection as with the edge experiments. The Harris-Depth baseline performs poorly because it looks for large changes in the gradient in all directions, which could result in false positives instead of real corners. Most of the grasp point selections from this baseline were on wrinkles and folds than on the cloth. The Harris-Color baseline performs better than depth, possibly because there are fewer false positives given the white on black input images. White cloth corners against the darker workspace surface can be easily detected; however, corners lying on top of the cloth are less likely to be detected. For our difficult randomly crumpled cloth configurations, the corners are not always cleanly visible against the surface, and often lie in configurations that are difficult to discriminate in 2D.

Table 4.2: Grasping Cloth Corners

Method	Grasp Success
Harris-Depth	0.05 ± 0.07
Harris-Color	0.33 ± 0.15
Our Method	$\mathbf{0.57 \pm 0.06}$

3 trials per method, 10 grasp attempts per trial

Ablations

We perform ablations on our method to determine the relative contribution of the different components of our method to grasp success. Our full method consists of segmenting cloth regions using a neural network (Sec. 4.3.3), determining the grasp direction for all segmented edge/corner pixels using their nearest segmented inner edge pixels (Sec. 4.3.4), and selecting a grasp point with the lowest grasp directional uncertainty (Sec. 4.3.4).

We perform the following ablations of our method:

- "No-Direction-Prediction" still uses the cloth segmentation network of Sec. 4.3.3. However, rather than determining the grasp direction using the methods of Sec. 4.3.4 and Sec. 4.3.4, this ablation determines the grasp direction by fitting a bounding box around the segmented outer edge pixels and setting the direction to be the vector pointing to the center of the box. Instead of using the point

with minimum directional uncertainty, it randomly selects the grasp point from the set of outer edge pixels.

- "No-Directional-Uncertainty" still uses the cloth segmentation network of Sec. 4.3.3 as well as the method of Sec. 4.3.4 for determining the grasp direction. However, rather than computing the grasp directional uncertainty to choose a grasp point as in Sec. 4.3.4, this ablation chooses a grasp point randomly.

The results are shown in Table 4.3. The ablations under-perform the full method, demonstrating that our method for estimating the grasp direction (Sec. 4.3.4) as well as our method for estimating directional uncertainty (Sec. 4.3.4) help to choose more robust grasps. We observe No-Direction-Prediction selecting grasp directions near-parallel to real edges instead of orthogonally, because it always chooses directions toward the center of the segmentation bounding box. The performance of No-Directional-Uncertainty vs. No-Direction-Prediction provides evidence that using the inner edge segmentation to determine the grasp direction improves grasp success. Comparing our full method with No-Directional-Uncertainty shows that selecting the grasp point with minimal directional uncertainty outperforms random grasp point selection.

Table 4.3: Ablations on Grasping Cloth Edges

Method	Grasp Success
No-Direction-Prediction	0.2
No-Directional-Uncertainty	0.4
Our Method	$\mathbf{0.7 \pm 0.20}$

1 trial per ablation, 10 grasp attempts in trial

Failure Cases

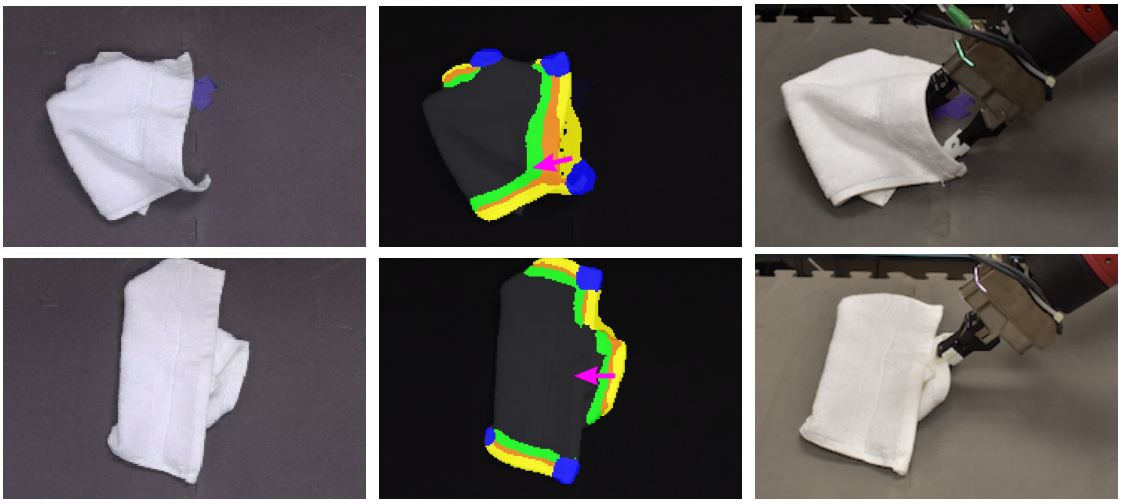
In this section we discuss the most frequent and notable failure cases. Examples of these cases are in Fig. 4.8 and the supplementary video.

Failures occurred when the segmentation produced by our method contained errors. Because the cloth is very thin and the depth images captured from our sensor are noisy, the network can fail to get accurate segmentation at cloth edges (see Fig. 4.8,

top row). This issue causes both false positives, in which pixels close to real edges are included in the segmentation, and false negatives, in which the segmentation does not include valid pixels. These segmentation errors affect the grasp selection component that takes the segmentation as input. As a result, we sometimes observed our method selecting grasp points on false positives, which were more likely to result in grasp failures.

Failures also occurred due to grasping areas with valid edges but problematic nearby cloth configurations. For example, overlapping edges can create the appearance of a continuous segmentation, and a grasp on that area will result in grasping both edges (see Fig. 4.8, bottom row). Developing a policy that can adapt to such challenging configurations is an area of future work.

Failures due to motion planning to reach commanded poses happened infrequently, such as when a selected grasp is in an unreachable robot configuration. These failures are easily detected, so we re-execute our method to choose a different grasp point in such cases.

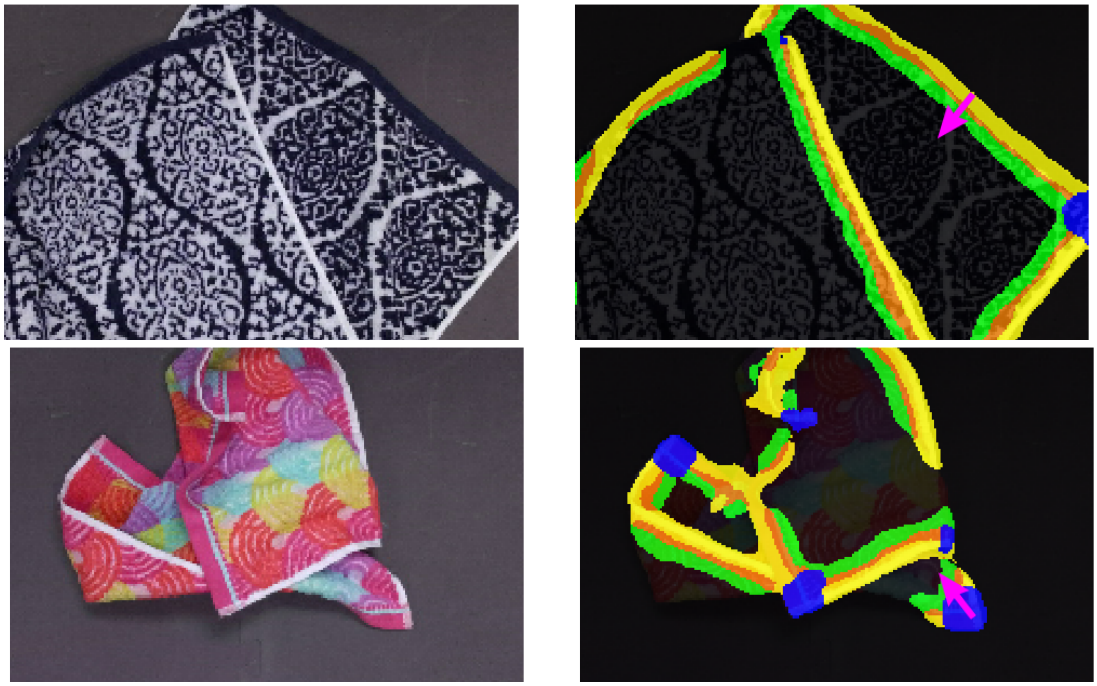


(a) RGB Image. (b) Segmentation and Grasp Prediction. (c) Grasp execution.

Figure 4.8: Failure cases. (top row) Segmentation bleeds over real cloth edge, leading to poor estimation of grasp height. (bottom row) Grasp fails to avoid grasping nearby folds and edges (note that misdetection has also occurred).

Robustness

We demonstrate that our network is robust to variations in visual texture and cloth size by grasping other cloths (see Fig. 4.9 and supplementary video). Our network can segment cloth with different colors and patterns because it only takes depth as input. It can also segment cloths of different dimensions due to its fully convolutional architecture.



(a) RGB Image. (b) Segmentation and Selected Grasp.

Figure 4.9: Our network is able to segment cloths of various sizes and visual texture. See the supplementary video for grasping demonstrations on these cloths.

4.5 Conclusion

We present a method to segment real edges and corners of cloth (as opposed to creases or folds) from depth images. Our method also determines a grasp configuration from these segmentations that accounts for directional uncertainty. We demonstrate a system that implements our approach to grasp cloths in crumpled configurations, and we show that our method outperforms various baselines in terms of grasp success rate

on grasping success.

Chapter 5

Learning to Singulate Layers of Cloth based on Tactile Feedback

Abstract

Robotic manipulation of cloth has applications ranging from fabrics manufacturing to handling blankets and laundry. Cloth manipulation is challenging for robots largely due to their high degrees of freedom, complex dynamics, and severe self-occlusions when in folded or crumpled configurations. Prior work on robotic manipulation of cloth relies primarily on vision sensors alone, which may pose challenges for fine-grained manipulation tasks such as grasping a desired number of cloth layers from a stack of cloth. In this thesis, we propose to use tactile sensing for cloth manipulation; we attach a tactile sensor (ReSkin) to one of the two fingertips of a Franka robot and train a classifier to determine whether the robot is grasping a specific number of cloth layers. During test-time experiments, the robot uses this classifier as part of its policy to grasp one or two cloth layers using tactile feedback to determine suitable grasping points. Experimental results over 180 physical trials suggest that the proposed method outperforms baselines that do not use tactile feedback and has better generalization to unseen cloth compared to methods that use image classifiers. Code, data, and videos are available at https://sites.google.com/view/reskin-cloth.

5.1 Introduction

Cloth manipulation remains an active research area in robotics with significant real world applications, including folding laundry [39, 119], assistive dressing [45, 46, 47, 207], bed-making [155], and manufacturing fabrics [175]. Cloth manipulation is challenging because it is difficult to infer the complete configuration of the cloth from robot observations when the cloth is in a crumpled or folded state, due to the high degrees of freedom and self-occlusions [12, 151].

In light of these challenges, researchers have recently proposed numerous data-driven methods for canonical cloth manipulation tasks such as smoothing [158, 197] and folding [95, 122, 188]. While showing promising results, many prior works focus on top-down grasping of one cloth. Such grasping may be ineffective for manipulation tasks involving multiple cloths, such as picking a desired number of layers of a stack of cloth, because performance is extremely sensitive to the height of the gripper when it grasps. Indeed, a common failure case reported in prior work [54, 188] is picking the wrong number of layers. Yet, manipulating a specific number of cloth layers is common in daily life, such as in folding and unfolding tasks, or handling piles of stacked clothing in stores. How, then, can robots achieve accurate grasping of multiple layers of cloth?

Incorporating tactile sensing is an under-explored direction for deformable object manipulation. While there has been recent work on optical-based tactile sensors such as GelSight [208] and DIGIT [93], these sensors have primarily been applied to cloth *perception* [108, 210] instead of cloth *manipulation*. Recent work on magnetometer-based sensors such as ReSkin [8] have benefits over optical sensors, such as lower-dimensional sensor readings, more direct measurements of normal and shear forces, and a compact form factor. However, research into the applications of *magnetometer-based* sensors for deformable object manipulation is currently limited.

In this thesis, we study the application of magnetometer-based tactile sensing for deformable cloth manipulation. We focus on precisely grasping and lifting layers of stacked cloth; due to the flexibility of cloth and unpredictable crumpling behavior, this task is challenging while being a well-defined manipulation problem. Furthermore, precise grasping of layers of cloth is a prerequisite for many downstream manipulation tasks (*e.g.,* folding cloth in half twice).

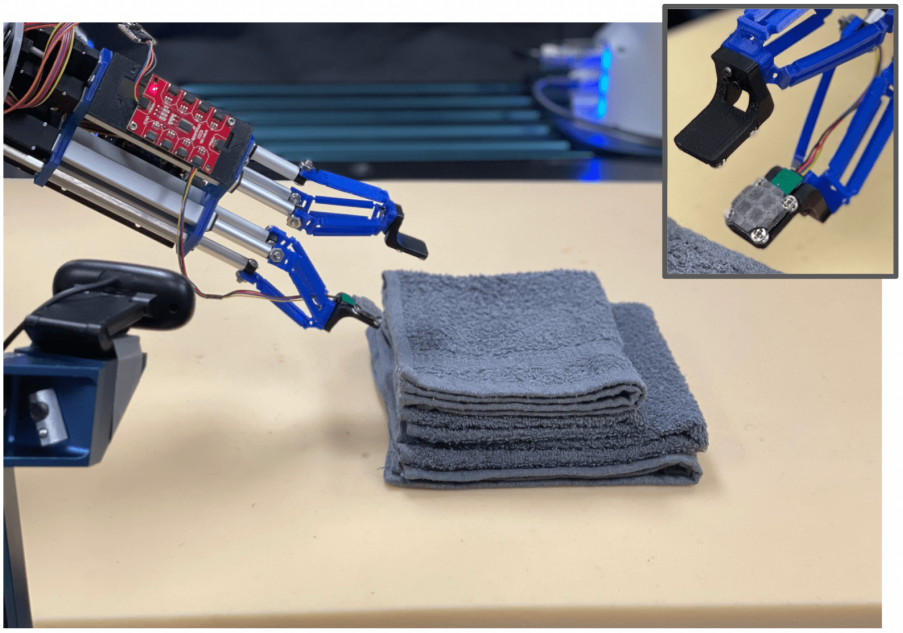


Figure 5.1: We present a tactile-based cloth manipulation system. The robot utilizes a ReSkin [8] sensor attached to the lower one of its two fingertips, which is visualized in more detail in the upper right inset. We train a classifier to distinguish among grasping different numbers of cloth layers from tactile feedback (no images are provided as input). The robot then uses this classifier at test time to determine suitable grasping points for obtaining a desired number of cloth layers.

We present a robotic system consisting of a 7-DOF Franka arm, a mini-Delta gripper [120], and a Reskin [8] sensor on the gripper finger to perform precise cloth grasping (see Fig. 5.1). The system uses a tactile classifier as feedback for a grasping policy. We show that simple approaches to both classifying tactile data and incorporating feedback into the policy (*e.g.,* as a termination condition) work surprisingly well.

This thesis chapter makes the following contributions:

1. A robot hardware system which incorporates ReSkin tactile sensors for cloth manipulation.
2. A training procedure for developing a classifier based on this hardware to use in a grasping policy.
3. Experiments showing success on the task of grasping a desired number of cloth layers.

5.2 Related Work

Manipulation of deformable objects such as cloth has a long history in robotics; see Yin et al. [206] and Zhu et al. [212] for representative surveys.

5.2.1 Cloth Manipulation Policies

In early research on cloth manipulation, a common strategy was to utilize a bimanual robot to grip cloth in midair to smooth it using gravity. This standardizes the configuration of cloth and exposes its corners, which can then facilitate planning subsequent manipulation tasks such as smoothing and folding [39, 119]. Other researchers have relied on using geometric features of cloth, such as by fitting polygon contours to clothing [127]. While these works showed impressive results, such approaches may be time-consuming or require strong assumptions on cloth configurations.

With the rise of deep learning, researchers have recently employed data driven techniques to obtain large amounts of interaction data with cloth to learn manipulation policies using powerful function approximators, often with the help of simulators [105, 174]. These works tend to learn quasi-static pick-and-place policies, which allow the cloth to settle between robot actions [54, 68, 106, 112, 144, 158, 159, 188, 197, 204]. Other researchers have learned continuous servoing policies [122], dynamic policies [60] or have explored learning cloth manipulation from purely real world interaction [95].

In contrast to these works which employ vision-manipulation policies, we focus on tactile sensing for cloth grasping.

5.2.2 Grasping for Cloth Manipulation

Perhaps the most important part of cloth *manipulation* tasks is cloth *grasping*, since a suitable grasp is necessary for subsequent actions such as dragging or lifting. Defining and identifying ideal cloth grasps remains challenging and is the subject of extensive research [12]. Early cloth manipulation research focused on vertically smoothing via gravity. A common such grasping strategy to reliably standardize cloth was to hold it with one gripper while iteratively grasping the lowest hanging corner with the other gripper [27, 87, 88, 119].

Other cloth grasping techniques do not require assuming that the cloth is lifted in midair. For example Ramisa et al. [146] and Sun et al. [165] determine suitable grasping points for cloth on a flat table by detecting wrinkles and edges using depth and classical computer vision techniques. Other applications of cloth manipulation may utilize specialized gripper designs [90] or may simplify the process by assuming that cloth is gripped in advance of the task [74].

Recently, Qian et al. [145] study how to robustly grasp cloth using dense segmentation of images to distinguish between edges and interior creases. Their method involves a self-supervised labeling procedure and a sliding grasp. Nonetheless, robustly grasping cloth remains challenging, particularly when the goal is to generalize to a wide variety of types and configurations of cloth. Prior work has reported that a typical failure cases is grasping the wrong number of cloth layers, particularly when unfolding [122, 158, 188]. Furthermore, many works employ heuristics such as hand-tuning the gripper design and grasp depth [54].

Prior work has also investigated learning to grasp one cloth from a stack using grasping and scooping actions from vision input only [29], as well as designing a robot system to turn a single book page using vision and force sensing [58]. In this work, we consider the novel task of grasping more than one cloth layer, and show the benefits of tactile sensing without requiring vision.

5.2.3 Tactile Sensor Hardware

The robotics community has developed numerous tactile sensors. Examples of sensors include the class of optical-based sensors such as GelSight [208], GelSlim [36], and DIGIT [93], which have been used for cloth perception. For example, Yuan et al. [210] demonstrate how to use active learning to identify where to grasp a garment to classify it among several categories of clothing, and Yuan et al. [209] and Luo et al. [108] study how to combine tactile information with vision to infer properties of cloth. Similarly, Khan et al. [82, 83] use tactile information to classify single fabrics into different material and clothing types using piezoelectric pressure sensors [30]. In addition, von Drigalski et al. [182] use tactile sensing and rubbing behavior to classify textiles into one of 18 categories, and can distinguish between 1 or 2 layers. In contrast to these works, we focus on cloth manipulation instead of pure perception,

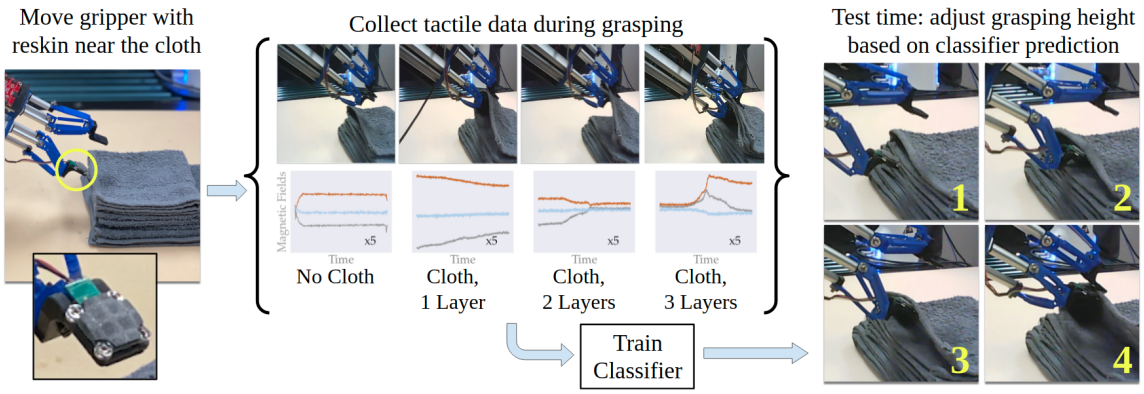


Figure 5.2: The proposed tactile-based cloth manipulation pipeline. A 7-DOF Franka robot uses a mini-Delta [120] gripper with two finger tips, the lower one of which has a ReSkin [8] sensor (see yellow circle and zoomed-in inset). Using this gripper, we collect tactile data from the ReSkin by performing grasps of different categories: grasping nothing, or pinching 1, 2, or 3 cloth layers (see Fig. 5.3 for more examples). The graphs above visualize the tactile time series data. At test time, the robot uses the trained tactile-based classifier to grasp a desired number of cloth layers.

and additionally focus on fine-grained manipulation which may be challenging with sensors such as the GelSight due to their relatively large size.

