Manipulation and Perception Policies for Robot Mechanical Search

Michael Danielczuk

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-98
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-98.html

May 13, 2022

Copyright © 2022, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Manipulation and Perception Policies for Robot Mechanical Search

by

Michael John Danielczuk

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Ronald Fearing
Assistant Professor Hannah Stuart

Spring 2022

Manipulation and Perception Policies for Robot Mechanical Search

Copyright 2022
by
Michael John Danielczuk

Abstract

Manipulation and Perception Policies for Robot Mechanical Search

by

Michael John Danielczuk

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

When operating in unstructured and semi-structured environments such as warehouses, homes, and retail centers, robots are frequently required to interactively search for and retrieve specific objects from cluttered bins, shelves, or tables, where the object may be partially or fully hidden behind other objects. The goal of this task, which we define as mechanical search, is to retrieve a target object in as few actions as possible. Robustly perceiving and manipulating objects is challenging in these scenarios due to the presence of sensor noise, occlusions, and unknown object properties. Because of these perception and manipulation challenges, learning end-to-end mechanical search policies from data is difficult. Instead, we break mechanical search policies into three modules, a perception module that creates an intermediate representation from the input observation, a set of low-level manipulation primitives, and a high-level action selection policy that iteratively chooses which low-level primitives to execute based on the output from the perception module. We explore progress made on manipulation primitives, such as pushing and grasping, segmentation of scenes with unknown objects and occupancy distribution predictions to infer likely locations of the target object. Additionally, we demonstrate that using simulated depth images or point clouds can allow rapid generation of large-scale training datasets for perception networks while allowing them to generalize to real-world objects and scenes. We show that integrating these components can result in an efficient mechanical search policy, improving the success rate by 15% and reducing the number of actions needed to extract the target object as compared to baseline policies in both bin and shelf environments across simulated and physical trials.

To Katie, who has been a constant source of encouragement, support, love and fun throughout my entire academic journey.

Contents

Acknowledgments

The research I have been so fortunate to do at Berkeley has been influenced, supported, and encouraged by so many mentors, colleagues, and friends along the way. To Mr. Barry Moran, my high school adviser, you taught me that math was a ton of fun through your magic and imparted so much sage life advice on me. To Mr. Ewen Ross and Ms. Julia Hinchman, my high school robotics coaches, you were the first to show me the world of robotics, and allowed Jarrod and I to build innovative and exciting FTC robots. No idea was too crazy to try, and you let us have full reign over the tools and resources we needed to explore how software and hardware could grow together (subject to the constraints of the game rules, of course). Without your leadership and expertise, I would never have developed the passion for robotics that I have today.

At Princeton, I never had much time for research until my senior thesis. Despite my lack of experience, I am extremely grateful that Professor James Sturm let me work under Josh Sanz-Robinson on flexible electronics research. From both of you, I learned the basics of being a graduate student and you taught me the excitement, joy, and fun of doing research on a daily basis. I also learned how to adjust project goals and how to assess options at a higher level when our initial ideas did not always work out. I would also like to thank the late Dr. Barry Burke and Daniel O'Mara for similarly taking a chance on me and allowing me to intern in the Advanced Imaging Technology group at MIT Lincoln Labs after my junior year of college. From you, I learned how to design experiments, test hypotheses, and dig deep into results when something seemed counterintuitive. Both of these research experiences were invaluable to me during my college career, and inspired me to continue on to a PhD. After college, I was fortunate enough to get back into robotics via VirtualApt; I am forever grateful to Bryan Colin and Filippo Alimonda for hiring me straight out of undergrad to be one of only a few employees at an ambitious robotics startup. Filippo, you were another incredible mentor for me and taught me how to manage numerous demo deadlines while teaching me how to be open to any and all new ideas that could improve our system.

When I arrived at Berkeley, I knew immediately that I wanted to do research in robotics, despite being admitted as an optoelectronics student. Professor Ken Goldberg was immediately enthusiastic about having me switch advisers and join his lab, recruiting me over a series of phone calls and breathlessly telling me about all of the exciting and unique project opportunities. Deciding to join the AUTOLab positively shaped my research career and will continue to as I transition into industry research. Ken, I am so thankful for your mentorship, advice, and willingness to get on a phone call at any hour to discuss how we could improve any and all of the projects I was working on. You taught me to think outside the box, encouraged me to connect ideas across different fields and research topics, gave me extremely thorough comments and advice on all of my papers, talks, and slides, and taught me how to develop themes and stories that wove through my research. You taught me the importance of applying all of our problems to the real world, running physical experiments, and being able to communicate all of our ideas to others in the lab, at Berkeley, and the public. For all of these reasons, I am very thankful to have had you as my PhD adviser.

After I initially joined the AUTOLab, I was extremely fortunate to be mentored by Jeff Mahler and collaborate with Matt Matl. Jeff immediately started me on a few projects that ended up defining much of my PhD and gave me an appreciation for the difficulty of robot grasping and manipulation, and Matt was always there for help whenever I needed it. Both of you were very selfless, excellent research role models, and impressed upon me how to create thorough, probing experiments and constantly come up with research questions. You are both incredible engineers, and I continue to aspire to reach your combined understanding of software and hardware, how to diagnose problems in both, and your understated efforts to maintain the lab's infrastructure and resources. In addition, you both knew not to take everything too seriously as a graduate student, whether by playing records during experiments or through lab happy hours, and encouraged me to balance my research with activities outside the lab like running, skiing, or backpacking. I'll never forget the trip to MARS 2018 to demo Dex-Net, and I look forward to continuing to work closely with you both, as well as Steve McKinley and David Gealy, at Ambi in the years to come!

In my time in the AUTOLab, I also had the opportunity to work closely with a huge number of PhD students, postdocs, undergraduates, and professors at Berkeley. In my mind, these interactions showed me why Berkeley is at the forefront of academic research in AI and robotics; I consistently had the pleasure of meeting with and picking the brains of incredible researchers and kind, thoughtful people. In addition to Jeff and Matt, I would especially like to thank Ashwin Balakrishna, Jeff Ichnowski, Raven Huang, Daniel Brown, Ajay Tanwani, Jingyi Xu, Chung Min Kim, Letian Fu, Chris Correa, Andrew Lee, and Andrew Li for their contributions to the material in this dissertation, as well as for making the lab a place I always looked forward to being in so I could chat and be around you all. To the undergraduates and masters students I mentored or co-mentored in addition to those above, including Katherine Li, Kate Sanders, David Tseng, David Wang, Zach Tam, Vishal Satish, Sona Dolasia, Zisu Dong, and Shivin Devgon, thank you for your patience as I learned how to balance projects and learn how to lead. You all taught me so much along the way and I am impressed by your dedication to research among all of your other commitments. To the PhD students in the lab that I did not work directly with, especially Carolyn Matl, Daniel Seita, Ryan Hoque, Brijen Thananjeyan, Bill DeRose, Michael Laskey, Sanjay Krishnan, Yahav Avigal, Justin Kerr, Ale Escontrela, Simeon Adebola, and Lawrence Chen, I enjoyed all of the times spent chasing deadlines together, your kindness in editing my papers, and the frequent banter in the lab. Having all of your support and seeing the excitement on everyone's faces each day helped make the tough days of research a little bit easier. Outside of the AUTOLab, I would like to thank Tae Myung Huh, Monica Li, Eric Chen, Debbie Liang, and Saurabh Gupta for your collaborations; you helped open me up to a variety of new topics that broadened my robotics knowledge significantly. I am also grateful to Ron Fearing and Hannah Stuart for serving on my qualifying exam and dissertation committees and I would like to thank the Berkeley EECS and BAIR administrators for their work to help me graduate on time and making my experience at Berkeley amazing, especially Angie Abbatecola and Shirley Salanio.

Outside of Berkeley, I had the pleasure of working with a large number of academic

and industry collaborators who helped to shape and contribute to my research journey. In particular, I would like to thank Andrey Kurenkov, Roberto Martín-Martín, Animesh Garg, and Silvio Savarese from Stanford, who worked closely with me to develop the first mechanical search pipeline, the inspiration for nearly the entirety of my dissertation. They helped to take the mechanical search project from an idea in a grant proposal to a fully-functioning real-world system and taught me so much about managing collaborations along the way. Similarly, Anelia Angelova, Brian Ichter, and Vincent Vanhoucke at Google helped me to take my research to another level; Anelia worked closely with me to come up with X-Ray and provided regular feedback and encouragement on the project and all three have continued to give me excellent feedback as we have extended X-Ray to lateral-access environments. My internship at NVIDIA would not have been successful without Arsalan Mousavian, Clemens Eppner and Dieter Fox; Arsalan in particular helped me to find a project that bridged our interests while connecting cutting-edge research in computer vision with robotics. Despite being remote the whole time, I enjoyed my internship immensely and I felt so lucky to work with such experts who could broaden the way I thought about mechanical search and the intersection of robot perception and manipulation. Juan Aparicio Ojea, Eugen Solowjow, Shirin Joshi, Eduardo M. C. Rocha and Nuttapong Chentanez also deserve acknowledgment for their insights and contributions to my projects on exploratory grasping and contact modeling. I would also like to acknowledge my funding sources, particularly my three years under the National Science Foundation Graduate Research Fellowship Program, Grant No. DGE 1752814, and support from the Berkeley AI Research (BAIR) Lab, and the CITRIS "People and Robots" (CPAR) Initiative, Google, Siemens, Toyota Research Institute and Autodesk.

Finally, I would like to acknowledge my friends and family for supporting me throughout my PhD; you all consistently kept me happy and healthy even through a pandemic! There are so many friends I would love to thank, especially Ryan Kaveh and Saavan Patel for running with me throughout my PhD, the Wololo crew for providing entertainment when we could not go outside, Kenny Rayner, Colin Gannon, Alex Mauro and the Strawberry Canyon Track Club for guiding me through training for my first Boston Marathon, and Campbell Weaver, Leo Tolias, David Balise, Andrew Shichman and Will Glockner, as well as the rest of the Tower crew. Last but not least, I would like to thank my parents and family and my fiancée Katie and her family for their unwavering support while encouraging me to enjoy life outside of research.

Chapter 1

Introduction

Searching for objects in constrained environments is a fundamental, daily problem for humans, whether it be looking for keys hidden within your house, a pen within a drawer, a can of soup on a cabinet shelf, or a box of clothes within a closet. At the same time, the dramatic expansion of e-commerce, fueled both by the COVID-19 pandemic and by growing expectations of convenient, diverse products that can be rapidly delivered, requires that desired objects must be retrieved efficiently from warehouse storage. In both cases, the desired object may be of a diverse appearance and shape, may be stored in bins or boxes or stacked on pallets or shelves, and is often in an unknown orientation. It may be hidden partially or fully by other objects of similar or completely different shapes, sizes, and textures. Thus, giving robots the ability to quickly assess where these objects are likely to be located, to interactively move and shift blocking objects aside, and to recover or extract the object of interest could have applications in home service of the elderly or disabled, warehousing, logistics and retail.

This dissertation investigates the problem of robotic "mechanical search" — identifying and retrieving desired objects from unordered collections of objects, typically performed by humans. Mechanical search lies at the intersection of robot perception and manipulation and thus faces the challenges associated with each of these fields in isolation as well as tradeoffs arising from their tight coupling. For example, perception pipelines must be robust to sensor noise and object occlusions, which can limit the robot's ability to infer precise object geometries and poses, or even to see some objects at all. Recent advances in object instance segmentation [91, 135, 267, 268] and object detection [84, 167, 214, 226, 286] that leverage deep learning have begun to address some of these challenges, but these methods may still struggle to generalize to unseen objects or partial observations. Similarly, manipulation policies must be robust to unknown object properties such as coefficients of friction and center of mass; coupled with imprecise robot actuation or calibration, robot grippers may not be able to precisely manipulate even a fully-observed object. Although there has been significant recent progress in task-agnostic grasping from point clouds or depth images even in the presence of these challenges [25, 79, 106, 110, 144, 166, 180, 201], these policies make multiple simplifying assumptions, such as (1) a grasp is successful if it lifts any object

in the environment, independent of object identity, purpose or intent, and (2) dynamic interactions between objects are negligible. By relaxing these assumptions in the context of the mechanical search problem, we require policies to decide when to trade off exploration (i.e., moving objects to reduce uncertainty about the desired object's pose) and exploitation (i.e., directly moving objects near the best estimate of the desired object's pose). Mechanical search requires that policies not only maintain a passive belief state over the current state of the objects in the scene, but also that policies actively influence their belief states by interacting directly with the objects to perturb them and reduce uncertainty while inferring contact locations, forces and torques from partial information.

Because of the above challenges, as well as the sparse-reward, long-horizon nature of the task, we argue that learning end-to-end mechanical search policies from data may be difficult. Instead, we propose a modular mechanical search policy made up of three components, a perception module that creates one or multiple intermediate representations from an input observation, a set of low-level manipulation primitives, such as grasping or pushing, and a high-level policy that selects low-level primitives. This dissertation contributes a number of methods for improving all aspects of a mechanical search policy, focusing on overhead-access environments, such as bins or tabletops, and lateral-access environments, such as shelves or cabinets.

1.1 Manipulation Primitives

To interact with the world around it, a robot must be able to robustly manipulate objects, whether by grasping, lifting, pushing or sliding them. Prior work has shown that grasping policies can be learned from data using labels gathered by humans [141, 180], by the robot itself through many interactions [110, 144, 201], or by dynamic simulation [106]. Recently, Mahler et al. [166] have shown that learning policies from labels generated by analytic grasp metrics in simulation can also generalize to real-world objects. However, these analytic grasp metrics do not consider the effects of increased contact area from compliant materials typically used for parallel-jaw grippers. In Chapters 2 and 3, we build on Mahler et al. [166] by improving the contact modeling techniques used to generate grasp labels. We introduce two computationally-efficient contact models that take into account the effects of compliant grippers deforming around rigid objects by modeling the full six-dimensional wrench applied at the contact.

Pushing objects is also a common primitive for singulating objects in cluttered environments so that they can be grasped [34, 60, 92] and can also be used to reorient objects that cannot be grasped in their current pose [74, 160, 172]. Prior work typically attempts to maximize separation between objects as a proxy for future grasp success rates. However, objects may not need to be completely separated to be grasped successfully, especially with vacuum-based suction grippers. In Chapter 4, we explore the effect of pushing primitives on grasp success rates directly by measuring predicted increases in parallel-jaw and suction grasp qualities. We then develop a set of pushing policies that attempt to maximize grasp

confidence over a single pushing action.

1.2 Perception Primitives

While manipulation primitives on their own provide a useful abstractions away from continuous control of the robot arm, they do not allow the robot to reason about relationships between objects, identify a target object, or reason about a target object's whereabouts. Perception components such as instance segmentation, object detection and localization, and target occupancy distributions can be viewed as similar abstractions for the input observation to the high-level policy, allowing it to understand object relationships without learning directly from raw pixels. While deep learning has had a major impact on these perception primitives in recent years, many state-of-the-art networks still rely on massive, hand-labeled datasets that are expensive to collect and may contain imperfect labels [70, 90, 91].

Chapters 5 and 6 discuss how we can leverage two insights to train perception networks that can robustly generalize to unseen real-world objects in the context of segmentation and collision prediction. The first insight is that simulation can provide cheap, accurate labels for many of these tasks; it can be used at scale to rapidly generate a large training dataset. The second insight is that simulated depth images can be used to ease transfer from simulation to reality. Taking inspiration from recent work by Mahler et al. [166], we show that training using simulated depth images as input can allow for seamless transition to depth images from real sensors without retraining or finetuning.

1.3 Mechanical Search Policies

Even given robust perception primitives and manipulation primitives, an action selection policy is needed to guide the mechanical search process. Action selection policies must reason about both past and future consequences of potential actions as well as relationships between objects that can be moved within or removed from the environment. Previous work considers only visual search [7] or known environments [266], or attempts to reconstruct object geometry [145, 204]. Similar policies have also been explored within environments with blocks or cylinders [188, 276, 282]. However, many of these policies may struggle to generalize to perceiving and manipulating unseen real-world objects.

Chapter 7 introduces a set of baseline action selection policies and shows how the manipulation and perception primitives discussed in Chapters 2 through 6 can be integrated as part of a mechanical search pipeline in extensive simulated and physical experiments. Additionally, we quantify how the mechanical search problem difficulty scales with the number of objects in the scene for these baseline policies in an overhead-access environment. Chapter 8 improves on these baseline policies by leveraging a more powerful intermediate representation: the target occupancy distribution. We show that directly reasoning about the target object pose and maximally reducing its uncertainty within the scene can result in a more

efficient and more successful mechanical search policy than the previous baselines. Then, in Chapter 9, we show how the concepts presented in Chapters 7 and 8 can be extended to the lateral-access settings (i.e., shelves) in cases where objects may not be removed from the environment.

Part I

Manipulation Primitives

Chapter 2

REACH: A Robust Efficient Area Contact Model

Inspired by the success of recent hybrid approaches for grasp planning that train deep neural networks on large datasets of simulated depth images with grasp labels generated by analytic grasp metrics, this chapter focuses on improving contact models to account for the area contacts produced by compliant gripper jaws. We hypothesize that using the less conservative and more accurate labels produced by efficient area contact models could lead to less conservative and more accurate grasp quality predictions, thus expanding the set of grasps on unknown objects in constrained settings like mechanical search.

In nearly all robotic grasping applications, gripper contacts are covered with compliant material to better resist disturbing wrenches [56, 80]. Compliant gripper fingertips also create contact areas that can deform around local surface geometry [147]. However, the majority of research on grasp planning uses a point contact model that does not take contact area into account [164]. Point contact models can produce a high false-negative rate, predicting failure for grasps that are successful, especially when grasping objects at edges and corners, as shown in Fig. 2.1. A high false negative rate may lead to a grasp planner being unable to find grasps on an object, even when they exist. Additionally, when training a deep network for grasp planning, a large dataset of true positives and true negatives are needed to avoid overly confident or conservative predictions [223]. Reducing the contact area to idealized surfaces [40, 97, 103, 274] or using the Finite Element Method (FEM) [41, 271, 273] are other approaches for modelling the interaction between a compliant gripper and objects, but the former does not model non-parametric surface geometries, while the latter is computationally expensive.

In this chapter, we address these issues and propose the REACH model, a novel Robust Efficient Area Contact Hypothesis model for robot grasping that estimates contact area through the constructive solid geometry intersection between the object mesh and the compliant volumetric model of each gripper jaw. REACH represents the pressure distribution using a linear model based on the deformation of the jaw and decomposes the contact surface into a triangular mesh to compute the friction constraints for each triangle based on the

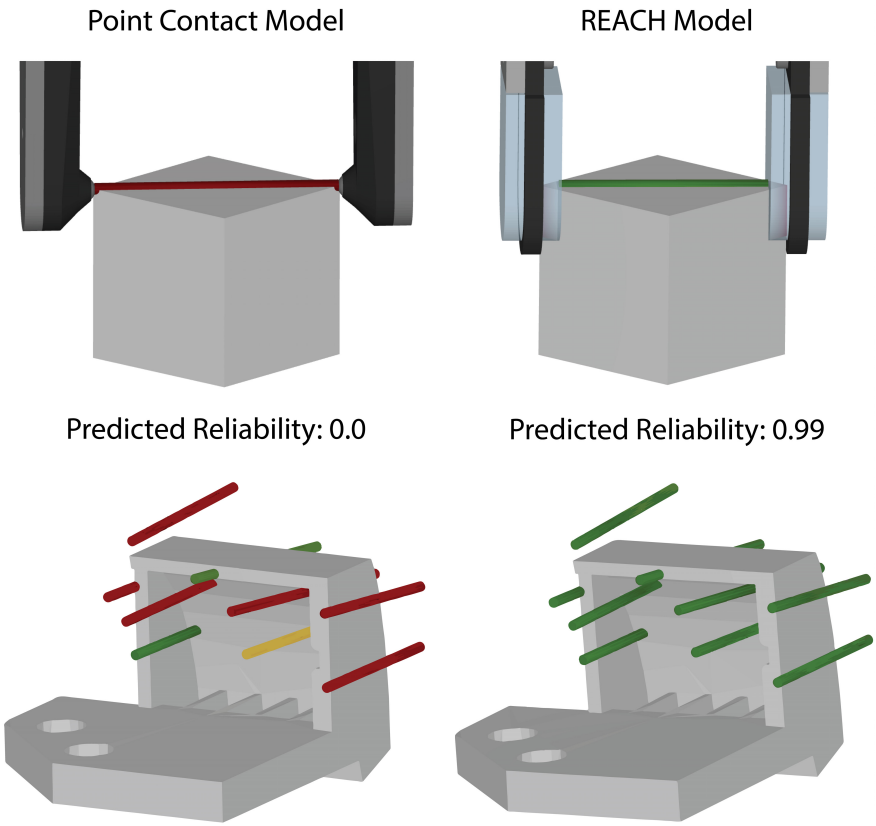


Figure 2.1: Top: An example of a grasp on the corner of a cube. With a point contact model (left) the grasp cannot resist the gravity wrench since the contacts slip; however, with the REACH model (right) the grasp can resist the gravity wrench under perturbations in pose and material properties as the gripper deforms around the object corners to form a sticking contact. Bottom: Predicted grasp reliability (red, yellow, green $\rightarrow$ low, medium, high) under the point contact model (left) and under the REACH model (right) for a bracket, illustrating that the REACH model is far less conservative. The REACH model is significantly more accurate in predicting the true outcome of physical grasp experiments on this bracket.

ellipsoidal limit surface [97]. "Hypothesis" refers to the inherent uncertainty in these constraints for multi-point frictional contacts. The wrench contributions from all triangles form the grasp wrench space, which is used to predict grasp reliability (probability of success). As in Dex-Net 1.0 [163], REACH uses Monte-Carlo sampling to evaluate grasp robustness under perturbations in gripper pose and material properties.

This chapter makes two contributions:

1. REACH, a novel and efficient area contact model that considers full six-dimensional wrenches applied at the contact by modeling frictional constraints at each triangle.

2. A labeled dataset from physical experiments with 2,625 grasps of 21 objects by an ABB YuMi robot with a compliant parallel-jaw gripper, with cross-validation experiments comparing REACH to point and dynamic contact models across results from physical experiments.

2.1 Related Work

There is substantial literature on contact modeling for robot grasping, with notable surveys by Kao, Lynch, and Burdick [114], Rimon and Burdick [215], and Bicchi and Kumar [20].

2.1.1 Point Contact Models

Point contact models provide constraints on forces and torques applied at the contact between a point and a surface. A point contact can be modeled as frictionless point contact, frictional, or soft [20, 222]. A frictionless point contact can only exert a force along the inward contact normal. A frictional point contact can additionally exert a tangential force, and a soft point contact also applies a pure torsional moment about the inward contact normal. The tangential force is commonly constrained by Coulomb's Law, and the torsional moment is similarly constrained, scaling with a *torsional friction coefficient* [172]. The tangential force and torsional moment that a soft point contact exerts can also be jointly constrained by the so-called *friction limit surface (FLS)* [76, 147, 186], which Howe et al. approximated with an ellipsoid for computational efficiency [97].

2.1.2 Area Contact Models

The first area contact models used Hertz's contact theory [94], which described the contact profile for two elastic bodies. Xydas et al. extended the model to include materials that are not linear-elastic [115, 274]. Other compliant contact models have also been introduced [103, 168]. Barbagli et al. [12] provide an excellent summary of several of these models, and how well each one approximates the human finger. However, these models assume a spherical or hemispherical object contacting a planar surface, and require finite-element analysis for nontrivial surfaces [202, 273]. Some work has focused on adapting these models to non-planar geometries. Ciocarlie et al. applied constraints from the ellipsoidal FLS model to area contacts by summing over elements in a FEM simulation [41] and by approximating the contact area with an ellipse and applying Hertzian and Winkler pressure distributions [40]. The 2D ellipse is later generalized to quadric surfaces in Tsuji et al. [248] to approximate the contact area. Xu et al. [271] analyze a 3D subspace of 6D friction constraints for curved contact area results from soft-finger grasping, where the contact profile is obtained from a FEM simulation. Ghafoor et al. applied a different model consisting of circular elastic point contact contours to find the 6D grasp stiffness matrix, which relates the force applied by the finger to the 6D wrench applied at the contact [71].

Of these, Ciocarlie et al.'s work is most similar to the work presented here, but our work precisely determines the area of contact without approximating the surface. Charusta et al. also find a patch contact by intersecting the gripper geometry with the object, but only consider spherical gripper pads and do not consider the pressure distribution formed by the patch contact [35]. In contrast, we consider arbitrary planar gripper geometries and explicitly model the pressure distribution created by the patch contact. Sinha and Abel analyze non-planar contact surfaces through discretization into finite area patches [232], but they applied their results only to objects with simple geometries (such as cubes and cylinders) and required a variational approach to solve for a quasi-static equilibrium. In contrast, we estimate the contact area without a smooth approximation of the object surface or FEM simulations. The proposed model also considers the full 6D wrench applied at the contact, whereas Ciocarlie et al. [40] and Charusta et al. [35] consider a 4D wrench and Xu et al. [271] study a 3D subspace.

2.1.3 Grasp Wrench Space Analysis

Our wrench-space analysis extends the analysis presented by Mahler et al. [165, 166] in that we characterize a grasp as successful if the grasp resists the gravity wrench. Previous work has considered many metrics including probability of force closure and ability to resist any applied disturbing wrenches [42, 216, 260]. Krug et al. perform a complete analysis of wrench-based grasp quality metrics [132]. In this work, we develop constraints for each patch contact applied to the object and solve a quadratic program to determine if the forces applied at the contacts can resist the gravity wrench applied to the object.

2.1.4 Grasp Datasets

Bohg et al. provided a survey of the most commonly-used grasp datasets [21]. Large-scale datasets for grasping are commonly created via hand-labeling, such as Team MIT-Princeton's dataset for the 2017 Amazon Picking Challenge [284] and Lenz et al.'s Cornell Grasping Dataset [141]. Other training datasets label successful robot grasps either in simulation [106] or on a physical system [143, 201]. Mahler et al. proposed a hybrid approach, using soft point contact models to efficiently generate a labeled dataset of millions of grasps for 3D models in simulation, and training a network on a labeled dataset of simulated grasps [164, 165, 166]. In contrast, to create our physical dataset, we sample grasps in simulation and gather ground-truth labels on the physical system.

2.2 Problem Statement

We consider the problem of predicting grasp reliability, or probability of grasp success, based on the ability of the grasp contacts applied by a robot gripper to resist the gravity wrench.

In this chapter, we consider a parallel-jaw gripper, but the model can be applied to grippers with any number of jaws.

2.2.1 Assumptions

We make the following assumptions:

1. Quasi-static physics (inertial terms are negligible) and Coulomb friction.
2. Objects to be grasped are rigid with known geometry.
3. The gripper has known geometry and two parallel jaws, each covered with a linear-elastic material whose geometry can be approximated by an extruded planar polygon.
4. The friction coefficient μ is constant over the contact area.
5. Both gripper jaws make contact simultaneously.
6. The jaw applies a force normal to each triangle's face in the contact patch.

2.2.2 Definitions

The state $\mathbf{x}$ of the grasping scene consists of the geometric, mass, and frictional properties describing an object $\mathcal{O}$, as well as the pose of the object $T_{\mathcal{O}}$. The set of actions are a set of grasps $\mathcal{U}$ that are available to a robot with a parallel-jaw gripper. Each grasp $\mathbf{u} \in \mathcal{U}$ is parametrized by a nominal grasp center $\mathbf{p} \in \mathbb{R}^3$ and an angle $\phi \in \mathcal{S}^3$. The jaws close with closing force f_C around the grasp center and are oriented according to the grasp angle. Success is measured through a binary reward function R, where $R = 1$ if the grasp successfully lifts the object and $R = 0$ otherwise. To model uncertainty in state, imprecision in robot control and imperfect knowledge of external wrenches, we define grasp reliability as a distribution $Q(\mathbf{x}, \mathbf{u}) = \mathbb{P}(R \,|\, \mathbf{x}, \mathbf{u})$ that describes the probability of success for a state $\mathbf{x}$ and grasp $\mathbf{u}$ [165]. We evaluate reliability in simulated environments by varying object pose, mass, and frictional properties, and by taking $Q(\mathbf{x}, \mathbf{u})$ to be the sample mean of N trials: $Q(\mathbf{x}, \mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} R_i(\mathbf{x}, \mathbf{u})$.

2.2.3 Objective

Given a nominal state $\mathbf{x}$ and grasp $\mathbf{u}$, we seek an accurate, efficiently-computable estimation of reliability $Q(\mathbf{x}, \mathbf{u})$, while optimizing average precision (AP). AP is defined as the weighted mean of precision values at each recall threshold, with the weights being the increase in recall from the previous threshold, and measures area under the precision-recall curve. This metric optimizes binary classification performance for an imbalanced dataset (e.g., more successes than failures) [221].

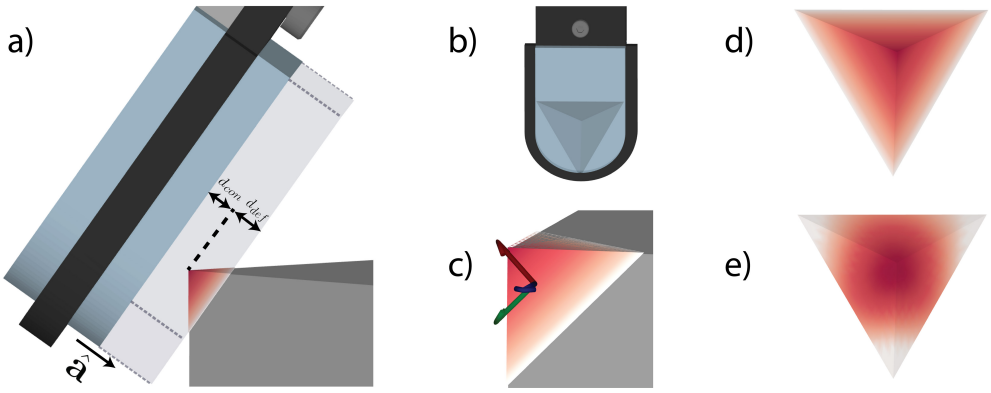


Figure 2.2: (a) Under the REACH model, the contact patch is formed via the constructive solid geometry intersection between the extruded pad polygon and the object mesh. $\hat{\mathbf{a}}$ indicates the approach vector of the jaw. Redder colors represent more deformation of the pad at that point and hence higher pressure. (b) The deformation of the gripper and (c) the triangle frame for a single triangle in the contact patch. The wrenches that can be applied at each triangle sum to form the total wrench applied to the object in its frame. (d) The pressure distribution of the patch under the REACH model and (e) as measured by a Weiss Robotics WTS tactile sensor [258].

2.3 REACH Model

To efficiently approximate grasp reliability, the REACH model estimates the contact wrenches a gripper can apply to the object. We first estimate the contact area, which we decompose into multiple triangles, and the pressure distribution. The wrenches that can be applied at each triangle are computed individually, and the sum of each triangle's wrenches forms the total wrench that the grasp can apply. Each triangle contributes frictional and normal wrenches, where the frictional wrenches are constrained with an elliptical FLS model [97, 186].

The wrench set that the contacts can exert in the object frame is given by $\Lambda = \{\mathbf{w} \in \mathbb{R}^6 \mid \mathbf{w} = G\boldsymbol{\alpha},\ \boldsymbol{\alpha} \in \mathcal{F}\}$, where $\boldsymbol{\alpha} \in \mathbb{R}^{6n}$ for n triangles, and $G \in \mathbb{R}^{6\times 6n}$ is a set of $6n$ basis wrenches in the object frame. If each triangle has m constraints, $\mathcal{F} \subseteq \mathbb{R}^{mn}$ is the set of constraints for the grasp. The contacts then can resist an external wrench $\mathbf{w}$ if $-\mathbf{w} \in \Lambda$ [132, 165, 186]. In practice, we solve this system using a quadratic program, which modern solvers can efficiently find an exact solution for given that the set $\mathcal{F}$ is defined by linear equality and inequality constraints. In this section, we describe the REACH model for efficient computation of the contact profile and the constraints $\mathcal{F}$.

2.3.1 Contact Area Computation

We model the object's geometry as a triangular mesh and define the contact area as the constructive solid geometry intersection of the extruded polygon of the jaw with the object. To model deformability of the jaw material, we find the gripper deformation depth $d_{def} = \kappa f_C$, where κ is a material constant specific to the jaw material and f_C is the closing force the jaw can apply. If the material has a physical limit for how much it can deform, we introduce this constraint by saturating d_{def} at this limit. We define the contact distance d_{con} as the minimum positive distance before contacting the object and slice the intersection mesh with planes parallel to the gripper at distances d_{con} and $d_{con} + d_{def}$ along the approach vector, as shown in Fig. 2.2. If the resulting surface patch(es) on the object have a finite non-zero area, then the jaw contacts the object.

We define a frame of reference for the object, where the origin lies at the center of mass of the object and the orientation depends on the object, and a frame of reference for each triangle in the contact patch, where the origin lies at the barycenter of the triangle, the z-axis is along the face normal, and the x-axis points to the first vertex. Fig. 2.2 shows these frames for one triangle in a contact patch. Then, for a deformation function $\delta : \mathbb{R}^3 \to \mathbb{R}$ that describes the distance the jaw presses into the object at each point on the contact surface, we consider a pressure distribution $p(\mathbf{r}) = k\delta(\mathbf{r})(\hat{\mathbf{n}} \cdot \hat{\mathbf{a}})$ that scales with deformation and with the angle between the object surface normal $\hat{\mathbf{n}}$ and approach vector $\hat{\mathbf{a}}$. The normalizing constant k is determined by integrating the pressure distribution over the contact area. The pressure distribution $p_i : \mathbb{R}^2 \to \mathbb{R}$ in the i-th triangle frame is given by transforming from object frame into the i-th triangle frame. Note that the pressure distribution in triangle frame varies only over two coordinates, as the triangle is planar. Fig. 2.2 shows that this pressure distribution reasonably models that measured by a Weiss Robotics WTS tactile sensor [258] for the given contact.

2.3.2 Wrench Space Constraints

Once we find the contact patch and its pressure distribution, we construct the ellipsoidal FLS for each triangle. The 6D wrench $\begin{bmatrix} f_{x_i} & f_{y_i} & f_{z_i} & \tau_{x_i} & \tau_{y_i} & \tau_{z_i} \end{bmatrix}^T$ for the i-th triangle in its local triangle frame is given in terms of the friction coefficient μ, the unit instantaneous velocities $\hat{\mathbf{v}}_i = \begin{bmatrix} v_{x_i} & v_{y_i} & v_{z_i} \end{bmatrix}^T$, and the pressure distribution $p_i(\mathbf{r}_i)$ over the triangle area A_i parametrized by coordinates $\mathbf{r}_i$ [114]:

$$
\begin{aligned}
f_{t_i} &= \begin{bmatrix} f_{x_i} \\ f_{y_i} \end{bmatrix} = -\int_{A_i} \mu \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} p_i(\mathbf{r}_i)\, dA_i, \\
f_{z_i} &= \int_{A_i} p_i(\mathbf{r}_i)\, dA_i, \\
\tau_{x_i} &= \tau_{y_i} = 0, \\
\tau_{z_i} &= -\int_{A_i} \mu \|\mathbf{r}_i \times \hat{\mathbf{v}}_i\| p_i(\mathbf{r}_i)\, dA_i.
\end{aligned}
$$

Note that τ_{x_i} and τ_{y_i} are constrained to be zero in the local triangle frame, but are typically non-zero after transformed into the object frame for the whole contact patch. The torsional moment and tangent friction force are jointly limited by a 3D ellipsoidal FLS:

$$\frac{\|f_{t_i}\|^2}{(f_{t_{i,\max}})^2} + \frac{|\tau_{z_i}|^2}{(\tau_{z_{i,\max}})^2} \leq 1, \tag{2.3.1}$$

where $f_{t_{i,\max}}$ and $\tau_{z_{i,\max}}$ are the maximal frictional force and torque, which are determined by maximizing the integral over the set of possible velocities $\hat{\mathbf{v}}_i$. Then the tangential force has maximum magnitude with $\hat{\mathbf{v}}_i = \begin{bmatrix} v_{x_i} & v_{y_i} & 0 \end{bmatrix}^T$ completely in the tangent plane:

$$f_{t_i} = \begin{bmatrix} f_{x_i} \\ f_{y_i} \end{bmatrix} \leq \mu \begin{bmatrix} f_{z_i} v_{x_i} \\ f_{z_i} v_{y_i} \end{bmatrix},$$
$$\|f_{t_i}\|_2^2 \leq (\mu f_{z_i})^2 = (f_{t_{i,\max}})^2, \text{ with } |f_{z_i}| \leq f_{z_i,\max},$$

where $f_{z_i,\max}$ is the fraction of the closing force f_C that is applied normal to the triangle. The torsional moment is maximized by $\hat{\mathbf{v}}_i = \begin{bmatrix} -y_i & x_i & 0 \end{bmatrix}^T / \sqrt{x_i^2 + y_i^2}$:

$$\tau_{z_{i,\max}} = -\int_{A_i} \mu \, \|\mathbf{r}_i \times \hat{\mathbf{v}}_i\| \, p_i(\mathbf{r}_i) \, dA_i \leq \mu \int_{A_i} \sqrt{x_i^2 + y_i^2} \, p_i(\mathbf{r}_i) \, dA_i.$$

Next, we compute the symbolic integrals for $f_{x_{i,\max}}$, $f_{y_{i,\max}}$, $f_{z_{i,\max}}$, and $\tau_{z_{i,\max}}$, so that they can be efficiently computed as a function evaluation at runtime. We determine the lower bound for $\tau_{z_{i,\max}}$, as its constraint integral can only be numerically integrated. This conservative approximation is used to limit false positives in grasp planning, and by avoiding numerical integration, we greatly increase efficiency in grasp prediction.

We start by transforming each integral to barycentric coordinates [259]. Given a function $f(\mathbf{r})$, with $\mathbf{r} = (x, y, z)$ in Cartesian coordinates, to integrate over a triangle T with vertices $\mathbf{t}_1 = (x_1, y_1, z_1), \mathbf{t}_2 = (x_2, y_2, z_2), \mathbf{t}_3 = (x_3, y_3, z_3)$, and area A,

$$\iint_T f(\mathbf{r}) \, dA = 2A \int_0^1 \int_0^{1-\lambda_2} f(\lambda_1 \mathbf{t}_1 + \lambda_2 \mathbf{t}_2 + (1 - \lambda_1 - \lambda_2)\mathbf{t}_3) \, d\lambda_1 d\lambda_2.$$

Then, we symbolically compute the maximal normal force of the i-th triangle $f_{z_{i,max}}$ based on its pressure distribution $p_i = a_i x_i + b_i y_i + d_i$, where a_i, b_i are based on the transform to triangle frame, and d_i is an offset from the first point of contact to the barycenter of the i-th triangle:

$$\begin{aligned} f_{z_{i,\max}} &= \int_{A_i} k(a_i x_i + b_i y_i + d_i) \; dA_i \\ &= \frac{A_i k}{3}(a_i(x_{i,1} + x_{i,2} + x_{i,3}) + b_i(y_{i,1} + y_{i,2} + y_{i,3}) + 3d_i). \end{aligned}$$

The lower bound of the maximal torsional moment $\tau_{z_{i,\max}}$ for the i-th triangle is computed with:

$$\begin{aligned}\tau_{z_{i,\max}} &= \int_{A_i} k(a_i x_i + b_i y_i + d)\sqrt{x_i^2 + y_i^2}\ dA_i \\ &\geq \frac{k}{\sqrt{2}}\left(\left|\int_{T_i} x_i(a_i x_i + b_i y_i + d)\ dA_i\right| + \left|\int_{T_i} y_i(a_i x_i + b_i y_i + d)\ dA_i\right|\right).\end{aligned}$$

The proof of the lower bound $\tau_{z_{i,\max}}$ and complete derivations and expressions for $f_{z_{i,\max}}$, $\tau_{z_{i,\max}}$ can be found in Appendix A.

Once these per-triangle integrals are complete, we form the final set of constraints for each triangle. First, the normalizing constant k is found using the condition that $f_C = \sum_{i=1}^{n} f_{z_{i,max}}$. This equality follows from the condition that the force applied must be equivalent to the integral of the pressure distribution over the contact area [114]. We additionally introduce equality constraints on k so that the pressure distribution is consistent across triangles. Next, we linearize the constraint in Equation 2.3.1 by approximating the ellipsoid with a series of linear constraints found by sampling j points on the ellipsoid [40]. For our implementation, $j = 18$, so there are $m = 23$ constraints for each triangle. In the experiments below, we use the ten triangles with the largest area for our constraints (for a total of 230 constraints per contact), as we empirically found these accurately describe the contact patch for nearly all objects without sacrificing prediction accuracy.

2.4 Experiments

2.4.1 Physical Experiments Dataset

To evaluate the precision and recall of the REACH model, we collected a set of 2,625 grasps across a set of 21 3D-printed objects (seen in Fig. 2.3) on a physical ABB YuMi robot with a compliant parallel-jaw gripper. These objects test generalization of our method to a diverse set of object geometries. To generate grasps for each object, we first computed all stable poses with a probability greater than 1% [73]. Then, in simulation, we placed each object in each stable pose on a flat workspace and sample parallel-jaw grasps on and around the object (within a radius equal to the width of the gripper). Grasps are sampled by choosing a point on the surface of the object and shooting rays along vectors within an angular threshold of the negative surface normal to find a second point on the surface. Grasps that only contact at one point or are wider than the maximum gripper width are discarded. Additionally, grasps that fall within a distance threshold of another grasp are discarded, along with stable poses that do not have any grasps that can be sampled (for example, the object is too wide for the gripper in the stable pose). Grasps within the gripper pad width of the object are sampled by adding uniform noise samples to the generated grasp centers, resulting in a set of sampled grasps such as those in Fig. 2.1.

When evaluating grasps on the physical system, a stable pose for the object is chosen from its stable pose distribution. A random grasp for the chosen stable pose is planned in

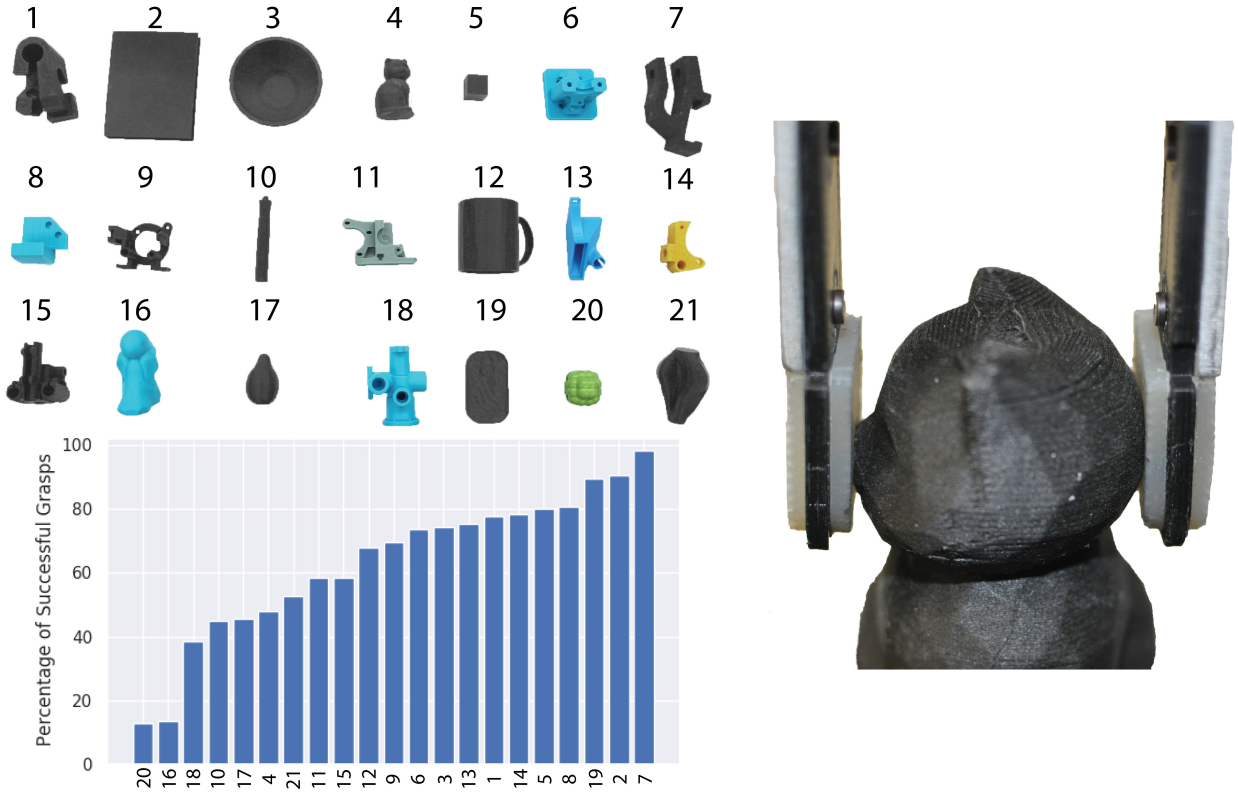


Figure 2.3: The 21 objects (top left) used to create the labeled dataset of real-world grasps, along with the percentage of sampled grasps that succeeded on the physical system for each object (bottom left). The objects are all 3D printed so that grasps could be indexed from simulation and were chosen to cover a wide range of geometries and grasp success rates. An example real-world grasp (right) with our compliant gripper, showing its ability to deform around complex geometries.

simulation and is executed on the physical system through a point-cloud registration system built around Super4PCS [173]. The transformation between the simulated point-cloud and the point-cloud captured on the physical system is applied to the simulated grasp pose, and grasp success is recorded by a human operator. A grasp is successful if the object is stably lifted (e.g., does not fall from the jaws during lifting). We sample 25 grasps for each object and each sampled grasp is repeated 5 times to measure robustness to small errors in registration and actuation. The 21 objects used and percentage of sampled grasps that succeeded are shown in Fig. 2.3. The complete dataset of 2625 object and grasp poses with success labels can be found at `https://sites.google.com/berkeley.edu/reach`, along with videos of each grasp trial.

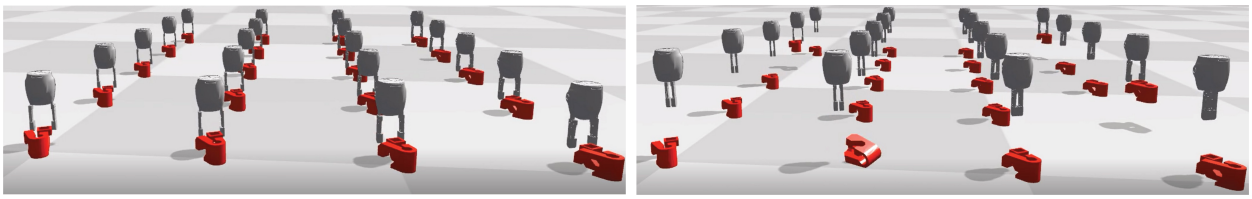


Figure 2.4: An example grasp on object 1 in the Flex dynamic simulator, with perturbations applied to each trial. We use a prototype GPU rigid-body global solver based on [161] to move to the grasp pose (left), close the gripper jaws, and lift (right). Success is recorded for each trial if the object remains in the gripper's jaws after lifting.

2.4.2 Benchmark Estimators

In addition to the model proposed here, we also benchmark three other models; two analytical models used in the literature, and a full rigid-body dynamic simulation implemented in NVIDIA Flex. Robust versions of each of the analytical models are also considered, and are calculated as described in Sec. 2.2.2 by taking the sample mean of N grasp trials over perturbations in object pose and frictional properties. Each object mesh contains between 12 and 3400 triangles, with an average of 1200 triangles.

Soft Point Contact Model

The soft point contact model applies a 4D wrench at the contact point, $[f_x \ f_y \ f_z \ \tau_z]^T$ [222]. The constraints on f_x and f_y are the linearized friction cone, determined by friction coefficient μ, and f_z is constrained by the closing force of the gripper. The torque τ_z applied at the contact is determined by a separate *torsional coefficient of friction* γ and the constraint is given by γf_z.

Elliptical Area Contact Model

The elliptical contact model is adapted from Ciocarlie et al. [40]. It is implemented here by first fitting a 2D ellipse to the extracted surface patch. The Hertztian pressure distribution is assumed to compute the maximal frictional force and torque; these constraints are solved for exactly by Ciocarlie et al. and depend only on the closing force of the gripper and the ellipse axis lengths. In this case, the linearized 3D ellipsoidal limit surface model for each contact forms the grasp wrench space.

Flex Simulation

We use a prototype version of a GPU-based rigid-body global solver [161] in NVIDIA Flex to run a full dynamic simulation of each grasp in the physical dataset. The solver uses a preconditioned conjugate residual method (PCR) to solve an approximated linear system for

each outer iteration. We use a time step of 1/60 sec with 2 sub-steps, 6 outer iterations and 40 PCR iterations per each outer iteration, with a relaxation factor of 0.75. Each gripper consists of 720 triangles, 300 for each finger and 120 for the base. Each trial consists of the gripper moving to the grasp pose, closing the gripper, and attempting to lift the object. Since the simulator allows for many simultaneous trials, we run 100 trials of each grasp in parallel, with perturbations of the same magnitude as we use for our robustness score applied to each trial. Fig. 2.4 shows an example grasping scene.

2.4.3 Metrics

Average precision (AP) and average recall (AR) are calculated using the ground-truth grasp labels and each model's predictions for the dataset of physical grasps. Average Precision is defined as the mean of the precision at each recall threshold, weighted by the difference between thresholds. Thus, AP is a measure of the area under the precision-recall curve and indicates overall performance of a binary classifier. AR is inversely related to the false negative rate (grasps predicted to fail but succeed) and is calculated as the recall averaged across 11 equally-spaced prediction thresholds for success from 0.0 to 1.0. Since each model has several parameters that can be tuned for each object, a cross-validation approach was used to avoid overfitting the model to the choice of objects. We sampled model parameters from uniform distributions: $U(0.3, 1.5)$ for the tangential friction coefficient, and $U(0.01, 1)$ for the point contact model's torsional friction coefficient or $U(0.0001, 0.01)$ for the area contact models' material constant. We performed leave-one-out cross-validation with 100 sets of sampled parameters, choosing the best performing model on the training objects and applying it to the held-out object. We also measure run time for each model on an Ubuntu 16.04 machine with a 12-core 3.7 GHz i7-8700k processor and a NVIDIA V100 GPU.

2.4.4 Discussion

The results in Tab. 2.1 suggest that the REACH model is able to correctly predict grasps that are classified as false negatives by the other methods, at the cost of several more false positive predictions. Full results for each object can be found in Appendix A. The non-robust REACH model achieves an absolute 17% higher AR than the soft point contact model while maintaining a similar AP. Adding robustness increases AP for both models, but the REACH model still increases AR by an absolute 11% over the point contact model. Flex outperforms both the robust soft point contact model and the REACH model in the AP metric by 4%, but at the cost of a run time that is 20x that of the REACH model and 250x that of the robust point contact model. Thus, despite similar AP performance to the robust soft point contact model, the REACH model shows promise for use in a grasp planner, as it can recall grasps that the other models are unable to recall.

These results also suggest that robustness can significantly increase average precision; the AP for both the robust soft point contact model and the REACH model is 6% and 5% higher than their non-robust counterparts. While these results are limited to our dataset of

Table 2.1: Average Precision (AP) and Average Recall (AR) for each model's predictions of grasp quality, for the 525 grasps collected on the physical robot. We perform leave-one-out cross validation, and average the AP and AR on each of the held-out folds. The non-robust REACH and REACH models achieve an absolute 17% and 11% higher AR than the non-robust and robust point contact models, respectively, with similar AP. Adding robustness increases AP and still allows the REACH model to recall grasps that the other models cannot. Flex outperforms all models in AP, but requires 20x more computation time. Full results for each object can be found in Appendix A.

Model	**AP**	**AR**	**Run Time (s/grasp)**
Point	0.76	0.55	**0.006**
Point Robust	0.82	0.55	0.040
Elliptical Area	0.73	0.47	0.024
Elliptical Area Robust	0.73	0.47	0.220
Flex Simulation	**0.86**	0.54	10.300
REACH (non-robust)	0.77	**0.72**	0.053
REACH	0.82	0.66	0.520

plastic 3D printed objects, adding this robustness could also help the model generalize to different object masses and frictional properties.

Additionally, we note that the sampled parameters that best fit our physical dataset for the REACH model often are unrealistic for the physical gripper that we use. In particular, the sampled material constant often leads to deformation depths much larger than the material can actually deform. We hypothesize that this choice of parameters leads to better performance because more deformation accounts for more dynamic grasps where the object may rotate slightly into alignment or where one jaw contacts the object first.

2.5 Conclusions

Accurately and efficiently modeling the area contacts produced by interactions between compliant parallel-jaw grippers and rigid objects is crucial for improving analytic grasp quality labels used to train hybrid grasp quality prediction networks. In this chapter, we introduced REACH, a Robust Efficient Area Contact Hypothesis model that estimates contact profile and contact wrench constraints by leveraging triangular mesh models and barycentric integration for computational efficiency to increase recall over traditional point contact models by 17%. The remaining false positives and false negative predictions from the REACH model mainly stem from dynamic effects, such as the object initially slipping before being grasped and the potential change of object pose due to two jaws not in contact simultaneously.

Chapter 3

6DFC: Efficiently Planning Soft Non-Planar Area Contact Grasps using 6D Friction Cones

This chapter continues the discussion from the previous chapter on improving contact models used in analytic grasp quality evaluation to account for the area contacts created by interactions between compliant parallel-jaw grippers and rigid objects. The REACH model in Chapter 2 can efficiently compute area contact wrench constraints for use in analytic grasp quality evaluation, but has two main drawbacks: (1) it does not scale gracefully to finer surface geometries (i.e., meshes with hundreds or thousands of triangles within the contact region) because it adds linear constraints for each contact triangle, and (2) it can overestimate the total frictional wrench applied at the contact since there is no coupling between contact triangles. In this chapter, we aim to further improve accuracy and efficiency of the REACH model by addressing these two drawbacks through the 6DFC (6D friction cone) model. 6DFC creates only one set of constraints per contact, allowing it to scale more easily to objects with fine surface geometries, and considers velocity vectors that are consistent across all triangles in the contact area to avoid overestimating the total contact wrench constraints.

As stated in the previous chapter, wrench-based grasp quality metrics rely on accurate estimation of the forces and torques applied at the contacts between the gripper and object to form the Grasp Wrench Space (GWS), or the set of wrenches that can be applied to the object by a set of gripper jaws. By computing the GWS, we can determine which wrenches the grasp can resist. In previous work, area contact models made up of multiple points or regions have been considered, but these either assume a planar contact area [85, 41, 40] or can be inefficient for fine surface geometries [50]. In this chapter, we consider non-planar soft area contacts from a compliant gripper and formulate constraints for the wrenches that can be applied at each contact, as shown in Figure 3.1. We extend the 6D ellipsoidal model proposed by Xu et al. [270] by more efficiently sampling wrenches on the 6D *friction limit surface (FLS)* without Finite Element Analysis (FEA) and combining the 6D FLS with the

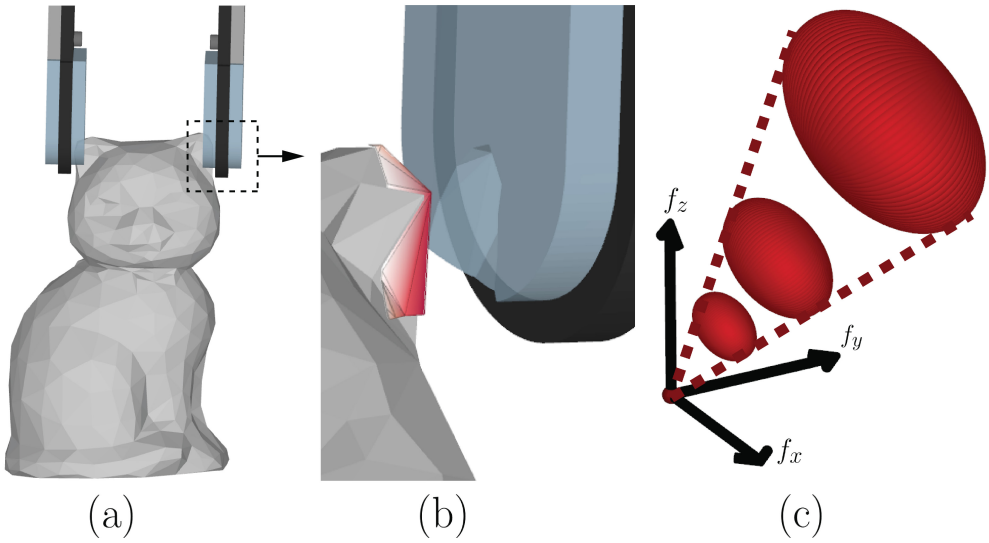


Figure 3.1: (a) A non-planar area contact is created when a compliant gripper jaw surface contacts a non-planar object's surface. (b) An enlarged view of the deformed jaw and the contact profile obtained by the REACH model [50]. The non-planar contact area consists of triangles and the redder colors represent higher pressure due to larger deformation of the jaw pad at that point. (c) A projection of the 6D friction cone that constrains the wrenches that can be applied at the contact. Each ellipsoid represents a projection of the friction limit surface for a given gripper closing force and its center corresponds to the wrench created by the contact pressure.

normal wrench imposed at the contact. We present a novel 6D friction cone (6DFC) that fully constrains the normal and frictional wrenches that can be applied at a contact by varying the grasp force. We then show how a linearized GWS can be formed from the 6D friction cones at each contact and can be directly used to evaluate grasp quality as a quadratic program. We believe that the 6DFC is particularly relevant to the family of compliant grippers both for grasp planning and for grasp robustness analysis when the exact grasp force is unknown. 6DFC subsumes both the friction cone created from planar contacts from rigid grippers and grasp analysis with known closing forces.

This chapter provides three contributions:

1. A generalization of the 3D friction cone to a 6D friction cone for grasp reliability computation.

2. The 6DFC sampling algorithm for efficiently constructing the 6D friction limit surface and 6D friction cone for a non-planar area contact.

3. Results comparing 1500 physical grasps of 12 3D printed non-planar objects on an ABB YuMi robot with predictions from 4 algorithms that suggest the 6DFC algorithm

can decrease false negatives by 17% over soft point contacts and 6% over a previously proposed area contact model.

3.1 Related Work

3.1.1 Contact Models

The contact between a robot gripper jaw and an object can be described as the jaw exerting a 6D wrench on the object with 3D force and 3D torque components, expressed in an object reference frame.

Among the many models introduced [222, 20, 114, 215], the most common models used in practice are point contact models with friction or soft point contact models [164, 12]. Under the Coulomb friction model, these contacts can exert forces in the plane tangent to the contact surface and a torsional moment (for soft point contacts) about the contact normal [97, 113, 115, 41, 40]. The tangential force and torsional moment that a soft point contact can exert can also be jointly constrained by the FLS [76, 147, 186], which Howe, Kao, and Cutkosky [97] approximated with an ellipsoid for computational efficiency.

Planar area contact models are used to construct a 3D ellipsoidal FLS [96, 245, 41]; these area contact models jointly constrain forces in the contact plane and torque about the normal, but do not consider non-planar area contacts. Xu et al. [271] analyzed a 3D subspace of 6D friction constraints for curved contact areas and generalized the 3D FLS ellipsoid to 6D to model friction for non-planar surfaces. The 6D FLS is computed by densely sampling body twists and fitting the downsampled wrenches with an ellipsoid via convex optimization [270]. We also use a sampling method to form the 6D FLS, but without densely sampling over the entire space for increased efficiency. Danielczuk et al. [50] considered non-planar soft contacts by discretizing the area contact as a triangular mesh, but the REACH model did not consider the coupling between contact triangles, instead formulating constraints for each triangle independently. This formulation also resulted in slow runtime due to the number of contact wrench constraints in the grasp optimization problem scaling with the number of triangles in the contact area.

3.1.2 Grasp Analysis

Evaluating grasp quality requires determining the grasp's ability to constrain the motion of the object by applying forces and torques at the contacts to resist external disturbances without violating the frictional constraints at each contact [19]. To evaluate this quality, many metrics have been developed based on the grasp wrench space (GWS), or the space of wrenches that the contacts can apply to the object [216]. For example, force-closure ensures the contacts can resist any external wrenches with arbitrarily high grasp forces [187, 177, 149]. While metrics analyzing the entire GWS can be useful for unknown tasks [260, 42, 216], force-closure grasps are conservative for many tasks, such as lifting an object, as they

require the grasp to be able to resist wrenches that will not be applied to the object during the task.

For unknown or complex tasks, formulating the 6D Task Wrench Space (TWS) ellipsoid can be complicated [149], but for simple tasks such as lifting an object, the task wrench space can be formulated as the wrenches applied to the surface of the object (object wrench space) or its center of mass (mass wrench space) that must be resisted [237, 23, 88, 153]. Mahler et al. used a mass wrench space of the gravity wrench, corresponding to a grasp's ability to lift and hold the object [164, 165]. We use the same metric as Mahler et al. in that we characterize a grasp as successful if the grasp can resist the gravity wrench.

3.1.3 Grasp Wrench Space Formulation

A common approximation of the GWS is to find the convex hull of the union or Minkowski sum of the discretized friction cones at each contact [177, 23, 66]. Krug, Bekiroglu, and Roa [132] noted that the independent contact bounds via the Minkowski sum more accurately represent fully-actuated grippers than that of the sum-bounded union. However, for soft point contact models or area contact models that consider a 3D friction limit surface, only the union or Minkowski sum of the ellipsoids created by the maximal normal force is considered (e.g., the ellipsoids generated by normal forces between zero and the maximal normal force are ignored) [132, 40, 42]. In contrast, we formulate the GWS using independent contact bounds and the full 6D friction limit surface ellipsoid for each normal force that can be applied at the contact. The 6D friction cone captures both normal wrench and frictional wrench constraints.

3.1.4 Contact Wrench Cones

In addition to grasping applications, contact models are also commonly used to form contact wrench cones in legged robotics. The contact wrench cone (CWC) describes wrenches acting on the center of mass of a humanoid robot, determined by the friction cones at discrete contact points where the robot contacts the ground [11, 209]. Caron, Pham, and Nakamura [30] built the 6D CWC for non-coplanar contacts using surface contact wrenches. Carpentier and Mansard [31] approximated the CWC with a 6D cone. While the algorithms in [30, 31] consider 3D wrench constraints at each point or planar area contact, the proposed 6DFC models 6D wrench constraints for each non-planar area contact from compliant grippers.

3.2 Problem Statement

We consider the problem of predicting grasp reliability, or probability of grasp success, by building a 6D friction cone of each non-planar area contact from compliant jaws.

3.2.1 Assumptions

We make the following assumptions:

1. Quasi-static physics (inertial terms are negligible) and Coulomb friction with constant friction coefficient μ over the contact area.
2. Objects to be grasped are rigid with known geometry.
3. The gripper has known geometry and two parallel jaws, each with a linear-elastic material at the tips.
4. Both gripper jaws make contact simultaneously.
5. Force is applied normally to the object surface at each point within the contact area.

3.2.2 Definitions

We define a state $\mathbf{x}$ that contains a single object $\mathcal{O}$ (including its geometric, material, and frictional properties) and its pose $T_{\mathcal{O}}$. We also define a parallel-jaw grasp action $\mathbf{u}$ parametrized by a nominal grasp center $\mathbf{p} \in \mathbb{R}^3$ and an angle $\phi \in \mathcal{S}^3$. The jaws close with force of magnitude f_C around the grasp center and are oriented according to the grasp angle. A binary reward function R describes the grasp success, where $R = 1$ if the grasp lifts the object (meaning the contacts resist the wrench applied to the object by gravity) and $R = 0$ otherwise. To account for uncertainty in the state as well as imprecision in control of the robot, we consider a grasp reliability distribution $Q(\mathbf{x}, \mathbf{u}) = \mathbb{P}(R \,|\, \mathbf{x}, \mathbf{u})$ that describes the probability of grasp success for a state $\mathbf{x}$ and action $\mathbf{u}$ [165]. We evaluate reliability in simulated environments by perturbing object pose, mass, and frictional properties and in physical experiments by repeating the same nominal grasp multiple times under uncertainty in the robot grasp pose accuracy and object registration. We approximate $Q(\mathbf{x}, \mathbf{u})$ with the sample mean of N Monte Carlo samples: $Q(\mathbf{x}, \mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} R_i(\mathbf{x}, \mathbf{u})$ [163].

3.2.3 Objective

We evaluate the accuracy on a dataset of physical grasp experiments. Specifically, we seek to maximize average precision (AP), defined as the mean of precision values at each recall threshold weighted by the difference in recall thresholds, which measures the area under the precision-recall curve. Additionally, we seek to maximize the average recall (AR) on the dataset for a given AP, as this metric indicates the ability to correctly predict positive grasps. Both metrics measure binary classification performance and are commonly used in computer vision for unbalanced datasets [64]. A combination of high AP and AR indicates that the algorithm predicts few false positives while also predicting few false negatives. We select AP and AR as metrics to evaluate the algorithm in reducing false positives and negatives.

3.3 Non-Planar Area Contact Constraints

3.3.1 Background

We define the 6D wrench that can be applied at the contact as $\mathbf{w} = \begin{bmatrix} f_x & f_y & f_z & \tau_x & \tau_y & \tau_z \end{bmatrix}^\top$, which corresponds to a force and torque that can be applied about each axis defined in the contact frame. The set of these contact wrenches (e.g., all wrenches that can be applied at the contact) forms the *contact wrench space (CWS)*. Additionally, we define a maximum closing force $f_{C,max} \in \mathbb{R}_+$ with which the jaw can contact the object such that $0 \leq f_C \leq f_{C,max}$.

For frictional point contacts using Coloumb friction, we can define the axis for the contact such that the z-axis is aligned with the negative surface normal and the x and y axes are in the plane. Then, $f_z = f_C$ and the CWS is defined by [186]:

$$\mathcal{W} = \left\{ \mathbf{w} \in \mathbb{R}^6 \mid 0 \leq f_z \leq f_{C,max},\ \sqrt{f_x^2 + f_y^2} \leq \mu f_z, \tau_x = \tau_y = \tau_z = 0 \right\} \tag{3.3.1}$$

Equation 3.3.1 limits the forces that can be applied at the contact with a 3D friction cone, as shown in Figure 3.2(a), where the tangential forces f_x and f_y that can be applied increase with f_z.

This idea can be extended to the case of soft point contacts, where the jaw can also apply a torque τ_z around the z-axis. In this case, the torque τ_z and the tangential forces f_x and f_y are jointly constrained by the so-called friction limit surface (FLS) [76, 147, 186], which can be approximated as a 3D ellipsoid [97]:

$$\mathcal{W} = \left\{ \mathbf{w} \in \mathbb{R}^6 \;\middle|\; \frac{f_x^2 + f_y^2}{\mu^2} + \frac{\tau_z^2}{\gamma^2} \leq f_{C,max},\ \tau_x = \tau_y = 0 \right\}$$

In previous work, the largest ellipsoid is used to constrain the wrench applied at the soft point contact [40, 132]. Thus, the CWS is a 3D ellipsoid. We note that this formulation can easily be expressed in the same way as the previous case by replacing $f_{C,max}$ with f_z and independently constraining $0 \leq f_z \leq f_{C,max}$, resulting in a 4D cone that extends along the f_z axis and 3D ellipsoidal cross sections at each value of f_C as shown in Figure 3.2(b).

In both cases, the CWS can be discretized into k samples:

$$\boldsymbol{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k]$$

and the GWS for n contacts is formed either through the convex hull of the union or Minkowski sum of the n discretized contact wrench spaces [66].

3.3.2 Friction Cones in 6D

In this section, we generalize the friction cone to the 6D space. When considering a non-planar area contact, the friction wrenches that can be applied at the contact and the wrench impressed by the normal pressure, defined as the normal wrench, are 6D, as the force and

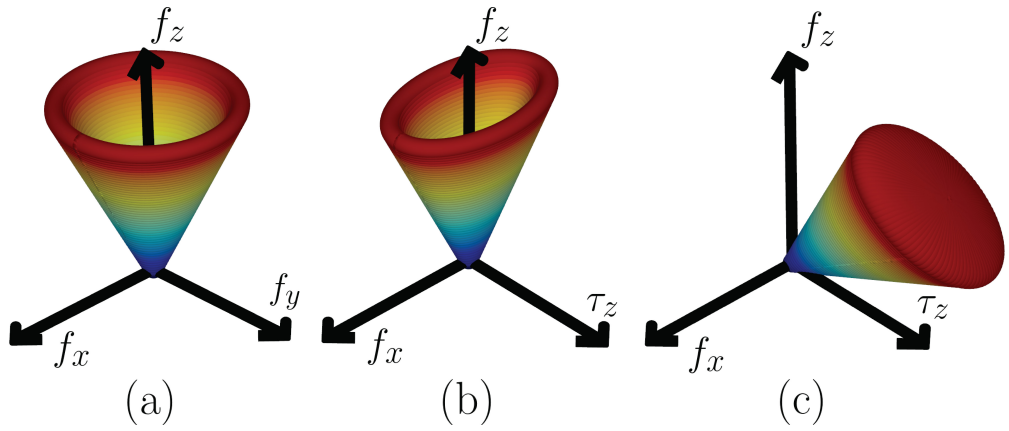


Figure 3.2: In each image, warmer colors indicate increasing gripper closing force magnitude f_C. (a) For frictional point contacts, the friction cone is circular and extends along the f_z axis. (b) For soft point contacts, the cone is 4D and elliptical; f_x, f_y and τ_z are jointly constrained at each value of f_z by an ellipsoid approximating the friction limit surface. Here we show a projection of the 4D cone. (c) In the non-planar area contact case, the cone is 6D and may not align with any axis, resulting in non-axis-aligned 6D ellipsoids approximating the friction limit surface at each value of the closing force. Here we show a projection of the 6D cone.

torque are both 3D. Xu et al. suggest that the friction wrenches are bounded with a 6D ellipsoid centered at the origin [270]. We propose that the total wrench applied at a contact with a grasp force f_C can be modeled with a 6D ellipsoid that is centered at the 6D normal wrench, so that varying the value of f_C results in a *6D friction cone*, whose center lies along the vector $\mathbf{f_N} \in \mathbb{R}^6$ and has 6D frictional ellipsoids as contours for each value of f_C, similar to the one shown in Figure 3.1. We can express this cone as:

$$(\mathbf{w} - f_C\mathbf{f_N})^\top A\,(\mathbf{w} - f_C\mathbf{f_N}) \leq f_C^2,\ 0 \leq f_C \leq f_{C,max} \tag{3.3.2}$$

Figure 3.2(c) shows a 3D projection of the 6D friction cone that is produced from a non-planar area contact.

3.3.3 Finding 6D Friction Limit Surface Cone Constraints

For the 3D frictional point contact or 4D soft point contact case discussed above, the ellipsoid that approximates the 2D or 3D friction limit surface can be easily found, since it is axis-aligned. Then, by finding the maximum values for f_x, f_y, and τ_z (in the soft point contact case), we can construct the ellipsoid. However, in the 6D case, the ellipsoid that approximates the friction limit surface may be rotated, since when one friction wrench component reaches its maximum value, the other dimensions might not be zero. This scenario can occur from an asymmetric pressure distribution over the contact area or from the geometry of the contact area itself [96]. Thus, we find an equation that describes the ellipsoid by explicitly sampling

its surface, then fitting an ellipsoid to the sampled wrenches. We describe this process in detail in the following sections.

Computation of Friction Wrench Samples

First, we extract the contact area patch on the object using the same method as in Danielczuk et al. [50]: the constructive solid geometry intersection between the gripper pad at its maximum depth of deformation and the object is the contact area patch. This resulting contact patch can be discretized into m triangles. However, unlike Danielczuk et al. [50], which treats each triangle in the discretized patch as a separate planar contact, we find the 6D friction limit surface for the entire patch.

To find the 6D friction limit surface for the patch contact, we sample points that lie on the surface, similar to Xu et al. [270]. However, unlike Xu et al. [270], we do not evenly sample axes of rotation at various distances from the contact, since finding extreme points on the surface requires sampling axes at both small distances from the contact (to maximize torques) and at infinite distances (to maximize forces) [96]. Thus, we sample wrenches with large torques or large forces separately.

First, we sample k_1 tuples consisting of a unit axis of rotation $\boldsymbol{\omega}_i$ and a 3D center point $\mathbf{c}_i$, where $\boldsymbol{\omega}_i$ is sampled uniformly from the unit sphere and $\mathbf{c}_i$ is sampled randomly within a radius r of the pressure center of the contact patch. For each tuple, we find the instantaneous unit velocity vector in the plane normal to $\boldsymbol{\omega}_i$ at each triangle. We project the unit velocity vector onto the surface of each triangle to find the projected velocity vector $\hat{\mathbf{v}}_{i,t}$ at the t-th triangle in the contact area patch. For wrenches with maximum forces, we uniformly sample k_2 unit velocities $\mathbf{v}_i$, similar to sampling $\boldsymbol{\omega}_i$, and project them onto each triangle plane to find $\hat{\mathbf{v}}_{i,t}$.

Then, we can calculate the magnitude of force that can be resisted in the triangle plane, depending on the force applied normal to that triangle, denoted as f_{N_t}. The force that can be applied in the frame of the triangle is $-\mu f_{N_t} \hat{\mathbf{v}}_{i,t}$. By transforming each of these forces into the contact patch frame using the triangle's adjoint matrix Ad_t, we can then form the full 6D wrench that can be applied at the contact:

$$\mathbf{w}_i = -\sum_{t=1}^{m} \mu Ad_t f_{N_t} \hat{\mathbf{v}}_{i,t}$$

Fitting the 6D Ellipsoid

Given the k samples $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$, we then fit a 6D ellipsoid to the data using linear least squares. To fit the ellipsoid, we find the positive semidefinite matrix A^*:

$$A^* = \arg\min_A \sum_{i=1}^{k} \|\mathbf{w}_i^\top A \mathbf{w}_i - 1\|_2^2$$

We can solve this equation exactly using least squares since it is linear in A. Note that this method of solving for A^* is not guaranteed to result in an ellipse if the matrix is not positive semidefinite (in fact, a hyperboloid could also be returned), so in practice, we first determine the dimensionality of the data by using principal component analysis (e.g., if the contact area is planar, then the ellipsoid will only be 3D), then fit an ellipsoid of the determined dimensionality to the data rotated to the PCA frame. We fill out the remaining dimensions with axes lengths of a small value $\epsilon > 0$, then rotate the fitted 6D ellipsoid back to the original non-PCA frame. Thus, this case subsumes the soft point contact case above; if the contact is planar, it reduces to the case where the normal wrench is along f_z and the FLS is 3D.

Linearizing the Ellipsoidal Constraints

Although the ellipsoidal constraint is itself a convex constraint, quadratic programs with linear constraints can be solved more efficiently. We formulate a quadratic program to determine if a grasp resists gravity. We approximate the ellipsoid with a set of linear constraints for computational efficiency.

To find these constraints, we first resample the ellipsoid. For many contacts, the initial random sampling process results in data that is not evenly sampled due to the geometry of the contact surfaces; thus, we resample points evenly over the surface of the ellipsoid to better approximate it. Then, for each resampled point $\mathbf{x}_i$ on the ellipsoid, its outward-facing normal vector is given by $A^*\mathbf{x}_i$. This normal vector defines the hyperplane tangent to the ellipsoid at $\mathbf{x}_i$; by finding many of these hyperplanes, we can construct a set of linear constraints that approximate the ellipsoid. The constraint set is of the form:

$$\mathcal{F} = \left\{\mathbf{z} \in \mathbb{R}^6 : \mathbf{z}^\top A^* \mathbf{x}_i \leq \mathbf{x}_i^\top A^* \mathbf{x}_i = 1, \forall i\right\}$$

Formulating Cone Constraints

Once the constraints are found for the unit closing force, we can shift the planar constraints along the normal force vector and scale them by an arbitrary closing force:

$$\mathcal{F} = \begin{Bmatrix} \mathbf{z} \in \mathbb{R}^6 : \mathbf{z}^\top A^* \mathbf{x}_i - f_C(1 + \mathbf{f_N} A^* \mathbf{x}_i) \leq 0, \ \forall i \\ f_C \in \mathbb{R} : 0 \leq f_C \leq f_{C,max} \end{Bmatrix}$$

Note that although the ellipsoidal constraint in Equation 3.3.2 is quadratic, by first approximating the unit closing force ellipse with a set of planes, we can take advantage of the fact that the ellipse grows linearly with increasing closing force to express the planar constraints linearly with f_C. This set of linear constraints can be used directly in a quadratic program to determine the ability of contacts that apply wrenches $\tilde{\mathbf{z}} = \begin{bmatrix}\mathbf{z}_1 & f_{C,1} & \dots & \mathbf{z}_n & f_{C,n}\end{bmatrix}^\top$ to resist the gravity wrench $\mathbf{t}$:

$$\min_{\tilde{\mathbf{z}}} \ \|G\tilde{\mathbf{z}} + \mathbf{t}\|_2^2 \quad \text{s.t.} \quad F\tilde{\mathbf{z}} \leq \mathbf{h} \tag{3.3.3}$$

Here, F and $\mathbf{h}$ are generated by concatenating the constraints $\mathcal{F}$ for each contact. The matrix $G \in \mathbb{R}^{6\times 7n}$ transforms $\tilde{\mathbf{z}}$ from the contact frames to the object frame.

3.4 Experiments

We evaluate the 6DFC algorithm proposed in Section 3.3.3 against three baseline algorithms that generate frictional and normal wrench constraints for each contact to determine which most accurately can predict grasp success.

3.4.1 Baseline Algorithms

Soft Point Contact

As described in Section 3.3.1, this algorithm constrains the wrench applied at the contact through a 3D ellipsoid representing the friction limit surface corresponding to the maximum normal force that can be applied at the contact. We linearize these constraints to solve the quadratic program in Equation 3.3.3.

REACH

The area contact model proposed in Chapter 2 discretizes the contact area into a triangular mesh and develops constraints of a similar form to the soft point contact model for each triangle. It can be modified to consider only a maximum number of triangles for each contact (e.g., the 10 largest triangles), which increases computational efficiency without significantly reducing accuracy.

Maximum Force Ellipsoid (MFE)

This algorithm constructs the 6D FLS ellipsoid as described in Section 3.3.3, but constrains the wrench applied at the contact to the ellipsoid generated by the maximum closing force. This method is similar to that proposed by Xu et al. [270].

3.4.2 Soft Non-Planar Area-Contact Physical Robot Grasps

To evaluate the precision and recall of 6DFC and the baseline algorithms, we use a subset of 1,500 grasps on 12 3D-printed objects from the Soft Area-Contact Physical Robot Grasp Dataset, collected on a physical ABB YuMi robot with a compliant parallel-jaw gripper [50]. The exerted forces are not actively controlled, as the gripper does not have force sensors. The subset of objects chosen from the dataset have non-planar contacts for all grasps. Note that the assumptions in Section 3.2 do not necessarily hold for the physical grasps that the algorithms are tested on (e.g., the jaws do not always make contact simultaneously, quasi-static assumptions do not hold).

Algorithm	**Precision**	**Recall**	**Runtime** (ms/grasp)
Point	0.80 ± 0.01	0.50 ± 0.01	$\mathbf{13.0 \pm 2.7}$
REACH	$\mathbf{0.83 \pm 0.01}$	0.61 ± 0.01	207.1 ± 35.3
MFE	$\mathbf{0.83 \pm 0.02}$	0.60 ± 0.01	247.9 ± 9.6
6DFC	0.82 ± 0.01	$\mathbf{0.67 \pm 0.01}$	251.6 ± 14.2

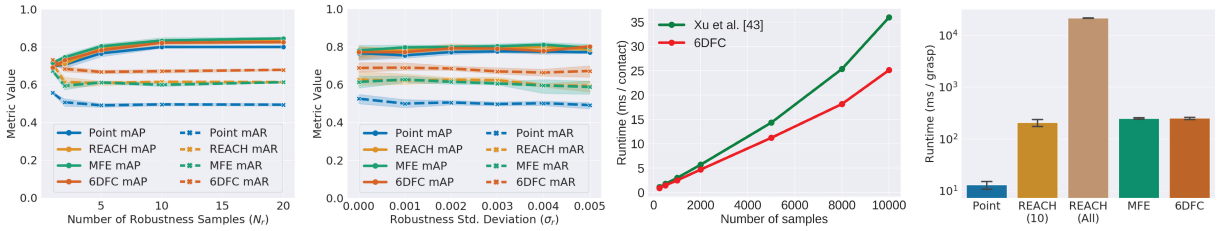
Table 3.1: Mean average precision (mAP) and mean average recall (mAR) and their standard deviations for each algorithm's predictions of grasp quality, for 5 runs of the 1,500 grasps collected on the physical robot. 6DFC outperforms the point, REACH, and MFE algorithms by 17%, 6%, and 7% in mAR, respectively, suggesting that cone constraints can reduce false negatives on objects with non-planar surfaces.