Other types of robotic tactile sensors include BioTac [166] and stretchable piezoresistive [7] sensors. These sensors are durable, but remain expensive and may not be easily replaceable. Research teams have also explored tactile sensing using a customized force-torque sensor [129] for manipulating deformable blocks [152]. To our knowledge, none of these sensors have been used for cloth manipulation tasks.

Recently, Bhirangi et al. [8] proposed the ReSkin, a class of magnetic sensors which is well suited to machine learning due to its low cost, durability, form factor, ability to cover a large area, and ease of replacement. The researchers demonstrate ReSkin on robotic grasping tasks that involve handling delicate objects such as blueberries and grapes. Due to these advantages and existing applications, we use the ReSkin for novel cloth manipulation tasks that involve fine-grained manipulation of cloth layers.

5.3 Problem Statement

We study the task of grasping a desired number of layers from a stack of cloths. Given a set of at least 3 cloth layers stacked on each other, the goal is to grasp the

top $k \in \{1, 2\}$ cloth layers. For each *trial* (a given instance of the task), we specify a target value for k. We assume a robot has a two-finger gripper where one of the gripper tips is equipped with a *tactile sensor*. We assume each trial begins with the robot's tactile sensor facing a set of edges from a stack of cloth layers, as shown in Fig. 5.1. A trial is a *success* when the robot grasps exactly k cloth layers and is able to lift its gripper upwards by 4 cm while preserving its grasp of the k layers.

5.4 Approach

This proposed system for tactile sensing involves designing hardware with tactile data (Sec. 5.4.1), training a classifier to distinguish grasping cloth layers (Sec. 5.4.1), then using this classifier for a grasping policy (Sec. 5.4.2). See Fig. 5.2 for the overall pipeline.

5.4.1 Hardware

The proposed system uses a ReSkin [8] sensor, which comprises of a soft magnetized skin and a circuit board with a 5-magnetometer array (see bottom-left inset of Fig. 5.2). The board sits beneath the skin, and any deformations caused by normal/shear forces are read via distortions in magnetic fields. For each of the 5 magnetometers, 3 magnetic flux values $\langle B_X, B_Y, B_Z \rangle$ are reported, corresponding to flux in the X-, Y-, and Z- magnetometer coordinate axes. Concatenating these values for a single time step t results in a 15-dim vector $\mathbf{B}^{(t)} \in \mathbb{R}^{15}$. ReSkin publishes these values at up to 400 Hz.

We attach ReSkin to a finger on a mini-Delta gripper [120]. We use the mini-Delta largely due to its length and form factor, since it facilitates grasping a layered stack of cloth folds by approaching it from the side, instead of top-down. The mini-Delta has 3 DOFs for each finger, and is compliant due to the 3D-printed soft links (blue component in Fig. 5.1), though in this work, we do not rely on the additional DOFs or on compliance. Our contribution centers on tactile sensing for grasping of cloth layers, and we leave investigation of exploiting additional DOFs and compliance for future work. The gripper and attached sensor are mounted on a 7-DOF Franka robot.

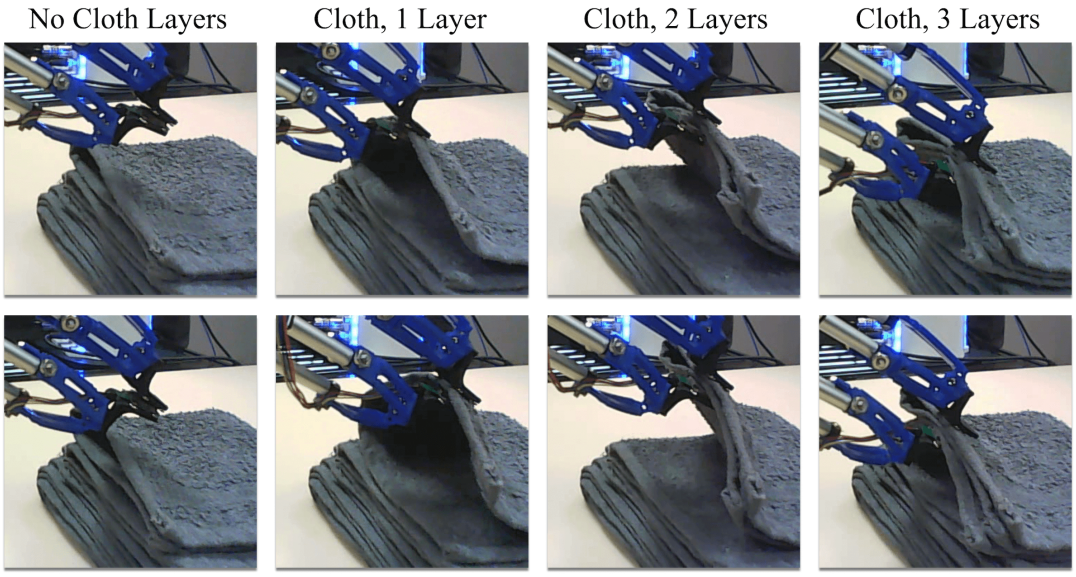


Figure 5.3: Examples of collecting data for tactile-based classification, with the ReSkin attached to the bottom gripper finger tip. From left to right, we show two examples each of collecting data with (1) contact, but without cloth, (2) 1 cloth layer, (3) 2 cloth layers, and (4) 3 cloth layers. The classifier only takes as input the data collected from the ReSkin sensor $\mathbf{B}^{(t)}$ at any give time step. As a baseline for comparison, we also train an image-based classifier which uses the RGB images above, which are collected with a webcam. See Sec. 5.4 for further details.

Grasp Classifier Training

We train a classifier to predict the number of cloth layers grasped to use as part of the grasp policy (Sec. 5.4.2). The classifier takes as input a tactile reading from a single time step $\mathbf{B}^{(t)}$. While analyzing sensory data across a time series seems natural for the tactile modality, we find that predictions based on point estimates are surprisingly effective, as we later show in Sec. 5.6. We do not take proprioceptive data as input, as this modality is not currently available with the mini-Delta gripper: the compliant links can bend from their commanded position given sufficiently high external force, and estimating proprioception for these types of compliant links is an area of active research.

The classifier uses the tactile readings to predict how the gripper is interacting with the cloth, among 4 classes: (1) pinching with no cloth between the fingers, (2) pinching 1 cloth layer, (3) pinching 2 cloth layers, and (4) pinching 3 cloth layers.

We limit the number of cloth layers under consideration to 3 to make classification tractable, while also allowing feedback-based policies to recover if they overshoot when grasping two layers. We leave classifying an arbitrary number of layers to future work.

We collect training data in the real world for the classifier due to the lack of a suitable simulator.[1] We define a single "training episode" as the process of getting a set of tactile data from one grasp. First, a human stacks several layers of cloth with edges facing the gripper. The height at which the robot approaches the cloth is uniformly sampled per attempt within a ± 2 mm range to collect a variety of grasps. The robot then approaches the cloth, closes its fingertips to grasp firmly, records ReSkin data during the grasp, then releases. Each training episode lasts roughly 5 seconds and produces approximately 350 sensor readings of 15 values each (3 per magnetometer). We visually inspect videos from the recorded data to determine the number of grasped cloth layers, and we label all points from a training episode with the same label, speeding up annotation time and effort. See Fig. 5.3 for example visualizations of training episodes for all classes.

We then use this collected data to train a classifier to distinguish the numbers of layers grasped from the tactile readings. We experimented with various types of classifiers, including k-Nearest Neighbor (kNN), SVM, Logistic Regression, and Random Forests, and we found the performance to be fairly similar across classifiers. For simplicity, we use a k-Nearest Neighbor (kNN) classifier with $k = 10$ neighbors.

5.4.2 Proposed Grasp Policy

Next we describe how we use the above trained classifier to enable the robot to grasp the desired number of layers. We divide the robot trajectory into three parts. First, the gripper moves vertically down by a distance d_{vert}, then horizontally towards the cloth stack by a distance d_{slide}, then lifts up by a distance d_{lift}, then closes its gripper tips (see Fig. 5.4 for a visualization). At this point, we record tactile data and classify the number of layers that are grasped. If the predicted number of grasped layers (according to the classifier) matches the target number of grasped layers, it lifts the

[1]While there has been progress in developing high-fidelity simulators for tactile sensors [161, 185] and for deformables [105, 114], simulating both is challenging and to our knowledge has not yet been shown.

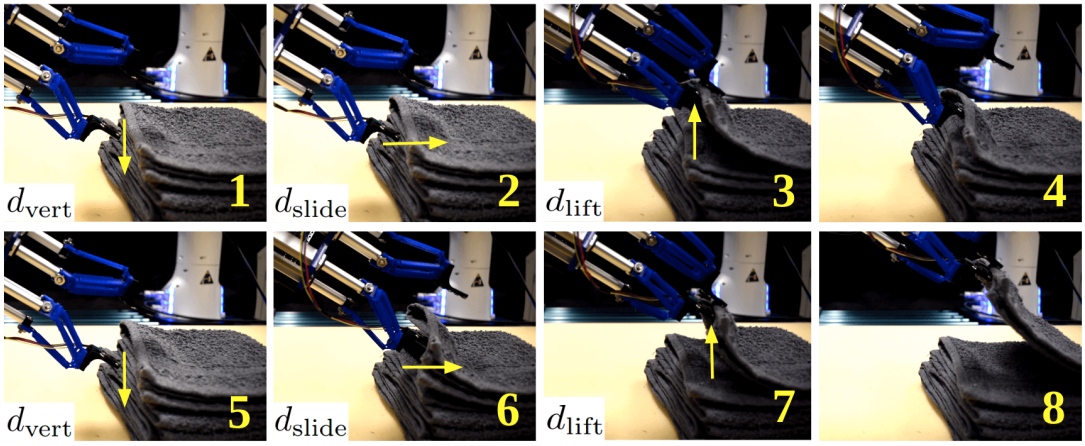


Figure 5.4: The proposed grasp policy parameterization (described in Sec. 5.4.2), visualized with a frame-by-frame overview of an example trial from the experiments. Each row, consisting of four frames, shows one action. The first part of an action (shown in frames 1 and 5) adjusts the initial gripper height by d_{vert}, possibly from prior tactile feedback. The second part of an action (shown in frames 2 and 6) moves towards the cloth stack by some distance d_{slide}. Then, the third part (frames 3 and 7) lifts upwards by d_{lift} and closes the grippers. At this point, the robot queries the classifier and may decide to release and re-attempt the grasp (frames 4 and 5) or the robot concludes that it has grasped the correct number of layers and further lifts the cloth to end the trial (frame 8).

gripper further by 4 cm to indicate the end of the trial; otherwise, it resets the gripper back to the starting position and tries again (see below for details). The values of d_{slide} and d_{lift} are tuned and fixed ahead of time by a human operator, while d_{vert} is determined by the policy, as explained below.

The grasping policy uses the output of the grasp classifier (Sec. 5.4.1) to determine the vertical distance that the gripper lowers before grasping, d_{vert}. For a target number of layers k to grasp, the robot begins at some height with the grippers open, moves towards the cloth stack, and attempts a grasp. If the grasp classifier determines that it has not grasped the correct number of layers, then the robot releases, moves back, and adjusts the gripper height (d_{vert}). If the classifier predicts that it has grasped too many layers, d_{vert} is decremented by a small value to decrease the grasp height; if it has grasped too few, d_{vert} is incremented by a small value. The policy continues until either the classifier determines that it has grasped the desired number of layers and ends the trial, or until the maximum number of grasp attempts is reached.

During each grasp attempt on the physical system, the classifier starts predicting

the class once the gripper closes, and stops predicting after the robot lifts by d_{vert}. This results in a set of about 160 separate predictions. We use the mode of all the predictions as the final prediction to determine whether to raise or lower the grasp height.

5.5 Physical Experiments

We evaluate the methods using the physical system described in Sec. 5.4.1. The experiments are designed to answer the following questions:

- Can magnetometer-based tactile sensing with ReSkin sensors provide sufficient information about grasping a target number of cloth layers?
- What are the benefits of the proposed method that uses tactile-based feedback to adjust the gripper height?
- Can a classifier trained on tactile feedback generalize to different cloths?

5.5.1 Experiment Protocol

We train our tactile classifier on a gray cloth; we then evaluate our system on the gray training cloth and on two other unseen cloths to measure the generalization of our method to new cloths (see Fig. 5.6). We use the same training data from the gray cloth for all of the tactile-based method variations described in Sec. 5.5.2. The tactile data collection results in a total of 18,838 such $\mathbf{B}^{(t)}$ readings. We train a classifier on 95 % of the training episodes (to allow for a small validation set). We normalize the tactile data so that each of the 15 features has mean 0 and variance 1 in the training set.

We perform two sets of experiments, in which we set the desired number of cloth layers to grasp as one layer and two layers, respectively. Each *trial* begins with a human arranging folded cloths on the workspace with edges exposed and facing the robot gripper. The number of folded cloths is the same across trials, but variations in the depth of the layers up to 1.5 mm can occur due to slight differences in the initial cloth configuration. We initialize the robot's gripper at an angle (30°) which increases the likelihood that a horizontal motion can slide the robot finger tips in between layers of cloth.

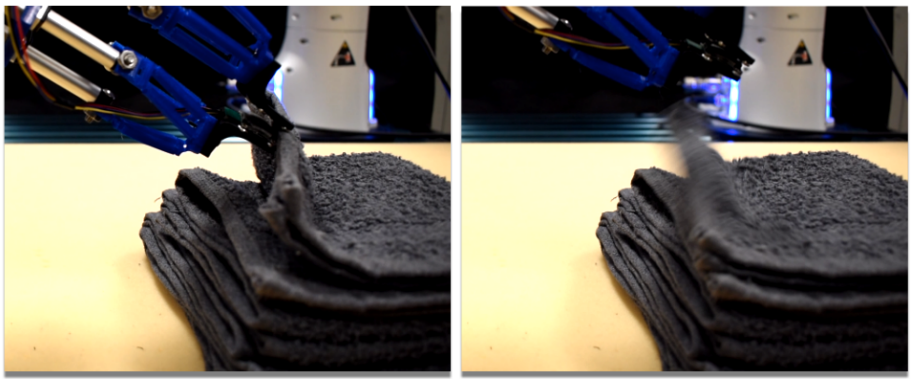


Figure 5.5: An example grasping failure case of the task. Due to an insufficiently robust grasp when lifting (left), the layers may slip out of the robot's control during the lifting portion (right).

Each experiment set consists of comparing several grasping methods (see Sec. 5.5.2). When running experiments, we randomly sample the method to run in the given trial *after* the cloths have been set, to reduce potential human bias in the data initialization. The robot employs the selected method to grasp the appropriate number of cloth layers. The robot is allowed up to $T = 10$ actions per trial, though it can terminate earlier if the classifier estimates that it has grasped the appropriate number of layers. Upon termination, the robot lifts the gripper by 4 cm and a human measures this as a success if the correct number of layers are still grasped. All other cases result in the trial as a failure.

We categorize failures into two types, *prediction* and *grasping* failures. Prediction failures are a result of mis-predictions by the trained classifier, where it either: (1) incorrectly predicts that the robot has grasped the desired number of layers and terminates the trial prematurely, or (2) the classifier incorrectly predicts that the robot has grasped the wrong number of layers, causing unnecessary regrasps and leading to the robot reaching the max number of attempts for the trial. Grasping failures are due to either failing to grasp the desired number of layers at the last time step in a trial, or failing to robustly grasp the cloth, such that cloth layers slip out of the robot's control when lifting (see Fig. 5.5).

Figure 5.6: The cloths we use for experiments. We use the gray towel (left) for training, and test on all 3 cloths for evaluation. The white towel and patterned cloth test generalizing to novel cloths. The cloths have thicknesses between 3-5 mm and variation in surface texture and stiffness.

5.5.2 Methods and Baselines

We evaluate the following methods for grasping 1 and 2 cloth layers:

1. **Fixed-Open-Loop**: Initialize the gripper at a fixed height, manually tuned for grasping 1 or 2 cloth layers: $d_{\text{vert}}^{(1)}$ and $d_{\text{vert}}^{(2)}$ respectively. This method terminates after a single trial as it has no access to feedback.
2. **Random-Tactile**: Randomly try different gripper heights within the range $\left[d_{\text{vert}}^{(2)} - 2\,\text{mm}, d_{\text{vert}}^{(1)} + 2\,\text{mm}\right]$ until the tactile classifier determines that the correct number of layers have been grasped.
3. **Random-Image**: Same as Random-Tactile, but uses an image classifier (instead of a tactile classifier) to determine when the correct number of layers has been grasped. The image classifier is an 18-layer ResNet [64] pre-trained on ImageNet and finetuned on the images from the same training episodes used to train the tactile classifier.
4. **Feedback-Image**: Same as Feedback-Tactile (our method, below) except with the image classifier.
5. **(Ours) Feedback-Tactile**: Initialize the gripper height to $d_{\text{vert}}^{(1)} + 2\,\text{mm}$; use the grasp policy described in Sec. 5.4.2 to adjust the height per grasp ($\pm 2\,\text{mm}$) based on the tactile classifier predictions.

Class \Prediction	0	1	2	3
0 (0 Layers)	1.000	0.000	0.000	0.000
1 (1 Layer)	0.000	0.999	0.000	0.001
2 (2 Layers)	0.030	0.003	0.866	0.100
3 (3 Layers)	0.128	0.256	0.138	0.478

Table 5.1: The average normalized confusion matrix from the cross-validation training results for the k-nearest neighbor classifier we use for tactile-based experiments.

5.6 Results

We first present results from training a classifier on ReSkin data followed by physical experiment results in which we run 10 trials for each method and condition.

5.6.1 The Tactile Classifier

To better understand the kNN performance, we perform 100 folds of cross-validation and average the validation performance. Each entire training episode is assigned to either the training or validation set.

Table 5.1 demonstrates the average normalized confusion matrix obtained from these 100 cross-validation runs, and also reports the average per-class accuracy. We also computed the average balanced accuracy metric [181] to consider the data imbalance and obtain $\mathbf{0.84 \pm 0.06}$. Inspecting the confusion matrix, we find that the tactile classifier can classify classes 0 (*i.e.,* pinching with no cloth between the fingers) and 1 (*i.e.,* pinching 1 cloth layer) with extremely high effectiveness. Results for classes 2 and 3 suggest that identifying 2 and 3 cloth layers is more challenging.

5.6.2 Grasping 1 Cloth Layer

In the first set of physical experiments, we report the success and failures of methods on grasping and lifting the top layer of cloth from a stack. See Table 5.2 for results. Our method, Feedback-Tactile, succeeds at grasping one layer of cloth in all 10 trials, whereas all competing ablations have lower success rates. Methods with the tactile classifier outperform those using the image classifier, with most failures attributed to mis-prediction rather than poor grasping.

Cloth Type	Method	Success Rate ↑	Prediction Failure	Grasp Failure	Attempts ↓
Gray Towel (Train)	Fixed-Open-Loop	6/10	-	4/10	1 (fixed)
	Random-Image	5/10	5/10	0/10	1.8±0.7
	Random-Tactile	6/10	3/10	1/10	4.8±2.8
	Feedback-Image	8/10	2/10	0/10	2.3±0.8
	Feedback-Tactile	**10/10**	0/10	0/10	3.1±1.0
White Towel (Generalization)	Feedback-Image	3/10	5/10	2/10	1.6±0.5
	Feedback-Tactile	**8/10**	0/10	2/10	2.3±0.8
Patterned Cloth (Generalization)	Feedback-Image	2/10	8/10	0/10	5.1±4.3
	Feedback-Tactile	**7/10**	2/10	1/10	4.6±3.2

Table 5.2: Results for the first set of physical experiments described in Sec. 5.6.2 with grasping at 1 cloth layer. We run all methods for 10 trials each and report the success rate, the failure types (grasping and prediction failures), and the average number of grasp attempts per trial.

The fixed-height open loop method (Fixed-Open-Loop) poorly handles variations in the initial cloth configuration. There can be up to 1.5 mm variation in the height of the cloth stack based on how they are placed at the start of the trial, which can lead to failures in the open loop grasping method. Both random grasping approaches, Random-Image (5/10) and Random-Tactile (6/10) have lower success rates compared to using feedback-based height adjustment with Feedback-Image (8/10) and Feedback-Tactile (10/10).

For testing generalization, Feedback-Tactile significantly outperforms Feedback-Image on the white towel and patterned cloth. Feedback-Tactile obtains 8/10 and 7/10 success rates for the white towel and patterned cloth, respectively, while Feedback-Image only succeeds in 3/10 and 2/10 trials.

We have analyzed the failure types of each method in Table 5.2. Grasping failures are rare for most methods on 1-layer grasping; grasping failures can occur if the robot does not robustly grip the cloth, and cloth slips out of the grasp when the robot lifts it (see Fig. 5.5). Our method (Feedback-Tactile), also has few prediction failures when generalizing to unseen cloths compared to Feedback-Image.