We measure both average precision (AP) and average recall (AR) for each object using the dataset's ground-truth physical grasp labels and each algorithm's predictions. We consider a physical grasp to be successful if it lifts and transports an object from a bin to a receptacle. We measure performance with mean average precision (mAP) and mean average recall (mAR), which are the AP and AR of the algorithm averaged over all objects to account for discrepancies in the number of successful grasps for each object. We also measure runtime per grasp computation for each model on the an Ubuntu 16.04 machine with a 12-core 3.7 GHz i7-8700k processor. For each algorithm, parameters such as the friction coefficient, elasticity coefficient, and robustness sample standard deviation were chosen using leave-one-out cross validation.

The results for each algorithm are shown in Table 3.1. These results suggest that formulating the 6DFC algorithm can increase the number of successful grasps on the physical system that can be recalled by as much as 17% over existing algorithms while maintaining a similar number of false positives predicted. We find that grasps found only by 6DFC are often of the kind shown in Figure 3.1, where the contact area is small with high surface curvature, and the grasp may include slight dynamic effects such as a bowing of the jaws. We hypothesize that allowing for reduced closing forces in this scenario could more accurately model the changing contact profile that occurs.

3.4.3 Grasp Planning Results

We also evaluate the reliability of each algorithm as part of a grasp planning policy via a second experiment. We hypothesize that our algorithm can find grasps in scenarios where a point contact algorithm would not return any high-quality grasps due to predicted collisions with other objects, environmental constraints, or motion-planning constraints. To test this hypothesis, we place each of the 12 objects in their 3 most probable stable poses [73] and attempt the top 3 grasps that each algorithm labels as the highest reliability grasps for that stable pose. We remove nearby grasps so that the algorithm cannot choose 3 similar grasps,



(a) (b) (c) (d)

Figure 3.3: Mean average precision (mAP) and mean average recall (mAR) for each algorithm as a function of (a) the number of robustness samples N_r, (b) the robustness standard deviation σ_r with error bars showing the standard deviation of 5 runs of each algorithm with the given parameter for the dataset of 1500 physical grasps. Adding robustness samples and spreading the samples increases mAP up to 10 samples and $\sigma_r = 0.003$ m. After this point, mAR continues to decline, but mAP remains constant or decreases (in the case of σ_r). (c) Runtime analysis of the sampling algorithms with different numbers of wrenches, averaged over 1000 runs. 6DFC is up to 30% faster than [270]. Timing analysis for each algorithm is shown in (d), averaged over 6000 grasps. The numbers below the REACH algorithm indicate the number of triangles considered in computation. The 6DFC algorithm is of similar time to the other algorithms that analyze area contacts, but is 85x faster when analyzing the same number of triangles as REACH.

simulating other objects blocking the grasp from being executed. If multiple grasps are labeled with the same probability, one of them is chosen at random.

We evaluated 875 unique grasps in total for 32 stable poses of the 12 objects. Both algorithms found successful grasps in at least one stable pose that the other could not (Figure 3.4 shows two examples), but overall the point grasps surprisingly succeeded more often (339 successes compared to 272 for the 6DFC algorithm). This result may be due to an inflation in grasp quality in the 6DFC algorithm; we found that many grasps that had large contact areas were rated highly but did not succeed, as shown in the bottom row of Figure 3.4. We will investigate this discrepancy further in future work.

3.4.4 Sensitivity Analysis

As each of the algorithms contains several parameters such as friction and elasticity coefficients, robustness parameters, we include an analysis of each algorithm's sensitivity to a subset of these parameters.

Effect of Robustness Parameters

All of the algorithms benefit from "robustness", or sampling grasp poses around the nominal pose and averaging the predicted grasp qualities. We sample a random 3D translation from a zero-mean Gaussian with variance σ_r^2 and a uniformly random 3D rotation with angle θ

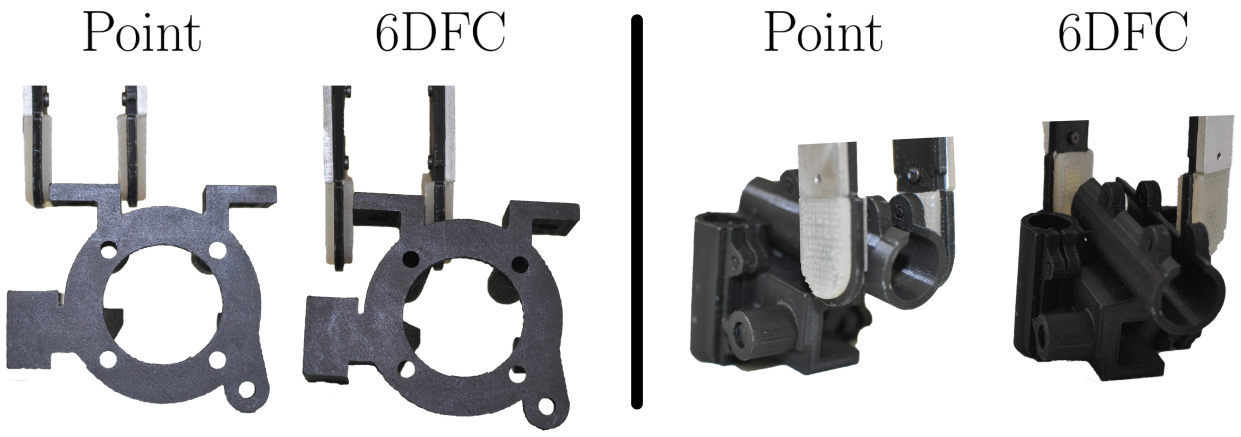


Figure 3.4: Left: An example where the 6DFC algorithm finds a robust grasp when the Point algorithm does not. Right: An example where the Point algorithm finds a robust grasp when the 6DFC algorithm does not. In the first case, the thin part of the object results in a low-quality prediction for the point contact algorithm, whereas in the second case, the large contact area produces a false positive prediction from the 6DFC algorithm.

proportional to σ_r^2 and apply them to the nominal grasp. Figure 3.3(a-b) shows sensitivity of the algorithms to the number of samples N_r and the standard deviation σ_r. At $N_r = 10$ and $\sigma_r = 0.003$, mAP and mAR are maximized. Adding more samples after this point does not increase mAR and may moderately increase mAP but does result in the standard deviation (shaded area) decreasing. Increasing σ_r results in lower mAR as sampled grasps no longer resemble the nominal grasp.

Effect of Number of Samples and Contact Triangles

Figure 3.3(c) shows the sampling runtime with different number of wrenches and suggests that the proposed algorithm described in Section 3.3.3 is up to 30% faster than [270]. The runtime of REACH strongly depends on the number of triangles in the area contact analyzed. The 6DFC algorithm runtime scales with the number of contacts as opposed to the number of triangles in each contact, resulting in the 85x faster runtime on the same surface patch mesh, as shown in Figure 3.3(d).

3.5 Discussion and Future Work

This chapter introduced 6DFC, an algorithm that generalizes 3D friction cones to non-planar soft area contacts, constraining both the contact normal and frictional wrenches. We sample the 6D friction cone using projections of instantaneous velocity vectors onto each triangle of the surface patch mesh. We show 6DFC outperforms baseline point contact and area contact

models on a dataset of physical grasps. We note that one reason for predicted false positives is that the 6DFC algorithm allows different contact forces of each jaw, which is not feasible with the current physical setup. In future work, we plan to evaluate 6DFC with a three-jaw gripper that allows fully controllable forces to further investigate this effect. We also plan to relax the simultaneous jaw contact assumption as part of a dynamic contact analysis and do a thorough comparison to recent work on fully-dynamic simulation of rigid and soft-body contacts using the IPC and Flex simulators [123, 62, 100].

Chapter 4

Linear Push Policies to Increase Grasp Access

While the previous two chapters have focused on improving parallel-jaw grasping, this chapter instead focuses on another useful manipulation primitive: pushing. Pushing is a vital primitive in both overhead-access and lateral-access environments, especially in cases where objects are large, heavy, or must be rotated or translated in order to be grasped. Pushing can also be an efficient action to take in the context of mechanical search since it can move multiple occluding objects away from the target object. Thus, it is important to consider policies that couple grasping actions with synergistic pushing actions that enable grasps at proceeding time steps. This chapter explores heuristic linear pushing policies in bin picking environments and measures pushing policy success by directly computing a difference of predicted grasp qualities before and after the pushing action is made, unlike previous work, which typically attempts to maximize object separation as a proxy for future grasp success rates.

Deep learning methods can enable robots to grasp objects in isolation on a flat workspace, and these methods have recently been extended to grasping in clutter [106, 141, 162, 164]. Perception in cluttered bin environments remains a difficult problem, and scenarios may arise where the robot is unable to execute a collision-free grasp due to object proximity to the bin walls or to other objects [79, 170]. Pushing can change the position or orientation of parts so that a grasp with a higher probability of success can be executed [34, 74, 109]. It may not be necessary to completely separate objects for robust grasps to become available, and a series of pushes may be inefficient. Previous work measures the success of pushes as the degree of separation between objects and attempts to minimize the number of push actions to achieve separation [34, 60]. We explore directly using grasp confidence metrics instead of an indirect metric such as object separation for comparison of push policies, and attempt to maximize grasp confidence over a single push action.

In this chapter, we explore pushing policies and analyze performance in bin picking environments. We simulate grasping and pushing policies over a set of 3D CAD models as sample objects using robust quasi-static analysis and the Bullet physics engine. By dropping

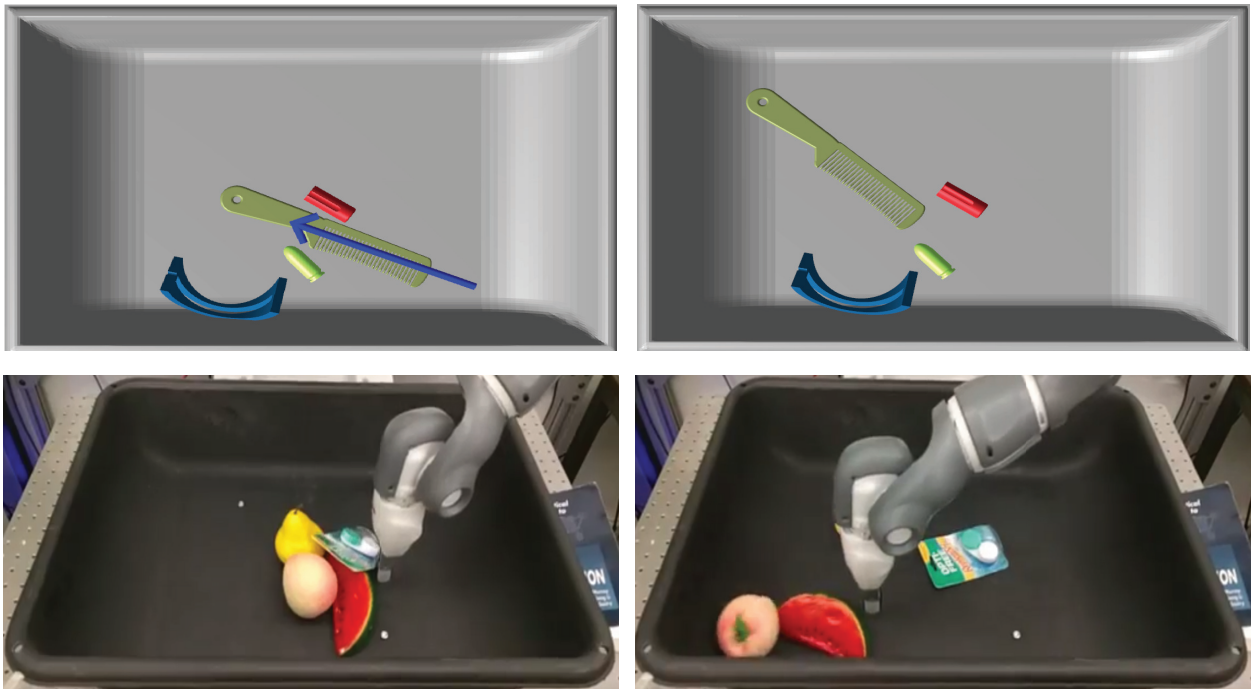


Figure 4.1: Before (left) and after (right) images of successful pushes in simulation (top) and in physical experiments with the ABB YuMi (bottom).

objects into a virtual bin and executing sequences of grasping actions, we generate a dataset of over 1,000 simulated scenarios in which pushing could potentially be useful. For each of these scenarios, we evaluate five pushing policies against a baseline policy using metrics that quantify changes in grasp availability.

This chapter makes three contributions:

1. Metrics to measure effectiveness of pushing actions,
2. Two novel push policies based on targeting free space and diffusing clusters,
3. Experimental data from 1,000 simulated and physical bin scenarios where all initial state grasps generated by Dex-Net 2.0 and 3.0 have low confidence [164, 165].

4.1 Related Work

Mason pioneered research on analytic models of push mechanics [171]. Lynch and Akella described the mechanics of stable pushing and described how to create a plan for controlled pushing of an object through a series of obstacles [4, 158, 159, 160]. Recently, both analytic

and learning techniques have been applied to learn physical pushing models for objects so they can be guided to a target location [2, 3, 16, 102, 131]. Goldberg and Brost showed that pushing objects to a location is desirable because it allows for objects to be grasped [26, 75]. Dogar and Srinivasa built on the idea of "push-grasping" to reduce uncertainty in grasping objects in clutter, using predefined 3D models of the objects to be grasped and attempting to estimate pose of each object in the scene before planning an action [54, 55]. Kehoe, Berenson, and Goldberg [119] extended these ideas to model uncertainty in the shape of the object. In the Amazon Picking Challenge, the Technische Universitat Berlin team used pushing to enhance suction [63]. Team MIT used "topple" and "push-rotate" primitives to aid in grasping and bin-clearing [278]. These primitives were designed for a specific scenario in the competition, where some objects were initially ungraspable.

Another class of related work focuses on pushing heaps of objects for interactive segmentation. Hermans, Rehg, and Bobick [92] reason about boundaries between objects and keep a "push history" to estimate likelihoods that each cluster in an image is a separate object. They plan pushes along the object boundaries to attempt to force two nearby objects apart. While they score potential pushes on workspace boundaries and likelihood of collision with other clusters, their analysis does not account for scenarios where objects may be lying on top of one another. Chang, Smith, and Fox [34] also consider interactive singulation of objects in clutter, pushing clusters of objects away from other clusters so that they can be grasped. They track clusters between actions to gain evidence of singulation and adapt their pushing strategy based on the results of the previous pushes. However, their approach relies on object tracking, which may be sensitive to sensor noise. Several other groups study pushing as a means of interactive perception or segmentation, where actions are used to inform segmentation of objects from the background [116, 121, 146]. These use fixed-length pushes to distinguish objects from each other or the background, but do not use pushing to aid in future grasping attempts. In all of these cases, multiple pushing actions are necessary to confirm singulation of the object. We seek to extend these previous pushing methods for bin picking applications by introducing a bounded workspace and new push metrics.

"Singulation" applies pushing mechanics and planning to the task of separating or extracting objects that lie close together, and it is often required for successful object recognition or grasping. Model-based approaches such as the one proposed by Cosgun et al. [45] planned a series of robot actions to clear space for a target object in two-dimensional, tabletop pushing scenarios. Without prior knowledge of the objects, it can be challenging to estimate object geometries, poses, and other physical properties of the objects and environment [243], which can affect the efficiency of pushing actions [279]. For this reason, recent work has focused on learning strategies to push objects apart. Laskey et al. have used learning from demonstrations to successfully singulate and grasp one object in clutter with continuous push trajectories [138, 139]. Omrčen et al. [190] used many examples of pushing objects to train a fully-connected two-layer neural network that predicted 3D object motion for proposed actions. They then used these predictions to push objects to the edge of a table where they could be grasped. Boularias, Bagnell, and Stentz [24] have also applied reinforcement learning to the task of pushing and grasping in clutter, extending their appli-

cation to two objects. Eitel, Hauff, and Burgard [60] explore singulation in clutter using a push proposal convolutional neural network, showing that they can separate up to 8 objects with at least a 40% success rate in an average of 11 push actions. In contrast to their work, which seeks to minimize the number of push actions to separate all objects, we find one push at each opportunity, take into account bin walls and corners, and analyze push success based on new metrics in addition to object separation distance.

4.2 Problem Statement

Given a depth image of one or more objects in a bin as input, find the push action that maximizes the probability of a successful grasp of an object from the bin. A push action is defined by a line segment at a constant height parallel to the workspace.

4.2.1 Assumptions

We assume a robot with a parallel-jaw gripper or suction end effector, and rigid objects in clutter resting in a bin. We assume quasi-static physics and soft finger point contacts. All objects considered are able to be grasped in at least one stable pose. We also assume known gripper and bin geometries and a single overhead depth camera with known intrinsics. For purposes of finding free regions in the bin and boundaries between objects, we approximate distances using object center of mass. This approximation allows for reasonable computational efficiency in simulation, as finding minimum distances between two meshes is an expensive operation. In addition, we assumed object distances to be the distance between each object's center of mass when determining which objects to push. In physical experiments we approximate the center of mass of each object to be the centroid of each object's points from the point cloud.

4.2.2 Definitions

States

Let $\mathbf{x} = (\mathcal{O}, T_{\mathcal{O}}) \in \mathcal{X}$ be the ground truth state of the heap, where $\mathcal{O}$ represents the geometries and inertial and material properties of all objects in the heap, and $T_{\mathcal{O}}$ represents the poses of the objects with respect to the camera.

Observations

Let $\mathbf{y} \in \mathcal{Y}$ be an observation of the state $\mathbf{x}$ which consists of a depth image of height H and width W taken by the overhead camera.

Actions

Let:

- $\mathbf{u}_j = (\mathbf{p}, \phi) \in \mathbb{R}^3 \times \mathcal{S}^1$ be a parallel jaw grasp defined by a center point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ between the jaws and an angle in the plane of the table $\phi \in \mathcal{S}^1$ representing the grasp axis,
- $\mathbf{u}_s = (\mathbf{p}, \phi, \theta) \in \mathbb{R}^3 \times \mathcal{S}^2$ be a suction grasp defined by a target point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and spherical coordinates $(\phi, \theta) \in \mathcal{S}^2$ representing the axis of approach,
- $\mathbf{u}_g \in \mathcal{U}_j \cup \mathcal{U}_s$ be a generic grasp action, either a suction grasp or parallel jaw grasp, and
- $\mathbf{u}_p = (\mathbf{q}, \mathbf{r}) \in \mathbb{R}^3 \times \mathbb{R}^3$ be a linear pushing action in 3D space defined by a start point $\mathbf{q} = (x, y, z)$ and an end point $\mathbf{r} = (x', y', z')$, with respect to the camera

Reward

Let the reward function $\mathcal{R} \in \{0, 1\}$ be a binary function that indicates whether an object has been successfully grasped and removed from the bin. Pushing actions receive a reward of 0.

Grasp Success Distribution

Let $p(\mathcal{R}|\mathbf{u}, \mathbf{x})$ be a grasp success distribution that models the ability of a suction or parallel jaw grasp to resist gravitational wrenches under uncertainty in sensing, control, and disturbing wrenches [164, 203].

Grasp Quality Function

Let the grasp quality function $Q(\mathbf{u}, \mathbf{x}) = \mathbb{E}[\mathcal{R}|\mathbf{u}, \mathbf{x}]$ be a function that takes as input a parallel jaw grasp $\mathbf{u}_j$ or suction grasp $\mathbf{u}_s$ and the state $\mathbf{x}$ and evaluates the probability of success for that action given the current state. The grasp quality function Q is a continuous function on the interval $[0, 1]$.

Composite Policy

A composite policy $\pi(\mathbf{x})$ receives as input a state $\mathbf{x}$ and returns either a grasp action $\mathbf{u}_g$ or a push action $\mathbf{u}_p$.

4.2.3 Objective

The high-level goal is to maximize mean picks per hour (MPPH), defined as:

$$\mathbb{E}[\rho] = \mathbb{E}\left[\frac{\sum_{i=0}^{N} \mathcal{R}(\pi(\mathbf{x}_i), \mathbf{x}_i)}{\sum_{i=0}^{N} \Delta(\pi(\mathbf{x}_i))}\right] \tag{4.2.1}$$

where $\Delta(\pi(\mathbf{x}_i))$ is the time spent sensing, planning, and executing the action $\pi(\mathbf{x}_i)$. If we assume a constant execution time for each of N push or grasp actions, Equation 4.2.1 can be simplified to:

$$\mathbb{E}[\rho] = \frac{\sum_{i=0}^{N} Q(\pi(\mathbf{x}_i), \mathbf{x}_i)}{N\bar{\Delta}} \tag{4.2.2}$$

where $\bar{\Delta}$ is the average time needed to execute an action.

Multiple consecutive unsuccessful grasps increase the total number of actions N without contributing a reward $\mathcal{R}$, decreasing $\mathbb{E}[\rho]$. When grasp quality is low, successful pushing actions lead to successful grasps, contributing a higher reward over the same number of actions. We focus on the subtask of choosing a push action to maximize probability of a robust grasp being available in the next timestep in scenarios where the initial grasp quality is low.

Let the function $f : \mathcal{X} \times \mathcal{U}_p \to \mathcal{X}$ define the state transition such that $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_{p,t})$ for a pushing action $\mathbf{u}_{p,t}$ at time t. Then we wish to find:

$$\mathbf{u}^*_{p,t} = \arg\max_{\mathbf{u}_{p,t} \in \mathcal{U}_p} Q(\mathbf{u}^*_{g,t+1}, \mathbf{x}_{t+1})$$

where $\mathbf{u}^*_{g,t+1}$ is the grasp that maximizes Q for the next state $\mathbf{x}_{t+1}$.

To consider when a push action would be more efficient than a grasp action at the current timestep $t = 0$, we analyze which action $\mathbf{u}_0$ maximizes the sum of expected rewards Ψ over a two-timestep period $t = \{0, 1\}$. At time $t = 0$, we measure the binary reward for grasp actions and assign a reward of 0 for push actions. At time $t = 1$ we measure the probability of success for the best grasp action given the new state of the heap, $Q(\mathbf{u}^*_{g,1}, \mathbf{x}_1)$:

$$\Psi(\mathbf{u}_0, \mathbf{x}_0) = \mathbb{E}[\mathcal{R}(\mathbf{u}_0, \mathbf{x}_0) + \mathcal{R}(\mathbf{u}^*_{g,1}, \mathbf{x}_1)] \approx \mathcal{R}(\mathbf{u}_0, \mathbf{x}_0) + Q(\mathbf{u}^*_{g,1}, \mathbf{x}_1)$$

By comparing the sum of expected rewards for both actions at times $t = 0$ and $t = 1$, we determine which action maximizes the total reward over the given period. We formalize this as follows:

$$\Psi_p = \Psi(\mathbf{u}_p, \mathbf{x}_0) \approx Q(\mathbf{u}^*_{g,1}, \mathbf{x}_{p,1})$$
$$\Psi_g = \Psi(\mathbf{u}_{g,0}, \mathbf{x}_0) \approx \mathcal{R}(\mathbf{u}_{g,0}, \mathbf{x}_0) + Q(\mathbf{u}^*_{g,1}, \mathbf{x}_{g,1})$$

We prefer the push action when $\Psi_p > \Psi_g$.

4.3 Push Action Metrics

We define four metrics to evaluate pushing policies on the dataset of generated heaps in simulation, where we have ground truth object poses. The first, *mean object separation gain*, is a metric based on previous work [34, 60] and rewards pushing policies that create as much separation between objects as possible. The other three metrics reward pushing policies that lead to available high-quality grasps in the next timestep, measured using robust gravitational wrench resistance analysis as in [164]. These metrics quantify the change in grasp quality (probability of a successful grasp) due to a pushing action. In all cases, we approximate inter-object distance and bin-object distance using centers of mass for computational efficiency.

4.3.1 Mean Object Separation Gain

Mean Object Separation Gain measures the average degree of separation between objects in the heap [60]. Prior work has focused on minimizing the number of pushes necessary to achieve a *minimum* separation between all objects in the heap; however our goal is to maximize the expected reward at the next timestep. Highly-successful pushing actions such as totally separating a single object from a heap might not increase the minimum separation at all, as other objects in the heap may still be touching. Therefore, we use a normalized change in the *mean* object separation for our first metric, and we call this adapted version of prior object separation metrics *mean object separation gain*. For n objects, mean object separation is defined as:

$$D = \frac{\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \|o_i - o_j\|_2 + \sum_{i=0}^{n-1} \min_{b_j \in \mathcal{B}} \|o_i - b_j\|_2}{\binom{n}{2} + n}$$

where o is a vector of object centroids, and $\mathcal{B}$ is the set of bin edges. D is an average of the pairwise distances between objects and each object's distance to the bin. Thus, mean object separation gain is defined as:

$$\Delta D = \frac{D_1 - D_0}{\max(D_0, D_1)}$$

where D_0 is the initial mean object separation and D_1 is the mean object separation after the push action. We normalize this quantity by the larger mean object separation of the initial and final states to get a value between -1 and 1.

4.3.2 Parallel Jaw Grasp Quality Gain

Parallel jaw grasp quality gain measures the change in best available parallel jaw grasp, allowing us to differentiate grasp probabilities of success by end-effector. The parallel jaw

grasp quality is defined as follows:

$$Q_j^* = \max_{\mathbf{u}_j \in \mathcal{U}_j} Q_j(\mathbf{u}_j, \mathbf{x})$$

where $\mathcal{U}_j$ is the set of all parallel-jaw grasps available on objects in the heap. We approximate the grasp set with a finite set of 100 sampled grasps per object. Parallel-jaw grasps are sampled from approximately antipodal pairs of points on the surface of each object. Parallel jaw grasp quality gain is then defined as:

$$\Delta Q_j^* = Q_{j,1}^* - Q_{j,0}^*$$

where $Q_{j,0}^*$ is the initial parallel jaw grasp quality and $Q_{j,1}^*$ is the parallel jaw grasp quality after the push action.

4.3.3 Suction Grasp Quality Gain

Suction grasp quality gain measures the change in best available suction grasp, again allowing us to differentiate grasp probabilities of success by end-effector. The suction grasp quality is defined as follows:

$$Q_s^* = \max_{\mathbf{u}_s \in \mathcal{U}_s} Q_s(\mathbf{u}_s, \mathbf{x})$$

where $\mathcal{U}_s$ is the set of all suction grasps available on objects in the heap. The grasp set is approximated in the same way as the parallel-jaw grasps, and suction grasps are sampled uniformly across the surface of each object with approach axes coinciding with surface normals. Suction grasp quality gain is then defined as:

$$\Delta Q_s^* = Q_{s,1}^* - Q_{s,0}^*$$

where $Q_{s,0}^*$ is the initial suction grasp quality and $Q_{s,1}^*$ is the suction grasp quality after the push action.

4.3.4 Overall Grasp Quality Gain

Overall grasp quality gain measures the change in best available grasp. Overall grasp confidence is defined as:

$$Q_o^* = \max\left(Q_j^*, Q_s^*\right)$$

Overall grasp quality gain is then defined as:

$$\Delta Q_o^* = Q_{o,1}^* - Q_{o,0}^*$$

where $Q_{o,0}^*$ is the initial overall grasp quality and $Q_{o,1}^*$ is the overall grasp quality after the push action.

4.4 Push Policies

We compare five policies, two from prior work, two novel and a baseline method. The baseline method is a quasi-random baseline policy similar to the one used by Hermans *et al.* [92]. All policies operate on both ground truth state information in simulation or the depth maps observations described in Section 4.2. For a full state input, we performed collision checking using the Flexible Collision Library and use the known object centers of mass and poses. For a depth image input, we transform the depth image to a point cloud, segment the bin and the objects using the Euclidean Clustering method described in the PointCloud Library [220], and execute an Image Clearance Checker that checks for object and gripper collisions. The object center of mass is approximated to be the centroid of the segmented cluster of points from the point cloud [142]. For the following section, references to "object" can be replaced with segmented cluster in the case of point cloud inputs. The "free space point" p_i is defined as the point that maximizes the minimum Euclidean distance from object i to the other objects in the bin and the bin walls, penalizing distance from the starting location with an L2 regularization term.

4.4.1 Quasi-Random Policy

The Quasi-Random Policy generates a linear push action using the following three steps:

1. Choose one object in the heap at random,
2. Choose a direction at random, and
3. Push for a fixed length at the center of mass toward the chosen object in the chosen direction.

The push action is clipped to the bounds of the bin so that the gripper will not collide when executing the action.

4.4.2 Boundary Shear Policy

The boundary shear policy is adapted from the pushing policy introduced in Hermans et al. in [92]. It aims to separate the two closest objects in the heap by pushing one of them along the boundary between the two objects.

1. Find the two closest objects in the heap with centers of mass c_i and c_j,
2. Construct a line $\overline{c_i c_j}$ connecting the centers of mass of the two closest objects projected to the plane of the bin bottom, and a line $\overline{c_i c_j}_\perp$ perpendicular to $\overline{c_i c_j}$ that defines the vector approximating the boundary of the two objects,
3. Generate four possible push vectors, two for each object, that extend through the centers of mass of the objects in the direction $\overline{c_i c_j}_\perp$, and

4. Choose the push direction which is closest to the direction of free space and is collision free.

4.4.3 Free Space Policy

The free space policy aims to separate the two objects in the heap with closest centers of mass by pushing one of them along a direction toward the most free space, taking into account bin walls and other objects. It generates the push action using the following steps:

1. Find the two objects in the heap with closest centers of mass c_i and c_j,
2. For each object, find the free space point p_i defined above,
3. Draw lines $\overline{c_i p_i}, \overline{c_j p_j}$ from each of the centers of mass of the two closest objects to the points p_1 and p_2, respectively, with each point projected to the plane of the bottom of the bin,
4. Generate two possible push vectors, one for each object, that extend through the centers of mass of the objects in the collision-free directions closest to $\overline{c_i p_i}$ and $\overline{c_j p_j}$, and
5. Choose from the two possible collision-free push actions based on the minimum distance from the current center of mass of object i to p_i.

4.4.4 Maximum Clearance Ratio Policy

The maximum clearance policy, defined by Chang, Smith, and Fox [34], analyzes the available space for an object to be pushed into and the cluttered area it is being pushed from.

1. Calculate clearance in front of and behind each object for 16 uniform directions spanning angles between 0 and 2π by moving the objects footprint in the given direction and checking for collisions with other objects or the bin, and
2. Choose push action that maximizes ratio of space in the forward direction to space in the backward direction and is collision free.

4.4.5 Cluster Diffusion Policy

The cluster diffusion policy groups objects into clusters based on their position. It considers pushes of objects away from their corresponding cluster centers, along the vector originating from the cluster center to the object center of mass.

1. Separate objects into clusters of one to three objects and find the centroid of each cluster m_i,

2. Define pushing vectors $\overline{m_i c_i}$ that connect center of cluster to center of mass c_i of each object in its cluster, and

3. Score each of the potential push actions as their cosine similarity with the direction of most free space for the given object, and execute the push action with the highest score.

4.5 Simulation Experiments

We generate heaps of 3D object meshes from the Thingiverse, and the KIT and 3DNet datasets in a bin. We initialize simulations by sampling over distributions of heap size, 3D object models, camera pose, and friction coefficients to get an initial state $\mathbf{x}_0$. We randomly drop objects into the bin, and repeatedly execute parallel jaw and suction grasps until the bin is cleared or the grasping policy described in [162, 164] fails n times in a row or has confidence below a threshold. If the bin is not cleared, we record the heap. We then roll out each push policy on the same set of heaps, and measure the performance of each policy using the metrics in Section 4.3. Algorithm 1 describes the process of grasping objects out of the bin and marking scenarios during the bin picking simulation when no high-quality grasp actions are available.

With the dataset of over 1000 pushing scenarios collected, each of the policies described in Section 4.4 were rolled out on the set of heaps, using PyBullet [46] for simulating gripper-object and object-object interactions in the bin, and the metrics in Section 4.3 were measured for each pushing action. We reject heaps where none of the policies (including the quasi-random policy), were able to generate positive values for Overall Grasp Confidence Gain either due to object geometry or extreme cases of object positioning in the bin (e.g. object lying directly in a corner of the bin). These heaps reduce the performance across all policies, and rejection sampling allows us to focus on cases that highlight differences between the pushing policies. The remaining 481 pushing scenarios, termed *improvable heaps* were used to compare policies to the baseline policy.

Figure 4.2 shows that all five policies studied had positive Overall Grasp Confidence Gain over the set of improvable heaps by 5 percent, even in cases where they performed poorly in terms of Mean Object Separation Gain. The Free Space and Boundary Shear policies performed the best on the improvable heaps, with average Overall Grasp Quality Gains of 17% and 18%, which outperformed the baseline by an absolute 7% and 8%, respectively. We note that the maximal clearance ratio policy performed best on the mean object separation gain metric, but the boundary shear and free space policies outperformed it by three times in overall grasp confidence gain. This result suggests grasp confidence gain is not always correlated with object separation and could better measure push action success in bin picking.

To analyze the distribution of Overall Grasp Quality Gain, we separately recorded the Suction Grasp Quality Gain and Parallel Jaw Quality Gain for each pushing action. These results can be seen in Figure 4.3 and imply that pushing affects the parallel jaw grasps more

Algorithm 1: Bin Dataset Generation

```
Sample over distributions of heap size, 3D object models, camera pose, and friction
  coefficients to create heap;
t = 0;
consecutive_failures = 0;
while consecutive_failures < max_failures && objects_in_heap > 0 do
    Randomly sample grasps over all objects in heap to create grasp set 𝒰;
    Prune 𝒰 using collision checking and proximity to other grasps;
    Find best grasps u_j* and u_s*;
    Find Q_o*;
    if Q_o* > threshold then
        Execute best grasp u*;
        if grasp_succeeded then
            consecutive_failures = 0;
        else
            increment consecutive_failures;
        Allow objects to settle;
        increment t;
    else
        Add heap to bin dataset;
```

than it affects suction grasps. Pushing actions typically move objects around the bin, but rarely topple them onto a new face or side. Suction relies on sampling grasps on the top faces of the objects; if the face does not change, then it is unlikely that the suction grasp confidence will change significantly. However, for the parallel jaws, grasp confidence depends strongly on available space around the object. Thus, pushing an object to a freer location can shift the parallel jaw grasp confidence more dramatically.

Next, we hypothesized that the policies would outperform the baseline policy on average and make higher confidence grasps available to the grasping policies at the next timestep. We used analysis of variance (ANOVA) and linear regressions to quantify the differences between policies for each metric. A one-way repeated measures ANOVA was run for each metric, and at least one policy was determined to be statistically different from the baseline policy for Mean Object Separation Gain ($F(3, 1907) = 19.01$, $p < 0.001$), Overall Grasp Confidence Gain ($F(3, 1205) = 24.81$, $p < 0.001$), and Parallel Jaw Grasp Confidence Gain ($F(3, 1205) = 21.38$, $p < 0.001$). However, none of the policies were determined to be statistically different from the baseline for the Suction Grasp Confidence Gain metric.

To further analyze the differences between policies statistically, we ran robust linear regressions over 1907 observations for each metric, controlling for differences between heaps. The results showed that the free space and boundary shear policies are statistically different

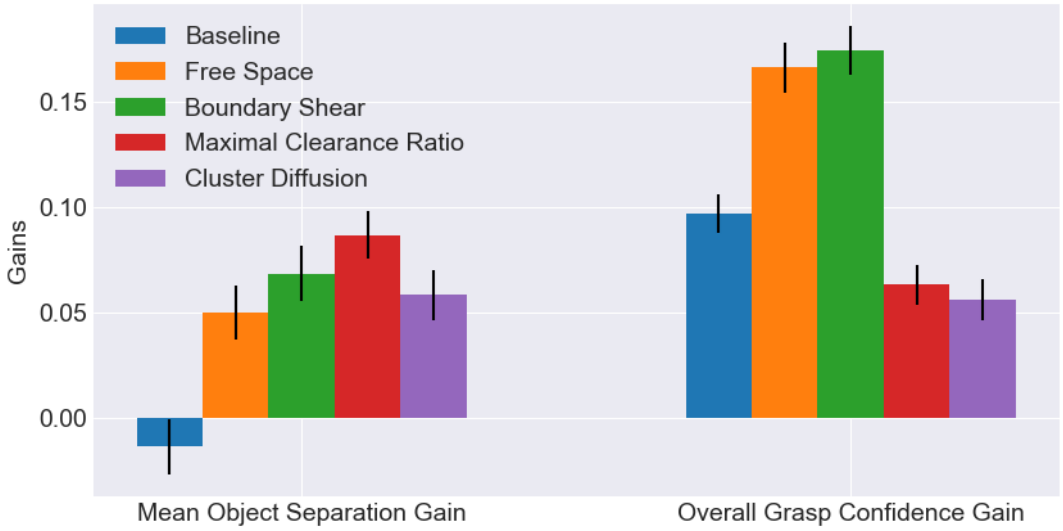


Figure 4.2: Means and standard errors of the mean for each policy and each metric. All policies have Overall Grasp Confidence Gain values above 0.1, but Mean Object Separation Gain values do not correspond to Overall Grasp Confidence Gain values, suggesting objects do not need to be separated to be grasped.

from the baseline in both Overall Grasp Confidence Gain and Parallel Jaw Confidence Gain ($p < 0.001$). Although they are statistically different from the baseline, we notice that the coefficient is about 0.1 units of grasp confidence, meaning that the baseline policy actually performs very well under many conditions. We hypothesize that the baseline policy performs well on average because it always contacts an object. Thus, it always changes the state of the heap, which can often generate grasps. Additionally, 337 of the 481 improvable heaps had fewer than four objects, so the baseline policy often planned similar actions to the other policies since it had fewer options to choose from. The heap sizes are often small since the original grasping policy which generated the heaps rarely fails consecutively on heaps with many objects to choose from.

We examined many individual heaps to understand the magnitude and variance of the policies' impact. Figure 4.4 shows an example where each policy outperformed the baseline, while Figure 4.5 depicts an example where the baseline performance exceeds that of each policy. In Figure 4.4, we can see that the non-baseline policies choose to push one of the two objects that overlap, and they all achieve a large increase in the parallel jaw metric by uncovering the red object initially lying underneath another object. The Boundary Shear and Free Space policies perform especially well, separating all of the objects. Note that the

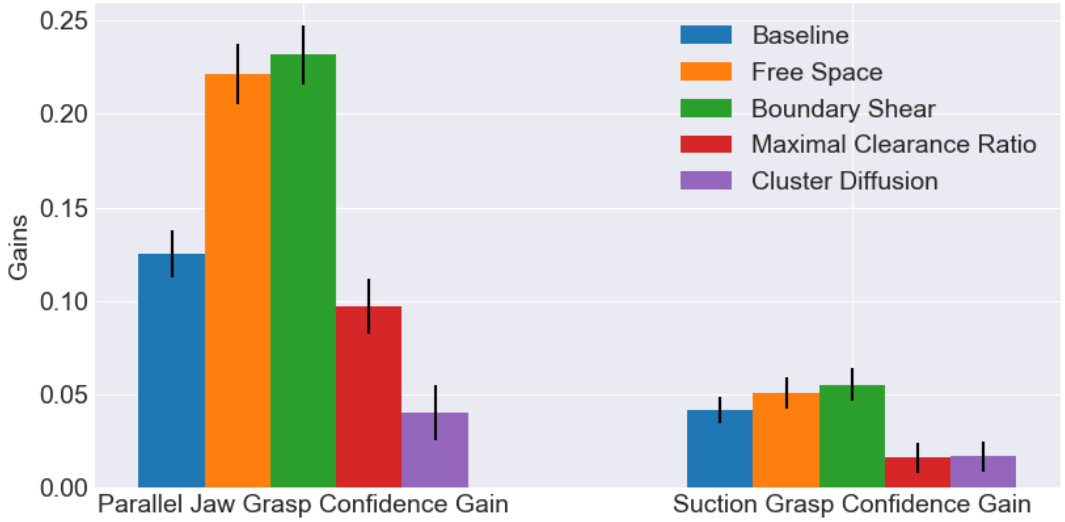


Figure 4.3: Means and standard errors of the mean for each policy and each type of end effector. These results suggest pushing has a larger effect on the parallel jaws. We speculate that this effect occurs due to suction grasps relying on faces of objects being available, and are thus less likely to be affected by pushing, whereas parallel jaw grasps are heavily affected by space around the object.