Cloth Type	Method	Success Rate ↑	Prediction Failure	Grasp Failure	Attempts ↓
Gray Towel (Train)	Fixed-Open-Loop	7/10	-	3/10	1 (fixed)
	Random-Image	6/10	1/10	3/10	5.3±3.0
	Random-Tactile	4/10	4/10	2/10	6.0±3.0
	Feedback-Image	**9/10**	0/10	1/10	4.7±0.9
	Feedback-Tactile	7/10	1/10	2/10	6.4±2.6
White Towel (Generalization)	Feedback-Image	0/10	8/10	2/10	9.2±1.8
	Feedback-Tactile	**4/10**	2/10	4/10	5.0±3.4
Patterned Cloth (Generalization)	Feedback-Image	0/10	10/10	0/10	10.0±0.0
	Feedback-Tactile	**1/10**	3/10	6/10	6.4±3.6

Table 5.3: Experimental results for grasping at the top 2 cloth layers as described in Sec. 5.6.3. Besides the change of 1 to 2 layers, the results are formatted in the same way as in Table 5.2.

5.6.3 Grasping 2 Cloth Layers

In the next set of experiments, we evaluate grasping and lifting the top two layers of cloth. The results in Table 5.3 suggest that the methods achieve success rates similar to 1-layer grasping (Table 5.2) for the gray towel, but performance is lower on the unseen cloths. While Feedback-Image performs slightly better than Feedback-Tactile on the gray towel, Feedback-Tactile performs slightly better on unseen cloths.

Table 5.3 shows that both prediction and grasping failures lead to errors for our method (Feedback-Tactile), though grasping failures are more common (accounting for 2/3 of our total failures). The higher incidence of grasp failures by our method in this experiment suggests that 2-layer grasping is more difficult than 1-layer grasping. Fig. 5.7 highlights some challenges with grasping two layers; for example, we observe that failures tend to occur due to crumpling the fabric when attempting to grasp 2 layers. Furthermore, the top layer of cloth can push downwards on the layer below it, which reduces the gap between the second and third layers; this reduced gap can make it difficult to grasp 2 layers. These observations and results suggest that further innovation on grasp policies may be necessary to improve 2-layer grasp performance on unseen cloths.

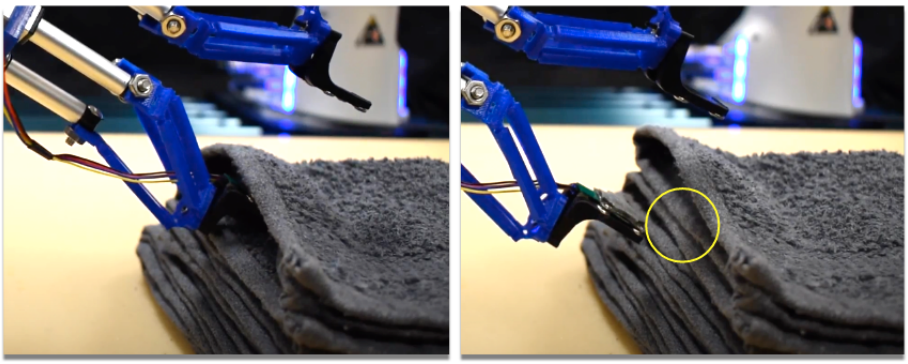


Figure 5.7: A qualitative example of how the task is challenging, particularly with grasping two layers. Because of the horizontal motion of the gripper, layers of cloth can be pushed apart (left), creating air pockets between the top and second layer after the action has finished (right). This gap makes it easier to grasp the top layer but harder to grasp the top *two* layers, due to a smaller gap between the second and third layers (see overlaid yellow circle).

5.7 Conclusion

In this thesis chapter, we present a robotic system that uses magnetometer-based tactile sensing for precisely grasping layers of cloth. We train a classifier on tactile sensor readings from a ReSkin sensor. At test time, the classifier determines the number of layers of cloth grasped, which informs the policy to adjust the height of the gripper for subsequent grasp attempts. The system obtains strong results with grasping the top 1 or 2 cloth layers out of a stack of cloth, and generalizes to unseen cloth. We hope this work motivates future research on tactile-based robotic policies that can manipulate a wide variety of complex objects.

Chapter 6

FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy

Abstract

We address the problem of goal-directed cloth manipulation, a challenging task due to the deformability of cloth. Our insight is that optical flow, a technique normally used for motion estimation in video, can also provide an effective representation for corresponding cloth poses across observation and goal images. We introduce FabricFlowNet (FFN), a cloth manipulation policy that leverages flow as both an input and as an action representation to improve performance. FabricFlowNet also elegantly switches between dual-arm and single-arm actions based on the desired goal. We show that FabricFlowNet significantly outperforms state-of-the-art model-free and model-based cloth manipulation policies. We also present real-world experiments on a bimanual system, demonstrating effective sim-to-real transfer. Finally, we show that our method generalizes when trained on a single square cloth to other cloth shapes, such as T-shirts and rectangular cloths. Video and other supplementary materials are available at: https://sites.google.com/view/fabricflownet.

6.1 Introduction

Cloth manipulation has a wide range of applications in domestic and industrial settings. However, it has posed a challenge for robot manipulation: compared to rigid objects, fabrics have a higher-dimensional configuration space, can be partially observable due to self-occlusions in crumpled configurations, and do not transform rigidly when manipulated. Early approaches for cloth manipulation relied on scripted actions; these policies are typically slow and do not generalize to arbitrary cloth goal configurations [6, 40, 118].

Recently, learning-based approaches have been explored for cloth manipulation [67, 136, 157, 158, 203], including model-free reinforcement learning to obtain a policy [96, 196]. For a cloth manipulation policy to be general to many different objectives, it must receive a representation of the current folding objective. A standard approach for representing a goal-conditioned policy is to input an image of the current cloth configuration together with an image of the goal [96, 157].

We will show a number of downsides to such an approach when applied to cloth manipulation. First, the policy must learn to reason about the relationship between the current observation and the goal, while also reasoning about the action needed to obtain that goal. These are both difficult learning problems; requiring the network to reason about them jointly exacerbates the difficulty. Additionally, previous work has used reinforcement learning (RL) to try to learn such a policy [96, 196]; however, a reward function is a fairly weak supervisory signal, which makes it difficult to learn a complex cloth manipulation policy. Finally, while many desirable folding actions are more easily and accurately manipulated with bimanual actions, previous learning-based methods for goal-conditioned cloth manipulation have been restricted to single-arm policies.

In this thesis, we introduce FabricFlowNet (FFN), a goal-conditioned policy for bimanual cloth manipulation that uses optical flow to improve policy performance (see Fig. 6.1). Optical flow has typically been used for video-related tasks such as object tracking and estimating camera motion. We demonstrate that flow can also be used in the context of policy learning for cloth manipulation; we use an optical flow-type network to estimate the relationship between the current observation and a sub-goal. We use flow in two ways: first, as an input representation to our policy;

second, after estimating the pick points for a pick-and-place policy, we query the flow image to determine the place actions. Our method is learned entirely with supervised learning, leveraging ground truth particles from simulation. Our method learns purely from random actions without any expert demonstrations during training and without reinforcement learning.

Our learned policy can perform bimanual manipulation and switches easily between dual and single-arm actions, depending on what is most suitable for the desired goal. Our approach significantly outperforms our best efforts to extend recent single-arm cloth manipulation approaches to bimanual manipulation tasks [67, 96]. We present experiments on a dual-arm robot system and in simulation evaluating our method's cloth manipulation performance. FabricFlowNet outperforms state-of-the-art model-based and model-free baselines, and we provide extensive ablation experiments to demonstrate the importance of each component of our method to the achieved performance. Our method also generalizes with no additional training to other cloth shapes and colors. This thesis chapter contributes:

- A novel flow-based approach for learning goal-conditioned cloth manipulation policies that can perform dual-arm and single-arm actions.
- A test suite for benchmarking goal-conditioned cloth folding algorithms encompassing and expanding on goals used in previous literature [53, 67, 96]; we perform extensive experiments using this test suite to evaluate FabricFlowNet (FFN), baselines [67, 96], and ablations, demonstrating that FFN outperforms previous approaches.
- Experiments to demonstrate that FFN generalizes to other cloth colors and shapes, even without training on such variations.

6.2 Related Work

Bimanual Manipulation. A large body of research exists on dual-arm, or bimanual, manipulation [163]. Dual-arm systems allow for more complex behaviors than single-arm systems at the cost of greater planning complexity [43, 160], leading to research on closed kinematic chain planning [11, 169], composable skill learning [20, 200], and rewarding synergistic behavior [21]. Prior work has also explored bimanual

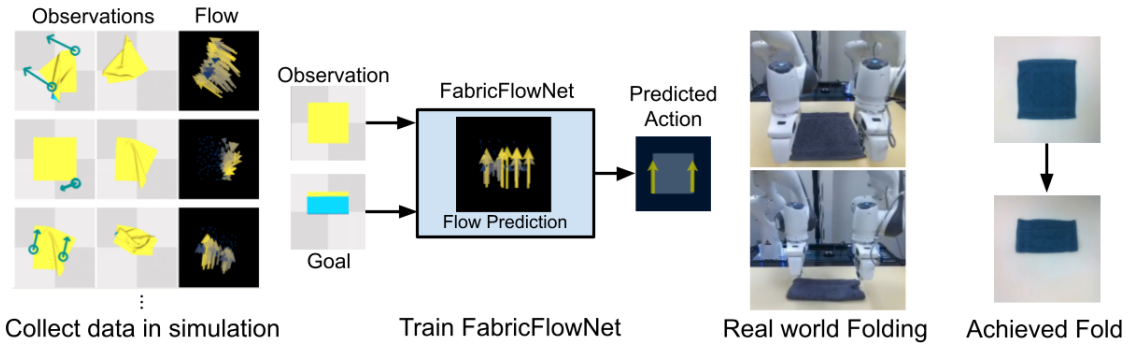


Figure 6.1: FabricFlowNet (FFN) overview. We collect a dataset of random actions and ground truth flow to train FFN. FFN learns to predict flow and uses it as both an input and action representation in a manipulation policy. FFN successfully performs single and dual-arm folding in the real world.

cloth manipulation [150], including establishing a diverse set of benchmark tasks [56]. Cloth manipulation is a highly underactuated task, and bimanual manipulation enables controlling multiple cloth points [13]. A common approach for cloth flattening is to lift a cloth with one arm and regrasp it with the other arm until it reaches the flattened configuration [6, 26, 40, 102, 118]. Previous work in this direction uses hard-coded policies [6, 40, 118], whereas we learn to achieve arbitrary folded configurations. Tanaka *et al.* [170] learn bimanual actions for goal-conditioned folding, using a voxel-based dynamics model to predict how actions will change the cloth state. However, optimizing this dynamics model can slow down inference time compared to our model-free approach. Dynamic bimanual manipulation has also been explored in simulation from ground-truth keypoints [73] and for unfolding cloth in the real world [60]; we perform real-world bimanual folding using depth image observations.

Learning for Cloth Manipulation. Prior works have proposed various hand-defined representations for cloth manipulation, such as parameterized shape models [126] or binary occupancy features [101]. Recent approaches use contrastive learning to learn pixel-wise latent embeddings for cloth [17, 53]. Both contrastive learning [53] and goal-conditioned transporter networks [157] have been applied to imitate expert demonstrations. Our approach doesn't require expert actions, just sub-goal states provided at test-time to define the task. In contrast to these previous representations, our method uses a flow-based representation, which we found to significantly outperform previous methods for goal-based cloth manipulation.

Other approaches have applied policy learning techniques to single-arm cloth smoothing [158, 196]. In contrast, we learn a policy that performs either single and dual-arm cloth manipulation; further, our focus is on goal-conditioned cloth folding, rather than smoothing. For cloth manipulation, Lee *et al.* [96] learns a model-free value function, but is limited by its discrete action space, and further, they do not use a flow-based representation, which we show leads to large benefits. Prior methods for learning goal-conditioned policies have used self-supervised learning to learn an inverse dynamics model for rope [136, 141] but such approaches have not been demonstrated for cloth manipulation. Lippi *et al.* [107] plan cloth folding actions in latent space, but do not demonstrate generalization to unseen cloth shapes. Other papers use an online simulator [102], or learn a cloth dynamics model in latent space [203], pixel-space [67], or over a graph of keypoints [111]. Unlike these model-based methods, our method is model-free and does not require online simulation or time-expensive CEM planning, leading to much faster inference. Further, we compare our approach to state-of-the-art approaches for cloth manipulation [67, 96] and show significantly improved performance.

Optical Flow for Policy Learning. Optical flow is the task of estimating per-pixel correspondences between two images, typically for video-related tasks such as object tracking and motion estimation. State-of-the-art approaches use convolutional neural networks (CNN) to estimate flow [37, 72, 171]. Optical flow between successive observations has previously been used as an input representation to capture object motion for peg insertion [35] or dynamic tasks [2]. Within the domain of cloth manipulation, Yamazaki *et al.* [202] similarly use optical flow on successive observations to identify failed actions. We use flow not to represent motion between successive images, but to correspond the cloth pose between observation and goal images, and to determine the placing action for folding. Argus *et al.* [3] use flow in a visual servoing task to compute residual transformations between images from a demonstration trajectory and observed images. In contrast, we learn a policy with flow to determine what cloth folding actions to take, not how to servo to a desired pose.

6.3 Learning a Goal-Conditioned Policy for Bimanual Cloth Manipulation

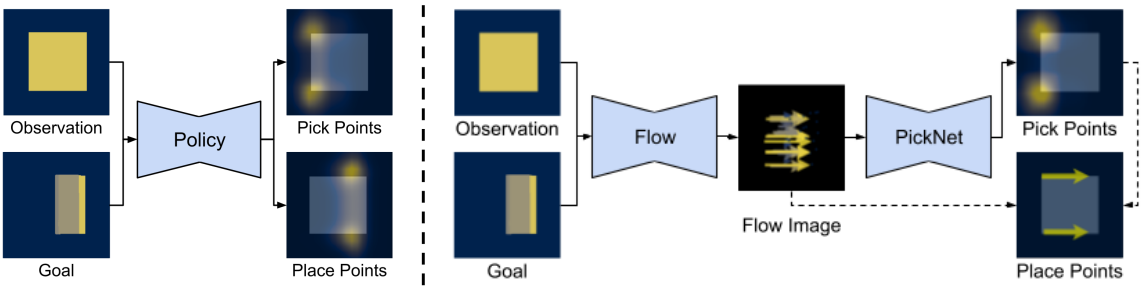
6.3.1 Problem Definition

Our objective is to enable a robot to perform cloth folding manipulation tasks. Let each task be defined by a sequence of sub-goal observations $\mathcal{G} : \{x_1^g, x_2^g, \ldots, x_N^g\}$, each of which can be achieved by a single (possibly bimanual) pick-and-place action from the previous sub-goal. We require sub-goals, rather than a single goal, because a folded cloth can be highly self-occluded such that a single goal observation fails to describe the full goal state. Defining a task using a sequence of sub-goals is found in other recent work [141]. Similar to prior work [136, 141], even if the sub-goals are obtained from an expert demonstration, we nonetheless do not assume access to the expert actions; this is a realistic assumption if the sub-goals are obtained from visual observations of a human demonstrator.

We assume that the agent does not have access to the sub-goal sequence $\mathcal{G}$ during training that it must execute during inference. Thus, the agent must learn a general goal-conditioned policy $a_t = \pi(x_t, \mathcal{G})$, where x_t is the current observation of the cloth and $a_t \in \mathcal{A}$ is the action selected by the policy. In our approach, we input each sub-goal x_i^g sequentially to our policy: $a_t = \pi(x_t, x_i^g)$. A goal recognizer [141] can also be used to decide which sub-goal observation to input at each timestep. For convenience, we will interchangeably refer to x_i^g as a goal or sub-goal.

6.3.2 Overview

A common approach for a goal-conditioned policy is to input the current observation x_t and the goal observation x_i^g directly into to a neural network representation of a policy [96, 141] or a Q-function [96, 196]. However, the network must reason simultaneously about the relationship between the observation and the goal, as well as the correct action to achieve that goal. Our first insight is that we can improve performance by separating these two components: we will learn to reason about the relationship between the observation and the goal, and separately use this relationship to reason over actions. Specifically, we represent this relationship using a "flow image"



(a) Naive system (b) FabricFlowNet (FFN)

Figure 6.2: (a) A naive approach to goal-conditioned policy learning is to input observation and goal images directly to the policy and predict the action. (b) FabricFlowNet separates representation learning from policy learning; it first estimates the correspondence between the observation and goal as a flow image. The flow is then used as the input to PickNet for pick point prediction, and as a way to compute place points without requiring additional learning.

f, which indicates the correspondence between the current observation x_t and sub-goal x_i^g. Thus we propose using the flow image f as an improved input representation of the policy, rather than directly inputting the observation x_t and goal observation x_i^g.

Our second insight is that we can also use flow in the output representation of the policy. We use a pick and place action space; prior methods that learn pick and place policies for deformable object manipulation predict place points using the policy network, either explicitly [136, 141, 157, 158, 196, 203] or implicitly by transforming the inputs to a Q-function [96]. Instead, we simplify the problem by leveraging flow: our policy network only learns to predict the *pick* points. For the place point, we query the flow image f for the flow vector starting at the predicted pick location, and use the endpoint of that vector as the place point.

We demonstrate that using flow in the two ways described above for our policy achieves significantly improved performance compared to prior work. Furthermore, our approach extends naturally to dual-arm manipulation, allowing us to easily transition between single and dual-arm actions.

A schematic overview of our system can be found in Fig. 6.2b. We first compute the flow f between the current observation x_t and goal x_i^g. Next, we input the flow f to a policy network (PickNet), which outputs pick points p_i. We then query the flow image $f(p_i)$ to determine the place points for each robot arm. Further details of our approach are described below.

6.3.3 Estimating Flow between Observation and Goal Images

We learn flow to use it as an input representation to our pick prediction network, and as an action representation for computing place points. Given an observed depth image x_t and desired goal depth image x_i^g, we estimate the flow $f = (f^1, f^2)$, mapping each pixel (u, v) in x_t to its corresponding coordinates $(u', v') = (u + f^1(u), v + f^2(v))$ in x_i^g. This task formulation differs from standard optical flow tasks as the input image pairs (x_t, x_i^g) are not consecutive images from video frames.

To capture the complex correspondences between x_t and x_i^g, we train a convolutional neural network to estimate the flow image f (see Appendix for details). The training loss we use to supervise the network is endpoint error (EPE), the standard error for optical flow estimation. EPE is the Euclidean distance between the predicted flow vectors f and the ground truth f^*, averaged over all pixels: $\mathcal{L}_{\text{EPE}} = \frac{1}{N}\sum_{i=1}^{N} \|f^* - f\|_2$. We use a cloth simulation to collect training examples with ground truth flow. The simulator provides the ground-truth correspondence between the particles of the cloth in different poses. The simulation cloth particles are not as dense as the depth image pixels; as a result, we only have ground-truth flow supervision for a sparse subset of the pixels that align with the cloth particles. Thus, we mask the loss to only supervise the flow for the pixels that align with the location of the cloth particles. We train the flow network using data collected from random actions. See Sec. 6.3.6 for more details on the simulator, data collection, and network training.

6.3.4 Learning to Predict Pick Points

Our bimanual action space $\mathcal{A}$ consists of actions $a = (p_1, p_2, q_1, q_2)$, where p and q are the pick and place points respectively, paired according to the subscripts. We train a neural network called PickNet to estimate the pick points p_1, p_2. Crucially, the input to PickNet is a flow image f, estimated between the current depth image x_t and the desired goal depth image x_i^g, as described in the previous section. The flow image indicates, for each pixel (u, v) in the current observation, the location $f(u, v)$ that the pixel has moved to in the goal observation. Our flow network (Sec. 6.3.3

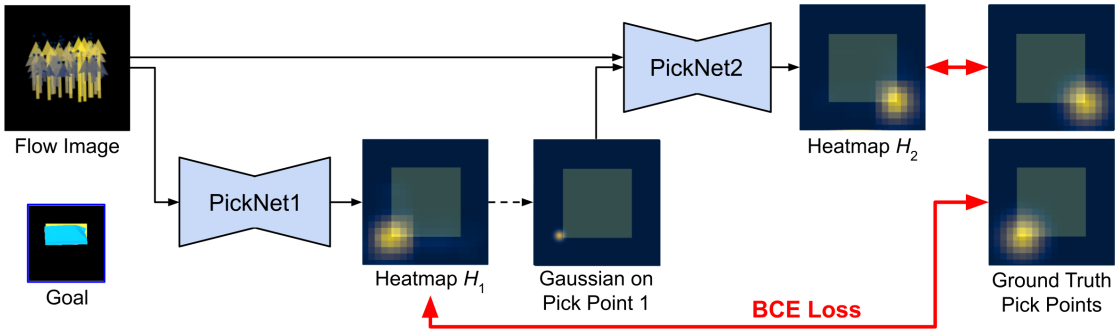


Figure 6.3: PickNet architecture. We utilize a two-network architecture for bimanual manipulation, where the second pick point is conditioned on the prediction of the first pick point.

above) reasons about the observation-goal relationship, so that the policy network (PickNet) only needs to reason about the action, specifically the two pick points (p_1, p_2); computing the place points is described in Sec. 6.3.5.