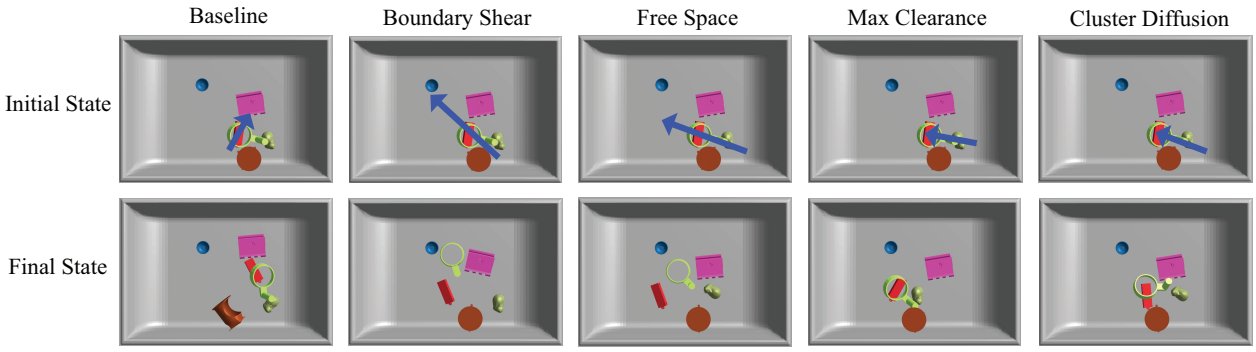


Figure 4.4: For this heap, each policy outperforms the baseline with respect to overall grasp confidence gain. The initial state with the planned action (top row) and final state after executing the planned action (bottom row) are shown for each policy. The blue arrow represents the planned push and the the initial gripper position is represented by the tail of the arrow, while the final position is represented by the head.

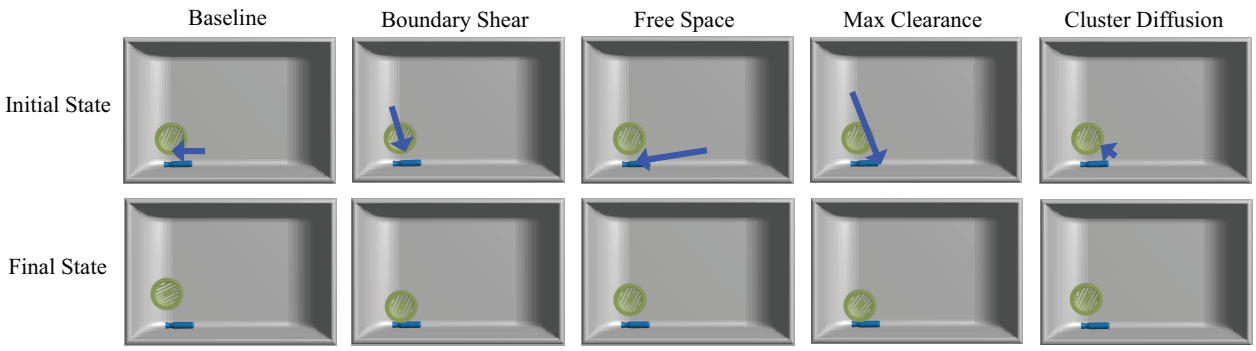


Figure 4.5: The baseline marginally outperforms all the other policies with respect to overall grasp confidence gain due to object placement. The initial state with the planned action (top row) and final state after executing the planned action (bottom row) are shown for each policy. The blue arrow represents the planned push and the the initial gripper position is represented by the tail of the arrow, while the final position is represented by the head.

object does not need to be completely uncovered for the grasp to be available. This reflects the difference between measuring grasping metrics and object separations because in this case, the objects are still touching but a parallel jaw grasp becomes available.

In contrast, in Figure 4.5, we can see that the non-baseline policies fail to find a collision-free push that can move one of the objects away from the corner of the bin. The baseline policy's action is clipped so that it does not collide with the bin, and results in it slightly increasing the parallel jaw grasp confidence by nudging the green object further from the other object. The non-baseline policies have no effect on the grasp confidence metrics. This figure illustrates one of the current failure modes with the pushing policies that we have implemented. By taking a conservative approach and avoiding collisions at all costs, we are sometimes unable to plan a push that moves the objects away from the bin edges.

4.6 Physical Experiments

We planned pushes for bin picking on an ABB YuMi using the Boundary Shear policy, the best performing policy in simulation, over 35 heaps of objects with varying geometry such as tools, toys, produce, and industrial parts. Each heap contained between two and ten objects in configurations with few accessible grasps, such as two bottles side-by-side. For each heap, the robot acquired a point cloud of the bin with a Photoneo PhoXi depth sensor, segmented the point cloud using Euclidean Cluster Extraction implemented in the Point Cloud Library [220], and planned a push using point clusters as objects and the cluster centroid as an estimate of the center of mass of each object. The robot then executed the push by closing the parallel jaw gripper and following the linear push trajectory. Each push took approximately 1.0 seconds to plan.

For each push, we measured the Overall Grasp Quality Gain of parallel-jaw and suction grasps planned by a Grasp Quality Neural Network policy [164]. We categorized performance based on the grasp quality for the best grasp in the initial heap (pre-pushing). Heaps with $Q_o < 0.25$ had an Overall Grasp Quality Gain of 0.24 ± 0.07, while heaps with $Q_o < 0.5$ had an Overall Grasp Quality Gain of 0.12 ± 0.06.

4.7 Discussion and Future Work

Analytic pushing methods can generalize well to heaps of different sizes and geometries on a tabletop environment, but with the bin as a workspace boundary, the action space for pushing is much more limited. Thus, in many cases, pushing any of the available objects in the bin in any direction will yield a change in the state of the heap large enough to change grasp access and affect grasp metrics. However, we have shown in this chapter that several policies are statistically different from a baseline when controlling for the difficulty of each heap, based on new grasp-centric metrics to measure effectiveness of pushing and given the assumptions in Section 4.2.

In future work, we will further explore how the approximations and assumptions made affect the results presented. For example, we will continue to benchmark the policies presented here, as well as other policies, on the physical robot in order to determine the effectiveness of each policy on the physical system and how closely our simulator can represent the physical system. Further experiments will also provide a larger sample size for statistical analysis on the physical system. We will also consider different metrics, such as *maximum* separation distance, which could better inform the state of each heap. For this analysis, we chose a two-step time horizon as an approximation of the normalized total expected reward over all time shown in Equation 4.2.2 because greedy grasping policies have been shown to perform very well in clutter, suggesting long-term rewards are not strongly correlated to grasp actions [162, 197]. In our next experiments, we will test this approximation by measuring the effect of pushing over the course of longer time horizons. Additionally, we made strong assumptions about the boundaries, geometries, and poses of the objects that were analyzed by representing them as points at their center of mass for finding free space in the bin. We seek to modify our simulations to calculate minimum distances between meshes more efficiently while still accounting for the entirety of the objects. We also will look to exploit quicker free space computation in image space as an alternative to our current object assumptions.

Furthermore, in this work, we assume that if none of the five policies were able to prove grasp quality, then the heap is not improvable. Some heaps may be improvable by some policy not tested in this work. In the future, we will determine why some heaps are not able to be improved and seek a method for determining when heaps can be improved without testing several policies on them. For example, when objects are entangled, or cannot easily be pushed due to object pose or shape, we could attempt a different push or grasp action.

As extensions to this work, Kurenkov et al. [137] identified and explored more complex push policies that include multiple linear segments and continuous motions in the plane and

out of plane (e.g., to topple or flip objects) as part of a visuomotor mechanical search pushing policy in tabletop environments. We will also explore how push policies can be learned from human demonstrations [139] and from automated execution data shared via the Cloud [120].

Part II

Perception Primitives

Chapter 5

Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data

In this part of the dissertation, we explore how intermediate representations such as instance segmentation and learned implicit collision functions can influence both high-level action selection policies and low-level manipulation primitives for mechanical search. In addition, we show that leveraging simulated depth images or point clouds in training not only eliminates costly hand labeling of datasets, but also can result in learned networks for these tasks that can robustly generalize to unseen real-world objects and images without finetuning or retraining.

In this chapter, we show that *category-agnostic instance segmentation*, or the ability to mask the pixels belonging to each individual object in a scene regardless of the object's class, has potential to enhance robotic perception pipelines for applications, such as instance-specific grasping, where a target object must be identified and grasped among potentially unknown distractor objects in a cluttered environment. For example, recent approaches to grasp planning on unknown objects have used deep learning to generate robot grasping policies from massive datasets of images, grasps, and reward labels [164, 196]. While these methods are effective at generalizing across a wide variety of objects, they search for high-quality grasp affordances across an entire scene and do not distinguish between the objects they are grasping. Nonetheless, these methods may be extended to plan grasps for a particular target object by constraining grasp planning to an object mask produced by category-agnostic instance segmentation. Instance-specific grasping can be seen as a subset of the mechanical search problem, since it encompasses instances of mechanical search where the target object is already visible.

Object segmentation without prior models of the objects is difficult due to sensor noise and occlusions. Computer vision techniques for generating category-agnostic object proposals [6, 130] often oversegment and require secondary pruning steps to find a set of valid independent objects. A variety of recent methods have demonstrated the ability to accu-

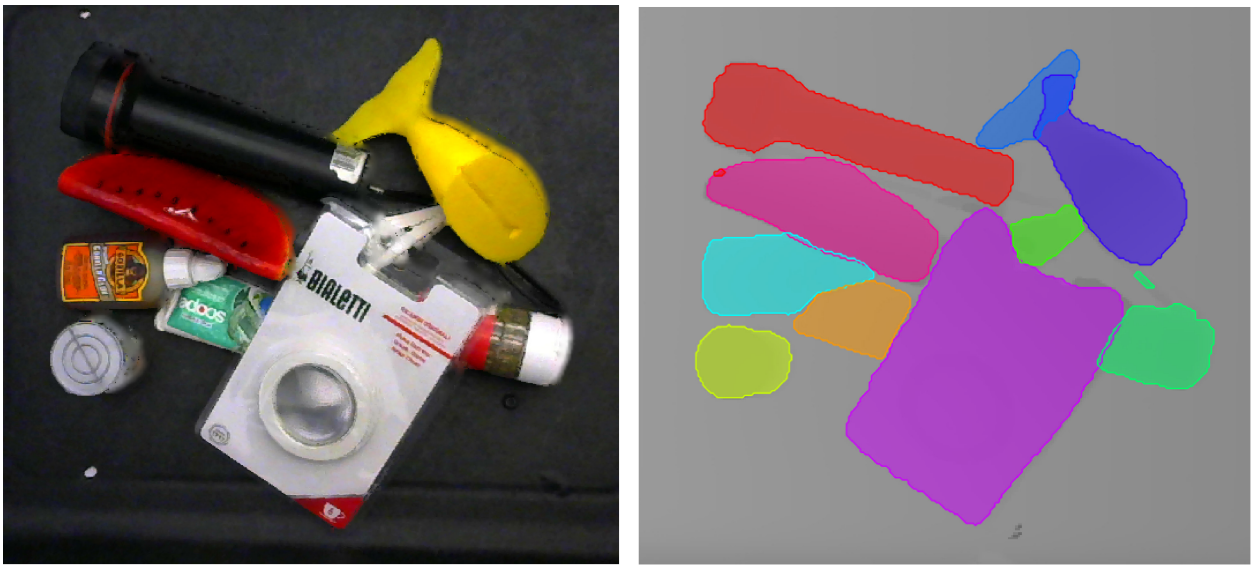


Figure 5.1: Color image (left) and depth image segmented by SD Mask R-CNN (right) for a heap of objects. Despite clutter, occlusions, and complex geometries, SD Mask R-CNN is able to correctly mask each of the objects.

rately segment RGB images into pre-defined semantic classes such as humans, bicycles, and cars by training deep neural networks on massive, hand-labeled datasets [155, 91]. These techniques require time-consuming human labeling to generate training data [152], and existing datasets consist of RGB images of natural scenes that are very different from the types of cluttered scenes commonly encountered in warehouses or fulfillment centers. Adding new object classes or generating data for new types of environments requires additional manual labeling. Thus, in robotics, pixel-wise object segmentation is often avoided [104, 283] or used for a small number of object classes, where semantic segmentation networks [181, 225] or predefined features [108] can be used.

To address these issues, we present a method and dataset for training Mask R-CNN [91], a popular instance segmentation network, to perform category-agnostic instance segmentation on real depth images without training on hand-labeled data – and, in fact, without training on real data at all. We build on recent research which suggests that networks trained on synthetic depth images can transfer well from simulation to reality in some domains [107, 164, 217] and that depth cues can enhance instance segmentation in simulated images [228]. To learn an instance segmentation network that transfers from simulation to reality, we propose to train on a large synthetic dataset of depth images with domain randomization [246] over a diverse set of 3D objects, camera poses, and camera intrinsic parameters.

This chapter contributes:

1. A method for rapidly generating a synthetic dataset of depth images and segmentation

masks using domain randomization for robust transfer from simulation to reality.

2. The Warehouse Instance Segmentation Dataset for Object Manipulation (WISDOM), a hybrid sim/real dataset designed for training and evaluating category-agnostic instance segmentation methods in the context of robotic bin picking.

3. Synthetic Depth Mask R-CNN (SD Mask R-CNN), a Mask R-CNN adaptation designed to perform deep category-agnostic object instance segmentation on depth images, trained on WISDOM-Sim.

4. Experiments evaluating the sim-to-real generalization abilities of SD Mask R-CNN and performance benchmarks comparing it against a set of baseline instance segmentation methods.

In an experimental evaluation on WISDOM-Real's high-resolution dataset, SD Mask R-CNN achieves significantly higher average precision and recall than baseline learning methods fine-tuned on WISDOM-Real training images, and also generalizes to a low-resolution sensor. We employ SD Mask R-CNN as part of an instance-specific grasping pipeline on an ABB YuMi bimanual industrial robot and find that it can increase success rate by 20% over standard point cloud segmentation techniques.

5.1 Related Work

This work builds on prior research in region proposal generation, neural architectures for image segmentation, and use of synthetic data for learning models in computer vision and robotics. The approach presented here is motivated and informed by robotic grasping, manipulation, and bin-picking tasks.

Box and Region Proposals Early work in computer vision focused on using bottom-up cues for generating box and region proposals in images [6, 130, 5, 61, 250]. Such techniques typically detect contours in images to obtain a hierarchical segmentation. Regions from such a hierarchical segmentation are combined together and used with low-level objectness cues to produce a list of regions that cover the objects present in the image. The focus of these techniques is on getting high recall, and the soup of output region proposals is used with a classifier to detect or segment objects of interest [32].

More recently, given advances in learning image representations, researchers have used feature learning techniques (specifically CNN based models) to tackle this problem of producing bounding box proposals [134, 214] and region proposals [199, 200]. Unlike bottom-up segmentation methods, these techniques use data-driven methods to learn high-level semantic markers for proposing and classifying object segments. Some of these learning-based region proposal techniques [200] have built upon advances in models for image segmentation and use fine-grained information from early layers in CNNs [155, 86] to produce high quality regions.

While most work in computer vision has used RGB images to study these problems, researchers have also studied similar problems with depth data. Once again there are bottom-up techniques [219, 220, 255, 210, 82, 84] that use low-level geometry-based cues to come up with region proposals, as well as more recent top-down learning-based techniques to produce proposals [37] in the form of image segments or 3D bounding boxes that contain objects in the scene. Shao, Tian, and Bohg [228] combined color and depth modalities, featurizing objects and clustering the features to produce object instance segmentation masks on simulated RGB-D images.

A parallel stream of work has tackled the problem of class-specific segmentation. Some of these works ignore object instances and study semantic segmentation [155, 200, 70, 151], while others try to distinguish between instances [91, 87]. Similar research has also been done in context of input from depth sensors [84, 207, 277, 36, 257].

Synthetic Data for Training Models Our research is related to a number of recent efforts for rapidly acquiring large training datasets containing image and ground truth masks with limited or no human labeling. The most natural way is to augment training with synthetic color and depth images collected in simulation. This idea has been explored extensively for training semantic segmentation networks for autonomous driving [107, 217] and for estimating human and object pose [230, 239]. Another approach is to use self-supervision to increase training dataset size by first hand-aligning 3D models to images with easy-to-use interfaces [169] or algorithmically matching a set of 3D CAD models to initial RGB-D images [283], and then projecting each 3D model into a larger set of images from camera viewpoints with known 6-DOF poses. In comparison, we generate synthetic training datasets for category-agnostic object segmentation in a robot bin picking domain.

Robotics Applications Segmentation methods have been applied extensively to grasping target objects, most notably in the Amazon Robotics Challenge (ARC). Many classical grasping pipelines consisted of an alignment phase, in which 3D CAD models or scans are matched to RGB-D point clouds, and an indexing phase, in which precomputed grasps are executed given the estimated object pose [43]. In the 2015 ARC, the winning team followed a similar strategy, using a histogram backprojection method to segment objects from shelves and point cloud heuristics for grasp planning [63]. In 2016, many teams used deep learning to segment objects for the alignment phase, training semantic segmentation networks with separate classes for each object instance on hand-labeled [226] or self-supervised datasets [283]. Team ACRV, the winners of the 2017 ARC, fine-tuned RefineNet to segment and classify 40 unique known objects in a bin, with a system to quickly learn new items with a semi-automated procedure [181, 176]. In contrast, our method uses deep learning for category-agnostic segmentation, which can can be used to segment a wide variety of objects not seen in training.

5.2 Problem Statement

We consider the problem of depth-based category-agnostic instance segmentation, or finding subsets of pixels corresponding to unique unknown objects in a single depth image.

To formalize category-agnostic instance segmentation, we use the following definitions:

1. *States:* Let $\mathbf{x} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m, \mathcal{B}_1, \ldots, \mathcal{B}_n, \mathcal{C}\}$ be a ground truth state which contains (A) a set of m foreground objects in the environment, (B) a set of n background objects (e.g. bins, tables), and (C) a depth camera. Here, each object state $\mathcal{O}_i$ or $\mathcal{B}_j$ is defined by the object's geometry and 6-DOF pose, while the camera state $\mathcal{C}$ is defined by its intrinsics matrix K and its 6-DOF pose $(R, \mathbf{t}) \in SE(3)$.

2. *Observations:* Let $\mathbf{y} \in \mathbb{R}_+^{H \times W}$ be a depth image observation of the state $\mathbf{x}$ generated from $\mathcal{C}$ with height H and width W. Let the pixel space $\mathcal{U} = [0, H-1] \times [0, W-1]$ be the set of all real-valued pixel coordinates in the depth image.

3. *Object Mask:* Let $\mathcal{M}_i \subseteq \mathcal{U}$ be a mask for foreground object $\mathcal{O}_i$, or the set of pixels in $\mathbf{y}$ that were generated by the surface of $\mathcal{O}_i$.

Every state $\mathbf{x}$ corresponds to a set of visible foreground object masks : $\mathcal{M} = \{\mathcal{M}_i : \mathcal{M}_i \neq \emptyset,\ \forall i \in \{1, \ldots, m\}\}$. The goal of category-agnostic object instance segmentation is to find $\mathcal{M}$ given a depth image $\mathbf{y}$.

5.3 Synthetic Dataset Generation Method

To efficiently learn category-agnostic instance segmentation, we generate a synthetic training dataset of N paired depth images and ground truth object masks: $\mathcal{D} = \{(\mathbf{y}_k, \mathcal{M}_k)\}_{k=1}^N$. The proposed dataset generation method samples training examples using two distributions: a task-specific state distribution, $p(\mathbf{x})$, that randomizes over a diverse set of object geometries, object poses, and camera parameters; and an observation distribution, $p(\mathbf{y}|\mathbf{x})$, that models sensor operation and noise.

To sample a single datapoint, we first sample a state $\mathbf{x}_k \sim p(\mathbf{x})$ using a dataset of 3D CAD models, dynamic simulation, and domain randomization [246] over the object states, camera intrinsic parameters, and camera pose for robust transfer from simulation to reality. Next, we sample a synthetic depth image $\mathbf{y}_k \sim p(\mathbf{y}_k \mid \mathbf{x}_k)$ using rendering. Finally, we compute the visible object masks $\mathcal{M}_j$ determining the set of pixels in the depth image with a corresponding 3D point on the surface of object $\mathcal{O}_j$. Specifically, we render a depth image of each object in isolation and add a pixel to the mask if it is within a threshold from the corresponding full-state depth image.

Figure 5.2: Dataset generation procedure for the WISDOM synthetic dataset. A subset of 3D CAD models from a training dataset of 1,600 objects are dropped into a virtual bin using dynamic simulation with PyBullet. A virtual camera captures both a synthetic depth image of the scene and object segmentation masks based on the pixelwise projection of each unique 3D object. This process is repeated to generate 50,000 images.

5.4 WISDOM Dataset

To test the effectiveness of this method, we generate the Warehouse Instance Segmentation Dataset for Object Manipulation (WISDOM), a hybrid sim/real dataset designed to train and test category-agnostic instance segmentation networks in a robotic bin-picking environment. WISDOM includes WISDOM-Sim, a large synthetic dataset of depth images generated using the simulation pipeline, and WISDOM-Real, a set of hand-labeled real RGB-D images for evaluating performance in the real world.

5.4.1 WISDOM-Sim

For WISDOM-Sim, we consider an environment for robotic bin picking consisting of a table and a bin full of objects imaged with an overhead depth camera. In general, $p(\mathbf{x})$ can be represented as a product over distributions on:

1. *Foreground and background object counts (m and n)*: We draw m from a Poisson distribution with mean $\lambda = 7.5$, truncated to a maximum of 10. We set $n = 2$ since we use two fixed background objects: a table and a bin.

2. *Background object states ($\{\mathcal{B}_j\}_1^n$)*: We set the geometry and pose of the background objects to fixed values.

3. *Foreground object states ($\{\mathcal{O}_j\}_1^m$)*: We sample the m foreground objects uniformly from a dataset of 1,664 3D triangular mesh models from Thingiverse, including objects augmented with artificial cardboard backing to mimic common packages. Object poses are sampled by selecting a random pose above the bin from a uniform distribution, dropping each object into the bin one-by-one in PyBullet dynamic simulation, and simulating until all objects come to rest [162].

4. *Camera state ($\mathcal{C}$)*: We sample camera poses uniformly at random from a bounded set of spherical coordinates above the bin. We sample intrinsic parameters uniformly at random from intervals centered on the parameters of a Photoneo PhoXi S industrial depth camera.

Because the high-resolution depth sensor we use has very little white noise, we fix $p(\mathbf{y}|\mathbf{x})$ to simply perform perspective depth rendering using an OpenGL z-buffer.

We used these distributions to sample a dataset of 50,000 synthetic depth images containing 320,000 individual ground-truth segmentation masks. Generating 50k datapoints took approximately 26 hours on a desktop with an Intel i7-6700 3.4 GHz CPU. The synthetic images are broken into training and validation sets with an 80/20 split, where the split is both on images as well as objects (i.e., no objects appear in both the training and validation sets). The training set has 40,000 images of 1,280 unique objects, while the validation set contains 10,000 images of 320 unique objects.

5.4.2 WISDOM-Real

Figure 5.3: Objects included in the WISDOM-Real dataset. 25 objects were used for fine-tuning, while a separate set of 25 were held out for evaluation.

Figure 5.4: Example bins from the WISDOM-Real test set. The number of objects in each bin was chosen from a Poisson distribution with mean 5, with a minimum of two objects per bin. The highly-varied geometry and occlusions make these bins challenging to segment.

To evaluate the real-world performance of category-agnostic instance segmentation methods and their ability to generalize to novel objects across different types of depth sensors, we collected a hand-labeled dataset of real RGB-D images. WISDOM-Real contains a total of 800 hand-labeled RGB-D images of cluttered bins, with 400 from both a high-resolution Photoneo PhoXi industrial sensor (1032x772 with 0.05 mm depth precision) and a low-resolution Primesense Carmine (640x480 with 1 mm depth precision). Missing depth values were filled in using fast inpainting [189] with an averaging kernel.

The objects in these bins were sampled from a diverse set of 50 novel objects with highly-varied geometry, all of which are commonly found around the home (see Figure 5.3),

and have no corresponding CAD model in WISDOM-Sim. This set of 50 objects was split randomly into even training and test sets. The training set is reserved for learning methods that require real training data, while the test set is used to test generalization to novel objects. We generated 100 bins containing objects from the training set and an additional 300 bins containing objects from the test set. For each bin, a truncated Poisson distribution ($\lambda = 5$) was used to determine the number of objects to be placed in the bin, and the objects were sampled uniformly at random from the appropriate subset. The sampled objects were shaken together in a plastic box to randomize their poses and dumped into a large black bin with identical geometry to the bin used in WISDOM-Sim, and once the objects settled, each camera took an RGB-D image from above. Sample bins are shown in Figure 5.4.

After dataset collection, all images were hand-labeled to identify unique object masks using the same tools used to label the COCO dataset [152]. We estimate that labeling the 800 real images took over 35 hours of effort due to time spent collecting images, labeling object masks, and data cleaning.

5.5 Synthetic Depth Mask R-CNN

To adapt Mask R-CNN to perform category-agnostic instance segmentation on depth images, we made several modifications:

1. We treat depth images as grayscale images and triplicate the depth values across three channels to match the input size of the original network.

2. We reduce the number of classes to two. Each proposed instance mask is classified as background or as a foreground object. Of these, only foreground masks are visualized.

3. We modify the network input to zero-pad the 512x384 pixel images in WISDOM-Sim to 512x512 images and set the region proposal network anchor scales and ratios to correspond to the 512x512 image size.

4. For efficiency, we swapped out the ResNet 101 backbone with a smaller ResNet 35 backbone.

5. We set the mean pixel value to be the average pixel value of the simulated dataset.

Training was based on Matterport's open-source Keras and TensorFlow implementation of Mask R-CNN from GitHub, which uses a ResNet 101 and FPN backbone [1]. This implementation closely follows the original Mask R-CNN paper in [91]. We made the modifications listed above and trained the network on WISDOM-Sim with an 80-20 train-val split for 60 epochs with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.0001 on a Titan X GPU. On our setup, training took approximately 24 hours and a single forward pass took 105 ms (average of 600 trials). We call the final trained network a Synthetic Depth Mask R-CNN (SD Mask R-CNN).

	High-Res		Low-Res	
Method	**AP**	**AR**	**AP**	**AR**
Euclidean Clustering	0.324	0.467	0.183	0.317
Region Growing	0.349	0.574	0.180	0.346
FT Mask R-CNN (Depth)	0.370	0.616	0.331	0.546
FT Mask R-CNN (Color)	0.384	0.608	***0.385***	***0.613***
SD Mask R-CNN	***0.516***	***0.647***	0.356	0.465

Table 5.1: Average precision and average recall (as defined by COCO benchmarks) on each dataset for each of the methods considered. SD Mask R-CNN is the highest performing method, even against Mask R-CNN pretrained on the COCO dataset and fine-tuned on real color and depth images from WISDOM-Real.

5.6 Experiments

We compare performance of SD-Mask-R-CNN with several baseline methods for category-agnostic instance segmentation on RGB-D images.

5.6.1 Baselines

We use four baselines: two Point Cloud Library (PCL) methods and two color-based Mask R-CNNs pre-trained on COCO and fine-tuned on WISDOM-Real images. For fine-tuning, the image shape and dataset-specific parameters such as mean pixel were set based on the dataset being trained on (e.g., either color or depth images).

Point Cloud Library Baselines

The Point Cloud Library (PCL), an open-source library for processing 3D data, provides several methods for segmenting point clouds [220]. We used two of these methods: Euclidean clustering and region-growing segmentation. Euclidean clustering adds points to clusters based on the Euclidean distance between neighboring points. If a point is within a sphere of a set radius from its neighbor, then it is added to the cluster [219]. Region-growing segmentation operates in a similar way to Euclidean clustering, but instead of considering Euclidean distance between neighboring points, it discriminates clusters based on the difference of angle between normal vectors and curvature [255, 210]. We tuned the parameters of each method on the first ten images of the high-res and low-res WISDOM-Real training sets.

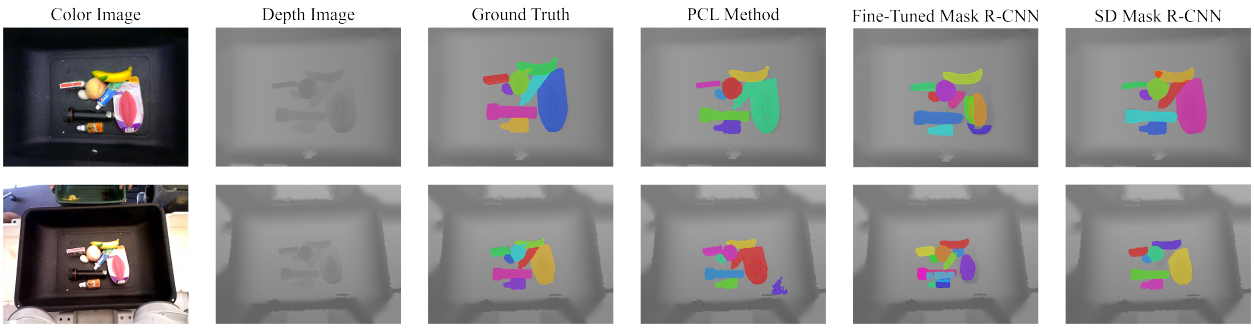


Figure 5.5: Images from both sensors with ground truth and object masks generated by each method. For the baseline methods, the better performing method was chosen for each scenario. While the baseline methods tend to undersegment (PCL) or oversegment (Fine-Tuned Mask R-CNN), SD Mask R-CNN segments the objects correctly.

Fine-Tuned Mask R-CNN Baselines

As deep learning baselines, we used two variants of Mask R-CNN, one trained on color images and one trained on depth images triplicated across the color channels. Both these variants were pre-trained on RGB images from the COCO dataset and then fine-tuned using the 100 color or depth images from the WISDOM-Real high-res training set. All images were rescaled and padded to be 512 by 512 pixels, and the depth images were treated as grayscale images. Both implementations were fine-tuned on the 100 images for 10 epochs with a learning rate of 0.001.

5.6.2 Benchmarks

We compare the category-agnostic instance segmentation performance of all methods using the widely-used COCO instance segmentation benchmarks [152]. Of the metrics in the benchmark, we report average precision (AP) over ten IoU thresholds over a range from 0.50 to 0.95 with a step size of 0.05, and we report average recall (AR) given a maximum of 100 detections. Averaging over several IoU thresholds rewards better localization from detectors, so we report this score as our main benchmark as opposed to simply the average precision for an IoU threshold of 0.50. All scores are for the segmentation mask IoU calculation.

5.6.3 Performance

We ran each of the methods on three test datasets: 2000 images from the WISDOM-Sim validation set and 300 real test images each from the Primesense and Phoxi cameras. All real test images were rescaled and padded to be 512 by 512 pixels. The results are shown in Table 5.1, and full precision-recall curves for each dataset can be found in Appendix B. The

SD Mask R-CNN network shows significant improvement over both the PCL baselines and the fine-tuned Mask R-CNN baselines, and is also robust to sensor noise.

An example of each method's performance on each of the real datasets can be seen in Figure 5.5. The visualizations suggest that the PCL baselines tend to undersegment the scene and cluster nearby objects as a single object. The fine-tuned Mask R-CNN implementations separate objects more effectively, but the color implementation may incorrectly predict multiple object segments on different colored pieces of the same object. In contrast, the SD Mask R-CNN network can group parts of objects that may be slightly discontinuous in depth space and is agnostic to color. It is able to segment the scenes with high accuracy despite significant occlusion and variation in shape. Table 5.1 also shows SD Mask R-CNN can perform similarly on low-res Primesense images, suggesting that the network can generalize to other camera intrinsics and poses.

5.6.4 Robotics Application: Instance-Specific Grasping

To demonstrate the usefulness of SD Mask R-CNN in a robotics task, we ran experiments using category-agnostic instance segmentation as the first phase of an instance-specific grasping pipeline. In this task, the goal is to identify and grasp a particular target object from a bin filled with other distractor objects. For these experiments, we used a randomly-selected subset of ten objects from WISDOM-Real's test set.

One approach to this problem is to collect real images of the items piled in the bin, label object masks in each image, and use that data to train or fine-tune a deep neural network for object classification and segmentation [181, 225]. However, that data collection process is time consuming and must be re-performed for new object sets, and training and fine-tuning a Mask R-CNN can also be time consuming. Instead, our experimental pipeline uses a class-agnostic instance segmentation method followed by a standard CNN classifier, which is easier to generate training data for and faster to train.

To train the classifier, we collected ten RGB images of each target object in isolation. Each image was masked and cropped automatically using depth data, and then each crop was augmented by randomly masking the crop with overlaid planes to simulate occlusions. From the initial set of 100 images, we produced a dataset of 1,000 images with an 80-20 train-validation split. We then used this dataset to fine-tune the last four layers of a VGG-16 network [231] pre-trained on Imagenet. Fine-tuning the network for 20 epochs took less than two minutes on a Titan X GPU, and the only human intervention required was capturing the initial object images.

We benchmarked SD Mask R-CNN against two segmentation methods across 50 independent trials. First, we compared against the PCL Euclidean Clustering method to evaluate baseline performance. Second, we compared with Mask R-CNN fine-tuned on the WISDOM-Real training dataset to evaluate whether SD Mask R-CNN is competitive with methods trained on real data.

During each iteration, one object at random was chosen as the target and all of the objects were shaken, dumped into the black bin, and imaged with the Photoneo PhoXi depth sensor.

We then segmented the depth image using either SD Mask R-CNN or one of the baseline methods, colorized the mask using the corresponding RGB sensor image, and then labeled each mask with the pre-trained classifier. The mask with the highest predicted probability of being the target was used as a constraint for a Dex-Net 3.0 [164] policy that planned suction cup grasps, and the planned grasp was executed by an ABB YuMi equipped with a suction gripper. Each iteration was considered a success if the target object was successfully grasped, lifted, and removed from the bin, then transported to a nearby receptacle.

The results of these instance-specific grasping experiments are shown in Table 5.2. SD Mask R-CNN achieved a success rate of 74%, significantly higher than the PCL Euclidean clustering baseline (56%). Furthermore, SD Mask R-CNN had performance on par with a Mask R-CNN that was fine-tuned on 100 real color images (78%). SD Mask R-CNN outperforms the PCL baseline and achieves performance similar to Mask R-CNN fine-tuned on real data, despite the fact that SD Mask R-CNN was training on only synthetic data. These results suggest that high-quality instance segmentation can be achieved without expensive data collection from humans or self-supervision and suggest that the effort needed to take advantage of object segmentation for new robotic tasks can be significantly reduced by leveraging synthetic data.

Method	**Success Rate** (%)	**Prec. @ 0.5** (%)	**# Corr. Targets**
Euclidean Clustering	56 ± 14	63 ± 19	35
FT Mask R-CNN (Color)	78 ± 11	85 ± 12	44
SD Mask R-CNN	74 ± 12	87 ± 11	39

Table 5.2: Results of semantic segmentation experiments, where success is defined as grasping and lifting the correct object. **(Success Rate)** Number of successful grasps of the correct object over 50 trials. **(Prec. @ 0.5)** Success rate when the classifier was $> 50\%$ certain that the selected segment was the target object. **(# Corr. Targets)** Number of times the robot targeted the correct object out of 50 trials.

5.7 Discussion and Future Work

This chapter introduced WISDOM, a dataset of images and object segmentation masks for the warehouse object manipulation environment, images that are currently unavailable in other major segmentation datasets. Training SD Mask R-CNN, an adaptation of Mask R-CNN, on synthetic depth images from WISDOM-Sim enables transfer to real images without expensive hand-labeling, suggesting that depth alone can encode segmentation cues. SD Mask R-CNN outperforms PCL segmentation methods and Mask R-CNN fine-tuned on real

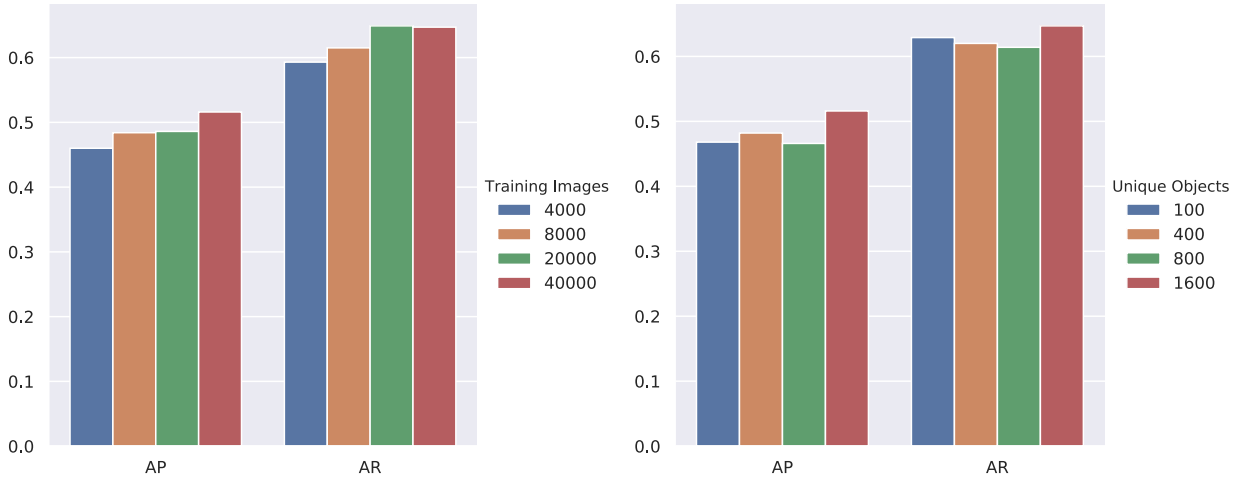


Figure 5.6: Effect of increasing synthetic training dataset size (left) and increasing the number of unique objects (right) on the performance of SD Mask R-CNN on the WISDOM-Real high-res test set. These results suggest that more data could continue to increase performance.

color and depth images for the object instance segmentation task, and can be used as part of a successful instance-specific grasping pipeline.

Figure 5.6 shows preliminary results of the effects of training image dataset size and training object dataset size on the performance of SD Mask R-CNN. We trained SD Mask R-CNN on random subsets of the training dataset with sizes $\{4k, 8k, 20k, 40k\}$ and on four $40k$ image datasets containing 100, 400, 800, and 1600 unique objects. Both AP and AR increase with image dataset size and number of unique objects used in training, although the image dataset size appears to have a stronger correlation with performance. These results suggest that performance might continue to improve with orders of magnitude more training data.

Chapter 6

Object Rearrangement Using Learned Implicit Collision Functions

Similar to the previous chapter, we provide another example of a robot task related to mechanical search (in this case, object rearrangement [13]) that can benefit from large-scale simulated datasets of labeled point clouds to predict collisions between objects efficiently. Rearrangement is a fundamental skill for robot mechanical search, since environments often may not allow for objects to be removed. Thus, robots must be able to reason about collisions between grasped objects and the rest of the scene and potential collision-free grasping and placement locations while receiving only partial observations. The majority of existing approaches rely on known models of the objects and environment to generate collision-free trajectories for grasping, moving, and placing objects in a scene [124, 98]. When only point cloud data for the scene and objects are available, these approaches may not correctly reason about occlusions or quickly react to a changing environment. We focus on a core ability that enables rearrangement of unknown objects in cluttered unknown environments: collision checking based on raw sensor measurements.

Existing techniques for collision checking between objects and scenes are limited in that they either rely on known object models or struggle to reason about occluded areas of a scene [72, 191, 192]. Recently, the computer vision community has introduced deep learning techniques with astonishing abilities to represent and reason about fine-grained 3D object geometries [195, 33, 105]. Unfortunately, these approaches are not efficient enough to handle the large number of collision queries necessary for efficient trajectory optimization and control in robotics. In this chapter, we introduce an approach that overcomes these limitations and provides robust collision checking on point clouds with occlusions at speeds that are beyond model-based collision checkers used in robotics.