For dual-arm actions, the pick points must be estimated conditionally, as the location of pick point p_1 on the cloth influences the optimal location of pick point p_2, and vice versa. To decouple this conditional estimation problem, we propose a two-network architecture, PickNet1 and PickNet2, to estimate the pick points (see Fig. 6.3). This architecture was inspired by Wu *et al.* [196], which used two networks for pick-conditioned placing; we instead use two networks to condition dual-arm picking. PickNet1 is a fully convolutional network that receives flow image f as input and outputs a single heatmap H_1 estimating the optimal pick points for arm 1. We compute the first pick point as $p_1 = \text{argmax}_p\, H_1(p)$. The second network, PickNet2, predicts the second arm's pick point p_2 conditioned on p_1; PickNet2 takes as input both the flow image f and an additional image with a 2D Gaussian centered on p_1, and is otherwise identical to PickNet1. PickNet2 outputs heatmap H_2, from which we compute the second pick point: $p_2 = \text{argmax}_p\, H_2(p)$. The two-network architecture decouples the conditionally dependent pick point predictions and does not require us to resort to heuristics to extract two pick points from a single heatmap. We refer to PickNet1 and PickNet2 together as "PickNet."

To train PickNet, we collect a dataset of random actions (see Sec. 6.3.6 for details) and record the current observation x_t, the bimanual action $a = (p_1, p_2, q_1, q_2)$, and the next observation x_{t+1}. We also estimate the flow f from x_t to x_{t+1}, as explained

in Sec. 6.3.3. We create ground truth pick heatmaps H_i^* for arm i using the recorded random action a, by placing a 2D Gaussian $\mathcal{N}(p_i, \sigma)$ on each ground truth pick location p_i. We then supervise PickNet using the binary cross-entropy (BCE) loss between predicted heatmaps H_1, H_2 and ground truth heatmaps H_1^*, H_2^*. However, it might be unclear to the network which pick point should be output by PickNet1 and which should be output by PickNet2. We compute the loss for both possible correspondences and use the minimum:

$$\begin{aligned} l_{\text{BCE}}(H_i, H_j, H_i^*, H_j^*) &= \text{BCE}(H_i, H_i^*) + \text{BCE}(H_j, H_j^*) \\ \mathcal{L}_{\text{Pick}} &= \min[l_{\text{BCE}}(H_1, H_2, H_1^*, H_2^*), l_{\text{BCE}}(H_2, H_1, H_1^*, H_2^*)] \end{aligned} \tag{6.1}$$

At inference time, PickNet outputs the pick points p_1, p_2, computed from the argmax of H_1, H_2 respectively, as described above.

6.3.5 Estimating the Place Points from Flow

After estimating the pick points p_1, p_2 from flow, the remaining step to predict a bimanual pick and place action $a = (p_1, p_2, q_1, q_2)$ is to estimate the place points q_1, q_2. A straightforward approach would be to train the network to predict place points q_1, q_2, similar to the pick points p_1, p_2 as described above (see Fig. 6.2a). Instead, our approach uses the flow image to find the place points, so that the place points do not have to be learned separately.

Our approach makes the assumption that, to achieve a desired subgoal configuration, the point picked on the cloth should be moved to its corresponding position in the goal image (which is estimated by the flow). This is a simplifying assumption, since it is possible that the picked point will shift slightly after it is released by the gripper; our method does not take into account such small movements. Using this assumption, to compute the place points q_1, q_2, we query the flow f at each pick point p_1, p_2 to estimate the delta between the pick point location in the observation image and the corresponding location of the pick points in the goal image. We use these predicted correspondences as the place points: $q_i = f(p_i) + p_i$, for each arm i.

Action predictions estimated by our approach can produce nearly overlapping pick and place points, indicating that arm 1 and arm 2 should perform identical actions. We observe this behavior from PickNet when the goal is best achieved with a

single-arm action, rather than a bimanual one. On a real robot, grippers are likely to collide if grasping points that are too close. Therefore, to switch between executing a single-arm or bimanual action, we compute the L2 pixel distance between pick points $d_{\text{pick}} = \|p_1 - p_2\|_2$ and place points $d_{\text{place}} = \|q_1 - q_2\|_2$. We use a single-arm action when either distance is smaller than a threshold α, which we set to 30 for all experiments.

6.3.6 Implementation Details

We use SoftGym [105], an environment for cloth manipulation built on the particle-based simulator Nvidia Flex, to collect training datasets. The simulator models cloth as particles connected by springs. We use pickers that simulate a grasping action by binding to the nearest cloth particle within a threshold to execute pick and place actions in SoftGym. We collect data by taking random actions, biased towards grasping corners of the cloth. We demonstrate that we are able to train our method in SoftGym and then transfer the policy to the real world. Details on the data collection, as well as the network architecture and training details, can be found in Appendix Sec. D.1.1.

6.4 Experiments

6.4.1 Simulation Experiments

Experiment Setup. We evaluate FabricFlowNet (FFN) and compare to state-of-the-art baselines in the SoftGym [105] simulator; real-world evaluations are below in Sec. 6.4.2. Our experiments focus on folding tasks, and we assume that a cloth smoothing method (*e.g.,* [60, 158]) is used to flatten the cloth before folding is executed. Our error metric is the average particle position error between the achieved and goal cloth configuration. We evaluate on two sets of goals: 40 *one-step* goals that can be achieved with a single fold action, and 6 *multi-step* goals that require multiple folding actions. The multi-step goals each consist of a sequence of sub-goal images, with the next sub-goal presented after each action. This protocol follows from our problem formulation in Sec. 6.3.1, and is similar to the protocol in Nair *et al.* [136].

Our goals include test goals from Ganapathi *et al.* [53] and Lee *et al.* [96] that are achievable with one arm, as well as additional goals more suitable for two-arm actions (see Appendix Fig. D.2 for the full set of goals).

We compare our method to Fabric-VSF [67], which learns a visual dynamics model and uses CEM to plan using the model. We only use Fabric-VSF with RGB-D input, as depth-only input performs poorly for folding tasks [67]. FabricFlowNet only uses depth and does not rely on RGB, which enables our method to transfer easily to the real world without extensive domain randomization. We also compare to Lee *et al.* [96], a model-free approach. We extend the the original single-arm method to a dual-arm variant and compare against both. For both our method and the baselines, we only allow each method to perform one pick-and-place action for each subgoal (e.g. one pick and place action for each single-step goals). Additional baseline details can be found in the Appendix.

Simulation Results

Table 6.1 contains our simulation results for all methods. We report average particle distance error (in mm) for one-step goals only, multi-step goals only, and over both one-step and multi-step goals. Our results show that FFN achieves the lowest error over all goals and has the fastest inference time.

Table 6.1: Mean Particle Distance Error (mm) and Inference Time (sec) on Cloth Folding Goals

Method	One Step (n=40)	Multi Step (n=6)	All (n=46)	Inference Time
Lee *et al.*, 1-Arm [96]	16.18 ± 08.38	26.20 ± 16.31	17.49 ± 10.10	~ 0.04
Lee *et al.*, 2-Arm	36.62 ± 14.51	47.71 ± 21.95	38.07 ± 15.82	~ 0.04
Fabric-VSF [67]	6.31 ± 06.55	$\mathbf{21.33 \pm 11.20}$	8.27 ± 08.90	~ 420
FabricFlowNet (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$	$\sim \mathbf{0.007}$

We also investigate whether using flow as a goal recognizer improves performance. When an observation closely matches the goal, the flow for all points is close to zero. We leverage this fact by evaluating FFN with "iterative refinement": we allow the policy to take multiple actions per subgoal to try to further minimize the flow between the observation and subgoal. When the average flow between observation and current subgoal reaches a minimum threshold, the policy moves forward to the

next subgoal. FFN with iterative refinement achieves a mean error of 6.62 over all goals vs. 7.14 without refinement. Additional details on iterative refinement can be found in the Appendix, along with additional results from baseline variants, crumpled initial configurations, and end-to-end training.

Ablations

We run series of ablations to evaluate the importance of the components of our system; results averaged over all 46 goals are in Table 6.2. Additional details and results are in Appendix Sec. D.4. Our ablations are designed to answer the following questions:

What is the benefit of using flow as input? We modify PickNet to receive depth images of the observation and goal as input to the network ("NoFlowIn"), as is commonly done in previous work on goal-conditioned RL [96, 157]. In this ablation, the PickNet needs to reason about both the relationship between the observation and the goal, as well as the action. In contrast, our method uses the flow network to compare the observation and goal; the picknet separately reasons about the action.

What is the benefit of using flow to choose the place point? In this ablation, we train a network to predict the place points directly ("NoFlowPlace"). This is in contrast to our approach where we use the flow field, evaluated at the pick point $f(p_i)$, to compute the place point q_i for arm i. Our approach leads to a 32.4% improvement, showing the benefit to using flow as an action representation.

What is the performance with no flow? We combine the above two ablations and remove flow entirely, ("NoFlow"; ours has 60.4% improvement). The above ablations all indicate the strong benefit of using flow as both an input and action representation for cloth manipulation.

What is the benefit of biasing the data collection to grasp corners? Our method uses prior knowledge about cloth folding tasks to bias the training data and pick at corners of the cloth. In this ablation, we choose pick points randomly ("NoCornerBias", ours has 35.5% better performance).

What is the performance with a simpler architecture? We also compare our architecture for PickNet (Sec. 6.3.4) to a simpler architecture that takes as input the flow image I_f and outputs a two heatmaps, one for each pick point ("NoSplitPickNet"; ours has 2.1% better performance).

Does the loss formulation in Eq. 6.1 improve performance? We compare our method to an ablation where the first ground-truth heatmap is used to supervise PickNet1 and similarly for the second, i.e. $\mathcal{L}_{\text{Pick}} = l_{\text{BCE}}(H_1, H_2, H_1^*, H_2^*)$. ("NoMinLoss"; ours has similar performance).

Table 6.2: Mean Particle Distance Error (mm) for Ablations over All Goals (n=46)

NoFlowIn	NoFlowPlace	NoFlow	NoCornerBias	NoSplitPickNet	NoMinLoss	FFN (Ours)
9.37	10.56	18.02	11.07	7.29	7.15	**7.14**

6.4.2 Real World Experiments

We evaluate FabricFlowNet in the real world and demonstrate that our approach successfully manipulates cloth on a real robot system.

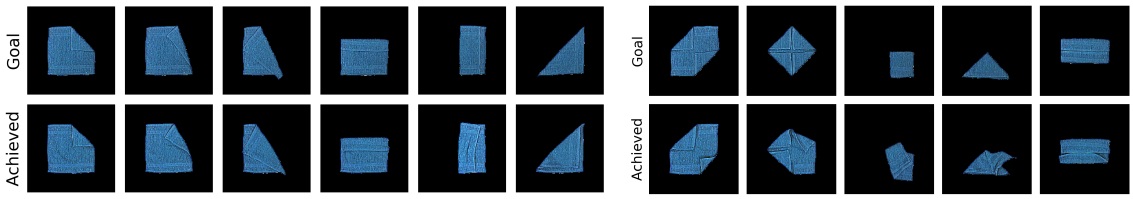
Experiment Setup. Our robot system consists of two 7-DOF Franka Emika Panda arms and a single wrist-mounted Intel RealSense D435 sensor (See Fig. 6.1). We plan pick and place trajectories using MoveIt! [22]. We evaluate on a 30x30 cm towel, using 6 single-step and 5 multi-step goals (see Fig. 6.4) that form a representative subset of our simulation test goals.

To transfer from simulation to the real world, we align the depth between real and simulated images by subtracting the difference between the average depth of the real support surface (i.e. the table) and the simulated surface. We mask the cloth by color-thresholding the background; see Appendix for details. We found that these simple techniques were sufficient to transfer the method trained entirely in simulation to the real world, because we use only depth images as input. Simulated depth images match reasonably well to real depth images, unlike RGB images.

Real World Results

Fig. 6.4 provides qualitative real world results, showing that we successfully achieve many of the goals. Our website (link in abstract) contains videos of these trials.

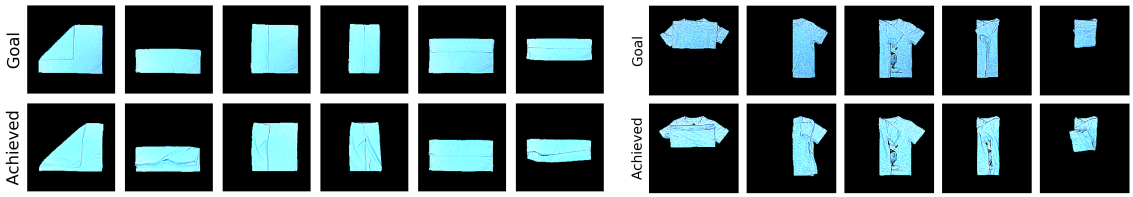
We compare FabricFlowNet to the NoFlow ablation from Sec. 6.4.1. Both methods used the same sim-to-real techniques described in the previous section. While we do not have access to the true cloth position error in the real world, Intersection-over-Union (IoU) on the achieved cloth masks serves as a reasonable proxy metric [96].



(a) One-step Square Cloth (b) Multi-step Square Cloth

Figure 6.4: Qualitative results for FFN on real world experiments. FFN only takes depth images as input, allowing it to easily transfer to cloth of different colors.

FFN achieves 0.80 mean IoU over 3 trials for the square cloth, compared to 0.53 for NoFlow. See the Appendix for additional details.



(a) Rectangular Cloth (b) Printed T-shirt

Figure 6.5: Generalization to new cloth shapes for FFN trained only on a square cloth in simulation. FFN achieves single and multi-step goals for rectangular fabric and a printed T-shirt.

Generalization. In addition to evaluating the folding policy on square cloth for various goal configurations, we also test the generalization of our method to other shapes of cloth. We evaluate the performance of FFN trained only on a square cloth on folding goals for a rectangular cloth as well as a T-shirt. These fabrics are also thinner than the square blue towel used in the real world experiments above. Fig. 6.5 shows that FFN trained on a square yellow cloth in simulation is able to generalize to other cloth shapes, textures, and colors (FFN only receives depth images as input). See Appendix for additional details.

6.5 Conclusion

In this work we present FabricFlowNet, a method which utilizes flow to learn goal-conditioned fabric folding. We leverage flow to represent the correspondence between

observations and goals, and as an action representation. The method is trained entirely using random data in simulation. Our results show that separating the correspondence learning and the policy learning can improve performance on an extensive suite of single- and dual-arm folding goals in simulated and real environments. Our experiments also demonstrate generalization to different fabric shapes, textures, and colors. Future work on flow-based fabric manipulation could incorporate actions beyond pick and place, such as parameterized trajectories or dynamic actions.

Chapter 7

Conclusions

This thesis presents visual action representations for manipulation tasks including grasping objects with non-Lambertian reflectance, manipulating fabric, and joint 6-DOF grasp and motion planning. Visual action representations enable manipulation under partial observability, without reconstructing the full state of the target object. These visual action representations forego explicit state estimation by grounding predicted actions on robot-object contact locations. The proposed methods then use these representations to reason about where to make contact, how to approach the desired contact location, and how to interact with an object after establishing contact.

7.1 Future Directions

The methods proposed in this thesis can be expanded upon to improve manipulation for tasks involving deformability, non-Lambertian reflectance, object shape generalization, and other challenges.

Volumetric Action Representations. Compared to 2D image-based representations, 3D volumetric representations are better able to reason about occlusions. Occlusions exist in nearly every scene due to limited camera views, object self-occlusions, or due to a robot's own embodiment. Research on manipulation under occlusion can potentially improve performance across many tasks.

Online Replanning with Multi-Modal Sensing. In this thesis, predicted

actions are executed without online replanning. Policies use closed-loop feedback, but feedback is provided only after the completion of each action. While executing actions without online replanning performed suitably for our task settings, manipulation in more dynamic environments will require online control. As robots often occlude objects once a manipulation action begins, the visual action representations proposed in this thesis can be extended to incorporate additional modalities such as haptic, tactile, and audio sensing to provide feedback when vision alone fails.

Reasoning about Robot Embodiment. The visual action representations proposed in this thesis assume that the robot used a parallel-jaw gripper to manipulate its environment. However, other robotic end-effectors exist, such as suction grippers and dexterous hands. Further, manipulation solely using the end-effector is overly restrictive, as a robot could use other parts of its embodiment to interact with the environment. We can extend policies and action representations to explicitly account for robot embodiment. Approaches that reason about robot morphology and geometry could perform non-prehensile manipulation actions like closing a door with an intermediate link of a robot arm instead of using an end-effector.

Appendix A

Chapter 2 Appendix

A.1 Network Architecture

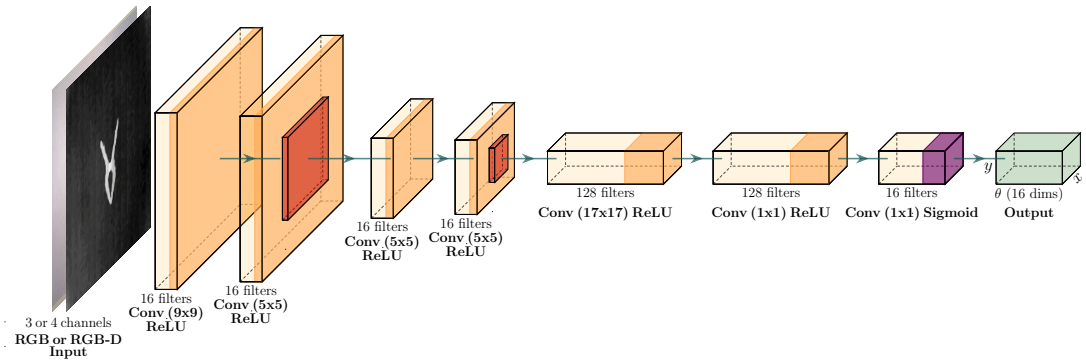


Figure A.1: Architecture diagram for supervision transfer networks, adapted from the FC-GQCNN [153] architecture. The input can be either 3-channel RGB input or 4-channel RGB-D input. The output is a 3D array of grasp quality scores over image coordinates x, y and rotation θ about the depth axis, discretized into 16 bins. The orange color accents correspond to ReLU activations and purple corresponds to sigmoid activation. The red layers are max pooling layers.

Fig. A.1 illustrates the architecture of the networks trained with supervision transfer.

A.2 Evaluations without random cropping

Table A.1 provides results for grasping in clutter without random cropping.

Table A.1: Performance on grasping in clutter by method without random cropping, averaged over five trials

Method	Opaque	Transparent	Specular
FC-GQCNN*	0.95 ± 0.05	0.26 ± 0.25	0.35 ± 0.23
RGB-ST†	0.77 ± 0.10	0.77 ± 0.15	0.68 ± 0.15
RGBD-ST†	0.62 ± 0.26	0.67 ± 0.19	0.75 ± 0.08
RGBD-M†	0.75 ± 0.13	0.60 ± 0.18	0.47 ± 0.28

*Trained on simulated grasps
†Trained on simulated grasps and opaque object images

Random cropping refers to sampling a 0.2m square crop from the input image and choosing the grasp with the highest probability from within the crop. Crops which have do not have any objects in them, as determined by whether the max grasp probability within the crop falls below a hand-defined threshold, are discarded and a new crop is sampled. This procedure helps prevent networks from repeatedly choosing highly rated false positive grasps. However, the cropping threshold must be tuned based on the performance of the grasping network. For our experiments, we used a threshold of 0.4.

A.3 Hyperparameters

Hyperparameters for networks trained with supervision transfer are:

- Learning rate: 1e-05
- Batch size: 64
- Number of rotation augmentations per image: 32
- Loss: Binary cross-entropy

The FC-GQCNN model we evaluated against was a pre-trained model from https://berkeleyautomation.github.io/gqcnn/.

Appendix B

Chapter 3 Appendix

B.1 Ablations for Neural Grasp Distance Fields

We perform ablation experiments for our trajectory optimizer (Table B.1). We compare using Adam [86] vs. a fixed step size ("No-Adam") for functional gradient descent. Unlike our method, CHOMP [214] originally uses a fixed or decaying step size, in the setting where the start and end trajectory configurations are not optimized (Sec. 3.4.2). In our setting, the end configuration is variable to allow optimization of the grasp pose. No-Adam converges slowly when the trajectory is far from a valid grasp pose, and overshoots when near the level set. We also evaluated using a decaying step size; while this mitigated the overshooting issue, convergence was still much slower, and the decay rate required tuning.

"No-Initial-IK" initializes the configuration at every timestep in the trajectory to the starting joint configuration, instead of using IK to initialize the trajectory as described in Sec. 3.5.2. We observe worse performance with No-Initial-IK as the initial trajectory is farther from the desired grasp trajectory, making it harder to plan.

Table B.1: Optimizer Ablation Results

	Grasp Execution ↑
NGDF, No-Adam	0.18
NGDF, No-Initial-IK	0.44
NGDF (Ours)	**0.61**

No-Adam uses CHOMP [214] with a fixed step size instead of Adam [86] optimization for the functional gradient update. No-Initial-IK initializes the trajectory so all steps in the plan start at the initial joint configuration. NGDF uses Adam and initializes the endpoint of the trajectory using inverse kinematics to achieve the best performance. 90 trials were performed as in Table 3.2.

B.2 Simulation Experiment Details

B.2.1 Camera Position in Simulation

Fig. B.1 shows the position of the four cameras in simulation.

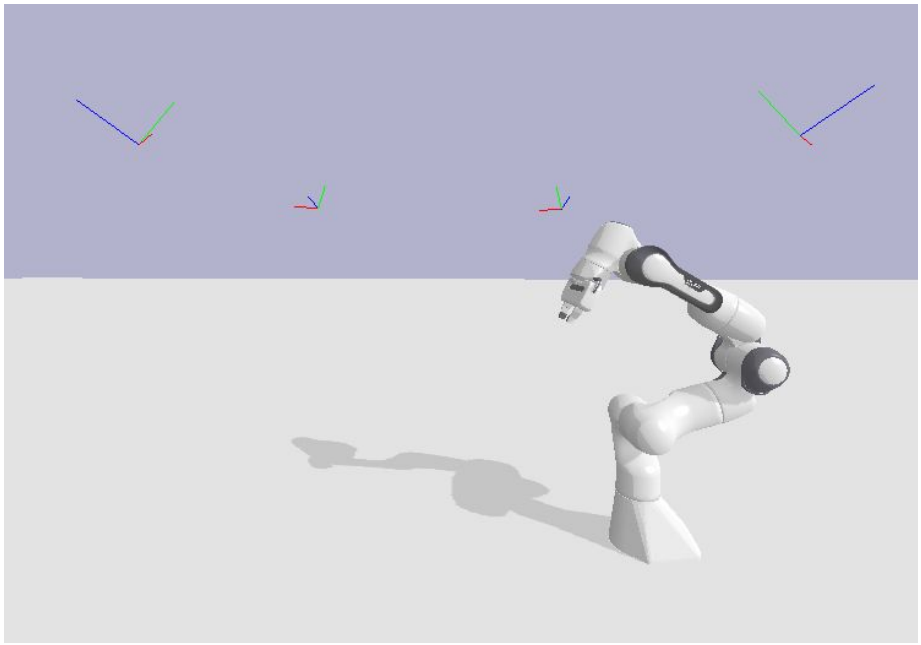


Figure B.1: Camera poses in simulation visualized as axes. The negative z axis (in blue) is the camera optical axis and points toward the robot workspace.

B.2.2 Qualitative Results

Fig. B.2 visualizes successful grasp trajectories in simulation for the reaching and grasping task from Sec. 3.5.1.

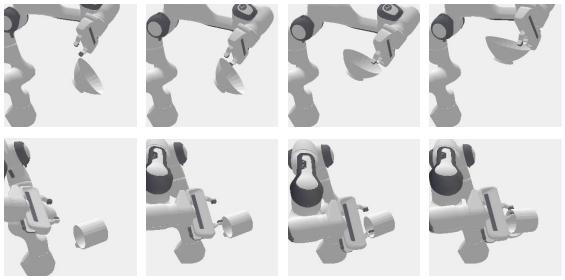


Figure B.2: Successful grasp trajectories (left-to-right) planned by our method for the bowl (top) and mug (bottom).

B.3 Real System Experiment Details

This section provides system implementation details for our real world experiments in Sec. 3.5.4. Our system consists of a 7-DOF Franka Panda robot and four Azure Kinect cameras (Fig. 3.5b), a similar setup to NDF [162].

B.3.1 Calibration

The Azure Kinect cameras were extrinsically calibrated using ColoredICP [139]. For camera intrinsics, the factory calibration was used. Robot-camera extrinsic calibration was performed using Tsai-Lenz [178]. The calibrated cameras produce a combined scene point cloud in the robot base frame.

B.3.2 Point Cloud Processing

To segment the object point cloud from the scene point cloud, we fit a table plane using RANSAC and remove points belonging to the plane. Outlier removal and DBScan [48] are used to refine the object point cloud. Our planner requires a signed distance field (SDF) of the object for collision avoidance, so we construct a mesh

from the object point cloud, then compute the SDF from the mesh using the tools provided in Wang *et al.* [183].

Even with four cameras, careful calibration, and point cloud processing, we recover partial point clouds with inaccuracies and noise (see Fig. B.3). Despite these deficiencies, our method achieves a high success rate on real objects in various configurations (Sec. 3.5.4), demonstrating robustness to perceptual errors.

B.3.3 Control

The output of the planner is a joint angle trajectory consisting of 30 timesteps. In order to execute the trajectory on the Franka Panda, the total duration of trajectory execution is set to 5 seconds, and the trajectory is interpolated using cubic spline interpolation to provide joint angles at 1 Hz. Impedance control [211] is then used to execute the high-frequency trajectory.

B.4 Experimental Setup

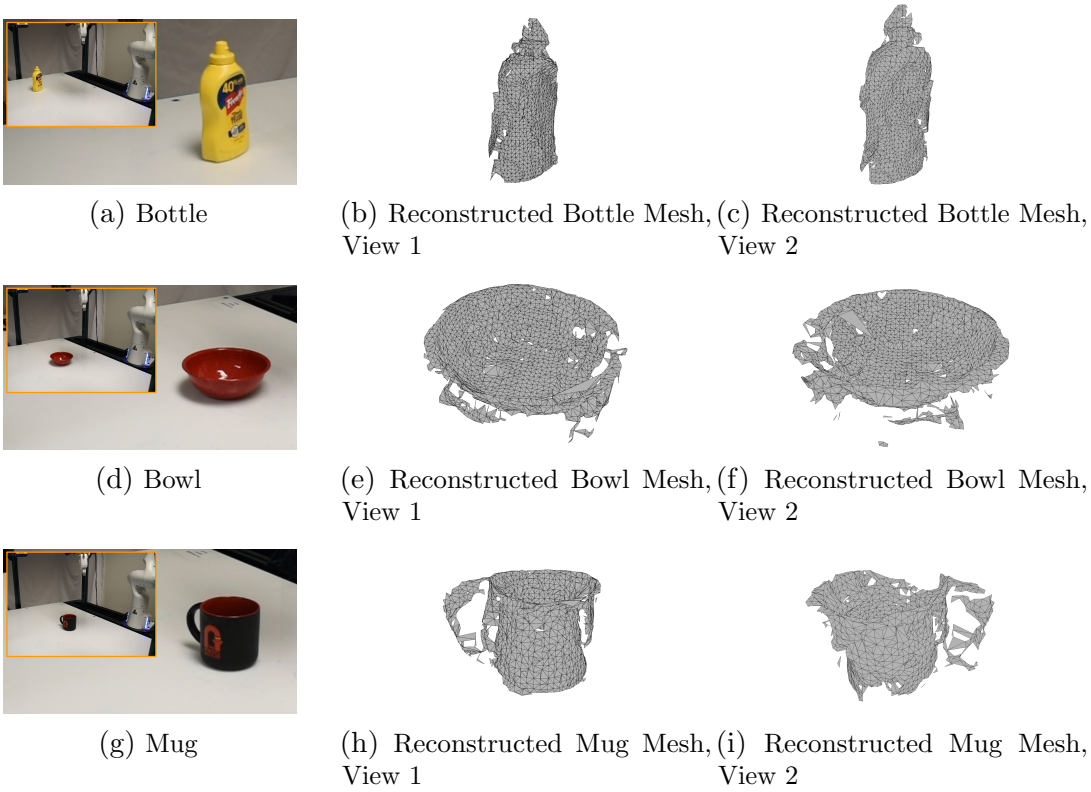
Table B.2: Real Object Pose Configurations

	Sampled Pose Configurations
Bottle 1 (Protein Drink)	Upright, Sideways
Bottle 2 (Mustard Bottle)	Upright
Bottle 3 (Coconut Water)	Upright, Sideways
Bowl 1 (YCB Bowl)	Upright
Bowl 2 (White Bowl)	Upright
Bowl 3 (Square Bowl)	Upright
Mug 1 (Black Mug)	Upright, Sideways, Upside Down
Mug 2 (YCB Mug)	Upright, Sideways, Upside Down
Mug 3 (Large Mug)	Upright, Sideways, Upside Down

The sampled pose configurations for objects in the real system evaluation. Objects are numbered left to right according to Fig. 3.5d. Some stable poses did not permit grasping and were omitted; for example, the mustard bottle cannot be grasped lying sideways as it is too wide.

Several of the test objects could be grasped via multiple stable pose configurations. For example, a mug can be grasped while upright, on its side, or upside down. For

objects with multiple graspable pose configurations, the configuration was randomly sampled for each trial. The 9 test objects (see Fig. 3.5d) had graspable stable pose configurations shown in Table B.2.



(a) Bottle

(b) Reconstructed Bottle Mesh, View 1

(c) Reconstructed Bottle Mesh, View 2

(d) Bowl

(e) Reconstructed Bowl Mesh, View 1

(f) Reconstructed Bowl Mesh, View 2

(g) Mug

(h) Reconstructed Mug Mesh, View 1

(i) Reconstructed Mug Mesh, View 2

Figure B.3: Meshes reconstructed during system evaluations. The first column shows the placement of the objects in each trial, along with a close-up image of the object itself. The second column shows the meshes reconstructed from the four depth cameras according to the procedure in Sec. B.3.2, posed roughly as they appear in the first column. The third column shows the back of each mesh. Note that these meshes and images are magnified for visual clarity and are not consistently scaled. Even with outlier removal and other mesh processing techniques, we observe inaccuracies in the reconstruction; however, our method is robust to these inaccuracies as demonstrated by our results in Sec. 3.5.4.

Appendix C

Chapter 4 Appendix

C.1 Network Architecture

Our network architecture is based on U-Net [148]. It consists of a downsampling part and an upsampling part. In the downsampling path, a step consists of two 3x3 unpadded convolutions, each with batch normalization and a rectified linear unit, followed by a 2x2 max pooling layer with stride 2. We apply four of these steps, doubling the number of feature channels each time. For the upsampling path, a step consists of a 2x2 up-convolution that halves the number of feature channels, a concatenation with a cropped feature map from the corresponding downsampled path, and two 3x3 convolutions, each followed by batch normalization and ReLU. A final 1x1 convolution is used to turn the feature map into 3 classes for corners, outer edges, and inner edges respectively.

The differences between our network and U-Net are that we add batch normalization, and our network takes a single channel depth image as input.

C.2 Network Training

We implemented the network in PyTorch. We use the Adam optimizer with a learning rate of 1e-5. We use a batch size of 8. To augment our data, we flip the image with 50 percent chance, and also rotate the image with 50 percent chance, sampling within

[-30 degrees, 30 degrees].

In our loss function, we set the per-class (corners, outer edges, and corners) weight w_k for balancing the loss on positive and negative predictions to 20 for all classes.

C.3 Grasp Direction Uncertainty Estimation

As described in Sec. 4.3.4.2, the uncertainty of the grasp direction for a single outer edge point $\mathbf{p}$ is the variance of the grasp directions predicted by its neighbors. Each neighbor is an outer edge pixel with its own grasp direction vector, computed as described in Sec. 4.3.4.1. We form the neighborhood by taking the k outer edge pixel points closest to $\mathbf{p}$, and set $k = 100$.

Appendix D

Chapter 6 Appendix

D.1 Additional Details and Results for FabricFlowNet

D.1.1 FabricFlowNet Implementation Details

Data Collection. We collect data in SoftGym by taking random pick and place actions on the cloth. The random actions are biased to pick corners of the cloth mask (detected using Harris corner detection [62]) 45% of the time, and "true" corners of the square cloth 45% of the time. If the true corners are occluded then Harris corners are used instead. For the remaining 10%, the pick actions are uniformly sampled over the visible cloth mask. After the pickers grasp the cloth, they lift to a fixed height of 7.5 cm.

We constrain the place points of the action so that both place points are offset in the same direction and distance from their respective pick points. The direction is orthogonal to the segment connecting the two pick points, and points towards the center of the image, so the cloth does not move out of the frame (similar to Lee *et al.* [96]). The distance between the pick point and the place point along this direction is uniformly sampled between $[25, 150]$ px. The distance is truncated if it exceeds a margin of 20 px from the image edge, again to prevent moving the cloth out of the frame. While these heuristics may seem to overly constrain the data we collect, we

observe that our data still contains highly diverse cloth configurations, as shown in Fig. D.1.

For each sample, we save the initial depth observation image, the dual-arm pick and place pixel locations of the action, the next depth observation resulting from the executed action, and the cloth particle positions of both observations (See Fig. D.1). The camera for capturing depth observations is fixed at 65 cm above the support surface. We mask the depth observations to only include the cloth by setting all background pixels to zero. The dataset for training both the flow and pick networks consists of 20k samples from 4k episodes, where each episode consists of five dual-arm pick and place actions.

Flow Network Training. We use FlowNet [37] as our flow network architecture. The input to FlowNet is the initial and next depth image from a sample in our dataset, stacked channel-wise. The ground truth flow for supervising FlowNet comes from the cloth particles used by the simulator to model the cloth's dynamics: we collect the cloth particle positions for each observation in our dataset and correspond them across observations to get flow vectors (See Fig. D.1). The ground truth flow is sparse because the cloth particles are sparse, so we train FlowNet using a masked loss that only includes pixels with corresponding ground truth flow. Similar to Lee *et al.* [96], we apply spatial augmentation of uniform random translation (up to 5 px) and rotation (up to 5 degrees) to augment the training data. We train the network using the dataset of 20k random actions described above. We use the Adam [85] optimizer, learning rate 1e-4, weight decay 1e-4, and batch size 8.

PickNet Network Training. PickNet1 and PickNet2 are fully-convolutional network architectures based on Lee *et al.* [96], with 4 convolutional layers in the encoder, each with 32 filters of size 5. The first three layers of the encoder have stride 2 and the last one has stride 1. The decoder consists of 2 interleaved convolutional layers and bilinear upsampling layers.

The input to the PickNet1 is a 200 $\times$ 200 flow image. PickNet2 receives the first pick point location (the argmax of the Picknet1 output, as described in the main text) as an additional input, represented as a 2D Gaussian $\mathcal{N}(p_1, \sigma)$ (where $\sigma = 5$). Similar to Nair *et al.* [136], the output of both networks is a 20 x 20 spatial grid. If the pick points predicted by PickNet are not on the cloth mask, we project them to the closest pixel on the mask using an inverse distance transform. In practice, we find

that the predictions are usually either on the cloth mask or very close to the mask. To train PickNet1 and PickNet2, we use the same dataset of 20k random actions described above. We use the Adam [85] optimizer, learning rate 1e-4, and batch size 10.

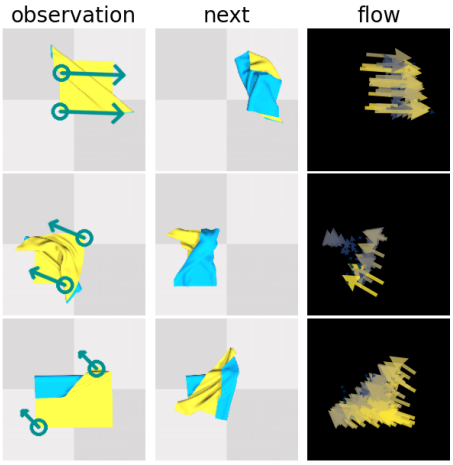


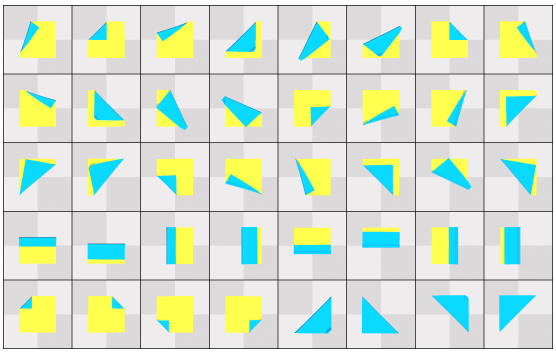
Figure D.1: Training data for FFN.

D.1.2 Additional Simulation Results for FabricFlowNet

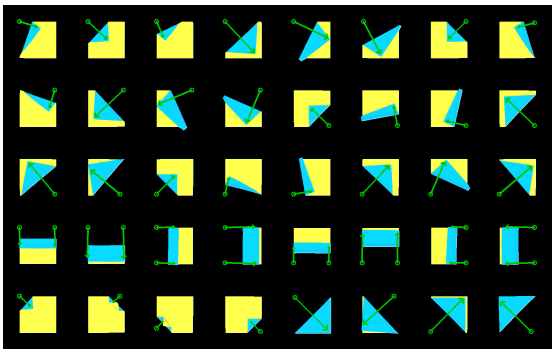
Fig. D.2b and Fig. D.2f show the cloth configurations achieved by FabricFlowNet for each of the one-step goals (Fig. D.2a) and multi-step goals (Fig. D.2e). Fig. D.1 provides examples of the data used to train FFN. Our policy is deterministic and the simulation is near-deterministic, so we only need 1 trial for our simulation experiments (unlike our real world experiments which use 3 trials).

D.1.3 Additional Real World Details and Results for FabricFlowNet

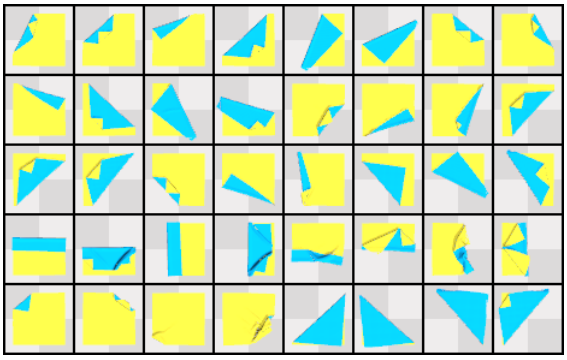
Cloth Masking. In simulation, we can obtain a perfect cloth mask. In the real world, we first obtain a background mask of the table using color-based HSV thresholding, which we can determine before the cloth is placed on the table. We then use the inverse of this background mask to obtain a mask of the cloth. Note that while we use background color of the table for cloth masking, the network itself only takes



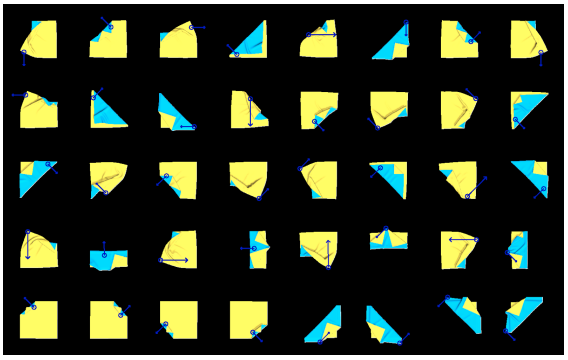
(a) One-step goals



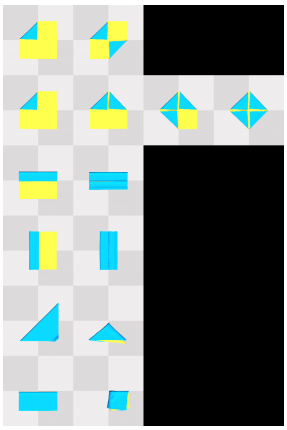
(b) One-step FFN performance



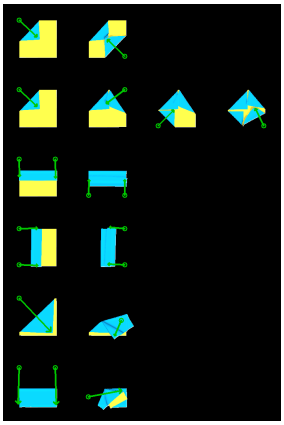
(c) One-step Fabric-VSF performance



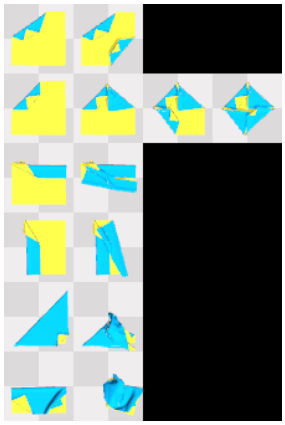
(d) One-step Lee et al. performance



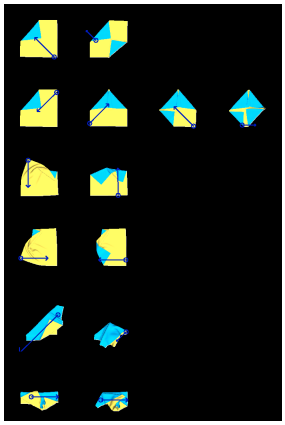
(e) Multi-step goals



(f) Multi-step FFN performance



(g) Multi-step Fabric-VSF performance



(h) Multi-step Lee et al. performance

Figure D.2: Goal configurations, achieved configurations, and training data in simulation. Arrows indicate the executed action. Fabric-VSF uses a lower camera height than FFN (45 cm vs. 65 cm), thus the cloth looks slightly larger.

depth input, allowing the network to be robust to colors and patterns on the cloth itself.

Results on Real Cloth Folding. Table D.1 provides mean IOU (mIOU) performance for NoFlow and FFN on real cloth goals. The NoFlow ablation performs considerably worse compared to FFN on real cloth folding. Qualitative results and the complete set of real square cloth goals are in Fig. D.3; the complete set of real rectangle and T-shirt goals are in the main text. We found that for FFN, using FlowNet weights from epochs at the start of convergence transferred better to the real world than using weights from epochs long after convergence.