We present a neural network that takes as input raw point clouds of both an object and a scene and a 6DOF pose of the object in the scene and outputs the likelihood that the object collides with the scene. We combine point features with voxel features to construct a scene representation that is both fast and memory efficient. We train the model entirely in simulation with 1 million randomly generated tabletop scenes and show it can generalize

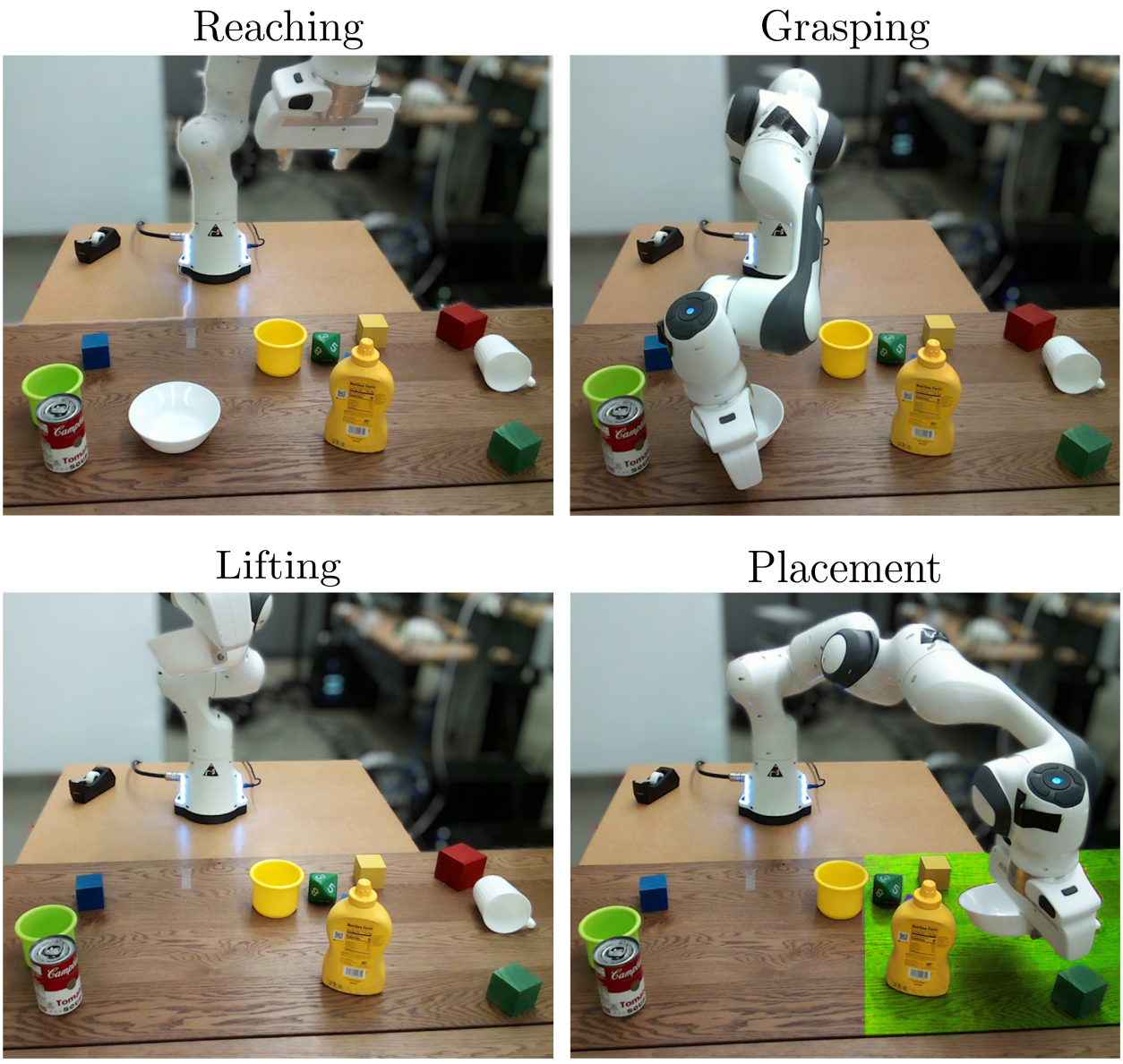


Figure 6.1: The rearrangement task consists of four subtasks: reaching for, grasping, lifting, and placing the object within a placement zone (overlaid in green). In each subtask, the robot must plan a kinematically feasible, collision-free path with only partial point cloud observations.

to real point cloud data. The resulting model can be used in any existing motion planning framework to generate collision-free motion plans; we demonstrate its capability for object rearrangement via pick-and-place actions based on point cloud measurements in a model predictive control framework with no additional learned parameters.

This chapter makes three contributions:

1. SceneCollisionNet: a model architecture and training procedure for collision checking between point clouds.
2. A rearrangement policy using SceneCollisionNet in a model predictive path integral controller.
3. Experimental results in simulation and on a real robot platform showing SceneCollisionNet achieves 93% accuracy on 2 million object-scene queries, taking only 10 μs per query, 75x faster than baselines.

6.1 Related Work

6.1.1 Robot Collision Detection from Point Clouds

When mesh models for objects in a scene are known, there exist fast and accurate methods for checking collisions between robot links and the scene or between objects themselves [191, 72]. However, in scenes containing unknown objects, only partial point cloud data may be available. One approach to collision checking for point cloud data is to expand each point as a sphere with a predefined radius [101], but the radius may be difficult to determine and may affect the resolution of the collision queries. Similarly, voxel-based approaches are memory-intensive [39] and can suffer from resolution errors due to discretization. Bounding volume hierarchies that attempt to capture a representation of the shape from the points have also been considered [67, 125], but again may not capture occluded areas and may not be robust to noise in the point cloud. Pan, Chitta, and Manocha [192] cast the problem as a binary classification problem, use an SVM to learn a boundary surface between the point clouds, and determine collision probability based on the probability of points crossing the boundary. In contrast, we encode the scene into a set of latent voxel vectors instead of checking collisions between raw point clouds and show an ability to reason about partially observable areas in real time.

6.1.2 Point Cloud Surface Representations

Another approach to point cloud collision detection is to derive a representation of the underlying surface and check collisions against that representation. Adaptive meshes [244] or alpha shapes [9, 59] convert an unstructured array of 3D points into a triangular or tetrahedral mesh. Berger et al. [18] provide an excellent survey of surface reconstruction

methods. Data-driven approaches also reconstruct underlying representations from point clouds or depth images [234, 77, 47]. They typically encode points using either point [207, 208] or voxel [263, 287] representations, or a combination of the two [154]. Several recent approaches use fully-connected neural networks to encode an implicit representation of the surface as a function in 3D space [195, 38, 174], showing an ability to reconstruct objects with fine geometries. Van der Merwe et al. [251] similarly reconstruct objects from partial point clouds without optimizing for a latent vector at run time. Jiang et al. [105] and Chabra et al. [33] further encode the surface into many latent vectors across discrete voxels as opposed to a single latent vector for the shape for better scene-level performance. We similarly discretize space into voxels and encode the points in each voxel into a latent vector, but optimize end-to-end for collision queries instead of reconstructing the underlying object geometry.

6.1.3 Accelerating Collision Detection

As collision checking is considered one of the bottlenecks in motion planning, several methods accelerate it using previously calculated collision results [193, 133]. Pan and Manocha [194] developed a GPU implementation of bounding volume test tree traversal that dramatically increases the speed of generating collision-free motion plans for a PR2 robot. Fastron [53] and ClearanceNet [122] generate $\mathcal{C}$-space models for collision checking. ClearanceNet also batches collision checks and does not need retraining when objects move. Tran, Denny, and Ekenna [247] use a contractive autoencoder and multi-layer perceptron to predict collisions in latent space between a robot and axis-aligned boxes. However, each of these methods assumes knowledge of object geometry whereas our method operates directly on point cloud data.

6.1.4 Robotic Object Rearrangement

There has been considerable work on planning for object rearrangement in tabletop scenarios [140, 229, 124, 98]; however, these approaches typically rely on known models of both the environment and the objects in the scene for finding collision-free grasping and placement motions. Recently, there have been advances in 6-DOF grasping [183, 185] and closed-loop grasping [233, 180]; Murali et al. [185] learn collisions between grippers and cluttered scenes centered around a target object, but their method does not easily lend itself to broader motion planning frameworks. Gualtieri and Platt [78] learn pick and place actions for block, mug, and bottle objects, but use top-down point clouds and do not account for workspace dynamics. The Amazon Picking Challenge has also focused development of pick and place systems [284]. In contrast, we integrate a learned point cloud collision checker into an existing motion planning framework for both grasping and placement actions. Zeng et al. [285], Yuan et al. [280], and Haustein et al. [89] learn policies to pick and place or rearrange objects via grasping or pushing directly from input images similar to our approach [281, 236]. How-

ever, they consider planar tasks that do not require the robot to reason about 3D collisions with other objects.

6.2 Problem Statement

We consider a problem setting where a robot with a parallel-jaw gripper iteratively grasps and places objects on a tabletop to rearrange them. The objective is to rearrange the objects as quickly as possible while reacting to changes in the environment. Observations of the scene are given by a single depth sensor with known camera intrinsics pointing toward the table and robot at an oblique angle.

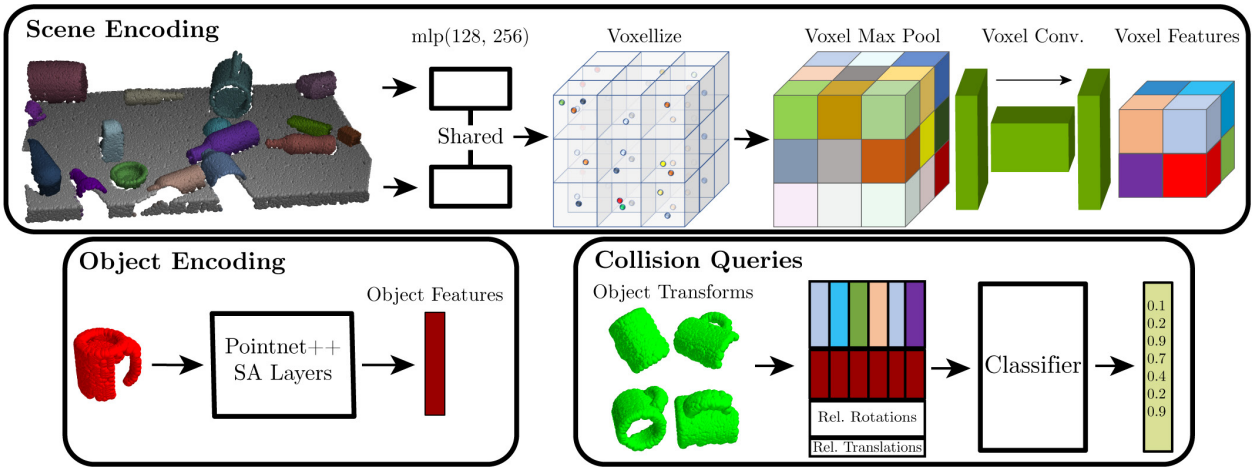


Figure 6.2: Network architecture for SceneCollisionNet, which predicts collisions between a scene point cloud and an object point cloud given a relative 6DOF pose of the object within the scene. Scene points are encoded by voxelizing, featurizing, and convolving the max-pooled voxels. Object points are encoded using Pointnet++ [208] layers. Collision queries are created by feeding the concatenated voxel features, object features, and the relative object transform into a small classifier that predicts the likelihood of collision. Object transforms are specified relative to the voxel frame such that collision queries across different voxels can be predicted simultaneously.

6.2.1 Definitions

We define the problem as having:

- **States** (S): A state $\mathbf{s}_k$ at time k consists of a valid robot joint configuration $\mathbf{q}_k$ and a tabletop containing N objects. No prior information is known about the N objects.

$S_{k,\text{free}} \subset S$ is the set of collision-free states at time k and $G_k \subset S$ is the set of goal grasp or placement states.

- **Observations** (O): An observation $\mathbf{y}_k \in \mathbb{R}^{n\times 3}$ at timestep k consists of a point cloud with n points from the camera pointing at the scene.
- **Actions** (A): Actions are defined as a change in the joint configuration of the robot $\mathbf{a}_k = \Delta\mathbf{q}_k$.
- **Transitions** (T): The transition model $T(\mathbf{s}_{k+1} \mid \mathbf{a}_k, \mathbf{s}_k)$ represents the dynamics of the scene and robot and is executed by Isaac Gym in simulation [150]. On the physical system, next states are determined by executing the action on a physical robot.
- **Cost Function** (C): The cost of a state $C(\mathbf{s}_k)$ is defined as the minimum L2 distance from the current robot joint configuration to the goal robot joint configuration: $C(\mathbf{s}_k) = \min_{\mathbf{g}_k} \|\mathbf{s}_k - \mathbf{g}_k\|_2 \; s.t. \; \mathbf{g}_k \in G_k$.

6.2.2 Objective

The rearrangement objective is to find a policy π that minimizes the total cost of the states visited during grasping and placement over a finite horizon H subject to kinematic constraints and that all states along the trajectory are collision-free:

$$\pi^* = \arg\min_{\pi} \; \mathbb{E}_{\mathbf{a}_k \sim \pi(\mathbf{s}_k)} \sum_{k=1}^{H} C(\mathbf{s}_k) \; s.t. \; \mathbf{s}_k \in S_{k,\text{free}}$$

6.3 SceneCollisionNet

To predict collisions between two point clouds, we propose SceneCollisionNet, a deep neural network inspired by recent work in implicit surface representations from point clouds. Similar to Jiang et al. [105] and Chabra et al. [33], we divide space into coarse voxels and use a local representation for each voxel based on the points contained within that voxel. However, our experiments show that for collision queries: (1) explicitly reconstructing the underlying surface within each voxel is unnecessary and (2) scene information must be shared between voxel representations. We also avoid the costly latent vector optimization, enabling real-time collision prediction.

Our model divides the scene point cloud into coarse voxels (side length of about $10\,cm$) and assigns points to the voxels, normalizing each point within its voxel by subtracting the voxel's center. We pass the points through a shared multi-layer perceptron and max-pool the features of the points per voxel, similar to Pointnet [207]. The max-pooled voxel features are passed through 3D convolution layers, similar to Liu et al. [154], incorporating global

information from neighboring voxels. The target object point cloud is featurized separately using Pointnet++ set abstraction layers [208]. Collision queries consist of the transform of the object relative to the nearest voxel center, the corresponding voxel features, and the object features. This approach means the scene and object features are generated only once per scene point cloud and a large number of collision queries can be made in a single forward pass through the classifier, which predicts the likelihood of collision for each transformation. Figure 6.2 shows the network architecture.

6.3.1 Dataset Generation and Training

We train SceneCollisionNet entirely using synthetic point clouds. For each scene, we place objects drawn from a dataset of 8828 3D mesh models [62] in one of their stable poses with a uniformly random rotation applied about the world z-axis on a planar surface. Object positions are chosen uniformly at random such that they do not collide with other objects. We draw the number of objects from a uniform distribution between 10 and 20. The camera, which renders a scene point cloud, is aimed at the origin of the scene and its extrinsics are taken from uniform distributions centered at their nominal values. A query object is also drawn from the dataset of mesh models; this object is placed at the origin in a random stable pose, where a point cloud is rendered using the same camera. We then generate q collision queries by moving the query object along t trajectories through the scene, recording its relative rotation, translation, and ground truth collisions with the scene using the flexible collision library (FCL) [191]. The trajectory is formed by linearly interpolating between the start and end object poses, which are chosen randomly. Generating one scene/target pair with $q = 2048$ queries over $t = 64$ trajectories takes roughly 2 seconds on an Ubuntu 18.04 machine with an Intel Core i7-7800X 3.50GHz CPU. Each epoch of training consists of 1,000 unique scene/object/trajectory inputs and we train each model for 1000 epochs, or a total of 1 million unique inputs and just over 2 billion total collision queries. We adopt a hard negative mining scheme, where we backpropagate the loss only from the 10% highest loss queries plus 10% random queries, which increases the true positive rate by 6% for similar accuracy. Training takes about 9 days on an NVIDIA V100 GPU. We use Stochastic Gradient Descent (SGD) with learning rate $1e-3$ and momentum 0.9.

6.3.2 Robot Collision Checking

For robot collision checking, we pre-sample points from the 3D mesh of each link in the robot's kinematic chain and featurize each set of points. This feature set is only generated once for a given robot. The set of link features and link poses (using forward kinematics for a given configuration) are input to SceneCollisionNet with the scene features at run time; collision predictions can then be generated for all links in a single forward pass. The same method can also be used to predict collisions between other known meshes and a partial scene point cloud, showcasing the flexibility of our method.

6.4 Object Rearrangement

Rearrangement of objects is a multi-stage task, so we incorporate a finite state machine into our policy with 5 states: reaching the pre-grasp pose, attempting the grasp, lifting the object, placing the object, and releasing the placed object. We use a model predictive path integral (MPPI) policy for the reaching and placing states and preset actions for reaching from the pre-grasp to final grasp pose, lifting, and releasing the object. We use SceneCollisionNet to find both placement positions and collision-free trajectories for grasping and placing.

6.4.1 Grasps and Placements

We modify Contact-GraspNet [240] to predict 6DOF grasps on a region of the raw point cloud in cluttered environments and the segmentation from Xiang et al. [264]. We use the Trac IK solver [14] to convert grasp poses to robot configurations. We accept a point cloud mask that represents an area of the scene where the object should be placed, which by default includes the entire workspace. Points are sampled uniformly at random within the placement zone and sorted by height in the scene; SceneCollisionNet classifies whether the object would be in collision at the given point and the lowest collision-free points are chosen as placement goals. Figure 6.3 shows placement candidates for both empty and cluttered placement zones for the object in hand. Final placement goals (purple) must be both collision-free and have an inverse kinematics solution. Orange points show placement candidates without inverse kinematics solutions; this decoupling allows for the same placements to be used with a different robot.

6.4.2 MPPI Policy

We leverage the parallelism provided by SceneCollisionNet in a model predictive path integral (MPPI) algorithm for object rearrangement in tabletop environments [261]. The advantages of MPPI in this setting are: 1) the task can be specified entirely in the joint space and joint constraints can be strictly enforced during trajectories, 2) trajectory costs can easily be specified using distances in joint space, 3) trajectory generation, cost calculation, collision checking, and forward kinematics can be parallelized on a GPU for the real-time capability necessary in closed-loop execution. In contrast, standard motion planning techniques, such as RRT or PRM, may provide guarantees of completeness and optimality, but are by nature sequential, require a nearest neighbor search for connecting nodes, and must be adapted for dynamic environments.

We adapt MPPI such that trajectories are generated by sampling around a linear trajectory between the start and goal joint configurations. Specifically, we create T vectors by perturbing the straight-line trajectory $\mathbf{d}$ with a vector drawn from a normal distribution and renormalizing: $\tilde{\mathbf{d}}_i = N(\mathbf{d} + \mathcal{N}(\mathbf{0}, \Sigma))$. Trajectories consist of H steps along $\tilde{\mathbf{d}}_i$; actions are clipped to the robot joint limits at each timestep.

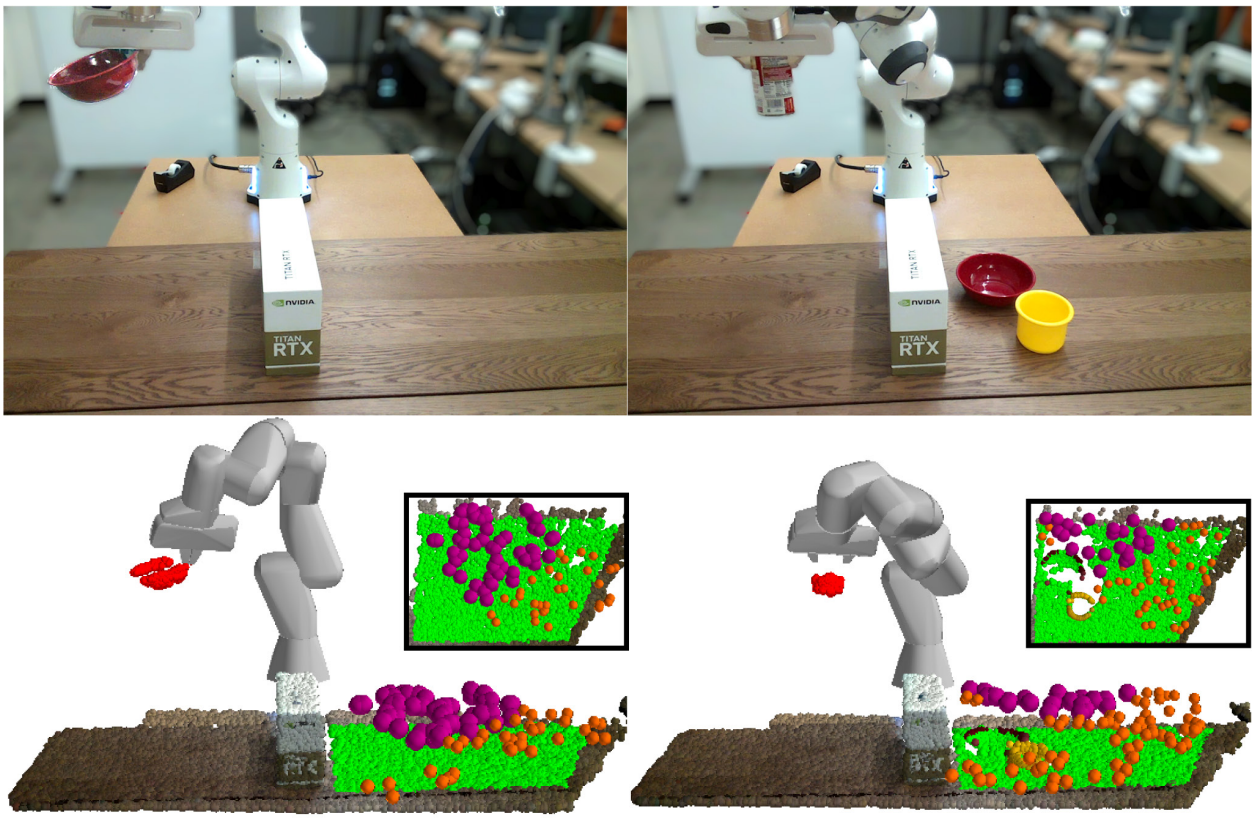


Figure 6.3: With an uncluttered (left) placement zone (green), collision-free placement goals with inverse kinematics solutions (purple dots) spread across the zone within reach of the robot. When the zone is cluttered (right), SceneCollisionNet predicts placements around the objects. Orange dots represent collision-free placements without IK solutions.

The cost of each trajectory is the cost of its final state as defined in Section 6.2.1. We check both robot-scene collisions as in 6.3.2 and robot self-collisions using a model that predicts distance to self-colliding configurations [212] at discrete intervals between each waypoint in each trajectory. Thus, at each policy call, we make $T \times H$ collision checks for each robot link, which can be computed in a single forward pass using SceneCollisionNet. If there is an object in hand, collisions between the object and the scene are also checked at each point in the trajectory. Then, we remove all waypoints after the first colliding waypoint and clip the trajectory to the waypoint with minimum cost. The minimal cost trajectory is executed until the policy is called again. Figure 6.4 shows a sampling of four trajectories and their associated costs.

Importantly, the scene points belonging to the robot or to the target object must be removed during placement; if they remain in the scene, they will cause all MPPI trajectories to be in collision. In physical experiments, we combine a learned robot point cloud segmen-

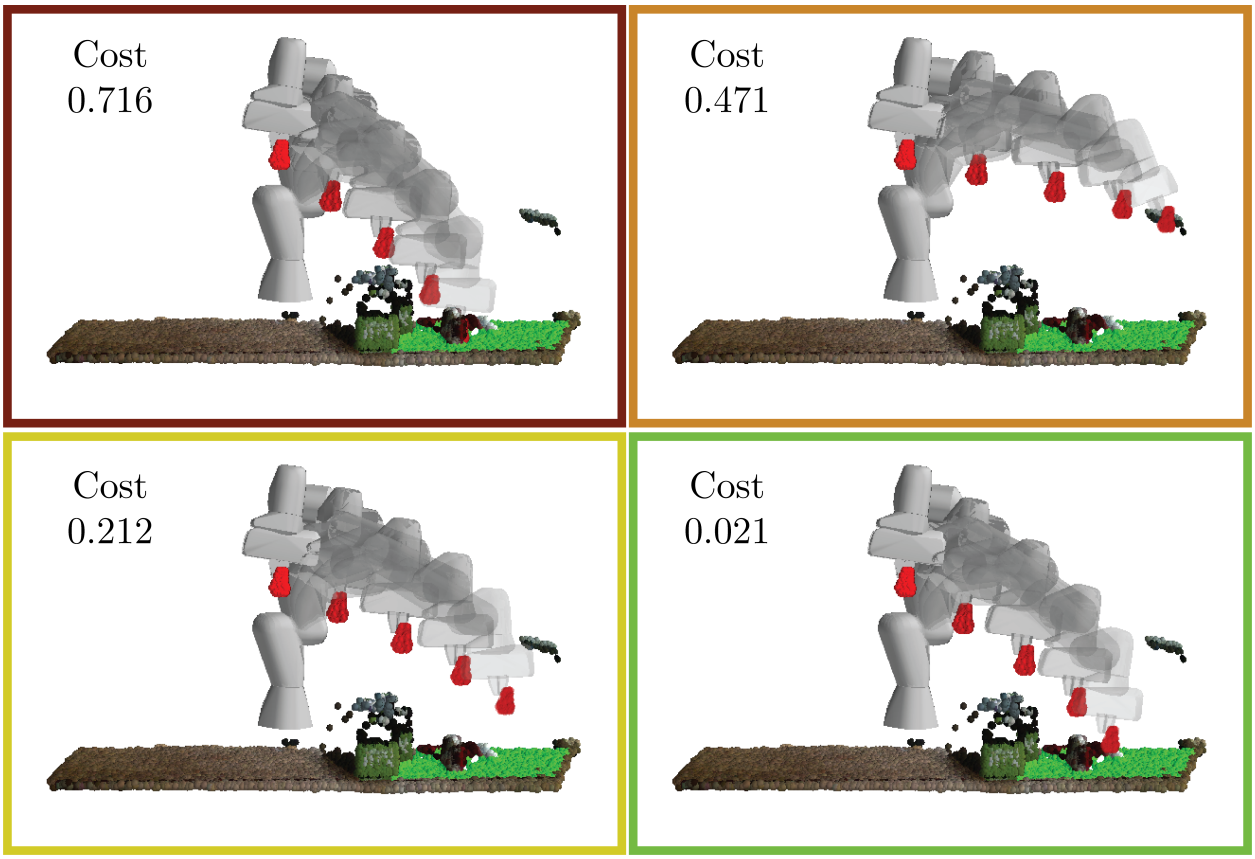


Figure 6.4: Sampled trajectories from the current robot configuration (solid) to the ending configuration (transparent) with outline colors indicating the cost of each trajectory. The lowest cost trajectory is collision-free and brings the object close to the placement area.

tation model and a particle filter to track the robot points and remove them. We segment the target object before grasping it and remove points within an object-tracking box that is transformed relative to the end-effector. Note that this approach does not account for any in-hand motion of the object, but avoids end effector occlusions after grasping. In simulation, we use ground truth segmentation masks for both the robot and the target object.

6.5 SceneCollisionNet Evaluation

We benchmark SceneCollisionNet against 4 baseline point cloud collision algorithms using synthetic data on two tasks: (1) a dataset of 1000 scene/object pairs with 2048 queries per scene/object pair where objects move on 16 linear trajectories through a scene, and (2) a dataset of 192,000 total grasps using the Franka Panda gripper [62], gathered from 5 scenes

and four object categories (mugs, cylinders, boxes, and bowls), where each scene has between 7 and 10 objects from the same category. In both tasks, ground truth collisions between the robot and the scene are calculated using FCL and the mesh models of the objects and robot in the scene. For each method and task, we compare the overall prediction accuracy and the computation time per query or grasp. We additionally report average precision (AP) scores, a weighted mean of precisions achieved at each recall threshold, for the trajectory benchmark and precision and recall for the grasp benchmark.

6.5.1 Baseline Algorithms

We benchmark SceneCollisionNet against both analytic and learned baselines. Marching Cubes baseline methods first create a mesh representation of the scene, object, or both from the raw point clouds [156]. Signed distance function (SDF) methods use the Kaolin library [69] for mesh to SDF conversion and GPU-based SDF evaluation. The baselines are:

1. **Marching Cubes + SDF Scene (MC+SDFS)**: The points belonging to the object point cloud are transformed and evaluated using the scene SDF. If any point has a negative distance, the object is in collision with the scene.

2. **Marching Cubes + SDF Object (MC+SDFO)**: The points belonging to the scene point cloud are transformed and evaluated using the object SDF. If any point has a zero or negative distance, the object is in collision with the scene.

3. **Marching Cubes + FCL (MC+FCL)**: Collisions between the scene meshes and object meshes are determined using the flexible collision library (FCL) [191]. For a fairer comparison, we parallelize this method across 10 processes. We also show performance when it receives points directly sampled from the underlying object meshes (FO).

4. **Pointnet Grid**: The scene is divided into coarse overlapping voxels; the object and each voxel are featurized using Pointnet++ set abstraction layers [208], but no voxel convolution layers, and trained on the same dataset used for SceneCollisionNet. Predictions are averaged across the 8 corresponding voxels. This algorithm can be viewed as an ablation of SceneCollisionNet that does not share information between voxels via voxel convolution.

6.5.2 Results

SceneCollisionNet outperforms all baselines in the linear trajectory collision environment, as shown in Table 6.1, with 9.8% and 15.8% gains in accuracy and AP score, respectively, over the FCL baseline. Additionally, SceneCollisionNet is nearly 20 times faster than the parallelized FCL baseline, taking only about 10 μs per collision query. SceneCollisionNet

Algorithm	Accuracy	AP	Time / Query (ms)
MC+SDFO	70.2%	0.651	27 ± 12
MC+SDFS	80.0%	0.781	24 ± 2
MC+FCL (10x)	75.4%	0.824	0.49 ± 0.06
MC+FCL (10x, FO)	83.4%	0.832	0.74 ± 0.13
PointNet Grid	76.7%	0.928	0.026 ± 0.035
SceneCollisionNet	**93.2%**	**0.990**	$\mathbf{0.010 \pm 0.002}$

Table 6.1: Benchmark results for 1000 scene/object pairs, with 16 linear trajectories and 2048 queries for each pair (2,048,000 total queries). SceneCollisionNet outperforms parallelized baselines that reconstruct meshes even from fully observed point clouds and a learned ablation that does not share information between voxels.

can predict over 500,000 queries in a single forward pass on an NVIDIA GeForce RTX 2080 Ti GPU, further reducing the time per query for large batches of queries.

The comparison with Pointnet Grid suggests the benefits of both the coarse voxel representation and the ability to share information between voxels via convolution. If the scene is encoded in the same way as the object (Pointnet++ layers only), the network fails to converge. When encoding coarse independent voxels using Pointnet++ set abstraction layers, but without using voxel convolutions to share information between them, the accuracy and AP scores are 16.5% and 6.2% lower, respectively.

Table 6.2 shows the results on the grasping benchmark. In addition to evaluating each model on the grasp poses, we evaluate the models on pre-grasp poses that are offset along the approach axis by 5 cm. The baseline methods slightly outperform or show similar performance to SceneCollisionNet on the grasp dataset with no offset, but SceneCollisionNet outperforms baselines on the 5 cm offset with a 1.5% improvement in accuracy as well as a 9.4% improvement in precision. These results suggest SceneCollisionNet can struggle to predict collisions for geometries that are very close to being in or out of collision, but dramatically improves with increasing distance between objects and learns underlying structure beyond the points in the scene, while the other methods are unable to account for gaps in point cloud data.

6.6 Policy Evaluation

We evaluate the proposed MPPI policy in both simulation and in physical tabletop scenes, recording the number of successful grasps and placements in each scenario as well as the time taken for picking and placing each object. We use $T = 300$, $H = 40$, and $\Sigma = 0.3 \cdot I$ and query the policy at 1 Hz.

Algorithm	Accuracy	Precision	Recall	Time (ms)
MC+SDFO	90.8/81.2	31.1/59.2	**95.4**/**98.9**	62
MC+SDFS	**94.4**/78.2	**32.0**/63.2	12.0/58.6	37
MC+FCL (10x)	**94.4**/80.4	27.8/63.6	10.8/68.8	0.27
SceneCollisionNet	92.4/**82.7**	21.2/**73.0**	19.3/71.8	**0.018**

Table 6.2: Benchmark results for 192,000 grasps across 20 scenes of 4 object categories and offsets of 0 cm / 5cm. The baselines slightly outperform SceneCollisionNet for the 0 cm offset, but SceneCollisionNet outperforms baselines in both accuracy and precision for the 5 cm offset while recalling over 70% of the collision-free grasps 15x faster.

Algorithm	Grasps	Placements	Time (min)
MC+FCL (10x)	109	92	164
SceneCollisionNet	**110**	**99**	**100**

Table 6.3: Simulation rearrangement results when using SceneCollisionNet and MC+FCL (10x) as collision checkers in the MPPI policy. SceneCollisionNet speeds up scene interaction and leads to more placements.

6.6.1 Simulation Evaluation

We compare SceneCollisionNet to MC + FCL (10x), the best performing baseline, as part of the MPPI policy in 10 simulated scenes with 10 objects each, drawn from a dataset of bowls, mugs, cylinders, and boxes. The objects are arranged randomly in a stable pose and not in collision, but may not be graspable. An object order is selected randomly, and the policy is given grasps on the specified target object to grasp and place that object in a different location on the table. The policy is given two attempts for each object, and if it is unable to pick or place the object, it moves on to the next target. In total, the policies interact with the scenes for 4.5 hours. Results in Table 6.3 suggest that SceneCollisionNet can dramatically speed up the MPPI policy, which can rearrange over half of the objects.

6.6.2 Physical Evaluation

We additionally evaluate the MPPI policy with SceneCollisionNet on a set of 10 physical tabletop scenes with a Franka Panda robot and an Intel RealSense LiDAR Camera L515. We divide the scenes into two categories: barrier scenes and rearrangement scenes, each with between 3 to 11 YCB objects. Examples are shown in Figures 6.1, 6.3 and 6.4. In barrier scenes, a tall box divides the scene and objects must iteratively be grasped and placed on

the opposite side of the barrier. Rearrangement scenes are similar to the simulated scenes, where objects are placed randomly on the table. However, in this case, placement is also restricted to a single side of the scene and both grasps and placements must be made among clutter. Placement becomes more difficult later in trials when the zone fills with objects.

In the four barrier scenes, the policy grasps 16/17 objects and places 12/17 objects successfully. Three failures were due to collisions in the trajectory or incorrect placement choice, one was due to object motion in the gripper, and one was due to the policy being unable to find a placement. In the rearrangement scenes, 25/27 grasps and 20/27 placements were successful. Of these failures, five were due to collision errors in the trajectory or placement position, with one failure each due to motion in the hand and no placements found.

The policy's grasping performance suggests it can consistently generate collision-free robot trajectories to specified goals in the presence of both clutter and a challenging divider that requires planning to significantly deviate from a straight-line trajectory. The placement performance indicates that the addition of checking collisions with an object in the hand makes finding collision-free trajectories more difficult, but the policy is still able to effectively reason about collisions along the trajectory and at the placement location.

6.7 Discussion

In this chapter, we presented a learned collision checking model that dramatically increases collision checking speeds between point clouds for motion planning in real-world rearrangement tasks. While we focus our evaluation on static scenes in this chapter, the supplementary video also shows a dynamic example where an obstacle is encountered during placement. While the MPPI and SceneCollisionNet framework can support a higher control frequency than 1 Hz, the policy reaction time is limited by the segmentation and point cloud processing; in future work, we will investigate the ability of the policy to react to more dynamic environments, other ways of generating candidate trajectories, and adaptation to constrained environments such as shelves and cabinets that are consistent with the lateral-access mechanical search problem.

Part III

Mechanical Search Policies

Chapter 7

Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter

Having explored improvements to both manipulation primitives and perception primitives relevant to robot mechanical search in the previous chapters, this part aims to connect the previous parts by formalizing the mechanical search problem and introducing high-level action selector policies. These high-level policies leverage both the low-level manipulation primitives discussed in Part I and the perception primitives discussed in Part II to decide which object or objects to interact with and how to manipulate them to efficiently reveal and extract the target object. In this chapter, we model the mechanical search problem as a partially-observed Markov decision process (POMDP) and introduce a set of baseline action selector policies to show how each of the primitives in the previous parts can be integrated as part of a mechanical search pipeline and quantify task difficulty in the overhead-access scenario.

Mechanical search describes a class of tasks where the goal is to locate and extract the target object, and poses challenges in visual reasoning, task, motion, and grasp planning, and action execution (see Figure 7.1). Significant progress has been made in recent years on sub-problems relevant to mechanical search. Deep-learning methods for segmenting and recognizing objects in images have demonstrated excellent performance in challenging domains [91, 175, 227] and new grasp planning methods have leveraged convolutional neural networks (CNNs) to plan and execute high-quality grasps directly from sensor data [141, 79, 164]. By combining object segmentation and recognition methods with action selectors that can effectively choose between different motion primitives in long horizon sequential tasks, multi-step policies can search for a target object and extract it from clutter.

In this chapter, we propose a framework that integrates perception, action selection, and manipulation policies to address a version of the mechanical search problem, with 3 contributions:

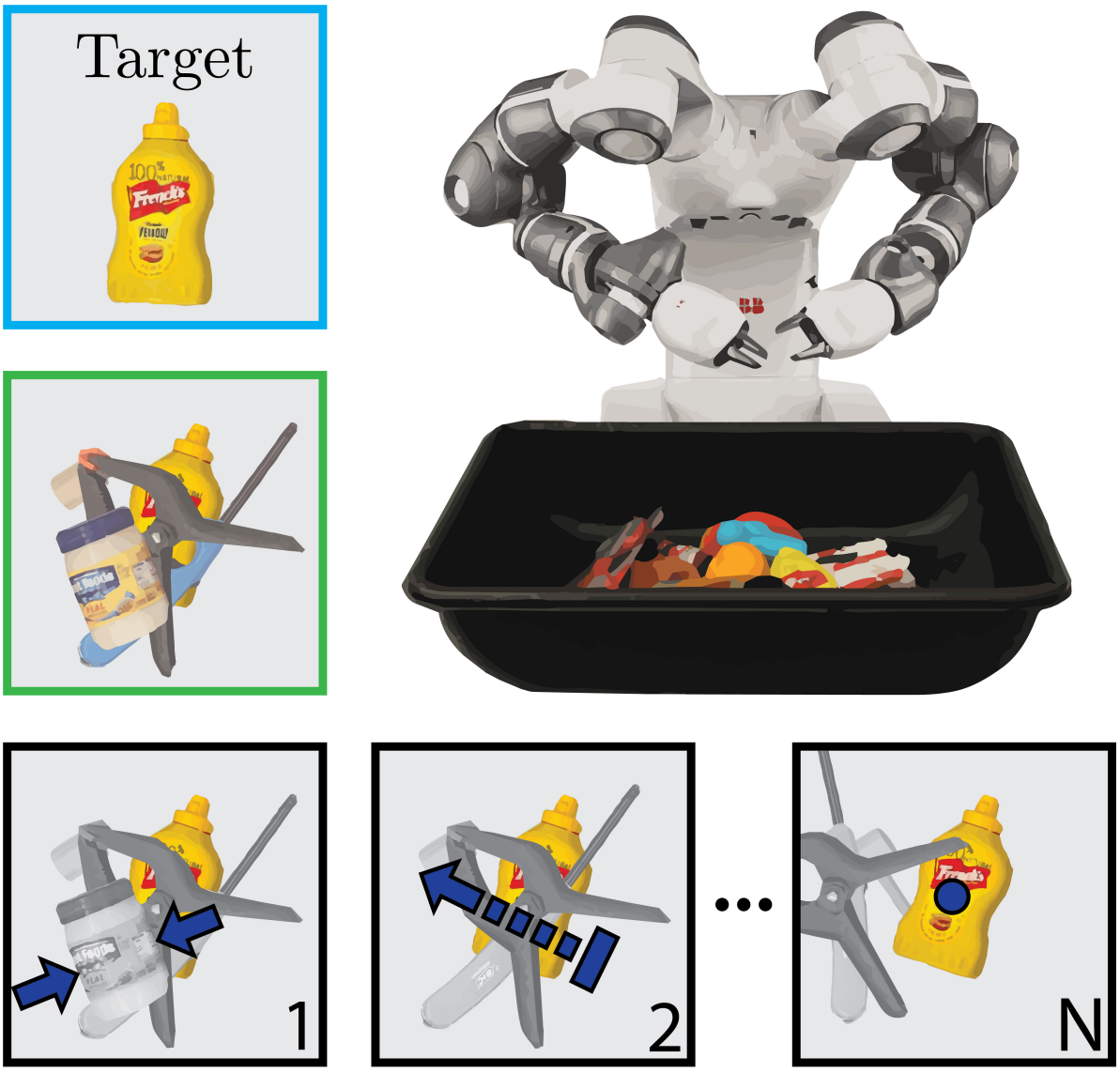


Figure 7.1: To locate and extract the target object from the bin, the system selects between 1) grasping objects with a parallel-jaw gripper, 2) pushing objects, or 3) grasping objects with a suction-cup gripper until the target object is extracted, a time limit is exceeded, or no high-confidence push or grasp is available.

1. A generalized formulation of the family of *mechanical search* problems and a specific version for retrieving occluded target objects from cluttered bins using a series of parallel jaw grasps, suction grasps and pushes.

2. An implementation of this version, using depth-based object segmentation, single-shot

image recognition, low-level grasp and push planners, and five action selection policies.

3. Data from simulation and physical experiments evaluating the performance of the five policies and that of a human supervisor. For simulated experiments, each policy was evaluated on a set of 1000 heaps of 10-20 objects sampled from 1600 3D object models; physical experiments used 50 heaps sampled from 75 common household objects.

7.1 Background and Related Work

Perception for Sequential Interaction Searching for an object of interest in a static image is a central problem in active vision [250, 213, 175]. There has also been work on optimizing camera positioning for improving visual recognition (i.e., *active perception* [10, 7]) and embodied interactions to explore (i.e., *interactive perception* [22, 81]). Mechanical Search differs from prior works in interactive perception in that it deals with long grasping sequences.

Recent deep learning based methods achieve remarkable success in segmentation of RGB [214, 200] and depth images [37], as well as in localizing visual templates in uncluttered [127, 254] and cluttered scenes [227, 175]. Furthermore, one-shot learning approaches using Siamese Networks for matching a novel visual template in images [127, 254] can translate well to pattern recognition in clutter [227, 175]. We build on Mask R-CNN [91] by training a variant for depth-image based instance segmentation and leverage a Siamese network for target template matching for localization.

Grasping and Manipulation in Clutter Past approaches to this problem can be broadly characterized as model-based with geometric knowledge of the environment [17, 235, 178] and model-free with only raw visual input [224, 117, 162]. Recent studies have leveraged CNNs for casting grasping as a supervised learning problem with impressive results [141, 164, 284, 104, 253, 65, 110]. Pushing and singulation can facilitate grasping in cluttered scenes [34, 92, 48]. Techniques for grasping in clutter, either as open-loop prediction or as closed-loop continuous control, have been studied but have not dealt with the multi-step plans that are critical to attain successful grasps on occluded or inaccessible target objects [110, 179]. In contrast, we formulate Mechanical Search as an interactive search problem in significant clutter, necessitating a multi-step process combining grasping and pushing actions.

Sequential Decision Making Sequential composition of primitives to enable long-term environment interaction has often been approached through hierarchical decomposition of control policies to manage task complexity. The idea of using hierarchical models for complex tasks has been widely explored in both reinforcement learning and robotics [242, 126, 241]. Training such multi-level models can be computationally expensive and has been limited to either simulated or elementary physical tasks [58, 269].

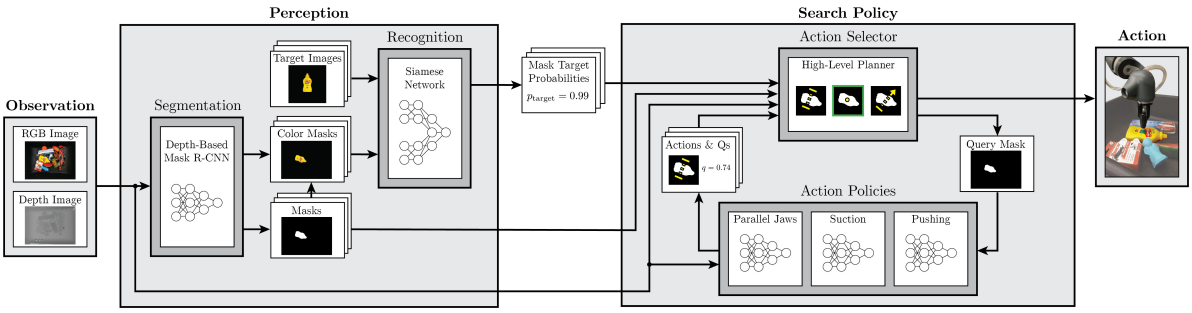


Figure 7.2: System architecture. At each timestep, the RGB-D image of the bin is segmented using a variant of Mask R-CNN trained on synthetic depth images. The colorized masks are each assigned a probability of belonging to the target object using a Siamese Network as a pattern comparator. These masks are then fed to an action selector, which chooses which object to manipulate and passes that object mask as context to all the action policies. These policies each compute an action with an associated quality score and pass them back to the action selector, which then chooses an action and executes it on the physical system. This process continues until the target object is retrieved. In simulation, the perception pipeline is removed and planners operate on full-state information rather than object masks.

Search Based Methods Traditional task planning approaches abstract away perception and focus on high-level task plans and low-level state spaces [68, 235]. For instance, in robotic applications, hierarchical methods have been used to learn task planning strategies while abstracting away low-level motion planning [198, 235, 262]. However, high-level planning requires complete domain specification a priori, and complex geometric and free space reasoning make this approach applicable only to uncluttered environments with few objects, such as a tabletop with one or two objects.

A similar problem has been studied in the context of mobility under problem domains of target-driven and semantic visual navigation [288, 184, 83]. These studies look at finding visual targets in unknown environments without maps through sensory pattern matching. The work by Gupta et al. [81] is the closest to the approach considered in this chapter. Their work also considers the problem of searching for a specific object using pushing and grasping actions, but when the objects are arranged in a shelf. We consider significantly more cluttered settings, while also executing temporally extended manipulation policies.

7.2 Mechanical Search: Problem Formulation

In mechanical search, the objective is to retrieve a specific target object (x^*) from a physical environment (E) containing a variety of objects X within task horizon H while minimizing time. The agent is initially provided with a specification of the target object in the form of images, text description, a 3D model, or other representation(s). We can frame the general

problem of mechanical search as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$.

- States ($\mathcal{S}$). A bounded environment E at time t containing N objects $\mathbf{s}_t = \{\mathcal{O}_{1,t}, \ldots, \mathcal{O}_{N,t}\}$. Each object state $\mathcal{O}_{i,t}$ includes a ground truth triangular mesh defining the object geometry and pose. Each state also contains the pose and joint states of the robot as well as the poses of the sensor(s).
- Actions ($\mathcal{A}$). A fixed set of parameterized motion primitives.
- Transitions ($\mathcal{T}$). Unknown transition probability distribution $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,\ \infty)$.
- Rewards ($\mathcal{R}$). Function given by $R_t = R(\mathbf{s}_t, \mathbf{a}_t) \to \mathbb{R}$ at time t that estimates the change in probability of successfully extracting the target object $x^* \in X$ within task horizon H.
- Observations ($\mathcal{Y}$). Sensor data, such as an RGB-D image, y_t from robot's sensor(s) at time t (see Figure 7.1).

In this chapter, we focus on a specific version of mechanical search: extracting a target object specified by a set of k RGB images from a heap of objects in a single bin while minimizing the number of actions needed. For this problem, we precisely specify the observations, the action set, and the reward function. All other aspects of the problem formulation are sufficiently captured by the general POMDP formulation above.

- Observations. An RGB-D image from an overhead camera.
- Actions.
  - **Parallel Jaw Grasping**: A center point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ between the jaws, and an angle in the plane of the table $\phi \in \mathbb{S}^1$ representing the grasp axis [164].
  - **Suction Grasping**: A target point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and spherical coordinates $(\phi, \theta) \in \mathbb{S}^2$ representing the axis of approach of the suction cup [165].
  - **Pushing**: A linear motion of the robot end-effector between two points $\mathbf{p}$ and $\mathbf{p}' \in \mathbb{R}^3$.
- Reward. Let v_t, derived from y_t, denote the estimated grasp reliability on the target object. An intuitive reward function would be the increase in estimated grasp reliability on the target object:

  $$R(s_t, a_t) = v_{t+1} - v_t$$

  The policies used in this chapter do not directly optimize this reward function because it is difficult to compute; instead, they continue to remove and push objects via heuristic methods until the target object is extracted. In future work, we will develop methods to approximate this function.

7.3 Perception and Decision System

As shown in Figure 7.2, we implement the system both in simulation and for physical experiments via a pipeline for perception and policy execution.

7.3.1 Perception

The system first processes the RGB-D image into a set of segmentation masks using an object instance segmentation pipeline trained on synthetic depth images. Then, a Siamese network is used to attempt to identify one of the masks as the target object, and a target mask is returned if a high confidence match is found. If no high confidence match is found, the perception system reports that no masks match the target object.

Object Instance Segmentation We first compute a mask for each object instance. Each mask is a binary image with the same dimensions as the input RGB-D image. These masks are computed with SD Mask R-CNN, a variant of Mask R-CNN trained exclusively on synthetic depth images [51]. It converts a depth image into a list of unclassified binary object masks, and generalizes well to arbitrary objects without retraining. Recent results suggest that depth cues alone may be sufficient for high-performance segmentation, and this network's generalization capabilities are beneficial in a scenario where only the target object is known and many unknown objects may be present.

Target Recognition Next, the set of masks is combined with the RGB image to create color masks of each object. Each of the m color masks is cropped, scaled, rotated, and compared to each of the k images in the target object image set using a Siamese network [127]. For each pair of inputs, the Siamese network outputs a recognition confidence value between 0 and 1, with a mask's recognition confidence score set to the maximum recognition confidence value over the k target object images. If the mask with the highest score has a score above recognition confidence threshold t_r, the mask is labeled as the target object. Otherwise, we report that no masks match the target object. See the appendix for training and implementation details.

7.3.2 Search Policy

Given the RGB-D image and the output of the perception pipeline, the system executes the next action in the search procedure by selecting the object to act on and the action to perform on it. Our approach to the version of Mechanical Search described in Section 7.2 for bin picking includes searching for actions in three continuous spaces (parallel jaw grasp, suction grasp and push). However, more complex versions of Mechanical Search (e.g., search for an object in a house) could have even more complex search spaces (e.g., navigation). To allow our method to scale to these more complex versions, we propose a hierarchical approach: (1) an action selector that queries a set of action policies on a specific object for

a particular action and associated quality metric and (2) action policies that correspond to the possible actions in the problem formulation.

Action Selection The search policy first determines which object masks to send to the action policies. Then, using the actions and associated quality metric returned by the low level policies, the high level planner determines whether to execute the action in the environment.

The action selector takes as input from the perception system the set of all m visible object masks ($[o_1, \ldots o_m]$), possibly including an object mask that is positively identified as the target object (o_T), from the perception system. It then selects an action policy and a goal object, o_{goal}, from $[o_1, \ldots o_m]$ and sends the action policy a query $q(o_{goal})$. The action policy p_i responds with an action $a_i = p_i(o_{goal})$ and a quality metric $Q(a_i, o_{goal})$ for the action, which is used to decide whether to execute the action.

Action Policies Each action policy p_i takes as input an object mask from the action selector (o_{goal}) and the RGBD image observation and returns an action $a_i = p_i(o_{goal})$ and a quality metric $Q(a_i, o_{goal})$. In simulation, the object masks and depth images are generated from ground-truth renderings of each object, while in physical experiments, depth images are obtained using a depth sensor and object masks are generated by the perception pipeline. The set of action policies in our system are:

Parallel Jaw Grasping In simulation, pre-computed grasps are indexed from a Dex-Net 1.0 parallel-jaw grasping policy [163], and the grasp with the highest predicted quality on o_{goal} is returned as the action along with an associated quality metric. For physical experiments, parallel-jaw grasps are planned using a Dex-Net 2.0 Grasp Quality CNN (GQ-CNN) [164]. To plan grasps for a single object in a depth image, grasp candidate sampling is constrained to the goal object's segmentation mask. The GQ-CNN evaluates each candidate grasp and returns the grasp with the highest predicted quality and its associated quality metric.

Suction Grasping For simulation experiments, grasp planning is done with a Dex-Net 1.0 suction grasping policy [163]. For physical experiments, suction cup grasps are planned with a Dex-Net 3.0 GQ-CNN [164], with mask-based constraints to plan grasps only on the goal object's segmentation mask. The GQ-CNN evaluates each candidate grasp and returns the grasp with the highest predicted quality and its associated quality metric.

Pushing The pushing action policy, similar to that in [48], selects $\mathbf{p}'$ as the most free point in the bin. This point is computed by taking the signed distance transform of a binary mask of the bin walls and objects, finding the pixel with the maximum signed distance value, and deprojecting that pixel back into $\mathbb{R}^3$. Given an object to push, $\mathbf{p}$ is then selected so that the gripper is not in collision at $\mathbf{p}$, the line from $\mathbf{p}$ to $\mathbf{p}'$ passes through the object's center of mass, and the push direction is as close as possible to the direction of the most free point in

the bin. The pushing policy returns the push satisfying the above constraints as its action if one exists. The returned quality metric is 1 if a valid push exists and 0 if not.

7.4 Action Selection Policies

All action selection methods use input from the perception system to generate a specific object priority list. Each action selection method generates a priority list in a different way but all have the same action execution criteria. For all action selection methods described here, a grasp action is executed if the quality metric returned by the action policy exceeds $t(o)$, the grasp confidence threshold for object mask o. The grasp confidence threshold for the object mask positively identified as the target object o_T is given by $t(o_T) = t_{\text{thresh}}$. For policies without pushing, $t(o) = t_{\text{thresh}}, \forall o$, while for policies with pushing, $t(o) = t_{\text{high}}, \forall o \neq o_T$. Policies with pushing can be more conservative in their choice of grasps, so they use a higher grasp confidence threshold t_{high} for non-target objects. A push action is performed if a valid push is found (quality 1). Details on parameters used can be found in the appendix.

Each action selection method iterates through its priority list, queries the grasping action policies for each object mask, and executes the returned action with the highest quality metric among the two grasping policies if it satisfies the action execution criteria. If the target object is grasped, the policy terminates and reports a success. If no grasping action satisfies the criteria and the policy does not have pushing, the policy terminates and reports a failure. If the policy does have pushing, it iterates through its priority list, queries the pushing action policy for each object mask, and executes the first action that satisfies the criteria. If no pushing action satisfies the criteria, or if a pushing action has been selected more than three consecutive times, the policy terminates with a failure.

Action Selection Methods The action selection methods are distinguished by whether or not they have pushing as an available action policy and by their generated object priority list:

1. **Random Search:** Prioritizes objects randomly, with no preference for the target object mask (o_T).

2. **Preempted Random Search (with and without pushing):** Always prioritizes o_T and prioritizes other objects randomly.

3. **Largest-First Search (with and without pushing):** Always prioritizes o_T and ranks the other objects by their visible area. If the target object isn't visible, this strategy will increase the likelihood of removing objects that may be occluding the target object.

Termination Criteria In addition to the termination criteria outlined above (terminate and return success if target object grasped, return failure if no good grasp/valid push found), we impose two more termination conditions on our policies which cause them to return a failure: (1) $2N$ timesteps have elapsed, where N is the initial number of objects in the bin and (2) The target object is inadvertently removed from the work space when another object is grasped or pushed.

7.5 Experiments

7.5.1 Simulation

Heap Generation Three datasets of simulated heaps are generated, each containing 1000 heaps of N objects, for $N \in \{10, 15, 20\}$. Then, using the Bullet Physics Engine [46], sampled objects are dropped one by one into the bin, and the target object is chosen to be the most occluded object. Please refer to the appendix for further details.

Rollouts To simulate grasp actions, we use the same approach as in [162]: using wrench space analysis, we determine whether or not an object can be lifted from the heap [203, 196]. If the object can be lifted, a constant upward force is applied to the object's center of mass until it leaves the bin, and the remaining objects are allowed to come to rest. To simulate push actions, we check that the gripper can be placed in the starting location without collisions, and only execute pushes if this is the case. Then, we place a 3D model of the closed gripper in the physics simulator and move it from the start point to the end point of the push, as in [48].

7.5.2 Physical

Heap Generation We randomly sample 50 heaps of 15 items each from a set of 75 common household objects with relatively simple shapes, such as boxes and cylinders, as well as more complex geometries, such as plastic climbing holds and scissors (see Figure 7.3). We also include several 3D-printed items, which present a challenge for both segmentation and target object recognition due to their unusual shapes and uniform texture. A target object is chosen at random from each 15 item heap. Then, in order to generate adversarial bin configurations, each rollout is initialized by first shaking the target object in a box to randomize its pose and dumping into the center of the bin, and then shaking the other fourteen objects and pouring them over the target object.

Policy Rollouts We execute pushing and grasping actions on an ABB YuMi robot equipped with suction-cup and parallel-jaw grippers (see Figure 7.3). Actions generated by the search policy are transformed into a sequence of poses for the robot's end-effectors, and we use ABB's RAPID linear motion planner and controller to execute these motions.

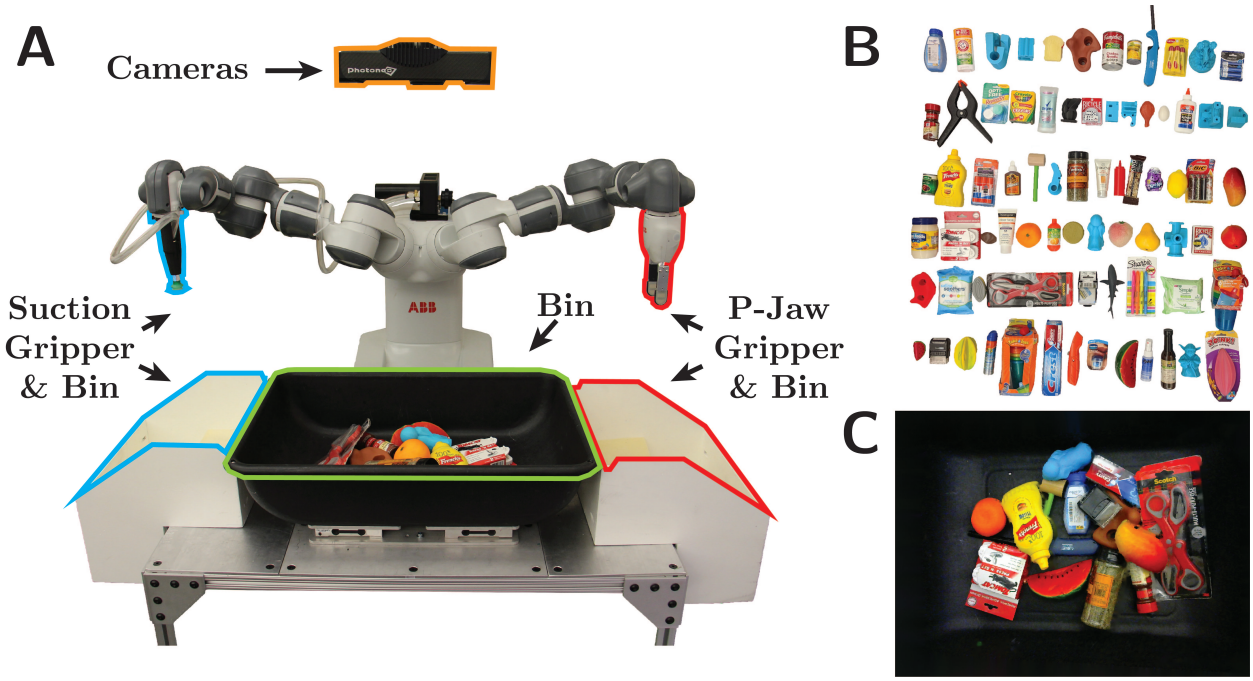


Figure 7.3: **(A)** Front view of the robot and bin setup. The black bin is the primary bin in which heaps are initialized, and the white bins provide space for the robot to deposit grasped items. **(B)** The 75 objects used in physical experiments. **(C)** A sample heap of 15 objects used in the physical experiments.

Human Supervisor Rollouts For comparison, we also benchmark a human supervisor's performance as an action selector. At each timestep, the human is asked to draw a mask in the scene on which to plan a push or a grasp. Then, grasps and pushes are planned and executed on the specified mask with the same action primitives described above (parallel jaw grasps, suction grasps, linear pushes). Thus, the human is limited by the available action primitives, but is allowed to use their own judgement for perceptual reasoning and high level action planning.

7.5.3 Evaluation Metrics

We evaluate each policy according to its reliability and efficiency in target object retrieval. Reliability is defined as the frequency at which the target object is successfully extracted, while efficiency is defined as the mean number of actions taken to successfully extract the target object. For each experiment, we recorded the number of successes and failures, as well as statistics regarding the number of actions taken.

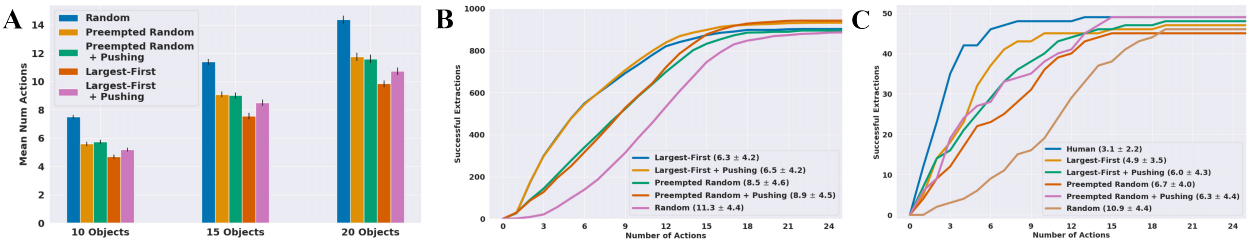


Figure 7.4: Performance of policies on **(A)** simulated heaps of 10, 15, and 20 objects over a total of 15,000 simulated rollouts and 300 physical rollouts, **(B)** simulated heaps of 15 objects, and **(C)** real heaps of 15 objects. The largest-first search policies are the most efficient, and are able to extract the target object in the least number of actions. All policies have similar reliability, although pushing shows potential to avoid more failures in simulation. The human was allowed to look at the RGBD image inputs and choose an object to push or grasp. Means and standard deviations for successful extractions are shown in parentheses for each policy.

7.6 Results

7.6.1 Simulation Results

We tested each action selection policy in simulation with heaps generated using the method described in Section 7.5 and running until termination on each heap. A total of 15,000 simulation experiments were conducted over all policies. The results for each of the five policies are shown in Figure 8.6, and a detailed breakdown of each policy can be found in the appendix. Figure 8.6(A) shows the mean number of actions needed for each policy as a function of heap size. These results suggest that extracting the target object becomes more difficult as heap size increases and the mean number of actions needed for each policy appears to scale linearly with heap size, although the rate of increase is not constant across policies.

Figure 8.6(B) shows cumulative successful extractions for a given amount of actions (number of successful extractions in that many actions or fewer). All policies have success rates of 90% or higher on 15 object heaps. The cumulative success plot suggests that grasping the target object when possible provides improvement over the random policy, and that prioritizing larger object masks when the target object is inaccessible further increases efficiency. The largest-first policies successfully extract the target object within 5 or fewer actions on 50% of the heaps, while the preempted random and random policies only do so for 30% and 10% of heaps respectively.

Results also suggest that pushing can increase overall success rate, as policies that included pushing succeeded on at least 3% more heaps than those without pushing. For policies with pushing, only 5% of all actions attempted are pushes, as opposed to 29% parallel jaw

grasps and 66% suction grasps, on ten object heaps. The percentage of push actions decreases further when increasing the number of objects to just 3% for 15 object heaps. We suspect that the reason pushes are selected so rarely is because the pushing primitive is designed to execute a sequence of linear pushes to singulate a particular object, rather than flatten a heap of objects. While the former directly addresses the objective of the push, the latter is much easier to achieve in practice, since in many cases, especially with many objects in the bin, it may be almost impossible to plan collision-free pushes that successfully singulate the object of interest.

In simulation, failures account for 6-12% of all rollouts. These failures fall into three categories: 1) the policy fails to plan an action (e.g., no actions are available on any remaining objects with a quality metric above the threshold), 2) the target object is inadvertently removed from the bin, despite an attempted action that was not a grasp on the target object, or 3) the policy reaches the maximum number of timesteps given to extract the object ($2N$ timesteps, where N is the initial number of objects in the heap).

Failure mode (1) accounts for 85-90% of all failures, while (2) accounts for nearly all of the remaining failures. Mode (1) typically happens because objects are moved to the corners of the bin, making it difficult to plan collision-free grasps. Mode (2) occurs more often for policies that include pushing actions, since pushing in clutter can occasionally lead to objects being pushed up and over the bin walls. It is also possible that the target object is removed from the work space when another object is grasped. Mode (3) never actually occurs in any of our experiments, and the maximum timesteps cutoff is intentionally set high to exhaust the policy of actions.

7.6.2 Physical Results

Figure 8.6(C) shows results on the physical system. A total of 300 physical experiments were conducted over all policies. All policies retrieved the target object within the given number of timesteps at least 90% of the time, and success rates were not statistically different between policies. However, the cumulative success curves suggest similar trends to those seen in the simulation results, with the largest-first policies outperforming the preempted random and random policies in terms of efficiency. The largest-first policies again successfully extract the target object within 5 or fewer actions on on 50% of the heaps, while the preempted random and random policies only do so for 40% and 10% of heaps respectively.

Stochasticity in the initial bin state in physical experiments can result in varying difficulty for different policies on the same heap. For example, a target object may be completely covered by other objects when one policy is presented with a given heap, but for another policy, the target may be partially or fully visible in the initial state. Thus, policies may occasionally get "lucky" or "unlucky" with respect to the target object visibility in the initial state, which may account for some increased variance in the physical results.

Failure cases for the physical heaps are very similar to those in simulation: 93% of failures arise from the policy being unable to plan an action. However, in physical experiments, out of the 300 heaps evaluated for all policies, only 1 rollout failed due to timing out. Failure

to plan actions is almost always due to the target object lying flat on the bottom of the bin (e.g., the dice, sharpie pens, or another blister-pack object), making it difficult to obtain accurate segmentation. Another common reason for failure to plan actions is when no mask is identified as the target object, which often occurs for 3D printed objects.

7.6.3 Action-Limited Human Supervisor

The human supervisor outperforms all policies presented here, requiring an average of just 3.1 actions to extract the target object due to more intelligent action selection. Specifically, we noticed that a human operator chose to push far more frequently (26% of all actions, compared to 6% for the other action policies with pushing), especially when objects were heaped in the center of the bin and the target was not visible. These pushes tend to spread many objects out over the bottom of the bin, as opposed to a grasping action that would remove only a single object from the top of the heap.

7.7 Discussion and Future Work

In this chapter, we presented a general formulation for mechanical search problems and described a framework for solving the specific problem of extracting a target object from a cluttered bin. While the best action selection method (largest-first) is much more efficient than random search and provides a solid baseline, a human selecting the low-level actions can still achieve 37% higher efficiency by pushing significantly more effectively and often. We will explore how reinforcement learning in simulation can address this gap. We conjecture that more effective push primitives can be learned from simulation.

Chapter 8

X-Ray: Mechanical Search for an Occluded Object by Minimizing Support of Learned Occupancy Distributions

Although the mechanical search policies in the previous chapter can efficiently extract a target object from cluttered heaps, the heuristic action selector policies lack an intuition for where the target object is likely to be hidden. For humans, much of this intuition comes from familiarity with everyday objects; our priors of object geometries and semantics are developed over years of interactions with a diverse set of objects. In this chapter, we aim to improve on the mechanical search policies presented in the previous chapter via the X-Ray policy, which builds an explicit representation for object occlusions and predicts a distribution over likely locations for the target object. We draw on recent work on shape completion to reason about occluded objects [252, 204] and work on predicting multiple pose hypotheses [167, 218]. X-Ray combines occlusion inference and hypothesis predictions to estimate an occupancy distribution for the bounding box most similar to the target object to estimate likely poses – translations and rotations in the image plane. X-Ray can efficiently extract the target object from a heap where it is fully occluded or partially occluded (Figure 8.1). We show that X-Ray shifts the mechanical search burden from the action selector policy towards perception: the occupancy distribution is a powerful intermediate representation that densifies the mechanical search reward and induces a simple policy based on an imagewise dot product of segmentation masks and occupancy distribution images.

This chapter makes four contributions:

1. X-Ray (maXimize Reduction in support Area of occupancY distribution): a mechanical search policy that minimizes support of learned occupancy distributions.
2. An algorithm for estimating target object occupancy distributions using a set of neural networks trained on a dataset of synthetic images that transfers seamlessly to real

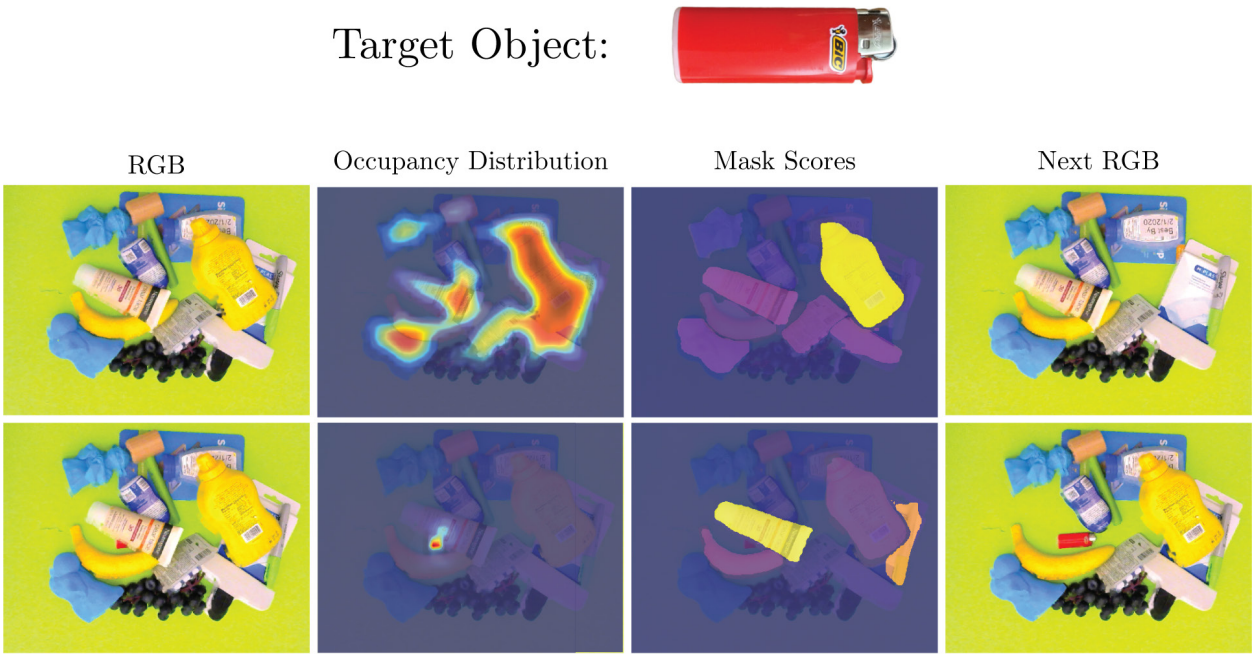


Figure 8.1: Mechanical search with a fully occluded target object (top row) and a partially occluded target object (bottom row). We predict the target object occupancy distribution, which depends on the target object's visibility and the heap (second column). Each pixel value in the distribution image corresponds to the likelihood of that pixel containing part of the target object. X-Ray plans a grasp on the object that minimizes the estimated support of the resulting occupancy distribution to minimize the number of actions to extract the target object. We show two nearly-identical heaps; in the fully occluded case, X-Ray grasps the mustard bottle whereas in the partially occluded case, the policy grasps the face lotion (third column), resulting in the respective next states (fourth column).

images.

3. A synthetic dataset generation method and 100,000 RGBD images of heaps labeled with occupancy distributions for a single partially or fully occluded target object, constructed for transfer to real images.

4. Experiments comparing the mechanical search policy against two baselines in 1,000 simulated and 20 physical heaps that suggest the policy can reduce the median number of actions needed to extract the target object by 20% with a simulated success rate of 87% and physical success rate of 100%.

8.1 Related Work

8.1.1 Pose Hypothesis Prediction

There is a substantial amount of related work in computer vision on 3D and 6D pose prediction of both known and unknown objects in RGB, depth, and RGBD images [118, 265, 148, 95]. Many of these papers assume that the target objects are either fully visible or have minor occlusions. In addition, many assume that there is no ambiguity in object pose due to self-occlusion or rotational symmetry of the object, as these factors can significantly decrease performance for neural network-based approaches [44]. Recent work has attempted to address the pose ambiguity that results from object geometry or occlusions by restricting the range of rotations [211] predicting multiple hypotheses for each detected object [218, 167]. Rupprecht et al. [218] find that refining multiple pose hypotheses to a 6D prediction outperforms single hypothesis predictions on a variety of vision tasks, such as human pose estimation, object classification, and frame prediction. Manhardt et al. [167] note that directly regressing to a rotation for objects with rotational symmetries can result in an averaging effect where the predicted pose does not match any of the possible poses; thus, they predict multiple pose hypotheses for objects with pose ambiguities to better predict the underlying pose and show Bingham distributions of the predicted hypotheses. However, only minor occlusions are considered and since ground truth pose distributions are not available for these images and objects, comparisons for continuous distributions can only be made qualitatively. Predicting multiple hypotheses or a distribution to model ambiguity has also been applied to gaze prediction from facial images [206], segmentation [128], and monocular depth prediction [275]. In contrast to these works, we learn occupancy distributions in a supervised manner.

8.1.2 Object Search

There has been a diverse set of approaches to grasping in cluttered environments, including methods that use geometric knowledge of the objects in the environment to perform wrench-based grasp metric calculations, nearest-neighbor lookup in a precomputed database, or template matching [17, 178, 163], as well as methods using only raw sensor data [117, 224], commonly leveraging convolutional neural networks [111, 104, 141]. While multi-step bin-picking techniques have been studied, they do not take a specific target object into account [162].

Kostrikov, Erhan, and Levine [129] learn a critic-only reinforcement learning policy to push blocks in a simulated environment to uncover an occluded MNIST block. Zeng et al. [282] train joint deep fully-convolutional neural networks to predict both pushing and grasping affordances from heightmaps of a scene containing multicolored blocks, then show that the resulting policy (VPG) can separate and grasp novel objects in cluttered heaps. The policy can be efficiently trained on both simulated and physical systems, and can quickly learn elegant pushes to expand the set of available grasps in the scene. Yang, Liang, and Choi

[276] train similar grasping and pushing networks as well as separate explorer and coordinator networks to address the exploration/exploitation tradeoff for uncovering a target object. Their policy learns to push through heaps of objects to find the target and then coordinate grasping and pushing actions to extract it, outperforming a target-centered VPG baseline in success rate and number of actions. Both approaches can generalize to objects outside the training distribution, although they are evaluated on a limited set of novel objects, and Yang, Liang, and Choi [276] separate the cases where the target object is partially occluded and fully occluded. Additionally, we focus only on grasping actions, as some mechanical search environments may be constrained or objects may be fragile.

Recently, several approaches to the mechanical search problem have been proposed, both in tabletop and bin picking environments. Price, Jin, and Berenson [204] propose a shape completion approach that predicts occlusion regions for objects to guide exploration in a tabletop scene, while Xiao et al. [266] implement a particle filter approach and POMDP solver to attempt to track all visible and occluded objects in the scene. However, 75% of the objects in Price, Jin, and Berenson [204]'s evaluation scenes are seen in training and Xiao et al. [266]'s method requires models of each of the objects in the scene. We benchmark our policy on a variety of non-rigid, non-convex household objects not seen in training and require no object models. In previous work, Danielczuk et al. [49] proposed a general mechanical search problem formulation and introduced a two-stage perception and search policy pipeline. In contrast, we introduce a novel perception network and policy based on minimizing support of occupancy distributions that outperforms the methods introduced in [49].

8.2 Problem Statement

We consider an instance of the mechanical search problem where a robot must extract a known target object from a heap of unknown objects by iteratively grasping to remove non-target objects. The objective is to extract the target object using the fewest number of grasps.

8.2.1 Assumptions

- One known target object, fully or partially occluded by unknown objects in a heap on a planar workspace.
- A robot with a gripper, an overhead RGBD sensor with known camera intrinsics and pose relative to the robot.
- A maximum of one object is grasped per timestep.
- A target object detector that can return a binary mask of visible target object pixels when queried.

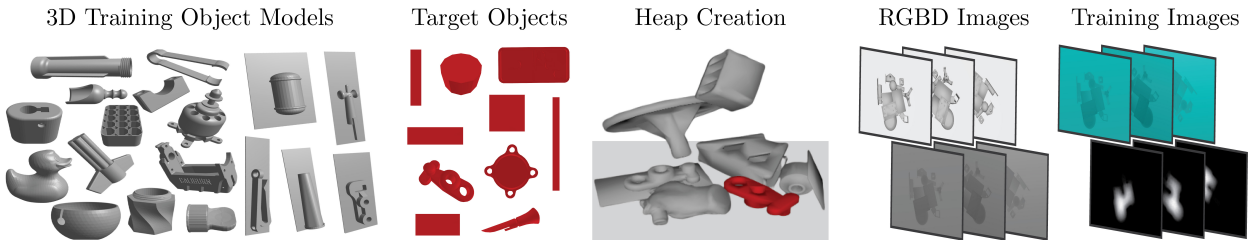


Figure 8.2: Training dataset generation for learning the occupancy distribution function. Each dataset image is generated by sampling $N = 14$ object models from a dataset of 1296 CAD models. The target object (colored red) is dropped, followed by the N other objects (colored gray), into a planar workspace using dynamic simulation. Camera intrinsics and pose are sampled from uniform distributions centered around their nominal values and an RGBD image is rendered of the scene. The augmented depth image (top right), consisting of a binary target object modal mask and a two-channel depth image, is the only input used for training for seamless transfer from simulation to real images. The ground truth target object distribution is generated by summing all shifted amodal target object masks whose modal masks correspond with the target object modal mask.

8.2.2 Definitions

We define the problem as a partially-observable Markov decision process (POMDP) with the 7-tuple $(S, A, T, R, \Omega, O, \gamma)$ and a maximum horizon H:

- **States** (S): A state $\mathbf{s}_k$ at timestep k consists of the robot, a static overhead RGBD camera, and a static bin containing $N + 1$ objects, target object $\mathcal{O}_t$ and distractor objects $\{\mathcal{O}_{1,k}, \mathcal{O}_{2,k}, \ldots, \mathcal{O}_{N,k}\}$. No prior information is known about the N distractor objects.
- **Actions** (A): A grasp action $\mathbf{a}_k$ at timestep k executed by the robot's gripper.
- **Transitions** (T): In simulation, the transition model $T(\mathbf{s}_{k+1} \mid \mathbf{a}_k, \mathbf{s}_k)$ is equivalent to that used by Mahler et al. [162] and uses PyBullet [46] for dynamics. On the physical system, next states are determined by executing the action on a physical robot and waiting until objects come to rest.
- **Rewards** (R): The reward $r_k = R(\mathbf{s}_k, \mathbf{a}_k, \mathbf{s}_{k+1}) \in \{0, 1\}$ is 1 if the target object is successfully grasped and lifted from the bin, otherwise the reward is 0.
- **Observations** (Ω): An observation $\mathbf{y}_k \in \mathbb{R}_+^{h \times w \times 4}$ at timestep k consists of an RGBD image with width w and height h taken by the overhead camera.
- **Observation Model** (O): A deterministic observation model $O(\mathbf{y}_k \mid \mathbf{s}_k)$ is defined by known camera intrinsics and extrinsics.

- **Discount Factor** (γ): To encourage efficient extraction of the target object, $0 < \gamma < 1$.

We also define the following terms:

- Modal Segmentation Mask ($\mathcal{M}_{m,i}$): the region(s) of pixels in an image corresponding to object $\mathcal{O}_i$ which are visible [112].
- Amodal Segmentation Mask ($\mathcal{M}_{a,i}$): the region(s) of pixels in an image corresponding to object $\mathcal{O}_i$ which are visible or invisible (occluded by other objects in the image) [112].
- The oriented minimum bounding box is the 3D box with the minimum volume that encloses the object, subject to no orientation constraints. We use this box to determine scale and aspect ratio for a target object.
- The *occupancy distribution* $\rho \in \mathcal{P}$ is the unnormalized distribution describing the likelihood that a given pixel in the observation image contains some part of the target object's amodal segmentation mask.

8.2.3 Objective

Given this problem definition and assumptions, the objective is to find a policy π_θ^* with parameters θ that maximizes the expected discounted sum of rewards:

$$\theta^* = \arg\max_\theta \ \mathbb{E}_{p(\tau|\theta)} \left[\sum_{k=0}^{H-1} \gamma^k R(\mathbf{s}_k, \pi_\theta(\mathbf{y}_k), \mathbf{s}_{k+1}) \right]$$

where $p(\tau \mid \theta) = \mathbb{P}(s_0)\Pi_{k=0}^{H-1} T(\mathbf{s}_{k+1} \mid \pi_\theta(\mathbf{y}_k), \mathbf{s}_k) O(\mathbf{y}_k \mid \mathbf{s}_k)$ is the distribution of state trajectories τ induced by a policy π_θ [162]. Maximizing this objective corresponds to removing the target object in the fewest number of actions.