Table D.1: mIOU for Folding Square Towel, Rectangular Cloth, and T-shirt

Method	1-Step Sq. ↑ ($n = 6$)	Multi-Step Sq. ↑ ($n = 5$)	All Sq. ↑ ($n = 11$)	Rect. ↑ ($n = 3$)	T-shirt ↑ ($n = 3$)
NoFlow	0.59 ± 0.04	0.45 ± 0.01	0.53 ± 0.02	0.65 ± 0.07	0.61 ± 0.06
FFN (Ours)	$\mathbf{0.89 \pm 0.01}$	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.80 \pm 0.03}$	$\mathbf{0.81 \pm 0.04}$	$\mathbf{0.82 \pm 0.02}$

Average of 3 rollouts. Higher mIOU scores are better; the max achievable score is 1.0.

Failure Cases. This work focused on high level actions with fixed primitives for picking and placing that may not be ideal for all cloth types, sizes, or folds. Causes of failures include the grasped portion of the cloth "flopping back" against the folding direction, undoing small folding actions or causing unwanted secondary folds (Fig. D.4a). Potential future work is to learn better pick and place primitives. Another source of failure was over- or under-estimating the fold distance due to slight inaccuracies in the flow prediction (Fig. D.4b). We also see some failures during multi-step folding; since we provide sub-goals in sequence and allow only one action per sub-goal, the discrepancy between the starting image of the demonstration and the observed image can result in poor predictions (Fig. D.4c). Allowing the policy to take multiple actions to achieve a sub-goal before proceeding may improve performance. For example, the flow can be recalculated after each action to determine if the observation is sufficiently close to the desired sub-goal configuration before proceeding to the next sub-goal.

D.2 Additional Details and Results for Fabric-VSF [67]

D.2.1 Fabric-VSF [67] Implementation Details

The original Fabric-VSF [67] paper uses single arm actions and a top-down close camera view such that the cloth covers the whole image. To match the camera view, we set the camera height to be 45 cm above the table in our case. The training dataset consists of 7115 trajectories, each with 15 random pick-and-place actions, totaling 106725 data points. Note that this dataset is 5x larger than the 20k samples we train FFN on. During training, Fabric-VSF takes as input 3 context frames and predicts the next 7 target frames.

We trained 8 variants of Fabric-VSF. Each variant differs in the following aspects: 1) whether it uses single arm or dual arms; 2) during data collection, whether the pick-and-place actions are randomly sampled, or use the corner biasing sampling strategy as described in Sec. D.1.1, and 3) whether it uses the original small action size ("Small Action", bounded to half of the cloth width) or a larger action size ("Large Action", bounded to the diagonal length of the cloth). Other than these three changes, we set all other parameters to be the same as in the original paper. Therefore, the variant with single arm actions, no corner biasing during data collection, and small action size is exactly how Fabric-VSF is trained in the original paper.

After the training, we plan with cross-entropy method (CEM) to find actions for achieving a given goal image. We use the exact same CEM parameters as in the original paper, *i.e.,* we run CEM for 10 iterations, each with a population size of 2000 and elite size of 400.

D.2.2 Additional Fabric-VSF [67] Results

The results for the Fabric-VSF variants are summarized in Table D.2. We note that the variant using single arm actions, corner biasing for data collection, and large action size performs the best out of all variants. This variant outperforms FFN on overall error and one-step error, but performs slightly worse than FFN on multi-step error (See Fig. D.2c and Fig. D.2g for qualitative results). However, we note that

Fabric-VSF was trained on 5x more data than FFN. Additionally, Fabric-VSF takes much longer to run at inference time, requiring ∼7 minutes of CEM iterations to compute a single action compared to ∼0.007 seconds for a forward pass through FFN. 7 minutes of CEM planning time is impractical for real-world folding. We also demonstrate in the following section that FFN generalizes to other cloth shapes better than Fabric-VSF.

Analyzing the performance between different Fabric-VSF variants, for single-arm actions, using large actions instead of small actions always leads to better performance. However, this is not true for the dual arm variants. Interestingly, we find that using dual arms tends to result in worse performance compared with using a single arm. The reason for this could be that during CEM planning, dual-arm variants double the action dimension, which increases complexity for CEM and makes it difficult to find optimal actions.

Table D.2: Mean Particle Distance Error (mm) and Inference Time (sec) for Fabric-VSF Variants

Baseline	1-Step (n=40)	Multi-Step (n=6)	All (n=46)	Inf. Time
1-Arm, No CB, Sm. Action	12.92 ± 13.00	46.05 ± 48.07	17.24 ± 23.93	∼420s
1-Arm, No CB, Lg. Action	10.13 ± 07.33	33.06 ± 12.46	13.12 ± 11.25	∼420s
1-Arm, CB, Sm. Action	14.09 ± 11.36	38.68 ± 27.72	17.30 ± 16.76	∼420s
1-Arm, CB, Lg. Action	6.30 ± 06.55	$\mathbf{21.33 \pm 11.20}$	8.27 ± 08.90	∼420s
2-Arm, No CB, Sm. Action	24.60 ± 14.69	50.26 ± 27.54	27.94 ± 19.00	∼420s
2-Arm, No CB, Lg. Action	10.98 ± 05.80	40.92 ± 18.06	14.89 ± 13.17	∼420s
2-Arm, CB, Sm. Action	16.21 ± 13.81	36.42 ± 26.51	18.84 ± 17.43	∼420s
2-Arm, CB, Lg. Action	15.58 ± 10.88	54.06 ± 26.68	20.60 ± 19.07	∼420s
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$	**∼0.007s**

CB: Corner Bias Sm. Action: Small Action Lg. Action: Large Action

D.3 Additional Details and Results for Lee *et al.* [96]

D.3.1 Lee *et al.* [96] Implementation Details

Lee *et al.* [96] learns a fabric folding policy for a discrete action space using a fully convolutional state-action value function, or Q-network. Observation and goal images are stacked channel-wise, then duplicated and transformed to form a batch of m image rotations and n scales to represent different pick and place directions and action lengths. The whole batch is input to the Q-network to compute the Q-value of executing an action for each rotation and scale at every point on the image. The action corresponding to the max Q-value from the outputs is executed. The discrete action space of m rotations and n action lengths for Lee *et al.* [96] enables efficient policy learning, but greatly limits the actions of the learned policy compared to FFN.

We extend Lee *et al.* [96] from a single-arm approach to a dual-arm one. To represent two pickers instead of one, we input two pairs of observation and goal images to the Q-network. When rotating and scaling the images to represent different actions, the images are constrained to have the same rotation, but are allowed to be scaled differently. In other words, the dual-arm actions are constrained to execute pick and place actions in the same direction, but can have different pick and place lengths. The Q-network outputs a pair (one for each arm) of Q-value heatmaps for every action in the discrete action space (*i.e.,* every rotation and scale). The max Q-value in each of the two heatmaps is averaged, and the heatmap pair with the highest averaged Q-value is selected from the set of all discrete rotations and scales. The picker action corresponding to the argmax of each heatmap is executed.

We train each Lee *et al.* variant below using hyperparameters similar to the original paper [96], training for 25k steps with learning rate 1e-4, batch size 10, and evaluating performance on test goals every 500 steps to find the best performing step.

D.3.2 Additional Lee *et al.* [96] Results

We trained variants of Lee *et al.* to compare single-arm vs. dual-arm performance, depth input vs. RGB input, collecting data with corner bias similar to FFN vs.

without bias, and using the original close-up image of the cloth ("Low Cam") vs. images from further away ("High Cam"). All variants were trained with 20k training examples. We also provide results for two variants of FFN trained on the same amount of data, one where actions are sampled from the discrete action space (*i.e.,* discretized action angles and lengths) in Lee *et al.* [96] ("Discrete Actions"), and the other where actions are sampled using our continuous action space described in Sec. D.1.1 ("Cont. Actions"). Lee *et al.* [96] is an inherently discrete approach and cannot be trained to output continuous actions, nor can it be trained on data with actions outside of its discrete action space.

Table D.3 shows that the performance of all Lee *et al.* variants is poor compared to FFN, particularly on 1-step goals (see Appendix Fig. D.2d and Appendix Fig. D.2h for qualitative results). FFN outperforms Lee *et al.* when trained on either the discrete action dataset or the continuous one. Training FFN on continuous actions results in better performance for 1-step goals, but the discrete action dataset also performs fairly well. These results indicate that the improved performance of FFN vs. Lee *et al.* cannot be solely explained by training on continuous vs. discrete action data, though other factors like outputting continuous actions instead of discrete ones may still play significant role in FFN's improved performance.

Table D.3: Mean Particle Distance Error for Lee *et al.* on 20k Training Examples

Baseline	1-Step (40)	Multi Step (6)	All (46)
Lee *et al.*, 1-Arm, D, No CB, LC	18.94 ± 16.43	24.18 ± 17.75	19.62 ± 16.49
Lee *et al.*, 1-Arm, D, No CB, HC	16.18 ± 08.38	26.20 ± 16.31	17.49 ± 10.10
Lee *et al.*, 1-Arm, D, CB, LC	20.99 ± 18.88	34.61 ± 31.35	22.77 ± 20.97
Lee *et al.*, 1-Arm, D, CB, HC	19.70 ± 09.37	38.91 ± 24.05	22.20 ± 13.53
Lee *et al.*, 1-Arm, RGB, No CB, LC	49.29 ± 18.10	52.03 ± 33.62	49.65 ± 20.26
Lee *et al.*, 1-Arm, RGB, No CB, HC	47.12 ± 21.04	64.48 ± 29.85	49.38 ± 22.75
Lee *et al.*, 1-Arm, RGB, CB, LC	33.89 ± 19.01	58.90 ± 43.34	37.15 ± 24.38
Lee *et al.*, 1-Arm, RGB, CB, HC	39.01 ± 25.36	55.46 ± 38.38	41.15 ± 27.43
Lee *et al.*, 2-Arm, D, No CB, LC	36.62 ± 14.51	47.72 ± 21.95	38.07 ± 15.82
Lee *et al.*, 2-Arm, D, No CB, HC	40.75 ± 13.22	52.88 ± 19.03	42.33 ± 14.45
Lee *et al.*, 2-Arm, D, CB, LC	47.18 ± 18.60	57.29 ± 28.65	48.50 ± 20.07
Lee *et al.*, 2-Arm, D, CB, HC	35.98 ± 24.60	64.75 ± 51.76	39.73 ± 30.30
FFN, 2-Arm, D, CB, HC, Discrete Actions	9.57 ± 06.07	$\mathbf{10.15 \pm 07.20}$	10.17 ± 07.34
FFN, 2-Arm, D, CB, HC, Cont. (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

D: Depth CB: Corner Bias LC: Low Camera HC: High Camera Cont: Continuous Actions

Lee *et al.* with and without Subgoals. FFN uses subgoals at inference time in order to fully specify the task; many cloth folding goals have final goal configurations in which large portions of the cloth are self-occluded. Subgoals are required to ensure the task is completed correctly and that the cloth is correctly folded. Lee *et al.* [96] demonstrated cloth folding without subgoals at inference time by relying on a learned Q-value heatmap to select actions toward a final end goal. We compare the performance of the best Lee *et al.* variant with and without subgoals at test-time. The results of this experiment are in Table D.4. While the performance on 1-step goals are similar because those tasks do not have subgoals, performance on multi-step goals is worse without subgoals.

Table D.4: Mean Particle Distance Error for Lee *et al.* With and Without Subgoals

Method	1-Step (40)	Multi Step (6)	All (46)
Lee *et al.*	16.92 ± 9.28	37.74 ± 38.99	19.71 ± 20.27
Lee *et al.*, With Subgoals	$\mathbf{16.18 \pm 8.38}$	$\mathbf{26.20 \pm 16.31}$	$\mathbf{17.49 \pm 10.10}$

D.4 Additional Details and Results for Ablations

D.4.1 Ablation Implementation Details

NoFlowIn The architecture for this ablation is identical to our main method, except that it takes depth images instead of flow images as input. We use a conditioned architecture with two PickNets; PickNet1 receives the observation and goal depth images as input both of size 200×200. The place point is computed by querying the flow image similar to our main method.

NoFlowPlace We predict the place points similarly to the pick points by using an additional place network. The place network architecture is identical to PickNet. The input is a flow image and the output is the place point predictions.

NoFlow This ablation is a combination of NoFlowIn and NoFlowPlace, where PickNet and PlaceNet both take observation and goal depth images as input.

NoCornerBias This ablation is the same as our main method except for the training dataset. We use a dataset that does not bias the data to pick corners (See

Sec. D.1.1). Instead, the pick actions are always uniformly sampled over the visible cloth mask. We still constrain the folding actions for both arms to be in the same direction and distance from their respective pick points and point towards the center of the frame.

NoSplitPickNet The architecture of PickNet is modified so that we only have one PickNet for both arms instead of the conditioned architecture used in our main method. The PickNet takes as input the flow image and outputs two heatmaps corresponding to the two pick points.

NoMinLoss The loss in Eq. 1 is replaced with the following:

$$\mathcal{L}_{\text{NoMin}} = \text{BCE}(H_1, H_1^*) + \text{BCE}(H_2, H_2^*) \tag{D.1}$$

D.4.2 Additional Ablation Results

We provide ablation results in Table D.5 grouped by single-step, multi-step, and all goals.

Table D.5: Mean Particle Distance Error for Ablations

Ablation	One Step (n=40)	Multi Step (n=6)	All (n=46)
NoFlowIn	5.14 ± 3.62	24.63 ± 21.30	9.37 ± 12.20
NoFlowPlace	7.61 ± 5.44	30.25 ± 17.62	10.56 ± 11.15
NoFlow	8.97 ± 7.45	28.79 ± 19.33	18.02 ± 20.34
NoCornerBias	9.79 ± 5.57	$\mathbf{19.61 \pm 17.52}$	11.07 ± 8.83
NoSplitPickNet	4.87 ± 2.61	23.41 ± 18.87	7.29 ± 9.56
NoMinLoss	5.10 ± 4.04	20.81 ± 17.57	7.15 ± 9.08
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

D.5 Additional Results on Unseen Cloth Shapes

We also evaluate Fabric-VSF and Lee *et al.* on generalization to unseen cloth shapes. FFN generalizes well to new shapes, as shown in the main text (see Fig. 5 and Sec. 4.2.1). Table D.6 provides quantitative results on the rectangle cloth and T-shirt for the best Fabric-VSF method and best Lee *et al.* method compared to FFN. FFN

outperforms both methods by a large margin. Fabric-VSF generalizes poorly, likely because it relies on planning with a learned visual dynamics model. Lee *et al.* also does not generalize well compared to FFN. Fig. D.5 provides a qualitative comparison.

Table D.6: Mean Particle Distance for Folding Unseen Cloth Shapes in Simulation

Method	Rectangle (n=6)	T-Shirt (n=3)
Lee *et al.* , 1-Arm, No Corner Bias, High Cam, 20k Actions	31.63 ± 18.04	86.65 ± 34.67
Fabric-VSF, 1-Arm, Corner Bias, Large Action	25.68 ± 11.21	45.25 ± 13.83
FFN (Ours)	$\mathbf{10.70 \pm 08.54}$	$\mathbf{20.91 \pm 11.28}$

D.6 End-to-End Variants of FFN

We investigate the effect of training our FFN architecture end-to-end. First, we train the FFN architecture with pick losses as well as the flow loss; all losses are allowed to backpropagate through the entire combined network, including through the FlowNet layers. The results on the square towel are presented in Table D.7 ("JointFFN"). This variant performs significantly worse than FFN (9.28 vs. 7.14 on all goals).

Table D.7: Mean Particle Distance Error (mm) for End-to-End Variants of FFN

Method	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
JointFFN	07.60 ± 05.62	$\mathbf{17.53 \pm 15.56}$	09.28 ± 09.39
JointPredictPlace	12.90 ± 11.67	35.25 ± 19.22	22.88 ± 23.24
JointFFN, No Flow Loss	32.41 ± 22.61	68.17 ± 50.35	37.07 ± 30.34
JointPredictPlace, No Flow Loss	16.31 ± 22.73	50.27 ± 31.44	24.39 ± 29.77
FFN (Ours)	$\mathbf{4.46 \pm 02.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

We also trained another variant which consists of a FlowNet, a PickNet, and a PlaceNet, trained end-to-end ("JointPredictPlace" in Table D.7). This is similar to our ablation "PredictPlace" in Table 6.2, which uses the same architecture but is not trained end-to-end. JointPredictPlace performs significantly worse than FFN (22.88 vs. 7.14 on all goals) and also underperforms compared to PredictPlace (10.56 on all goals). Overall, this result, as well as the one in the paragraph above, indicate that end-to-end training leads to significantly worse performance for this task. Our

intuition for this is that the flow network should be trained only with the flow loss, and that backpropagating the gradients from the pick loss into the flow network adds noise and reduces its performance.

Lastly, we evaluated variants of the above two architectures with the flow loss removed, to see if we could train these architectures end-to-end with just a single loss at the end, instead of using an intermediate flow loss. The results, shown in Table D.7, are worse for both variants, showing the importance of the intermediate flow loss.

D.7 FFN Performance with Crumpled Starting Configurations

Our experiments focused on folding tasks, and we assume that a previous method was used to flatten the cloth before our method is executed. To evaluate the robustness of our method to imperfect smoothing, we evaluate the performance of FFN in simulation on slightly crumpled initial cloth configurations. We generated crumpled configurations by taking the flat cloth and executing a random pick and place action with a maximum translation of 10 pixels. The three configurations used in our experiments are shown in Fig. D.6.

For each crumpled configuration, we evaluated FFN on the full set of 46 evaluation goals, where the starting configuration of the cloth was set to the given crumpled configuration. The results of these evaluations are in Table D.8. The particle distance error is slightly higher with the crumpled starting configurations, but the qualitative results in Fig. D.7 show that FFN still produces actions that are very close to the intended goals.

Table D.8: Mean Particle Distance Error (mm) for FFN with Different Start Configurations

Starting Config	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
FFN, Crumpled 0	12.40 ± 4.82	24.82 ± 24.81	14.01 ± 10.86
FFN, Crumpled 1	10.68 ± 2.89	23.54 ± 22.56	12.36 ± 9.61
FFN, Crumpled 2	10.68 ± 4.29	$\mathbf{21.05 \pm 14.70}$	12.03 ± 7.51
FFN, Flat	$\mathbf{4.46 \pm 2.62}$	25.04 ± 22.88	$\mathbf{7.14 \pm 11.06}$

D.8 FFN Performance with Iterative Refinement

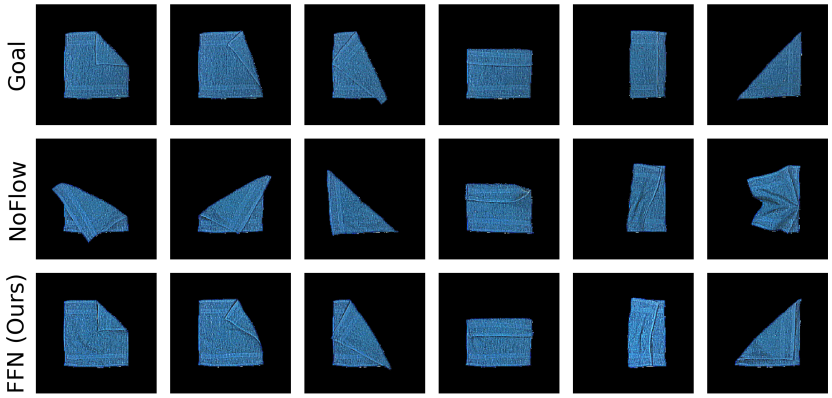
Table D.9: Mean Particle Distance Error (mm) for FFN with Iterative Refinement

Starting Config	1-Step (n=40)	Multi-Step (n=6)	All (n=46)
FFN, No Refinement	$\mathbf{4.46 \pm 2.62}$	25.04 ± 22.88	7.14 ± 11.06
FFN, Iterative Refinement	4.54 ± 2.58	$\mathbf{20.47 \pm 19.49}$	$\mathbf{6.62 \pm 9.17}$

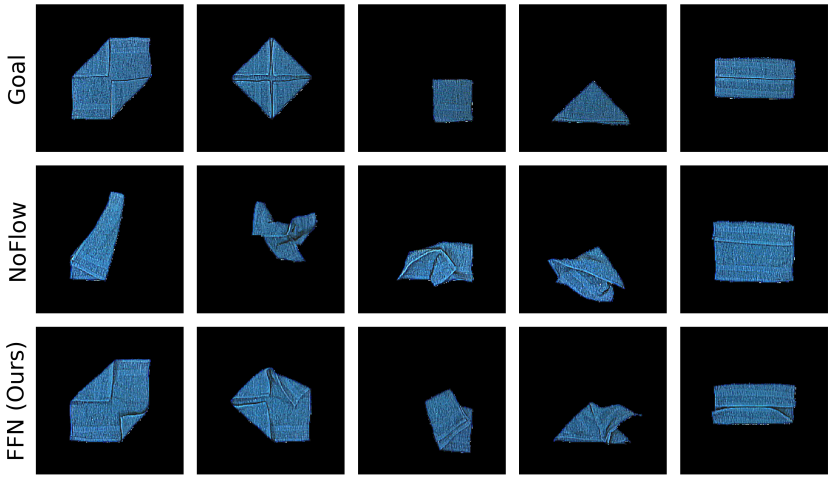
In our normal evaluations, each goal or subgoal is attempted only once by each method. With a single attempted action for each subgoal, FFN is able to achieve a diverse set of goals, as demonstrated in this work. However, we find that FFN can achieve even better performance when attempting goals multiple times, using the flow to compare the current observation with the goal and taking actions that move the observation closer to the goal if it has not yet been reached. We evaluate the benefit of using this "iterative refinement" procedure in simulation. FFN moves to the next subgoal when a minimum threshold for the average flow is achieved, so the flow acts as a goal recognizer. The policy is allowed a maximum of 3 iterative actions per subgoal to limit potential divergence. The results in Table D.9 show that iterative refinement can improve performance, particularly on multi-step goals, where reaching the current subgoal accurately is important for achieving subsequent goals.

D.9 FlowNet Performance

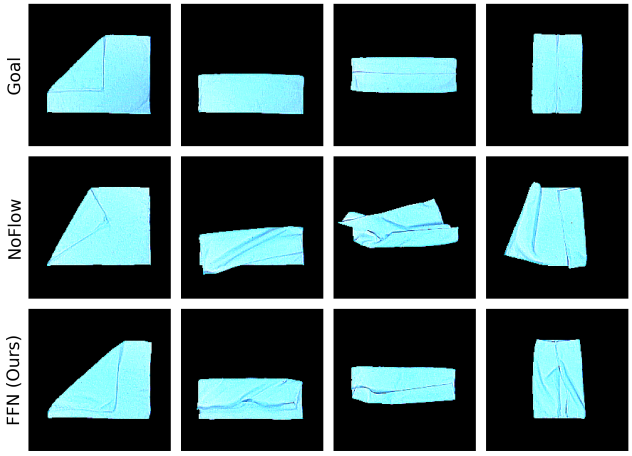
FlowNet achieves an average endpoint error (EPE) of 1.0268 on the set of simulated test goals. The test goals are not seen during training. Fig. D.8 provides qualitative examples of FlowNet performance on simulated test goals.



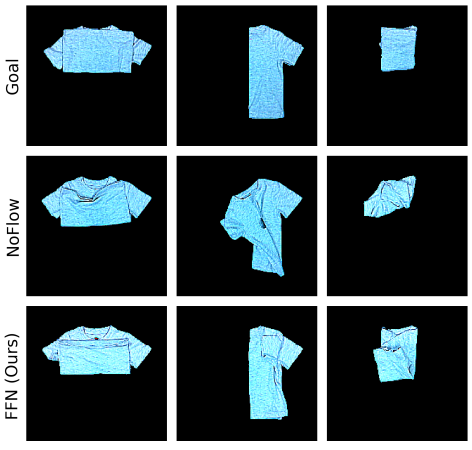
Single-step Square Towel



Multi-step Square Towel

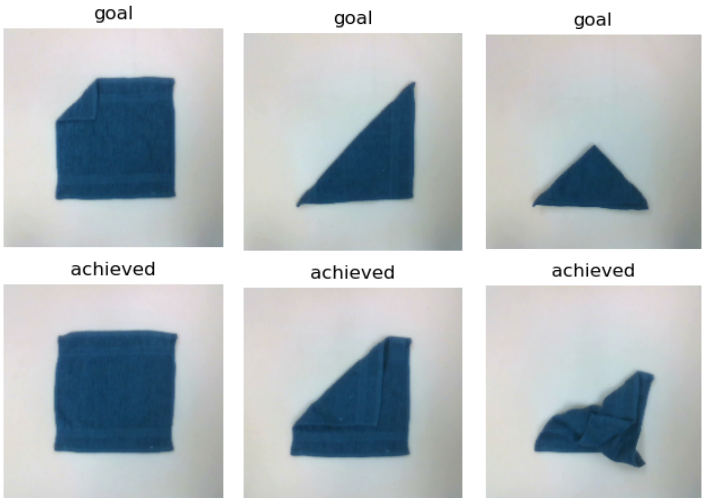


Rectangular Cloth



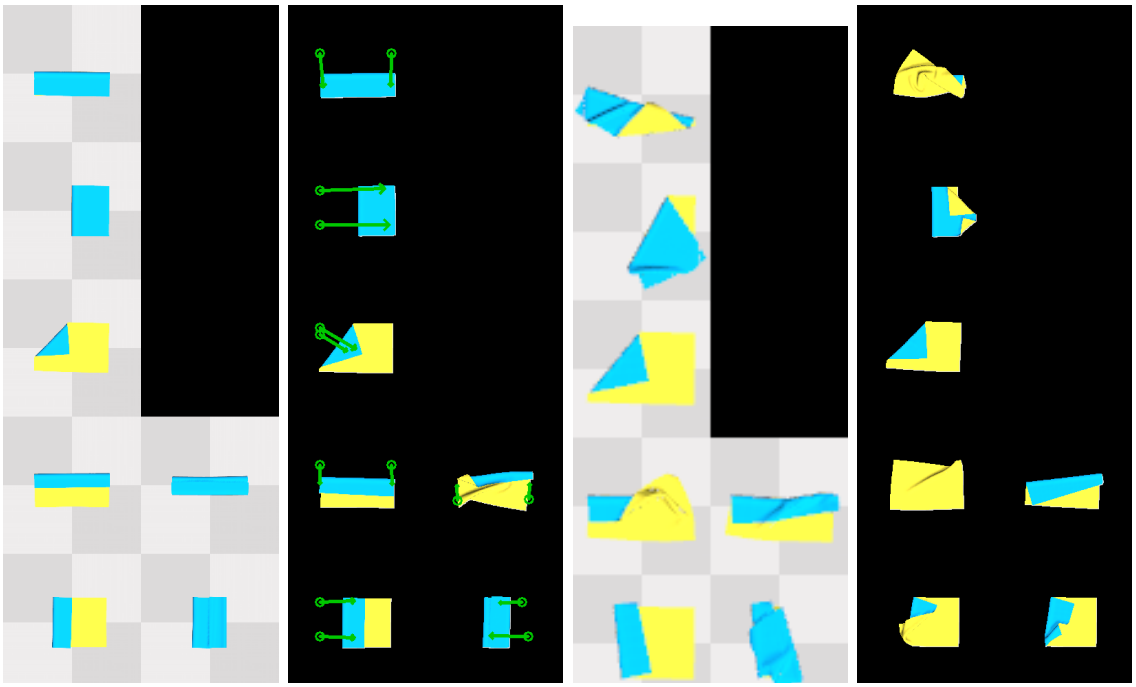
T-shirt

Figure D.3: Qualitative performance of FFN and NoFlow on real cloth. The trial corresponding to the best achieved IOU is shown for each example. For multi-step goals, only the final goal is shown. FFN only takes depth images as input, allowing it to easily transfer to cloths of different colors. Contrast and brightness have been adjusted to enhance visibility.



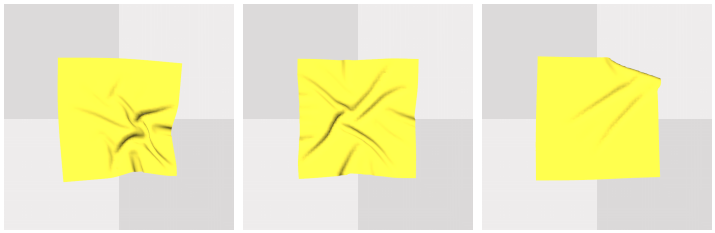
(a) Flopping back (b) Undershooting (c) Poor prediction

Figure D.4: Examples of failure cases



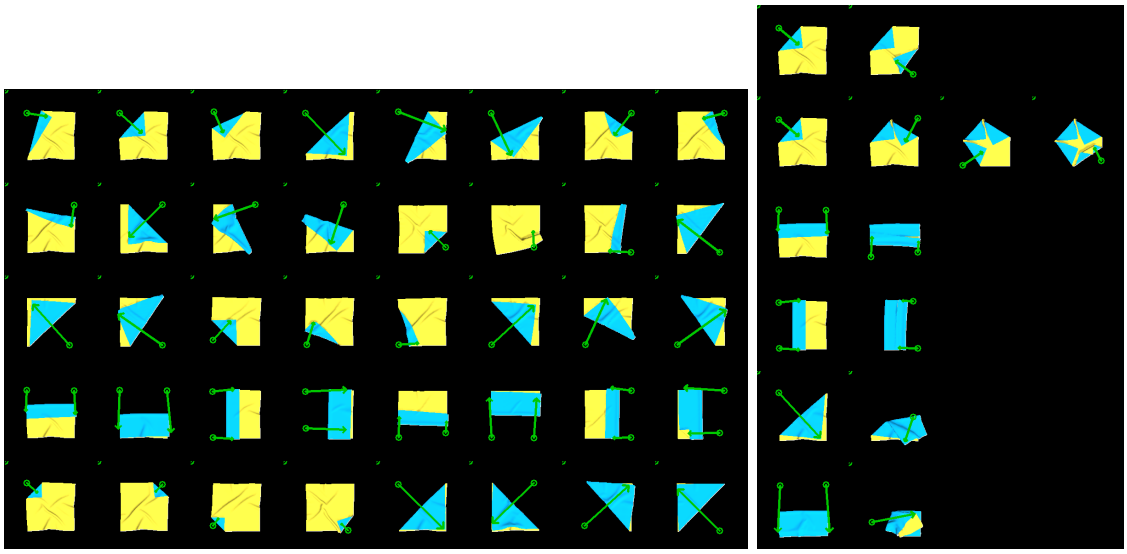
(a) Rect. cloth goals (b) FFN achieved (c) Fabric-VSF achieved (d) Lee *et al.* achieved

Figure D.5: Qualitative performance of FFN, Fabric-VSF, and Lee *et al.* on rectangular cloth.



(a) Crumpled 0 (b) Crumpled 1 (c) Crumpled 2

Figure D.6: Crumpled initial cloth configurations



(a) Crumpled one-step FFN performance (b) Crumpled Multi-step FFN performance

Figure D.7: Configurations achieved by FFN when starting from the "Crumpled 1" configuration for each attempt (compare with Fig. D.2)

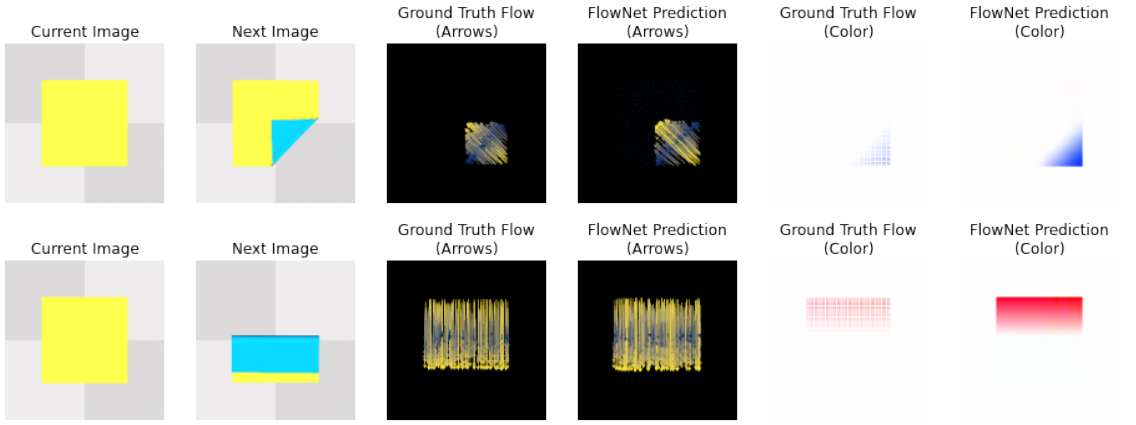


Figure D.8: FlowNet Qualitative Performance. Two types of visualizations are provided: representing the flow vector as arrows, and representing the flow vector using RGB channels. FlowNet outputs a dense flow image but is trained on sparse ground truth flow. FlowNet takes only depth images as input; RGB images are shown as a visual aid only.

Bibliography

[1] Faraj Alhwarin, Alexander Ferrein, and Ingrid Scholl. Ir stereo kinect: improving depth images by combining structured light with ir stereo. In *Pacific Rim International Conference on Artificial Intelligence*, pages 409–421. Springer, 2014. 14

[2] Artemij Amiranashvili, Alexey Dosovitskiy, Vladlen Koltun, and Thomas Brox. Motion perception in reinforcement learning with dynamic objects. In *Conference on Robot Learning*, pages 156–168. PMLR, 2018. 99

[3] Max Argus, Lukas Hermann, Jon Long, and Thomas Brox. Flowcontrol: Optical flow based visual servoing. *arXiv preprint arXiv:2007.00291*, 2020. 99

[4] Dmitry Berenson, Siddhartha Srinivasa, and James Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *The International Journal of Robotics Research*, 30(12):1435–1460, 2011. xvi, 35, 36

[5] C. Bersch, B. Pitzer, and S. Kammel. Bimanual robotic cloth manipulation for laundry folding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1413–1419, Sep. 2011. 54

[6] Christian Bersch, Benjamin Pitzer, and Sören Kammel. Bimanual robotic cloth manipulation for laundry folding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1413–1419, 2011. doi: 10.1109/IROS.2011.6095109. 96, 98

[7] Tapomayukh Bhattacharjee, Advait Jain, Sarvagya Vaish, Marc D. Killpack, and Charles C. Kemp. Tactile Sensing over Articulated Joints With Stretchable Sensors. In *World Haptics Conference (WHC)*, 2013. 80

[8] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. ReSkin: versatile, replaceable, lasting tactile skins. In *Conference on Robot Learning (CoRL)*, 2021. xviii, 76, 77, 80, 81

[9] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2013. 1, 14

[10] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014. doi: 10.1109/TRO.2013.2289018. 36

[11] Ricard Bordalba, Lluís Ros, and Josep M Porta. A randomized kinodynamic planner for closed-chain robotic systems. *IEEE Transactions on Robotics*, 2020. 97

[12] Julia Borras, Guillem Alenya, and Carme Torras. A Grasping-centered Analysis for Cloth Manipulation. *arXiv preprint arXiv:1906.08202*, 2019. 76, 78

[13] Júlia Borràs, Guillem Alenyà, and Carme Torras. A grasping-centered analysis for cloth manipulation. *IEEE Transactions on Robotics*, 36(3):924–936, 2020. 98

[14] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, pages 536–551. Springer, 2014. 1

[15] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015. 22

[16] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 65, 66

[17] Cheng Chi and Shuran Song. Garmentnets: Category-level pose estimation for garments via canonical space shape completion. *arXiv preprint arXiv:2104.05177*, 2021. 98

[18] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2020. 34, 37, 38

[19] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020. 34, 37, 38

[20] Rohan Chitnis, Shubham Tulsiani, Saurabh Gupta, and Abhinav Gupta. Efficient bimanual manipulation using learned task schemas. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1149–1155. IEEE, 2020. 97

[21] Rohan Chitnis, Shubham Tulsiani, Saurabh Gupta, and Abhinav Gupta. Intrin-

sic motivation for encouraging synergistic behavior. *International Conference on Learning Representations*, 2020. 97

[22] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012. 108

[23] Walon Wei-Chen Chiu, Ulf Blanke, and Mario Fritz. Improving the kinect by cross-modal stereo. Citeseer. 14

[24] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021. 41, 43

[25] Brian Curless and Marc Levoy. Better optical triangulation through spacetime analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 987–994. IEEE, 1995. 14

[26] Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James F. O'Brien, and Pieter Abbeel. Bringing clothing into desired configurations with limited perception. In *2011 IEEE International Conference on Robotics and Automation*, pages 3893–3900, 2011. doi: 10.1109/ICRA.2011.5980327. 98

[27] Marco Cusumano-Towner, Arjun Singh, Stephen Miller, James F O'Brien, and Pieter Abbeel. Bringing Clothing Into Desired Configurations with Limited Perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011. 78

[28] Satonori Demura, Kazuki Sano, Wataru Nakajima, Kotaro Nagahama, Keisuke Takeshita, and Kimitoshi Yamazaki. Picking up one of the folded and stacked towels by a single arm robot. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1551–1556. IEEE, 2018. 54, 55

[29] Satonori Demura, Kazuki Sano, Wataru Nakajima, Kotaro Nagahama, Keisuke Takeshita, and Kimitoshi Yamazaki. Picking up One of the Folded and Stacked Towels by a Single Arm Robot. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018. 79

[30] Simone Denei, Perla Maiolino, Emanuele Baglini, and Giorgio Cannata. On the Development of a Tactile Sensor for Fabric Manipulation and Classification for Industrial Applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 79

[31] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021. 41

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet:

A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 16

[33] Amaury Depierre, Emmanuel Dellandréa, and Liming Chen. Jacquard: A large scale dataset for robotic grasp detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3511–3516. IEEE, 2018. 15

[34] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5882–5889. IEEE, 2018. 3

[35] Siyuan Dong, Devesh K Jha, Diego Romeres, Sangwoon Kim, Daniel Nikovski, and Alberto Rodriguez. Tactile-rl for insertion: Generalization to objects of unknown geometry. *arXiv preprint arXiv:2104.01167*, 2021. 99

[36] 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 *IEEE Robotics and Automation Letters (RA-L)*, 2018. 79

[37] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 99, 126

[38] Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, and Sotiris Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 987–993, 2014. 54

[39] Andreas Doumanoglou, Andreas Kargakos, Tae-Kyun Kim, and Sotiris Malassiotis. Autonomous Active Recognition and Unfolding of Clothes Using Random Decision Forests and Probabilistic Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 76, 78

[40] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimír Petrík, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. Folding clothes autonomously: A complete pipeline. *IEEE Transactions on Robotics*, 32(6):1461–1478, 2016. doi: 10.1109/TRO.2016.2602376. 96, 98

[41] Anca D. Dragan, Nathan D. Ratliff, and Siddhartha S. Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 4582–4588,

2011. doi: 10.1109/ICRA.2011.5980538. 36, 40, 45

[42] Danny Driess, Jung-Su Ha, Marc Toussaint, and Russ Tedrake. Learning models as functionals of signed-distance fields for manipulation planning. In *Conference on Robot Learning*, pages 245–255. PMLR, 2022. 37

[43] Aaron Edsinger and Charles C Kemp. Two arms are better than one: A behavior based control system for assistive bimanual manipulation. In *Recent progress in robotics: Viable robotic service to human*, pages 345–355. Springer, 2007. 97

[44] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 41, 42, 44, 46, 48

[45] Zackory Erickson, Henry Clever, Greg Turk, C. Karen Liu, and Charles Kemp. Deep Haptic Model Predictive Control for Robot-Assisted Dressing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 76

[46] Zackory Erickson, Maggie Collier, Ariel Kapusta, and Charles Kemp. Tracking Human Pose During Robot-Assisted Dressing using Single-Axis Capacitive Proximity Sensing. In *IEEE Robotics and Automation Letters (RA-L)*, 2018. 76

[47] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp. Assistive Gym: A Physics Simulation Framework for Assistive Robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 76

[48] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 119

[49] Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 11444–11453, 2020. 35, 36

[50] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kuryenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *Robotics: Science and Systems (RSS)*, 2018. 1

[51] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022. 37, 38

[52] Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5

(2):492–499, 2019. 2

[53] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph E Gonzalez, Nawid Jamali, et al. Learning dense visual correspondences in simulation to smooth and fold real fabrics. *arXiv preprint arXiv:2003.12698*, 2020. 97, 98, 106

[54] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph Gonzalez, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 76, 78, 79

[55] Irene Garcia-Camacho, Martina Lippi, Michael C Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, Carme Torras, Alessandro Marino, Guillem Alenyà, et al. Benchmarking bimanual cloth manipulation. *IEEE Robotics and Automation Letters*, 5(2):1111–1118, 2020. 64

[56] Irene Garcia-Camacho, Martina Lippi, Michael C. Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, Carme Torras, Alessandro Marino, Guillem Alenyà, and Danica Kragic. Benchmarking bimanual cloth manipulation. *IEEE Robotics and Automation Letters*, 5(2):1111–1118, 2020. doi: 10.1109/LRA.2020.2965891. 98

[57] Marcus Gualtieri, Andreas Ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2016. 12, 15

[58] Yuhao Guo, Xin Jiang, and Yunhui Liu. Deformation Control of a Deformable Object Based on Visual and Tactile Feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021. 79

[59] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2827–2836, 2016. 16, 17

[60] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. *arXiv preprint arXiv:2105.03655*, 2021. 78, 98, 105

[61] Kyoko Hamajima and Masayoshi Kakikura. Planning strategy for task of unfolding clothes. *Robotics Auton. Syst.*, 32:145–152, 1997. 54

[62] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. 125

[63] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988. 67

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 87

[65] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 19–34, 2018. 1

[66] Judy Hoffman, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell. Cross-modal adaptation for rgb-d detection. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5032–5039. IEEE, 2016. 12, 16, 17

[67] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.034. xiv, 96, 97, 99, 106, 130

[68] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Robotics: Science and Systems (RSS)*, 2020. 78

[69] Jeffrey Ichnowski*, Yahav Avigal*, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *Conference on Robot Learning (CoRL)*, 2020. 37, 45

[70] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. Gomp: Grasp-optimized motion planning for bin picking. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 5270–5277. IEEE, 2020. 36

[71] Ivo Ihrke, Kiriakos N Kutulakos, Hendrik PA Lensch, Marcus Magnor, and Wolfgang Heidrich. Transparent and specular object reconstruction. In *Computer Graphics Forum*, volume 29, pages 2400–2426. Wiley Online Library, 2010. 12, 13

[72] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 99

[73] Rishabh Jangir, Guillem Alenyà, and Carme Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636. IEEE, 2020. 98

[74] Biao Jia, Zhe Hu, Jia Pan, and Dinesh Manocha. Manipulating Highly Deformable Materials Using a Visual Feedback Dictionary. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 79

[75] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311. IEEE, 2011. 15

[76] Zhenyu Jiang, Yifeng Zhu, Maxwell Svetlik, Kuan Fang, and Yuke Zhu. Synergies between affordance and geometry: 6-dof grasp detection via implicit representations. 2021. 37

[77] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019. 1

[78] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 22

[79] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018. 22

[80] Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael J Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps. In *2020 International Conference on 3D Vision (3DV)*, pages 333–344. IEEE, 2020. 37

[81] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439. 35

[82] A. A. Khan, M. Hassan Tanveer, Tahir Rasheed, and Abdul Azees Ajmal. Tactile Discrimination of Fabrics Using Machine Learning Techniques. In *IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS)*, 2017. 79

[83] A.A. Khan, M. Khosravi, S. Denei, P. Maiolino, W. Kasprzak, F. Mastrogiovanni, and G. Cannata. A Tactile-based Fabric Learning and Classification

Architecture. In *IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, 2016. 79

[84] Ninad Khargonkar, Neil Song, Zesheng Xu, Balakrishnan Prabhakaran, and Yu Xiang. Neuralgrasps: Learning implicit representations for grasps of multiple robotic hands. *Conference on Robot Learning*, 2022. 37

[85] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 126, 127

[86] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 42, 117, 118

[87] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. Clothes State Recognition Using 3D Observed Data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009. 78

[88] Yasuyo Kita, Toshio Ueshiba, Ee Sian Neo, and Nobuyuki Kita. A Method For Handling a Specific Part of Clothing by Dual Arms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009. 78

[89] Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. A survey on learning-based robotic grasping. *Current Robotics Reports*, 1(4): 239–249, 2020. 36

[90] Panagiotis N. Koustoumpardis, Kostas X. Nastos, and Nikos A. Aspragathos. Underactuated 3-finger Robotic Gripper for Grasping Fabrics. In *International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2014. 79

[91] J.J. Kuffner and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2, 2000. doi: 10.1109/ROBOT.2000.844730. 35

[92] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32, 2019. 1

[93] Mike Lambeta, Po-Wei Chou, Stephen Tian, Brian Yang, Benjamin Maloon, Victoria Rose Most, Dave Stroud, Raymond Santos, Ahmad Byagowi, Gregg Kammerer, Dinesh Jayaraman, and Roberto Calandra. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor with Application to In-Hand Manipulation. In *IEEE Robotics and Automation Letters (RA-L)*, 2020. 76, 79

[94] Michael Laskey, Chris Powers, Ruta Joshi, Arshan Poursohi, and Kenneth Y.