8.2.4 Surrogate Reward

Because the reward defined in Section 8.2.2 is sparse and the transition function relies on complex inter-object and grasp contact dynamics, it is difficult to directly optimize for π_θ. Thus, we instead introduce a dense surrogate reward $\tilde{R}$ describing the reduction of the support of the target object's occupancy distribution:

$$\tilde{R}(\mathbf{y}_k, \mathbf{y}_{k+1}) = |\text{supp}(f_\rho(\mathbf{y}_k))| - |\text{supp}(f_\rho(\mathbf{y}_{k+1}))|,$$

where $f_\rho : \Omega \longrightarrow \mathcal{P}$ is a function that takes an observation $\mathbf{y}_k$ and produces the corresponding occupancy distribution ρ_k for a given bounding box and $\text{supp}(\rho) = \{(i,j) \in \{0,\ldots,h-1\} \times \{0,\ldots,w-1\} \mid \rho(i,j) \neq 0$ is the *support* of the occupancy distribution. Then, $|\text{supp}(\rho)|$ is the number of nonzero pixels in ρ. Section 8.3 discusses a data-driven approximation for the function f_ρ while Section 8.4 discusses a greedy policy using the learned f_ρ and $\tilde{R}$.

8.3 Learning Occupancy Distributions

We describe a method for estimating the function f_ρ via a deep neural network. Each pixel in the occupancy distribution $\rho \in [0,1]^{h \times w}$ has a value representing the likelihood of it containing part of the target object's amodal segmentation mask, or the likelihood that some part of the object, in some planar translation or rotation, would occupy that pixel without any occlusions from other objects. We train this pixelwise distribution network on a dataset of augmented depth images and ground-truth occupancy distributions.

Aspect	**Test**		**Lid**		**Domino**		**Flute**	
Ratio	Bal. Acc.	IoU	Bal. Acc.	IoU	Bal. Acc.	IoU	Bal. Acc.	IoU
1:1	98%	0.91	**93%**	**0.70**	92%	0.74	71%	0.30
2:1	97%	0.90	79%	0.44	**96%**	0.81	84%	0.44
5:1	97%	0.90	66%	0.23	96%	**0.83**	**86%**	**0.49**
10:1	97%	0.87	84%	0.49	82%	0.58	82%	0.41

Table 8.1: Balanced accuracy (Bal. Acc.) and Intersection over Union (IoU) metrics for networks trained on various aspect ratio target boxes. The first column is the respective set of 2,000 test images for the network's training dataset. The other columns show how the networks can generalize to unseen objects outside the training distribution. Each dataset contains 1,000 test images for the lid, domino, and flute objects, respectively. These objects are shown in Figure 8.4 and have approximate aspect ratios of 1:1, 2:1, and 5:1, respectively. Each network performs very well when estimating distributions for its training target object and makes reasonable predictions for target objects with similar bounding box aspect ratios, even for novel target objects at different scales and in the presence of new occluding objects. However, a network trained on a small aspect ratio does not generalize well to higher aspect ratio objects, as it tends to overestimate the occupancy distribution.

8.3.1 Dataset Generation

We generate a dataset of 10,000 synthetic augmented depth images labeled with target object occupancy distributions for a rectangular box target object. We choose 10 box targets of various dimensions ranging from $3cm \times 3cm \times 5mm$ to $9.5cm \times 0.95cm \times 5mm$ (aspect ratios varying from 1:1 to 10:1) with equal volume and generate a dataset for each, resulting in a total of 100,000 dataset images. We choose a relatively small thickness for the target so that it is more likely to be occluded in heaps of objects, as it tends to lie flat on the workspace. We sample a state $\mathbf{s}_0$ by uniformly sampling a set of N 3D CAD models as well as a heap center and 2D offsets for each object from a 2D truncated gaussian. First, $\mathcal{O}_t$ is dropped from a fixed height above the workspace, then the other N objects are dropped one by one from

a fixed height and dynamic simulation is run until all objects come to rest (all velocities are zero). Any objects that fall outside of the workspace are removed. N is drawn from a Poisson distribution ($\lambda = 12$) truncated such that $N \in [10, 15]$. The 3D CAD models are drawn from a dataset of 1296 models available on Thingiverse, including "packaged" models, where the original model has been augmented with a rectangular backing, as in [166]. The camera position is drawn from a uniform distribution over a viewsphere and camera intrinsics are sampled uniformly from a range around their nominal values. We use the Photoneo Phoxi S datasheet intrinsics and a camera pose where the camera points straight down at the heap at a height of $0.8m$ for the nominal values. An RGBD image is rendered and augmented depth images are created by concatenating a binary modal mask of the target object with the depth image. Note that if the target object is not visible, the image is equivalent to a two-channel depth image, as the first channel is all zeros. We find that training on these images, as opposed to training on RGBD images directly, allows for seamless transfer between simulated and real images.

To generate the ground-truth occupancy distribution, we find the set of translations and rotations in the image plane for the target object such that an image rendered from the same camera pose with all other objects in the scene in the same respective poses will yield the same target object modal segmentation mask. Thus, when the object is fully visible, the distribution's support collapses to the pixels of the target object modal segmentation mask. However, when the object is partially or fully occluded, then multiple target object translations or rotations may result in the same image and the distribution will spread to reflect where the target could hypothetically be hiding. In practice, we generate this distribution by discretizing the set of possible translations into a 64×48 grid (every 8 pixels in the image) and rotations into 16 bins, then shifting and rotating a target-only depth image to each point on the grid, offsetting by the depth of the bottom of the workspace at that point. By comparing the depths for the set of these shifted and rotated depth images to original depth image, we can determine the modal segmentation mask for the target object as if it were at each location. Any location for which there is intersection-over-union (IoU) greater than 0.9 (or, in cases where the target object has a blank modal mask due to full occlusion, any location for which the modal mask is also blank) is considered to result in the same image. Then, the amodal target object masks from all locations resulting in the same image are summed and the resulting normalized single-channel image is the ground truth occupancy distribution. A visualization of this process is shown in Figure 8.2. Dataset generation for 10,000 images took about 5 hours on an Ubuntu 16.04 machine with a 12-core 3.7 GHz i7-8700k processor.

8.3.2 Occupancy Distribution Model

We split each dataset of 10,000 images image-wise and object-wise into training and test sets (8,000 training images and 2,000 test images, where objects are also split such that training objects only appear in training images and test objects only appear in test images). We train a fully-convolutional network with a ResNet-50 backbone [155] using a pixelwise

mean-squared-error loss for 40 epochs with a learning rate of 10^{-5}, momentum of 0.99, and weight decay of 0.0005. The input images were preprocessed by subtracting the mean pixel values calculated over the dataset and transposing to BGR. Training took approximately 2.5 hours on an NVIDIA V100 GPU and a single forward pass took 6 ms on average as compared to 1.5 s for generating the ground-truth distribution.

8.3.3 Simulation Experiments for Occupancy Distributions

We benchmark the trained model on the full set of 2,000 test images as well as on 1,000 images with three other simulated target objects shown in Figure 8.4 - a lid, a domino, and a flute - to test generalization to object shapes, aspect ratios and scales not seen during training. We chose these target objects due to their diversity in scale and object aspect ratio (e.g., the flute is longer, thinner, and deeper, while the lid is nearly square and flat). We report two metrics: balanced accuracy, the mean of pixelwise accuracies on positive and negative pixel labels, and intersection-over-union, the sum of positive pixels in both the ground truth and predicted distribution divided by the sum of total positive pixels in either distribution. We consider true positives as the ground truth pixel having normalized value greater than 0.1 and the predicted value being within 0.2 of the ground truth value. Similarly, we consider true negatives as the ground truth pixel having normalized value less than 0.1 and the predicted value being within 0.2 of the ground truth value. Results are shown in Table 8.1.

Target Object Scale. For objects of different scale than the training target object, we scale the input image by a factor equal to the difference in scale between the box target object and the other target object, feed it through the network, and then rescale the output distribution. We find that this scaling dramatically improves performance with minimal preprocessing of the input image; for example, when testing on the lid object, which is about twice as large as the training box object, we increase balanced accuracy and IoU from 63.0% and 0.186 to 93.1% and 0.697, respectively.

Target Aspect Ratios. We found that, while our network performed well on objects with similar aspect ratios, longer and thinner objects with higher aspect ratios resulted in the model overestimating the support of the distribution. This effect can be seen in Figure 8.3, which shows ground truth occupancy distributions for target objects of different aspect ratios in the same heap image. Table 8.1 suggests that the trained networks can accurately predict occupancy distributions for target objects that have similar aspect ratios to the training boxes, but do not perform as well when tasked with predicting a distribution for objects with dramatically different aspect ratios. In particular, the network trained with a 1:1 box target object tends to overestimate the support for target objects with high aspect ratios, leading to a drop in metrics. This effect is especially visible along corners of occluding objects, where more rotations of a low aspect ratio object are possible, while only one or two rotations of a high aspect ratio object are possible.

Figure 8.4 shows occupancy distribution predictions with ground truth distributions for the three unseen objects using the network trained on the closest aspect ratio target object

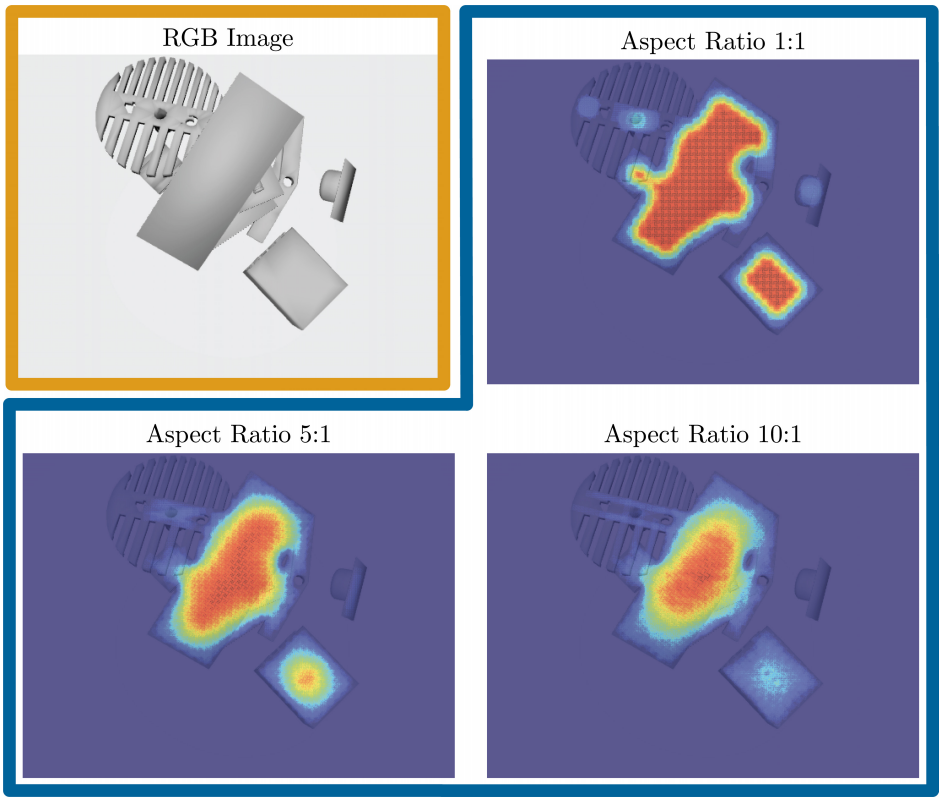


Figure 8.3: The ground truth occupancy distributions for a target object of various aspect ratios for the same heap image.

and scaled appropriately. Results suggest that the network is able to accurately predict diverse distributions when occluding objects not seen in training are present. Figure 8.4 suggests not only that the network can predict the correct distribution spanning multiple occluding objects in unimodal and multimodal cases when the target object is fully occluded, but also that it can correctly collapse the distribution to a small area around the visible part of the target object when it is only partially occluded.

8.4 X-Ray: Mechanical Search Policy

Using the learned occupancy distribution function f_ρ, we propose X-Ray, a mechanical search policy that optimizes for the objective and surrogate reward $\tilde{R}$ defined in Section 8.2. We create both simulated and physical object heaps and generate overhead camera images using an observation model based on the Photoneo PhoXi S depth camera. The heap RGBD image and target object are inputs to the perception system, which uses the network trained on the most similar bounding box to the target object to predict an occupancy distribution for the

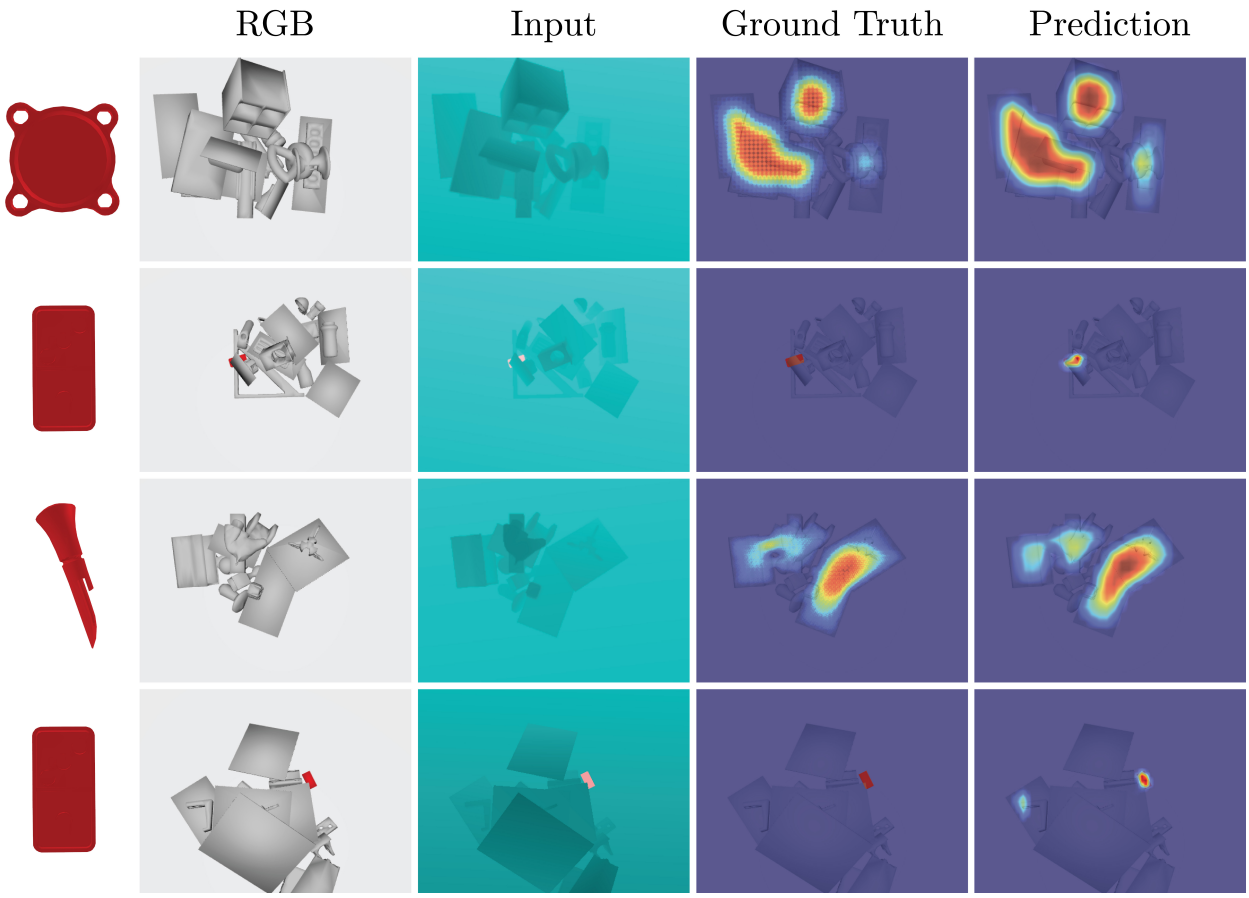


Figure 8.4: Example predicted target object occupancy distributions for three target objects, a lid, domino, and flute, unseen during training (far left). Warmer colors indicate a higher likelihood of that pixel containing part of the target object's amodal mask. The network is able to accurately predict a distribution across many objects, a collapsed distribution when the object is partially visible, and multimodal distributions when there are gaps between objects (top three rows). The final row shows a failure mode where the network spuriously predicts an extra mode for the distribution when the target object is partially occluded.

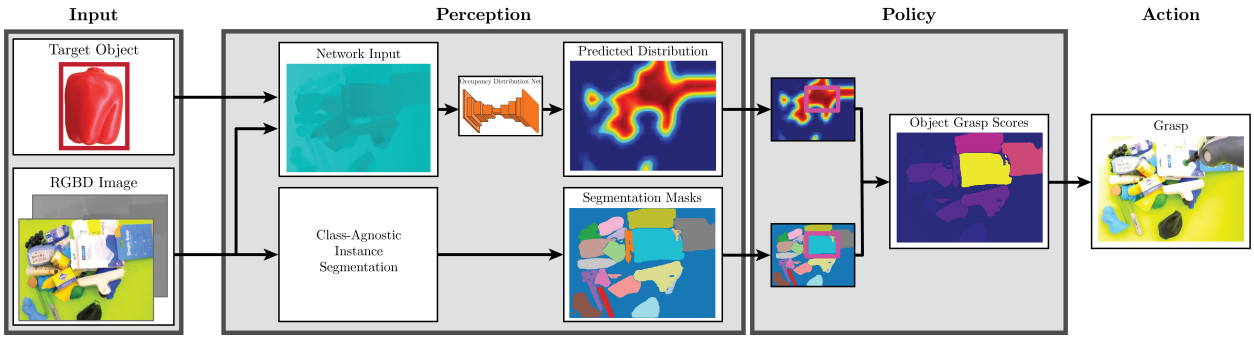


Figure 8.5: The perception stage takes as input an RGBD image of the scene and outputs an occupancy distribution prediction using a network based on the target object bounding box dimensions and the created augmented depth image. The perception stage also produces a set of segmentation masks. The X-Ray mechanical search policy then finds the mask that has the most overlap with the occupancy distribution (colored yellow in the grasp scores image) and plans a grasp on that mask.

target. The policy takes the predicted distribution and a set of modal segmentation masks for the scene and computes a grasping action that would maximally reduce the support of the subsequent distribution. Specifically, the policy takes an element-wise product of each segmentation mask with the predicted occupancy distribution and sums over all entries in the resulting image, leading to a score for each of the segmentation masks. The policy then plans a grasp on the object mask with the highest score and executes it, as shown in Figure 8.5.

8.4.1 Simulation Experiments with X-Ray

We first evaluate the mechanical search policy with simulated heaps of novel objects. To further test the ability of the learned network to generalize to unseen occluding objects, we use a set of objects unseen in training and validation: 46 YCB objects [28] and 13 "packaged" YCB objects (augmented in the same way as described in Section 8.3). Initial states were generated as explained in Section 8.3, first dropping the target object, followed by the other N objects. We use $N = 14$ so each heap initially contained 15 total objects, colorreda similar or larger size to previous bin-picking work [162, 179]. As the focus of this work was not instance segmentation or target detection, we use ground truth segmentation masks and target binary masks in simulation, although we note that any class-agnostic instance segmentation network [135, 51] or object detection network [286] can be substituted. For each grasp, either a parallel jaw or suction cup grasp, we use wrench space analysis to determine whether it would result in the object being lifted from the workspace under quasi-static conditions [203, 163, 164]. If the grasp is collision-free and the object can be lifted, the object

Policy	**Success Rate**	**Number of Actions Quartiles**		
Random	42%	4	7	9
Largest	67%	4	**5**	7
X-Ray	**82%**	**3**	**5**	**6**

Table 8.2: Evaluation metrics for each policy over 1,000 simulated rollouts. The lower quartiles, medians, and upper quartiles for number of actions are reported for successful rollouts. X-Ray extracts the target at a higher success rate with significantly fewer actions.

is lifted until the remaining objects come to rest using dynamic simulation implemented in PyBullet, resulting in the next state. Otherwise the state remains unchanged.

In addition to the policy proposed here, we evaluate two previously proposed baseline policies, **Random** and **Largest** [49]. The **Random** policy that first attempts to grasp the target object, and, if no grasps are available on the target object, grasps an object chosen uniformly at random from the bin. The **Largest** policy that first attempts to grasp the target object, and, if no grasps are available on the target object, iteratively attempts to grasp the objects in the bin according to the size of their modal segmentation mask.

Each policy was rolled out on 1,000 total heaps until either the target object was grasped (successful rollout) or the horizon $H = 10$ was reached (failed rollout). We benchmark each policy using two metrics: success rate of the policy and mean number of actions taken to extract the target object in successful rollouts. Table 8.2 and Figure 8.6 show these metrics and the distribution of successful rollouts over the number of actions taken to extract the target object, respectively.

While the Random and Largest policies occasionally are able to quickly extract the target object, X-Ray consistently extracts the target in fewer actions and succeeds in 15% more heaps than the best-performing baseline. Largest is a reasonable heuristic for these heaps, as shown in [49], as large objects typically have a greater chance of occluding the target, but X-Ray combines this intuition with superior performance when the object is partially occluded. X-Ray outperforms the Largest policy on heaps where the target object is partially occluded by a thin or small object (such as a fork or dice) at some point during the rollout. In these scenarios, a robust grasp is often not available on the target object, and while X-Ray can correctly identify that the occluding object should be removed, the Largest policy will often grasp a larger object further from the target object. In scenarios where there are many large objects, but some are lying to the side, X-Ray will typically grasp objects that are in the more cluttered area of the bin, since they are more likely to reveal the target object. This behavior is a function of weighting the object area by the predicted distribution, which encourages the policy to ignore solitary objects.

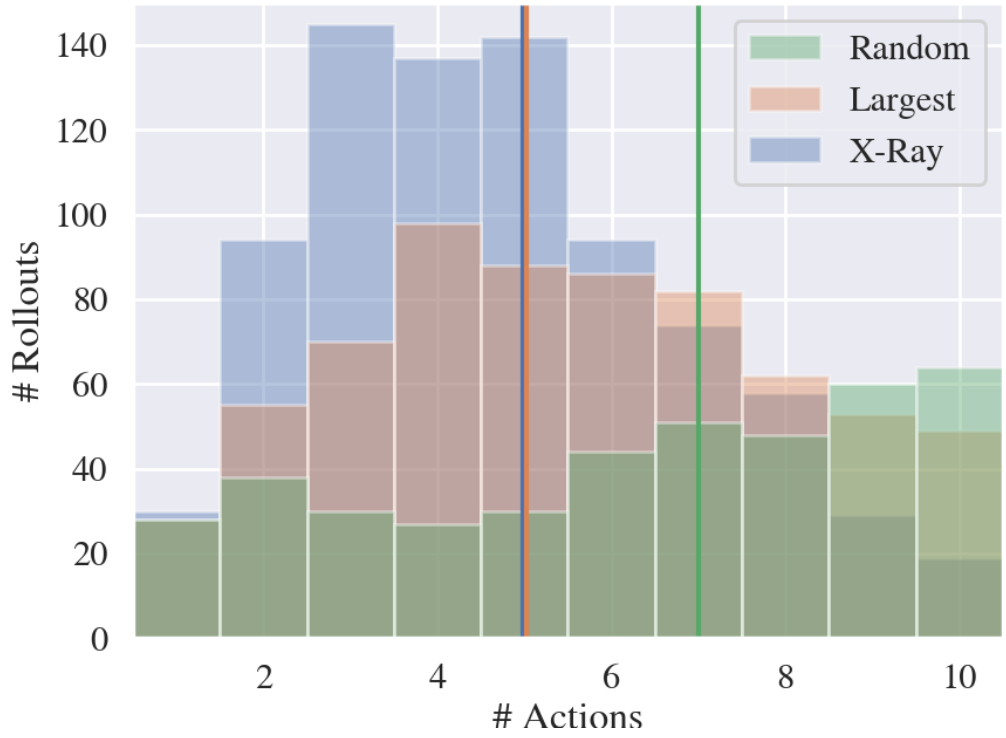


Figure 8.6: Histogram of the number of actions taken to extract the target object over the 1,000 simulated rollouts for the three policies tested. The median number of actions for each policy is shown by the corresponding vertical line.

8.4.2 Physical Experiments with X-Ray

We also evaluate X-Ray with heaps of novel household objects on a physical ABB YuMi robot with a suction cup and parallel jaw gripper, using two target objects. Some examples of the objects used can be seen in Figures 8.1 and 8.5. Initial states were generated by placing the target object on the workspace, filling a bin with the N other objects, and then dumping the bin on top of the target object. In these heaps, $N = 24$ was used so that each heap initially contained 25 total objects. We chose 25 total objects because it has been commonly used in cluttered bin-picking environments [166] and objects tend to disperse further on the physical setup. For segmentation masks, we used the class-agnostic instance segmentation network from [51], and for grasp quality analysis, we used FC-GQCNN [223]. To generate binary target masks, we use HSV color segmentation from OpenCV and use red target objects. While we make this assumption for simplicity, we note that we could substitute this process with a target object segmentation method that uses visual features, semantics and shape, such as the one described in [51].

We perform 20 rollouts for each of the three policies. Each policy was rolled out until either the target object was grasped (successful rollout) or the horizon $H = 10$ was reached

Policy	**Success Rate**	**Number of Actions Quartiles**		
Random	85%	**4**	6	7
Largest	85%	4	6	7
X-Ray	**100**%	**4**	**5**	**5.25**

Table 8.3: Evaluation metrics for each policy over 20 physical rollouts. The lower quartiles, medians, and upper quartiles for the number of actions are reported across successful rollouts. X-Ray extracts the target with significantly fewer actions, always extracting it within 10 actions.

(failed rollout). We report the same metrics as in the simulated experiments in Table 8.3.

We find that X-Ray outperforms both baselines, extracting the target object in a median 5 actions over the 20 rollouts as compared to 6 actions for the Largest and Random policies while succeeding in extracting the target object within 10 actions in each case. These results suggest that X-Ray not only can extract the target more efficiently than the baseline policies, but also has lower variance. The Largest policy performed comparatively worse with more objects in the heap than in simulation, as it relies heavily on accurate segmentation masks. However, when objects are densely clustered together, segmentation masks are often merged, leading to grasps on smaller objects that do not uncover the target. In this case or in the case of spurious segmentation masks that do not cover objects, X-Ray reduces this reliance on accurate segmentation masks, as the occupancy distribution and segmentation are combined to create a score for the mask. This property of X-Ray causes it to compare favorably to a policy that directly scores segmentation masks based on their relationship to the target object geometry. X-Ray also reduces reliance on the target object binary mask being accurate; if the detector cannot see enough of the target object to generate a detection even when it is partially visible, X-Ray will continue to try and uncover it according to the fully occluded occupancy distribution until more of the target is revealed.

8.5 Discussion and Future Work

In this chapter, we presented X-Ray, a mechanical search algorithm that minimizes support of a learned occupancy distribution. We showed that a model trained only on a synthetic dataset of augmented depth images labeled with ground truth distributions learns to accurately predict occupancy distributions for target objects unseen in training. We benchmark X-Ray in both simulated and physical experiments, showing that it can efficiently extract the target object from challenging heaps containing 15-25 objects that fully occlude the target object in 82% - 100% of heaps using a median of just 5 actions.

In future work, we will address some of the failure modes of the system, especially for objects that are significantly non-planar. Currently, the assumption that the object is flat

can result in incorrect occupancy distributions for taller objects. Additionally, we will look to add memory to the policy so that if objects shift into previously free space, the distribution will not cover that area, and explore reinforcement learning policies based on a reward of target object visibility.

Chapter 9

Mechanical Search on Shelves using a Novel "Bluction" Tool

While the majority of the dissertation has focused on applying manipulation primitives, perception primitives, and action selection policies to overhead-access mechanical search in bins or tabletop environments, this chapter extends these ideas to lateral-access environments such as shelves. In contrast with the overhead-access bin settings explored in prior work [49, 137, 276, 282], where objects can be heaped in arbitrary poses, objects on shelves consistently rest in stable poses. If a target object is occluded on a shelf, other objects must be carefully moved without toppling to reveal the target. Lateral-access environments may restrict available parallel jaw grasps, as there may not be sufficient space between objects to insert gripper jaws. Prior work on mechanical search in lateral-access environments has used pushing actions, executed with a wrist-mounted blade [99]. This chapter introduces a novel end-effector that combines a blade for pushing and a suction cup for grasping. The blade and suction cup are mounted on a thin shaft that can be rotated to maximize camera visibility. We call this end-effector a *bluction* tool (a pormanteau of "blade" and "suction").

The expanded action set afforded by the bluction tool is beneficial when objects are close together or close to the shelf wall. In prior work [99], these objects would likely be immovable as the robot would have no space to place its end-effector to reach and push them, and some pushes could result in irrecoverable failures. Thus, even moderately dense scenes could make target extraction impossible. Suction grasping can resolve these issues, but requires new planning algorithms that aim to balance pushing and suction grasping actions to efficiently reveal the target object.

This chapter introduces *Suction Lateral Access maXimal Reduction in support Area of occupancY distribution* (SLAX-RAY), which extends the lateral-access mechanical search policies introduced in LAX-RAY [99] to include both suction and pushing actions executed with the novel bluction tool.

This chapter makes four contributions:

1. The bluction tool: the design and evaluation of a novel robot end-effector.

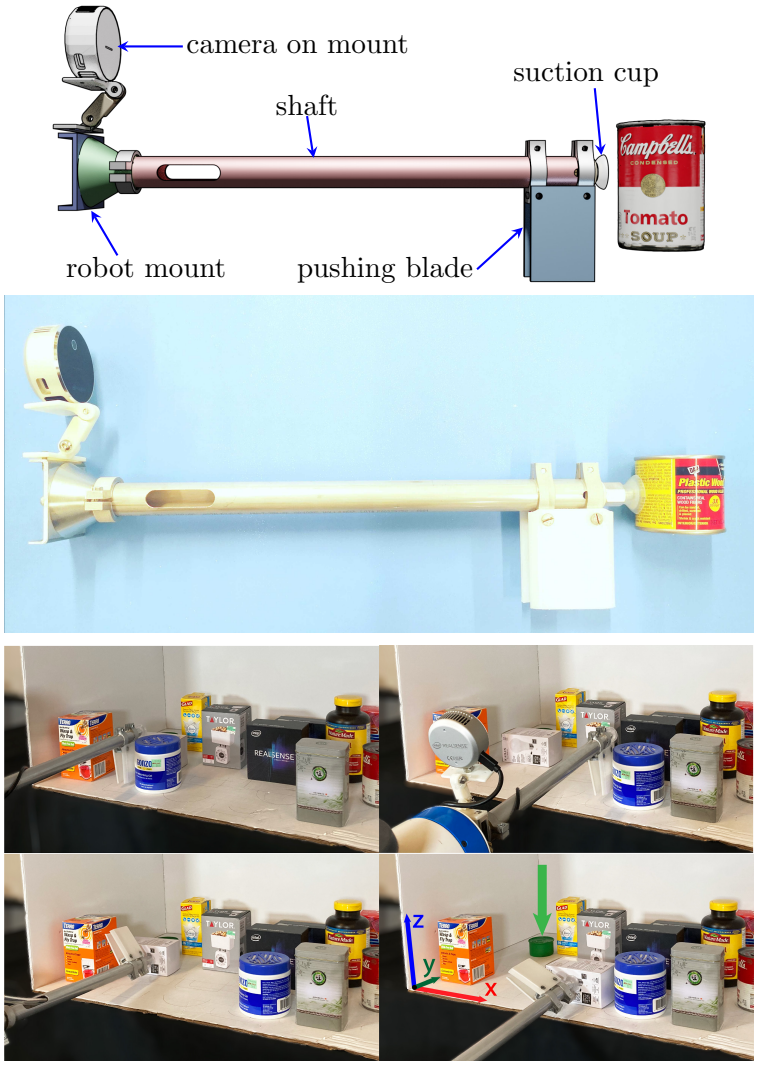


Figure 9.1: **Bluction Tool**: CAD (top) and physical implementation (middle). Using the bluction tool, the SLAX-RAY policy performs pushing and suction actions (bottom) to reveal the green cylinder target object among occluding objects within the shelf. Suction actions increase the robot search efficiency and can reveal the target object in scenes where no pushing actions are available. The shelf coordinate system is shown in the bottom right.

2. A simulation pipeline with 300 % speedup compared to LAX-RAY [99] that uses ray-casting and 2D Minkowski sums to guarantee a fully-occluded target across a dataset of 100,000 random shelf configurations.

3. Bluction-DAR: a lateral-access mechanical search policy using the bluction tool, which uses a combination of suction and pushing actions to recover the target object.

4. Experimental data from 2000 simulated trials with 4 – 10 occluding objects and 18 physical trials with a Fetch robot with 8 – 12 occluding objects, that suggest the SLAX-RAY search policy with a bluction tool can improve the average success rate in absolute values by 26 % in simulation and 67 % in physical compared to pushing-only baseline policies.

9.1 Related Work

9.1.1 Mechanical Search

For mechanical search for a specific target object in an overhead-access environment (e.g., a tabletop or bin), the robot uses an overhead camera to acquire color, depth, or RGBD images of the scene and performs top-down grasps then pushes to reveal and extract the target object. Danielczuk et al. [49] formulate the mechanical search problem and introduce a two-stage perception and search policy pipeline that uses heuristic policies to guide pushing and grasping within the bin. Kurenkov et al. [137] extend this work by introducing a learned, non-linear pushing action to uncover the target. Zeng et al. [282], Novkovic et al. [188], and Yang, Liang, and Choi [276] jointly learn coordinated pushing and grasping strategies. One approach for mechanical search is to statistically estimate target object locations. Xiao et al. [266] and Price, Jin, and Berenson [204] attempt to model the object locations using particle filters and shape completion. Danielczuk et al. [52] learn such a distribution from a synthetic dataset of depth images and target object occupancy distributions. The target occupancy distributions represent all locations of the target object that would result in the rendered depth image. They train a neural network to estimate this distribution for multiple target object aspect ratios and show it can transfer to physical scenes as part of an overhead-access mechanical search policy.

However, the lateral-access shelf environment introduces new challenges and constraints, particularly for motion planning. Zeng et al. [283] use pick-and-place actions in a shelf environment during the Amazon Picking Challenge, but do not rearrange objects *within* the shelf. Many prior works [182, 15] focus on scenarios where the target object is partially observable or the robot camera sweeps freely across the shelf opening, whereas we consider the case where the target object is initially occluded and the robot camera position is fixed. Lou, Yang, and Choi [157] estimate collisions between the robot and the environment with a learned Collision-Aware Reachability Predictor. Wang, Miao, and Bekris [256] propose a new solver for rearranging objects in shelves with a global planner. Li, Hsu, and Lee [145] use

a POMDP solver in sparse simulated shelf environments. Gupta et al. [81] introduce a multi-step object search algorithm that discretizes objects placements within shelf and searches for objects using pushing or pick-and-place actions; in contrast, we do not discretize the shelf for objects placements. Huang et al. [99] introduce LAX-RAY with two pushing policies for the shelf environment. The LAX-RAY policies leverage a history encoding of target occupancy distributions and multi-step lookahead to efficiently reveal a target object on a shelf with up to eight occluding objects. This chapter introduces the bluction tool for suction grasping actions in addition to pushing actions and a bluction-based policy that maximally reduces distribution area at each timestep.

9.1.2 Suction Grasping

Suction grasps have been heavily utilized in prior robotics works ranging from the Amazon Picking Challenge [283, 181] and grasping [165, 166, 249] to underwater manipulation [238] and wall climbing [8]. Suction-based end-effectors apply vacuum force on a single point of contact with the target object. Most works consider suction grasps in an overhead scenario, where the suction cup's vacuum force mainly resists the gravitational force along its grasp approach axes [166]. However, shear forces are rare. In this chapter, we instead focus on suction grasps with approach axes perpendicular to gravity, which results in higher shear forces that the suction cup must resist.

Suction grasps can be directly planned on a point cloud of the scene, using heuristic methods or by sampling a range of candidate grasps and ranking them using a quality metric. For the former, common approaches are grasping near estimated centroids of flat surfaces [278], grasping along inward surface normals towards an object centroid [93], or pushing objects from the top or side until a suction seal is formed [63].

Suction grasp planning can also leverage geometric models: Domae et al. [57] use a geometric model that assesses planarity by convolving a contact template with the image. Cartman [181], which won the 2017 Amazon Picking Challenge, ranked grasps according to their distance from object boundaries. Zeng et al. [283] and Cao et al. [29] use large, labeled datasets of real images to train a neural network that predicts grasp affordances directly from RGBD images or point clouds. Similarly, Mahler et al. [165] use a hybrid approach where suction grasps are modeled in wrench space and labeled on simulated depth images. A network trained on these simulated grasps is applied to real depth images.

9.2 Problem Statement

A set of non-stacked objects rest in stable poses within a shelf, with one object designated as the target to be revealed. Only the target is of known color and geometry. The robot views the shelf from the open side using an RGBD camera that is attached to its arm.

We additionally assume:

- The shelf and camera poses are static with respect to the robot.

- All objects are cylinders or rectangular prisms with a flat face resting on the shelf surface. All rectangular prisms are axis-aligned with the shelf.
- Actions will not topple objects or move multiple objects simultaneously.

At any time $t \in \{1, \ldots, H\}$, where H is a user-specified time limit, let $y_t \in \mathbb{R}^{w \times h \times 4}$ be the observation from an RGBD camera of the arrangement of objects on the shelf. At time t, the robot performs an action $a_t \in \mathcal{A}$, where $\mathcal{A} = \mathcal{A}_p \cup \mathcal{A}_s$, the union of pushing and suction actions. The search terminates in *failure* after H steps, and in *success* when the robot observes a threshold visibility $v\%$ of the target object.

Pushing actions in $\mathcal{A}_p$, parametrized as $\mathbf{a}_t = (\mathbf{O}, d_x)$, start with the blade at left or right edge of object $\mathbf{O}$ and push a signed distance $d_x \in \mathbb{R}$ along the x-axis of the shelf frame, which is shown in Fig. 9.1.

Suction actions in $\mathcal{A}_s$, parametrized as $\mathbf{a}_t = (\mathbf{O}, d_x, d_y)$, start with the robot forming a seal between the suction cup and object $\mathbf{O}$, followed by four linear motions: (1) lifting the object along the shelf z-axis, (2) pulling the object towards the camera along the shelf $-y$-axis, (3) translating the object along the shelf x-axis by d_x, and (4) pushing the object along the shelf y-axis into its final placement pose at a distance $d_y \in \mathbb{R}$ into the shelf.

The goal is to terminate the search in success by revealing $v\%$ of the target, while minimizing the total action cost $n_p + \psi n_s$, where n_p and n_s are the number of pushing and suction actions respectively. $\psi > 1$ is an action cost ratio that accounts for the increased motion planning and execution time for suction actions as compared to pushing actions. By measuring action planning and execution times during physical experiments, we empirically estimate $\psi = 1.3$.

9.3 Bluction Tool

The bluction tool is a combination of a blade and a suction cup. It attaches to the robot wrist, enabling pushing actions with the blade, and suction actions with the suction cup, as shown in Fig. 9.1. Our specific design for the Fetch robot includes a 0.40 m aluminum vacuum tube with a diameter of 0.03 m that allows the robot to reach deep into a shelf without losing pressure while avoiding undesirable collisions between the robot wrist and the environment. The blade has a width of 0.065 m and a height of 0.075 m. The suction cup at the end has a diameter of 0.03 m. The bluction tool includes a manually-adjustable camera mount, and allows for autonomous rotation to facilitate camera visibility. Attached to the camera mount is a RealSense L515 LiDAR Camera.

The bluction tool enables a wider set of actions than considered by suction-only [49] and push-only [99, 137] mechanical search policies. To push an object with the blade, the robot inserts the flat face of the blade next to the object and pushes along the x-axis of shelf. In contrast to pushing using the shaft, the blade flat face increases contact area, thus reducing object rotation during pushing. To perform a suction action, the robot rotates the pushing blade upwards to avoid collision with the shelf, and contacts object with the suction cup.