Goldberg. Learning robust bed making using deep imitation learning with dart. *ArXiv*, abs/1711.02525, 2017. 54

[95] Robert Lee, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience. In *Conference on Robot Learning (CoRL)*, 2020. 76, 78

[96] Robert Lee, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. Learning arbitrary-goal fabric folding with one hour of real robot experience. *Conference on Robot Learning*, 2020. xiv, 96, 97, 99, 100, 101, 106, 107, 108, 125, 126, 132, 133, 134

[97] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015. doi: 10.1177/0278364914549607. URL https://doi.org/10.1177/0278364914549607. 15

[98] 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. 2

[99] 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. 15

[100] Guanbin Li, Yukang Gan, Hejun Wu, Nong Xiao, and Liang Lin. Cross-modal attentional context learning for rgb-d object detection. *IEEE Transactions on Image Processing*, 28(4):1591–1601, 2018. 16, 17

[101] Yinxiao Li, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K. Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1046–1052, 2014. doi: 10.1109/IROS.2014.6942687. 98

[102] Yinxiao Li, Danfei Xu, Yonghao Yue, Yan Wang, Shih-Fu Chang, Eitan Grinspun, and Peter K. Allen. Regrasping and unfolding of garments using predictive thin shell modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 98, 99

[103] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 37

[104] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Görner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3629–3635. IEEE, 2019. 35, 36

[105] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation. In *Conference on Robot Learning (CoRL)*, 2020. 2, 78, 83, 105

[106] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2021. 78

[107] Martina Lippi, Petra Poklukar, Michael C Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626. IEEE, 2020. 99

[108] Shan Luo, Wenzhen Yuan, Edward Adelson, Anthony G Cohn, and Raul Fuentes. Vitac: Feature Sharing Between Vision and Tactile Sensing for Cloth Texture Recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 76, 79

[109] Ilya Lysenkov and Vincent Rabaud. Pose estimation of rigid transparent objects in transparent clutter. In *2013 IEEE International Conference on Robotics and Automation*, pages 162–169. IEEE, 2013. 14

[110] Ilya Lysenkov, Victor Eruhimov, and Gary Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. *Robotics*, 273, 2013. 14

[111] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, 2021. 99

[112] Xiao Ma, David Hsu, and Wee Sun Lee. Learning Latent Graph Dynamics for Deformable Object Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022. 78

[113] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33 (4), jul 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601152. URL https://doi.org/10.1145/2601097.2601152. 41

[114] Miles Macklin, Matthias Muller, Nuttapong Chentanez, and Tae-Yong Kim. Unified Particle Physics for Real-Time Applications. *ACM Trans. Graph.*, 33 (4), July 2014. 83

[115] K. Maeno, H. Nagahara, A. Shimada, and R. Taniguchi. Light field distortion feature for transparent object recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2786–2793, June 2013. doi: 10.1109/CVPR.2013.359. 14

[116] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017. 15, 22

[117] Jeffrey Mahler, Rob Platt, Alberto Rodriguez, Matei Ciocarlie, Aaron Dollar, Renaud Detry, Maximo A Roa, Holly Yanco, Adam Norton, Joe Falco, et al. Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics. *IEEE Transactions on Automation Science and Engineering*, 15(4): 1440–1442, 2018. 14, 21, 26

[118] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315, 2010. doi: 10.1109/ROBOT.2010.5509439. 96, 98

[119] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *2010 IEEE International Conference on Robotics and Automation*, pages 2308–2315. IEEE, 2010. 54, 55, 76, 78

[120] Pragna Mannam, Avi Rudich, Kevin Zhang, Manuela Veloso, Oliver Kroemer, and F. Zeynep Temel. A Low-Cost Compliant Gripper Using Cooperative Mini-Delta Robots for Dexterous Manipulation. In *Robotics: Science and Systems (RSS)*, 2021. xviii, 77, 80, 81

[121] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *Conference on Robot Learning*, 2020. 1

[122] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-Real Reinforcement Learning for Deformable Object Manipulation. *Conference on Robot Learning (CoRL)*, 2018. 76, 78, 79

[123] 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. 34, 37, 38

[124] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 34, 37, 45

[125] Andrew T Miller, Steffen Knoop, Henrik Iskov Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. 2003. 15

[126] S. Miller, J. V. D. Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and

P. Abbeel. A geometric approach to robotic laundry folding. *The International Journal of Robotics Research*, 31:249 – 267, 2012. 98

[127] Stephen Miller, Jur van den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A Geometric Approach to Robotic Laundry Folding. In *International Journal of Robotics Research (IJRR)*, 2012. 78

[128] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021. 1

[129] K. Mohy el Dine, J. Sanchez, J. A. Corrales, Y. Mezouar, and J.-C. Fauroux. Force-torque Sensor Disturbance Observer Using Deep Learning. In *International Symposium on Experimental Robotics (ISER)*, 2018. 80

[130] Yusuke Moriya, Daisuke Tanaka, Kimitoshi Yamazaki, and Keisuke Takeshita. A method of picking up a folded fabric product by a single-armed robot. *ROBOMECH Journal*, 5:1–12, 2018. 54, 55

[131] Douglas Morrison, Peter Corke, and Jürgen Leitner. Learning robust, real-time, reactive robotic grasping. *The International Journal of Robotics Research*, 0 (0):0278364919859066, 0. doi: 10.1177/0278364919859066. URL https://doi.org/10.1177/0278364919859066. 15

[132] Douglas Morrison, Juxi Leitner, and Peter Corke. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.021. 12, 28

[133] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019. 35, 36, 39

[134] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340, 2018. 35

[135] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8688–8697, 2019. 3

[136] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017. 96, 99, 100, 101, 105, 126

[137] Peiyuan Ni, Wenguang Zhang, Xiaoxiao Zhu, and Qixin Cao. Pointnet++ grasping: Learning an end-to-end spatial grasp generation algorithm from sparse point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3619–3625, 2020. doi: 10.1109/ICRA40945.2020.9196740. 35, 36

[138] John Oberlin and Stefanie Tellex. Time-lapse light field photography for perceiving transparent and reflective objects. 14

[139] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 143–152, 2017. 119

[140] 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. 34, 37, 38, 41, 45

[141] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. In *ICLR*, 2018. 99, 100, 101

[142] Luis Pineda, Taosha Fan, Maurizio Monge, Shobha Venkataraman, Paloma Sodhi, Ricky Chen, Joseph Ortiz, Daniel DeTone, Austin Wang, Stuart Anderson, et al. Theseus: A library for differentiable nonlinear optimization. *arXiv preprint arXiv:2207.09442*, 2022. 42

[143] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. 15

[144] Kavya Puthuveetil, Charles C. Kemp, and Zackory Erickson. Bodies Uncovered: Learning to Manipulate Real Blankets Around People via Physics Simulations. In *IEEE Robotics and Automation Letters (RA-L)*, 2022. 78

[145] Jianing Qian, Thomas Weng, Luxin Zhang, Brian Okorn, and David Held. Cloth Region Segmentation for Robust Grasp Selection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 6, 79

[146] Arnau Ramisa, Guillem Alenya, Francesc Moreno-Noguer, and Carme Torras. Using Depth and Appearance Features for Informed Robot Grasping of Highly Wrinkled Clothes. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012. 79

[147] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322, May 2015. doi: 10.1109/ICRA.2015.7139361. 15

[148] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 63, 123

[149] Caner Sahin, Guillermo Garcia-Hernando, Juil Sock, and Tae-Kyun Kim. A review on object pose recovery: From 3d bounding box detectors to full 6d pose estimators. *Image and Vision Computing*, 96:103898, 2020. 1

[150] J. Sanchez, J. Corrales, B. Bouzgarrou, and Y. Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37:688 – 716, 2018. 98

[151] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: a Survey. In *International Journal of Robotics Research (IJRR)*, 2018. 76

[152] Jose Sanchez, Kamal Mohy El Dine, Juan Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Blind Manipulation of Deformable Objects Based on Force Sensing and Finite Element Modeling. In *Frontiers in Robotics and AI*, 2020. 80

[153] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks. *IEEE Robotics and Automation Letters*, 4(2):1357–1364, 2019. xv, xx, 12, 15, 16, 19, 20, 23, 26, 113

[154] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27 (2):157–173, 2008. 15

[155] Daniel Seita, Nawid Jamali, Michael Laskey, Ron Berenstein, Ajay Kumar Tanwani, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making. In *International Symposium on Robotics Research (ISRR)*, 2019. 76

[156] Daniel Seita, Nawid Jamali, Michael Laskey, Ajay Kumar Tanwani, Ron Berenstein, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making. In *International Symposium on Robotics Research (ISRR)*, 2019. 54, 55, 58

[157] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. *arXiv preprint arXiv:2012.03385*, 2020. 96, 98, 101, 107

[158] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen,

Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep Imitation Learning of Sequential Fabric Smoothing From an Algorithmic Supervisor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. 76, 78, 79, 96, 99, 101, 105

[159] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 78

[160] Ruhizan Liza Ahmad Shauri and Kenzo Nonami. Assembly manipulation of small objects by dual-arm manipulator. *Assembly Automation*, 2011. 97

[161] Zilin Si and Wenzhen Yuan. Taxim: An example-based Simulation Model for GelSight Tactile Sensors. In *IEEE Robotics and Automation Letters (RA-L)*, 2022. 83

[162] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. *arXiv preprint arXiv:2112.05124*, 2021. 37, 41, 46, 119

[163] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V. Dimarogonas, and Danica Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous Systems*, 60(10):1340–1353, 2012. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2012.07.005. 97

[164] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020. 37

[165] Li Sun, Gerardo Aragon-Camarasa, Simon Rogers, and J. Paul Siebert. Accurate Garment Surface Analysis using an Active Stereo Robot Head with Application to Dual-Arm Flattening. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015. 79

[166] Balakumar Sundaralingam, Alexander Lambert, Ankur Handa, Byron Boots, Tucker Hermans, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Robust Learning of Tactile Force Estimation through Robot Interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 80

[167] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Fox Dieter. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. xvi, 35, 36, 44, 45

[168] Giovanni Sutanto, Austin Wang, Yixin Lin, Mustafa Mukadam, Gaurav

Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm. volume 120 of *Proceedings of Machine Learning Research*, pages 804–813, The Cloud, 10–11 Jun 2020. PMLR. URL http://proceedings.mlr.press/v120/sutanto20a.html. 42

[169] Raúl Suárez, Jan Rosell, and Néstor García. Using synergies in dual-arm manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5655–5661, 2015. doi: 10.1109/ICRA.2015.7139991. 97

[170] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. Emd net: An encode–manipulate–decode network for cloth manipulation. *IEEE Robotics and Automation Letters*, 3(3):1771–1778, 2018. doi: 10.1109/LRA.2018.2800122. 98

[171] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 99

[172] Andreas ten Pas, Marcus Gualtieri, Kate Saenko, and Robert Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, 36(13-14):1455–1473, 2017. 15

[173] Sashank Tirumala, Thomas Weng, Daniel Seita, Oliver Kroemer, Zeynep Temel, and David Held. Learning to singulate layers of cloth using tactile feedback. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7773–7780. IEEE, 2022. 6

[174] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. 78

[175] Eric Torgerson and Fanget Paul. Vision Guided Robotic Fabric Manipulation for Apparel Manufacturing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 1987. 76

[176] Dimitra Triantafyllou and Nikos A. Aspragathos. A vision system for the unfolding of highly non-rigid objects on a table by one manipulator. In Sabina Jeschke, Honghai Liu, and Daniel Schilberg, editors, *Intelligent Robotics and Applications*, pages 509–519, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-25486-4. 54

[177] Dimitra Triantafyllou, Ioannis Mariolis, Andreas Kargakos, Sotiris Malassiotis, and Nikos A. Aspragathos. A geometric approach to robotic unfolding of garments. *Robotics Auton. Syst.*, 75:233–243, 2016. 54

[178] Roger Y Tsai, Reimar K Lenz, et al. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transactions on robotics and automation*, 5(3):345–358, 1989. 119

[179] Julen Urain, Niklas Funk, Georgia Chalvatzaki, and Jan Peters. Se (3)-diffusionfields: Learning cost functions for joint grasp and motion optimization through diffusion. *arXiv preprint arXiv:2209.03855*, 2022. 37

[180] Ulrich Viereck, Andreas ten Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017. 22, 24

[181] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2. 88

[182] Felix von Drigalski, Marcus Gall, Sung-Gwi Cho, Ming Ding, Jun Takamatsu, Tsukasa Ogasawara, and Tamim Asfour. Textile Identification Using Fingertip Motion and 3D Force Sensors in an Open-source Gripper. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2017. 79

[183] Lirui Wang, Yu Xiang, and Dieter Fox. Manipulation trajectory optimization with online grasp synthesis and selection. In *Robotics: Science and Systems (RSS)*, 2020. xvi, 35, 36, 44, 45, 120

[184] Lirui Wang, Xiangyun Meng, Yu Xiang, and Dieter Fox. Hierarchical policies for cluttered-scene grasping with latent plans. *IEEE Robotics and Automation Letters*, 7(2):2883–2890, 2022. doi: 10.1109/LRA.2022.3143198. 37

[185] Shaoxiong Wang, Mike Lambeta, Po-Wei Chou, and Roberto Calandra. TACTO: A Fast, Flexible, and Open-source Simulator for High-Resolution Vision-based Tactile Sensors. In *IEEE Robotics and Automation Letters (RA-L)*, 2022. 83

[186] David Watkins-Valls, Jacob Varley, and Peter Allen. Multi-modal geometric learning for grasping and manipulation. *arXiv preprint arXiv:1803.07671*, 2018. 15

[187] Thomas Weng, Amith Pallankize, Yimin Tang, Oliver Kroemer, and David Held. Multi-modal transfer learning for grasping transparent and specular objects. *IEEE Robotics and Automation Letters*, 5(3):3791–3798, 2020. doi: 10.1109/LRA.2020.2974686. 5

[188] Thomas Weng, Sujay Bajracharya, Yufei Wang, Khush Agrawal, and David Held. FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy.

In *Conference on Robot Learning (CoRL)*, 2021. 7, 76, 78, 79

[189] Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural grasp distance fields for robot manipulation. *arXiv preprint arXiv:2211.02647*, 2022. 45, 46

[190] Thomas Weng, David Held, Franziska Meier, and Mustafa Mukadam. Neural grasp distance fields for robot manipulation. 2023. 5

[191] Gordon Wetzstein, Ramesh Raskar, and Wolfgang Heidrich. Hand-held schlieren photography with light field probes. In *2011 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2011. 14

[192] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. 2022. 37

[193] B. Willimon, S. Birchfield, and I. Walker. Model for unfolding laundry using interactive perception. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4871–4876, Sep. 2011. 54, 55

[194] Ruihai Wu and Yan Zhao. Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects. In *International Conference on Learning Representations (ICLR), 2022*, 2022. 3

[195] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *ArXiv*, abs/1910.13439, 2019. 54, 55

[196] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.065. 96, 99, 100, 101, 103

[197] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Robotics: Science and Systems (RSS)*, 2020. 76, 78

[198] Yueh-Hua Wu, Jiashun Wang, and Xiaolong Wang. Learning generalizable dexterous manipulation from human grasp affordance. *Conference on Robot Learning*, 2022. 37

[199] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *Robotics: Science and Systems*, 2018. 1

[200] Fan Xie, Alexander Chowdhury, M. Clara De Paolis Kaluza, Linfeng Zhao, Lawson Wong, and Rose Yu. Deep imitation learning for bimanual robotic manipulation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2327–

2337. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/18a010d2a9813e91907ce88cd9143fdf-Paper.pdf. 97

[201] K. Yamazaki. Gripping positions selection for unfolding a rectangular cloth product. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 606–611, Aug 2018. 54

[202] Kimitoshi Yamazaki, Ryosuke Oya, Kotaro Nagahama, Kei Okada, and Masayuki Inaba. Bottom dressing by a dual-arm robot using a clothing state estimation based on dynamic shape changes. *International Journal of Advanced Robotic Systems*, 13(1):5, 2016. doi: 10.5772/61930. 99

[203] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *Robotics: Science and Systems*, 2020. 96, 99, 101

[204] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning Predictive Representations for Deformable Objects Using Contrastive Estimation. In *Conference on Robot Learning (CoRL)*, 2020. 78

[205] Wei Yang, Chris Paxton, Arsalan Mousavian, Yu-Wei Chao, Maya Cakmak, and Dieter Fox. Reactive human-to-robot handovers of arbitrary objects. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 37

[206] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, Learning, Perception, and Control Methods for Deformable Object Manipulation. *Science Robotics*, 6(54), 2021. 78

[207] Wenhao Yu, Ariel Kapusta, Jie Tan, Charles C. Kemp, Greg Turk, and C. Karen Liu. Haptic Simulation for Robot-Assisted Dressing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 76

[208] Wenzhen Yuan, Siyuan Dong, and Edward H. Adelson. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors*, 17(12), 2017. URL https://www.mdpi.com/1424-8220/17/12/2762. 76, 79

[209] Wenzhen Yuan, Shaoxiong Wang, Siyuan Dong, and Edward Adelson. Connecting Look and Feel: Associating the Visual and Tactile Properties of Physical Materials. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 79

[210] Wenzhen Yuan, Yuchen Mo, Shaoxiong Wang, and Edward Adelson. Active Clothing Material Perception using Tactile Sensing and Deep Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 76, 79

[211] Kevin Zhang, Mohit Sharma, Jacky Liang, and Oliver Kroemer. A modular robotic arm control stack for research: Franka-interface and frankapy. *arXiv*

preprint arXiv:2011.02398, 2020. 120

[212] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, Jia Pan, Wenzhen Yuan, and Michael Gienger. Challenges and Outlook in Robotic Manipulation of Deformable Objects. *arXiv preprint arXiv:2105.01767*, 2021. 78

[213] Luyang Zhu, Arsalan Mousavian, Yu Xiang, Hammad Mazhar, Jozef van Eenbergen, Shoubhik Debnath, and Dieter Fox. Rgb-d local implicit function for depth completion of transparent objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4649–4658, 2021. 37

[214] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013. 35, 39, 40, 42, 45, 46, 117, 118