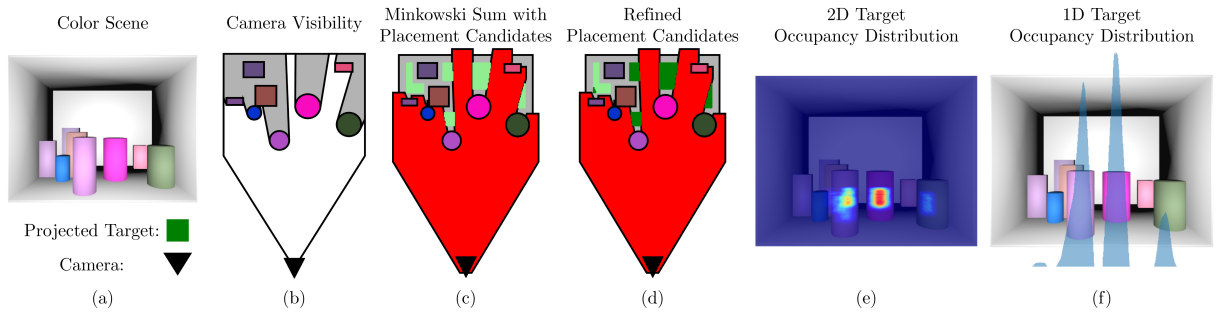


Figure 9.2: SLAX-RAY Simulation Pipeline: (a) A random set of occluding objects are placed in stable poses such that they do not collide with each other or the shelf. (b) We find the visibility polygon given a projected scene from the camera perspective. (c) We calculate the Minkowski sum of the visibility polygon and the projected target object, which yields a set of candidate target object placements (light green). (d) By casting 3D rays from the camera to all non-colliding candidate target placements, we find the set of all fully hidden target object placements (dark green). (e-f) We project the target object points from each hidden placement into the camera to get a 2D distribution over target locations in image space, then project this 2D distribution to a 1D distribution along the image x-axis.

To take an image of the shelf, the robot rotates the shaft to avoid occlusion of the view by the blade.

9.4 Methods

X-RAY (maXimize Reduction in support Area of occupancY distribution) [52] and LAX-RAY (Lateral Access X-RAY) [99] introduce mechanical search policies that attempt to maximally reduce either support area or entropy of an estimated target occupancy distribution. The target occupancy distribution encodes target object likelihoods within an RGBD image observation, and is estimated using a deep neural network trained on a dataset of simulated depth images and target object segmentation masks with corresponding ground-truth occupancy distributions. To perform lateral-access mechanical search with the bluction tool, we propose Suction LAX-RAY (SLAX-RAY). SLAX-RAY improves the full LAX-RAY pipeline, including scene generation, perception system, and search policies. First, SLAX-RAY simulates scenes that resemble cluttered shelf scenarios it might encounter. SLAX-RAY then uses these scenes to train a perception system that predicts the occupancy distribution for a target object. The search policies use this occupancy distribution to determine which action to execute.

To assess the effects of state uncertainty, object arrangement, and available actions on the complexity of the lateral-access mechanical search problem, we also propose an oracle

policy that uses full state knowledge to upper-bound policy performance given a scene and available action set.

9.4.1 Lateral-Access Simulation

The SLAX-RAY simulator uses scene generation both to train the perception system and to evaluate policies in simulation experiments. The intent is to generate a set of scenes with random clutter on a shelf, and to render depth images and their corresponding target object occupancy distributions, as in prior work [99]. SLAX-RAY improves on prior work by increasing the computational efficiency of the scene generation process and guaranteeing that target objects are completely occluded in all generated scenes. This improvement in scene generation results both in a more accurate occupancy distribution model and a larger simulated evaluation dataset. This new method uses a hybrid approach that relies on 2D polygon visibility calculations, Minkowski sums, and 3D ray-casting.

To generate a dataset, the scene generator first samples $N \in [2 \,..\, 12]$ 3D cuboids or cylinders from 6 cm to 20 cm in width, depth, and height. It places these objects in one of their stable poses on the shelf iteratively, drawing their 2D positions uniformly at random and rejecting positions that are in collision with the shelf or previously placed objects. Next, it projects the objects to the support surface of the shelf and calculates the 2D visibility of the scene from the perspective of the camera using the CGAL [205] implementation of Bungiu et al. [27] (Fig. 9.2(b)). Given that it knows the projected target object, it calculates a Minkowski sum of the visibility polygon and the convex hull of the target's projected vertices (the red shaded area in Fig. 9.2(c)). By sampling in the remaining 2D positions of the shelf, it generates a set of candidate target placements (light green area in Fig. 9.2(c)). To take the target object height into account, candidate placements are refined by casting rays from the camera to a set of 3D points sampled from the target mesh transformed to each location; if any ray from the camera to the transformed point does not first intersect with the scene, the target object would be visible at that location. The non-intersecting locations are removed resulting in the final set of target locations in dark green shown in Fig. 9.2(d). Given this set of 2D target locations and a set of sampled mesh points that are transformed to each possible target location, we can directly generate a 2D distribution over possible target object locations in camera frame by projecting the set of target points into the camera (Fig. 9.2(e)). Then, by further projecting this 2D distribution to the image x-axis, we can reduce it to a 1D distribution (Fig. 9.2(f)). In comparison to prior work [99], which exhaustively transformed the object across a grid of 3D locations in the shelf, this method is over 300 % faster, with generation of a SLAX-RAY training dataset of 100,000 scenes taking only 4 hours on an Ubuntu 20.04 machine with an Intel i7 12-core processor and NVIDIA Titan X GPU. The training dataset is generated with random camera positions sampled from a sphere centered around the shelf center.

Figure 9.3: **Bluction tool camera view with overlaid occupancy distribution and actions.** At each timestep, Bluction-DAR computes the minimum of the predicted target occupancy distribution at the current timestep and that from the previous timestep (left, light blue overlay). It executes the pushing action (top, green) or suction action (bottom, blue) that maximally reduces the support area of the distribution. The dotted arrows represent the motion of the end-effector from the lighter colored box to the darker color. Suction actions are advantageous when moving objects that are very close. They follow the piecewise linear trajectory described in Sec. 9.2, as in the bottom row.

9.4.2 SLAX-RAY Perception System

Based on the neural network architecture and training pipeline in Danielczuk et al. [52], the SLAX-RAY perception system provides the search policy with information on where the target object may be hidden by computing a target occupancy distribution from a depth image. It also segments the scene into object instances using the input depth image. We assume that target objects can have three different aspect ratios (target objects of size $0.06\times0.06\times0.03\,\mathrm{m}^3$, $0.06\times0.06\times0.06\,\mathrm{m}^3$, and $0.06\times0.06\times0.12\,\mathrm{m}^3$). The neural network is trained on a dataset of 100,000 depth image and occupancy distribution pairs for those three target aspect ratios. Each training example contains a single target object and loss for each predicted distribution is only enforced on the network head that outputs the prediction of the aspect ratio corresponding to the target. By training in this way, the network learns to predict a target occupancy distribution for a set of target aspect ratios for each input image. Training takes approximately 18 hours on a Titan X GPU. At run time, the network takes in a depth image from the physical system and we take the output that corresponds to the aspect ratio of the known target object. SLAX-RAY projects the 2D image-space target occupancy distribution into a 1D distribution along the image x-axis, using the assumption that objects rest on the shelf (Sec. 9.2). Specifically, the 1D distribution is $P_t(x) = \sum_{y=0}^{h-1} p_t(x, y)$, where $p_t(x, y)$ is the 2D occupancy distribution at pixel location (x, y).

9.4.3 SLAX-RAY Mechanical Search Policy

We propose a novel mechanical search policy: *Bluction-DAR* (Distribution Area Reduction). Bluction-DAR takes actions at each timestep that attempt to maximally reduce the target

occupancy distribution until $v\,\%$ of the target is revealed. The policy operates on the full action space of pushing $\mathcal{A}_p$ and suction $\mathcal{A}_s$ actions using the bluction tool. The policy considers the suction action space $\mathcal{A}_s$ to be all combinations of a collision-free grasp, extraction and placement, which expands the total action space of the problem beyond the pushing actions previously considered and allows for revealing the target object in scenarios where the blade cannot be inserted between any pair of objects. The policy rejects actions that would cause collisions among the gripper, objects and the shelf. For pushing actions, an object can only be pushed next to the closest object on its left or right such that it does not collide with other objects in the shelf or result in simultaneous pushing of multiple objects, as stated in Sec. 9.2. For suction actions, we generate the d_x, d_y candidates by discretizing the x and y axis of the shelf uniformly into 50 and 8 segments, respectively. We restrict suction actions to the piecewise-linear action described in Sec. 9.2 to prevent collisions between objects in the shelf. After all actions are generated, we check for collisions between the shelf environment (pointcloud from depth image) and the bluction tool trajectory.

To track search progress and help inform subsequent actions, as in Huang et al. [99], we encode the history of previous observations via the minimum of the current 1D predicted distribution and the previous encoding: $P'_t(x) = \min\left\{P_t(x), P'_{t-1}(x)\right\}$. For the first timestep, we set $P'_0(x) = P_0(x)$. We define the support of the occupancy distribution of each object $\mathbf{O}_i$, $r_t(\mathbf{O}_i) = \sum_{x=l_i}^{r_i} P'_t(x)$, where l_i, r_i is the x coordinate of left and right edge of the object segmentation mask. We further define the reduction of support as $\Delta r_t(\mathbf{O}_i) = r_t(\mathbf{O}_i) - r_{t+1}(\mathbf{O}_i)$, where $r_{t+1}(\mathbf{O}_i)$ is calculated with the segmentation mask of object $\mathbf{O}_i$ after applying the action. The updated segmentation mask is obtained by deprojecting into the camera frame, applying the action and projecting back to the image frame. Note that if an action increases the support of the distribution, Δr_t may be negative. We measure Δr_t for each action candidate, weighting suction actions by $1/\psi$ to account for the action cost ratio defined in Sec. 9.2. Thus, for each suction action, $\Delta r_t^s = \Delta r_t/\psi$, while the reduction of support for the pushing action is kept the same: $\Delta r_t^p = \Delta r_t$. Bluction-DAR returns the list of actions sorted by decreasing $\Delta r_t^{(\cdot)}$, and executes the first kinematically feasible action from the list.

9.4.4 Oracle Policies

We quantify the difficulty of each generated shelf environment and upper-bound policy performance for a given action set by measuring the number of actions taken by oracle policies that use full-state information to reveal the target object. Although policy success is defined in Sec. 9.2 as target object visibility above v, the oracle policies search for a series of actions that fully reveals the target. Since all objects are assumed to be cylinders or cuboids, clearing a visibility triangle (from the camera to the front face of the target object in a projected overhead view) is equivalent to solving the 3D visibility problem. Therefore, for computational efficiency, the oracle policies project the 3D shelf scene into a 2D overhead representation and evaluate actions using the projected representation. While oracle policies have full-state knowledge, their action sets are equivalent to the non-oracle policies (e.g., they will not push unreachable objects).

		Oracle		Huang et al. [99]		Bluction
No.	Metric	Oracle-P	Oracle-P+S	DAR	DER-3	Bluction-DAR
4	Success Rate	100 %	100 %	77 %	83 %	**96 %**
	Median Steps (IQR)	1.5 (1.0 – 2.0)	1.5 (1.0 – 2.0)	2.0 (1.0 – 3.0)	2.0 (1.0 – 3.0)	2.0 (1.0 – 2.5)
6	Success Rate	100 %	100 %	68 %	72 %	**92 %**
	Median Steps (IQR)	3.0 (2.0 – 3.0)	2.0 (2.0 – 3.0)	3.0 (1.8 – 5.0)	3.0 (1.0 – 4.3)	2.0 (2.0 – 4.0)
8	Success Rate	99 %	100 %	45 %	50 %	**88 %**
	Median Steps (IQR)	3.0 (2.0 – 4.0)	2.5 (2.0 – 3.0)	3.0 (1.0 – 4.0)	3.0 (2.0 – 4.0)	3.0 (2.0 – 5.0)
10	Success Rate	94 %	100 %	24 %	34 %	**67 %**
	Median Steps (IQR)	4.0 (3.0 – 6.0)	3.0 (2.0 – 4.0)	4.5 (2.0 – 6.0)	3.0 (2.0 – 5.0)	6.0 (3.0 – 8.0)
All	Success Rate	98 %	100 %	54 %	60 %	**86 %**
	Median Steps (IQR)	3.0 (2.0 – 4.0)	2.0 (1.0 – 3.0)	2.0 (1.0 – 4.0)	3.0 (1.0 – 4.0)	3.0 (2.0 – 4.0)

Table 9.1: Simulated policy rollouts: 100 trials were executed for 4, 6, 8, and 10 occluding objects; in total, 2000 trials. For each, we show the **Success Rate** followed by the **Median** and **Interquartile Range (IQR)** for the number of steps taken to reveal the target. Results suggest that suction actions allow the Bluction-DAR policy to succeed between 13% and 38% more often in densely cluttered environments than the best-performing pushing baselines.

We define two versions of the oracle policy: **Oracle-P** (pushing actions only), and **Oracle-P+S** (pushing and suction actions). Given the visibility triangle to be cleared, the policy exhaustively generates all valid action trees (capped at a maximum of H actions) that would result in this area being free of objects. The action trees are sorted by the number of distinct actions and the policy executes the action tree with the fewest actions. Suction actions are weighted by ψ.

9.5 Experiments

We evaluate SLAX-RAY in both simulated and physical shelf environments. In simulation, the shelf is 0.60 m wide by 0.60 m deep by 0.60 m high. In physical environments, the shelf is 0.80 m wide by 0.50 m deep by 0.50 m high. In both experiments, we use the perception pipeline in Sec. 9.4.2 to estimate the target occupancy distribution.

9.5.1 Simulation Experiments

The simulated experiments use the First Order Shelf Simulator (FOSS) from Huang et al. [99] where the actions are deterministic. We generate 400 total environments with 4, 6, 8, and 10 occluding objects. For these experiments, we set the maximum number of steps to $H = 2n$, where n is the number of objects.

We evaluate Bluction-DAR on the set of 400 shelves along with two baselines from [99]: **DAR** (Distribution Area Reduction) and **DER-3** (Distribution Entropy Reduction over 3

steps) are lateral-access mechanical search policies based on target occupancy distribution predictions. DAR is an ablation of Bluction-DAR that greedily reduces the target object distribution support area and DER-3 is a policy with 3-step lookahead that takes the action with least predicted entropy after the 3 steps. Both DAR and DER-3 only use pushing actions. For these experiments, we use the target object of size $0.06 \times 0.06 \times 0.06\,\mathrm{m}^3$ and set the target visibility threshold to $v = 80\,\%$.

Table 9.1 shows the results. Bluction-DAR succeeds in revealing the target more than baselines across all scenes, with a 13 % and 33 % improvement over the best-performing pushing-only baseline for 4 object scenes and 10 object scenes, respectively. Bluction-DAR also maintains success rates of at least 67 % even in 10 object scenes, suggesting it scales better to more dense arrangements than baselines. Bluction-DAR gains a considerable advantage in denser scenes because of the additional object mobility. Since objects may not be pushed behind other objects (as to not risk collisions with obscured objects), pushing policies are extremely limited in scenes with many objects. Neighboring objects can block large parts of the scene and it may be difficult or impossible to align the small window of visibility with the target object. Suction actions enable the movement of occluders without having to clear a substantial pushing path. As compared to previous work [99], DAR and DER-3 success rates are lower due to the relaxation of the assumption that objects must be separated from each other and the shelf walls by at least the blade width.

The performance of the oracle policies in Table 9.1 suggests that while pushing actions can reveal the target object reliably in less cluttered shelves of 4, 6, or 8 occluding objects, as shelves become more densely packed, the upper bound of pushing policies begins to drop. However, when using both pushing and suction actions, it remains possible to succeed in 100 % of environments. These experiments further explain the performance gain of Bluction-DAR over pushing-only baselines and suggest that the bluction tool and associated new actions may be necessary to reveal target objects in increasingly dense scenes. The Oracle-P+S policy outperforms Bluction-DAR by 12 % and 33 % in scenes with 8 and 10 objects, respectively, which suggests that the Bluction-DAR policy can be further improved to approach success rates over 90 % even in denser scenes.

9.5.2 Physical Experiments

We evaluate SLAX-RAY and the DER-3 policy in 9 physical shelf environments with 8, 10, and 12 objects like the one shown in Fig. 9.1. We randomly place the household kitchen and bathroom objects with an occluded target of size $0.06 \times 0.06 \times 0.03\,\mathrm{m}^3$ on a shelf and execute the pushing and suction actions using a Fetch robot with the bluction tool. We use the Intel RealSense LiDAR Camera L515 for the RGBD observations. The results in Table 9.2 suggest that Bluction-DAR performs consistently better than DER-3 across these scenes. A benefit of the suction action is that Bluction-DAR is more robust to the perception noise. For 3 out of 9 trials, DER-3 fails to find the available actions when the segmentation mask is narrower than the actual object since it would produce a false collision detection. Bluction-DAR still finds available actions via suction actions. Bluction-DAR is also more robust to manipulation

No.	Metric	DER-3	Bluction-DAR
8	Successes	0/3	**3/3**
	No. Steps	(-)	3, 4, 4
10	Successes	2/3	**3/3**
	No. Steps	7, 4	4, 5, 4
12	Successes	1/3	**3/3**
	No. Steps	4	6, 7, 7
All	Success Rate	33 %	**100 %**
	Median Steps (IQR)	4.0 (4.0 – 5.5)	4.0 (4.0 – 6.0)

Table 9.2: Physical experiment results. We evaluate DER-3 and Bluction-DAR across 3 scenes with 8, 10, and 12 objects each. Bluction-DAR outperforms DER-3 in physical scenes consistently.

or trajectory planning deviations. When the robot accidentally pushes objects against other objects or to the wall, Bluction-DAR searches for viable suction actions. Bluction-DAR can find the target in scenes which are unsolvable for pushing-only policies, such as scenes where the target is hidden behind two objects with no large gap in between for blade insertion.

9.6 Conclusion and Future Work

This chapter presented SLAX-RAY, a lateral-access mechanical search pipeline. SLAX-RAY uses the bluction tool with a new search policy Bluction-DAR, which uses both pushing and suction actions to reveal a target object in a shelf environment. Simulation and physical experiments demonstrate that the increased action set improves the average success rate by 26 % and 67 %, respectively, over pushing-only baselines. In future work, we will explore actions that move multiple objects simultaneously, plan collisions between objects to "nudge" occluders out of the way, extend to objects beyond cylinders or rectangular prisms and explore scenes with stacked objects. We will also improve SLAX-RAY searching policies by incorporating future predictions and exploring different action selection policies such as information gain instead of support reduction.

Part IV

Conclusion and Future Work

Chapter 10

Discussion

10.1 Overview

This dissertation formalizes the problem of robot mechanical search and contributes to three areas of a mechanical search pipeline: manipulation primitives, perception primitives, and action selection policies. We show that while impressive progress has been made towards grasping singulated objects or objects in clutter as well as in object detection and segmentation, mechanical search policies can benefit from taking a modular approach and improving along each of these vectors.

In Chapters 2 and 3, we presented REACH and 6DFC, two area contact models that can efficiently provide contact wrench constraints for the 6D wrench applied by a compliant gripper jaw on the surface of a rigid object. We showed that explicitly modeling these soft area contacts results in higher recall for grasp quality predictions than baselines with a moderate increase in computation cost. As these contact models have been successfully incorporated into training pipelines for hybrid grasping models, their improvement could lead to more precise grasping policies in the mechanical search context, especially in cases where few grasps are available. In Chapter 4, we showed that coupling pushing policies with predicted grasp qualities can lead to successful pushes even in cases where objects may not be fully singulated. We also explored trading off pushing actions with grasping actions based on a success threshold for grasping.

In Chapter 5, we delved into category-agnostic instance segmentation and trained SD Mask R-CNN using a large dataset of simulated depth images. In comparison to baseline methods, our method shows impressive generalization both across the sim-to-real gap and to real objects unseen in training, with some capability to generalize to novel cameras as well. Chapter 6 again showed that leveraging ideas from computer vision on learning implicit functions, combined with training on simulated data, can result in fast, collision-free motion plans even with unknown objects in hand.

Finally, in Chapters 7, 8, and 9, we showed how the modules introduced in the earlier parts of the dissertation could be integrated together as part of a real-world mechanical

search pipeline in tabletop, bin, and shelf environments. We showed that shifting policy burden onto perception by predicting an occupancy distribution for the target object can increase success rate while decreasing the number of actions taken. When considering shelf environments, we explored new challenges and constraints afforded and introduced history encodings based on a 1D occupancy distribution prediction to account for object movement within the shelf.

10.2 Takeaways

A few common threads run through the mechanical search policies for bins, shelves, and tabletops presented in this dissertation. Here, we revisit some of the themes of this dissertation that can be applied outside of the mechanical search task.

10.2.1 Learning from Simulated Depth Data

In chapters 5, 6, and 8, we leverage large simulated datasets of labeled depth images or point clouds to train deep neural networks. Although we only extend these hypotheses as far as the robots, cameras, and objects that we trained and evaluated on, the success of each of these methods suggests that training with simulated depth image or point cloud data: (1) can ease transfer from simulated data to real-world sensor data, (2) can readily provide generalization to unseen objects of diverse colors and textures, and (3) can significantly decrease human costs associated with labeling datasets while allowing for customization to different tasks and settings. These three positives make training using simulated data attractive for many vision-based tasks. However, some tasks may still benefit from being trained solely on real-world data or a mixture of simulated and real-world data, especially in cases where ground-truth labels are difficult to specify in simulation or simulation capabilities diverge too far from real-world behavior.

10.2.2 Intermediate Representations and Manipulation Primitives

Chapter 8 shows the power of an effective intermediate representation for mechanical search, as the concept of a target occupancy distribution captures much of the intuition behind searching for a hidden object from a geometric perspective. The object instance segmentation network discussed in Chapter 5 also provides a useful abstraction and a higher-level understanding of objects in the scene than raw pixel observations. In this dissertation, we argue that providing structure for an action selection policy may be beneficial due to the long horizon and sparse reward in mechanical search. By learning intermediate representations and manipulation primitives, we can leverage the progress in multiple robotics subfields and more efficiently reuse data from other tasks for a high-level mechanical search policy.

10.3 Opportunities for Future Work

There are a number of exciting areas to explore in future work related to each of the chapters of this dissertation. We discuss future work related to efficient compliant contact models in Section 10.3.1, grasping in Section 10.3.2, pushing policies for mechanical search in Section 10.3.3, perception and manipulation primitives in Section 10.3.4, and mechanical search policies and environments in Section 10.3.5.

10.3.1 Contact Modeling

Although the area contact models presented in this dissertation are some of the first to model interactions between compliant gripper jaws and rigid objects in computationally-efficient manner, they remain limited to a quasistatic analysis and do not consider contacts between deformable objects and compliant grippers. Some work has already begun on incorporating dynamics [123] and grasping deformable objects [272, 100], but can these models be made more efficient? What role can improved simulation have in grasp quality prediction using hybrid training methods? Furthermore, we observed in experiments that the plastic support behind each gripper jaw slightly cantilevers outward when grasping, which may be modeled with Bernoulli beam theory.

10.3.2 Dex-Net 5.0

A large part of my graduate study has been devoted to robot grasping, especially viewed through the lens of contact modeling. Recently, I have led an effort to update, build up, and maintain the code infrastructure for the publicly-available GQ-CNN, Dex-Net, and contact modeling repositories. Our goal is to allow seamless generation of grasp datasets with analytic quality labels, generation of labeled grasp images, training of GQ-CNN models, and execution of Dex-Net grasp planning policies across a variety of objects, contact models, sensors, and robots. I hope that the code and documentation for this project will be available soon so that researchers across the globe can benefit from and contribute to the next generation of the Dex-Net project.

10.3.3 Pushing and Other Action Types

Pushing policies can be difficult to analyze, especially in bins and on tabletops, where pushing affects multiple objects. However, Chapter 7 suggests that it can be a valuable primitive in mechanical search, increasing success rates by creating access to more grasps as well as increasing efficiency in cases where a push can dislodge multiple occluding objects. Continuous rummaging or sweeping actions that more closely mirror those of humans may also benefit mechanical search policies. Can the complexity of the analysis around these actions be reduced or simplified so that they can be incorporated into future mechanical search policies? Are there other actions (perhaps with different grippers or robot morphologies) that

would further increase efficiency for this task? Can tactile data be effectively incorporated into these actions to increase control and robot's abilities to react to changing contact forces?

10.3.4 Perception for Mechanical Search

Although this dissertation introduces several intermediate representations for mechanical search, there is certainly still room for growth in this area. Is an occupancy distribution the most efficient way to represent likely locations for the target object? Can we extend the representations introduced in this dissertation to the three-dimensional space? Is it in fact best to allow an action selection policy to learn its own representations from raw observations?

10.3.5 Mechanical Search Environments

While this dissertation has explored mechanical search on tabletops and on bins and shelves, the task of mechanical search can encompass a much wider array of environments. Searching rooms, entire homes or office spaces, or environments with dynamics might require a hierarchy of policies [136] as well as entirely new abstractions and policies. Semantics may also play a role in these policies as they scale to larger and larger environments; humans often structure their homes and warehouses in semantically meaningful ways. How can semantic reasoning be incorporated into mechanical search policies? Some progress has already been made on this front [136], but robots can still gain a greater understanding of semantic object relationships to inform search.

10.4 A Broader View of Mechanical Search and Robot Manipulation

While significant progress has been made over the last few years towards robot manipulation of unknown and challenging objects, open problems still remain. Unlike in the fields of computer vision or natural language processing, where large datasets of real world data are common and systems can often be learned from data, real world labeled data is often expensive or impossible to collect for many robotics tasks. While I have hope that deep learning can continue to improve robotic manipulation policies, I think that this dissertation suggests that integrating deep learning at crucial stages in a pipeline might be an efficient way of applying it to robotics. Additionally, scalable data collection for these tasks might be possible through simulation, and continual improvement and investment into building realistic simulators augmented with automated data generation pipelines may combat the relative lack of real-world data available. I hope that this dissertation provides both a base for future research on the mechanical search problem, which uniquely integrates robotic perception and manipulation, as well as inspiring future work on scalable data collection and coupling machine learning with analytic methods.

Part V

Appendices

Appendix A

REACH: A Robust Efficient Area Contact Model

This appendix provides full derivations and full results for each test object for the material in Chapter 2.

A.1 Derivations

For the i-th triangle, we show full derivations for the maximal normal force $f_{z_{i,max}}$, the maximal tangential friction force $f_{t_{i,max}}$, and the maximal torsional moment $\tau_{z_{i,max}}$ and its lower bound in the following sections.

A.1.1 $f_{z_{i,max}}$ Derivation

This derivation gives an example of the process used to compute $f_{z_{i,max}}$, $f_{x_{i,max}}$, $f_{y_{i,max}}$, and $\tau_{z_{i,max}}$ for each triangle. First, the integral is converted to barycentric coordinates, then evaluated. The final result is a symbolic representation that can be evaluated with the triangle vertices and transform at run time.

$$
\begin{aligned}
f_{z_{i,max}} &= \int_{A_i} k(a_i x_i + b_i y_i + d_i)\ dA_i \\
&= 2A_i k \int_0^1 \int_0^{1-\lambda_2} \lambda_1(a_i x_{i,1} + b_i y_{i,1}) + \lambda_2(a_i x_{i,2} + b_i y_{i,2}) \\
&\qquad\qquad + (1-\lambda_1-\lambda_2)(a_i x_{i,3} + b_i y_{i,3}) + d_i\ d\lambda_1 d\lambda_2 \\
&= 2A_i k \int_0^1 \frac{1}{2}(1 - 2\lambda_2 + \lambda_2^2)(a_i x_{i,1} + b_i y_{i,1}) + \left(\lambda_2 - \lambda_2^2\right)(a_i x_{i,2} + b_i y_{i,2}) \\
&\qquad\qquad + \frac{1}{2}\left(2 - 4\lambda_2 + \lambda_2^2\right)(a_i x_{i,3} + b_i y_{i,3}) + (1-\lambda_2)\, d_i\ d\lambda_2 \\
&= \frac{A_i k}{3}(a_i(x_{i,1} + x_{i,2} + x_{i,3}) + b_i(y_{i,1} + y_{i,2} + y_{i,3}) + 3d_i)
\end{aligned}
$$

A.1.2 $f_{t_{i,max}}$ Derivation

This derivation shows how we extract bounds for the tangential frictional force f_{t_i} the triangle can resist in terms of the previously defined f_{z_i}. f_{t_i} is maximized when the instantaneous velocity vector $\hat{\mathbf{v}}_i$ is entirely in the tangent plane:

$$
\begin{aligned}
f_{t_i} = \begin{bmatrix} f_{x_i} \\ f_{y_i} \end{bmatrix} &= -\int_{A_i} \mu \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} p_i(\mathbf{r}_i)\, dA_i \leq \mu \begin{bmatrix} f_{z_i} v_{x_i} \\ f_{z_i} v_{y_i} \end{bmatrix} \\
\|f_{t_i}\|_2^2 &\leq (\mu f_{z_i} v_{x_i})^2 + (\mu f_{z_i} v_{y_i})^2 \\
&= (\mu f_{z_i})^2 = (f_{t_{i,\max}})^2,\ |f_{z_i}| \leq f_{z_i,\max}
\end{aligned}
$$

A.1.3 $\tau_{z_{i,max}}$ Derivation

This derivation shows how we extract bounds for the torsional friction τ_{t_i}. τ_{t_i} is maximized when the instantaneous velocity vector $\hat{\mathbf{v}}_i$ is equal to $\begin{bmatrix} -y_i & x_i & 0 \end{bmatrix}^T / \sqrt{x_i^2 + y_i^2}$:

$$
\begin{aligned}
\tau_{z_{i,\max}} &= -\int_{A_i} \mu \|\mathbf{r}_i \times \hat{\mathbf{v}}_i\| p_i(\mathbf{r}_i)\, dA_i \\
&\leq \mu \int_{A_i} \| \begin{bmatrix} x_i & y_i & 0 \end{bmatrix}^T \times \begin{bmatrix} -y_i & x_i & 0 \end{bmatrix}^T / \sqrt{x_i^2 + y_i^2} \| p_i(x_i, y_i)\, dA_i \\
&= \mu \int_{A_i} \sqrt{x_i^2 + y_i^2}\, p_i(x_i, y_i)\, dA_i
\end{aligned}
$$

A.1.4 $\tau_{z_{i,max}}$ Lower Bound

It is non-trivial to integrate $\tau_{z_{i,max}}$ symbolically, as we did for $f_{z_{i,max}}$ in Sec. A.1.1, since it contains an elliptic integral. We therefore compute its lower bound as a conservative

approximation:

$$\begin{aligned}
\tau_{z_{i,\max}} &= \int_{A_i} k(a_i x_i + b_i y_i + d_i)\sqrt{x_i^2 + y_i^2}\ dA_i \\
&\geq \frac{k}{\sqrt{2}}\left(\left|\int_{A_i} x_i(a_i x_i + b_i y_i + d_i)\ dA_i\right| + \left|\iint_{T_i} y_i(a_i x_i + b_i y_i + d_i)\ dA_i\right|\right) \\
&= \frac{1}{\sqrt{2}}\left(|\tau_{y_{i,\max}}| + |\tau_{x_{i,\max}}|\right)
\end{aligned}$$

where

$$\begin{aligned}
(\tau_x)_{i,max} &= \int_{A_i} k y_i(a_i x_i + b_i y_i + d_i)\ dA_i \\
&= \frac{A_i k}{12}(2a_i x_{i,1} y_{i,1} + a_i x_{i,1} y_{i,2} + a_i x_{i,1} y_{i,3} + a_i x_{i,2} y_{i,1} + 2a_i x_{i,2} y_{i,2} \\
&\qquad + a_i x_{i,2} y_{i,3} + a_i x_{i,3} y_{i,1} + a_i x_{i,3} y_{i,2} + 2a_i x_{i,3} y_{i,3} + 2b_i y_{i,1}^2 \\
&\qquad + 2b_i y_{i,1} y_{i,2} + 2b_i y_{i,1} y_{i,3} + 2b_i y_{i,2}^2 + 2b_i y_{i,2} y_{i,3} + 2b_i y_{i,3}^2 \\
&\qquad + 4d y_{i,1} + 4d y_{i,2} + 4d y_{i,3}) \\
(\tau_y)_{i,max} &= \int_{A_i} k x_i(a_i x_i + b_i y_i + d_i)\ dA_i \\
&= \frac{A_i k}{12}(2a_i x_{i,1}^2 + 2a_i x_{i,1} x_{i,2} + 2a_i x_{i,1} x_{i,3} + 2a_i x_{i,2}^2 + 2a_i x_{i,2} x_{i,3} \\
&\qquad + 2a_i x_{i,3}^2 + 2b_i x_{i,1} y_{i,1} + b_i x_{i,1} y_{i,2} + b_i x_{i,1} y_{i,3} + b_i x_{i,2} y_{i,1} \\
&\qquad + 2b_i x_{i,2} y_{i,2} + b_i x_{i,2} y_{i,3} + b_i x_{i,3} y_{i,1} + b_i x_{i,3} y_{i,2} + 2b_i x_{i,3} y_{i,3} \\
&\qquad + 4d x_{i,1} + 4d x_{i,2} + 4d x_{i,3})
\end{aligned}$$

through the same process that we used in Sec. A.1.1. We give the proof of this lower bound in the following proposition.

Proposition $\frac{1}{\sqrt{2}}\left(|(\tau_y)_{i,max}| + |(\tau_x)_{i,max}|\right) \leq (\tau_z)_{i,max}, (a_i x_i + b_i y_i + d_i) > 0$

Proof.

$$
\begin{aligned}
(\tau_z)_{i,max} &= \int_{T_i} k(a_i x_i + b_i y_i + d_i)\sqrt{x_i^2 + y_i^2}\ dA_i \\
&\geq \frac{1}{\sqrt{2}} \int_{A_i} k(a_i x_i + b_i y_i + d_i)\left(|x_i| + |y_i|\right)\ dA_i \\
&\qquad \text{(follows from Cauchy-Schwarz)} \\
&= \frac{1}{\sqrt{2}} \int_{A_i} k\left(|x_i(a_i x_i + b_i y_i + d_i)| + |y_i(a_i x_i + b_i y_i + d_i)|\right)\ dA_i \\
&\qquad \text{for } a_i x_i + b_i y_i + d_i > 0 \\
&\geq \frac{1}{\sqrt{2}} \left(\left| \int_{A_i} k x_i (a_i x_i + b_i y_i + d_i)\ dA_i \right| + \left| \iint_{T_i} k y_i (a_i x_i + b_i y_i + d_i)\ dA_i \right| \right) \\
&= \frac{1}{\sqrt{2}} \left(|(\tau_y)_{i,max}| + |(\tau_x)_{i,max}| \right) \qquad \square
\end{aligned}
$$

$\square$

A.2 Per-Object Experimental Results

Tab. A.1 contains an expanded version of the results, with AP and AR metrics for each of the objects tested. Here, EACH is the non-robust REACH model and REACH is the proposed model.

Object	Point		Point Robust		Ellipse		Ellipse Robust		EACH		REACH		Flex	
	AP	**AR**	**AP**	**AR**	**AP**	**AR**	**AP**	**AR**	**AP**	**AR**	**AP**	**AR**	**AP**	**AR**
Bar Clamp	0.94	0.74	0.96	0.71	0.82	0.30	0.84	0.32	0.84	0.95	0.86	0.95	1.00	0.91
Book	0.99	0.89	0.99	0.86	0.91	0.09	0.86	0.33	0.91	1.00	0.95	1.00	1.00	0.93
Bowl	0.98	0.91	1.00	0.88	0.75	0.09	0.75	0.09	0.75	0.38	0.88	0.47	1.00	0.94
Cat	0.57	0.24	0.70	0.28	0.54	0.55	0.56	0.53	0.70	0.47	0.72	0.32	1.00	0.93
Cube	0.89	0.50	0.92	0.41	0.83	0.73	0.88	0.82	0.81	1.00	0.74	0.93	1.00	0.92
Endstop Holder	1.00	0.86	1.00	0.84	0.99	0.09	0.99	0.15	0.99	0.82	0.99	0.83	1.00	0.95
Engine Part	0.83	0.46	0.92	0.52	0.78	0.49	0.82	0.41	0.72	0.64	0.82	0.62	0.98	0.84
Fan Extruder	0.88	0.40	0.90	0.40	0.87	0.51	0.87	0.79	0.82	0.82	0.88	0.76	0.97	0.69
Gearbox	0.80	0.40	0.89	0.43	0.82	0.45	0.83	0.49	0.81	0.87	0.89	0.82	0.94	0.55
Large Marker	0.60	0.33	0.79	0.25	0.62	0.74	0.86	0.91	0.86	0.79	0.92	0.77	0.93	0.41
Mount1	0.69	0.57	0.80	0.55	0.74	0.45	0.78	0.52	0.61	0.67	0.80	0.67	0.90	0.20
Mug	0.92	0.94	0.99	0.94	0.69	0.09	0.68	0.10	0.83	0.51	0.89	0.40	0.91	0.23
Nozzle	0.81	0.28	0.81	0.32	0.91	0.86	0.89	0.78	0.95	0.81	0.95	0.78	0.87	0.34
Part1	0.86	0.45	0.89	0.49	0.79	0.31	0.87	0.41	0.90	0.64	0.91	0.56	0.83	0.34
Part3	0.73	0.52	0.80	0.45	0.61	0.59	0.68	0.70	0.69	0.75	0.78	0.57	0.72	0.34
Pawn	0.43	0.60	0.39	0.55	0.32	0.49	0.33	0.52	0.43	0.60	0.62	0.55	0.80	0.40
Pear	0.66	0.94	0.81	0.87	0.69	0.80	0.71	0.69	0.70	0.81	0.86	0.67	0.73	0.33
Pipe Connector	0.67	0.54	0.77	0.61	0.46	0.76	0.54	0.63	0.76	0.65	0.74	0.56	0.68	0.17
Sardines	0.94	0.41	0.97	0.56	0.91	0.46	0.96	0.46	0.93	0.92	0.98	0.93	0.78	0.25
Sulfur Neutron	0.13	0.09	0.13	0.09	0.40	0.38	0.40	0.38	0.67	0.66	0.26	0.16	0.86	0.21
Vase	0.72	0.49	0.81	0.49	0.64	0.30	0.72	0.41	0.68	0.46	0.74	0.49	0.87	0.22

Table A.1: Per-object AP and AR scores for each of the models tested.

Appendix B

Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data

This appendix contains dataset statistics and experiments that supplement the segmentation dataset generation and network discussed in Chapter 5.

B.1 WISDOM Dataset Statistics

The real dataset has 3849 total instances with an average of 4.8 object instances per image, fewer than the 7.7 instances per image in the Common Objects in Context (COCO) dataset and the 6.5 instances per image in WISDOM-Sim, but more instances per image than both ImageNet and PASCAL VOC (3.0 and 2.3, respectively). Additionally, it has many more instances that are close to, overlapping with, or occluding other instances, thus making it a more representative dataset for tasks such as bin picking in cluttered environments. Since it is designed for manipulation tasks, most of the objects are much smaller in area than in the COCO dataset, which aims to more evenly distribute instance areas. In the WISDOM dataset, instances take up only 2.28% of total image area in the simulated images, 1.60% of the total image area on average for the high-res images, and 0.94% of the total image area for the low-res images. Figure B.1 compares the distributions of these metrics to the COCO dataset.

B.2 Precision-Recall Evaluation

We performed additional experiments to analyze the precision-recall performance of SD Mask R-CNN along with the baseline methods for category-agnostic instance segmentation on RGB-D images: RGB object proposals [6, 130], cluster-based geometric segmentation

Figure B.1: Distributions of the instances per image and instance size for the WISDOM dataset, with comparisons to the COCO dataset. Average number of instances are listed in parentheses next to each dataset. The number of instances and relative object size make this dataset more applicable to manipulation tasks.

methods from PCL [220], and Mask R-CNN fine-tuned for instance segmentation from the WISDOM-Real dataset. We also include a variant of SD Mask R-CNN fine-tuned on real depth images from WISDOM-Real. We evaluate performance using the widely-used COCO instance segmentation benchmarks [152].

The RGB object proposal baselines were based on two algorithms: Geodesic Object Proposals (GOP) [130] and Multi-scale Combinatorial Grouping (MCG) [6]. GOP identifies critical level sets in signed geodesic distance transforms of the original color images and generates object proposal masks based on these [130]. MCG employs combinatorial grouping of multi-scale segmentation masks and ranks object proposals in the image. For each of these methods, we take the top 100 detections. We then remove masks where less than half of the area of the mask overlaps with the foreground segmentation mask of the image and apply non-max suppression with a threshold of 0.5 Intersection-over-Union (IoU).

Figure B.2 shows precision-recall curves on three test datasets: 2000 images from the WISDOM-Sim validation set and 300 test images each from the Primesense and Phoxi cameras.Our learning-based method produces a ranked list of regions, that can be operated at different operating points. Not only does SD Mask-RCNN achieve higher precision at the same operating point than PCL, it is able to achieve a much higher overall recall without any compromise in precision at the cost of only a handful of extra regions.

We also evaluate the performance of SD Mask R-CNN and several baseline on the WISDOM-Sim test set.

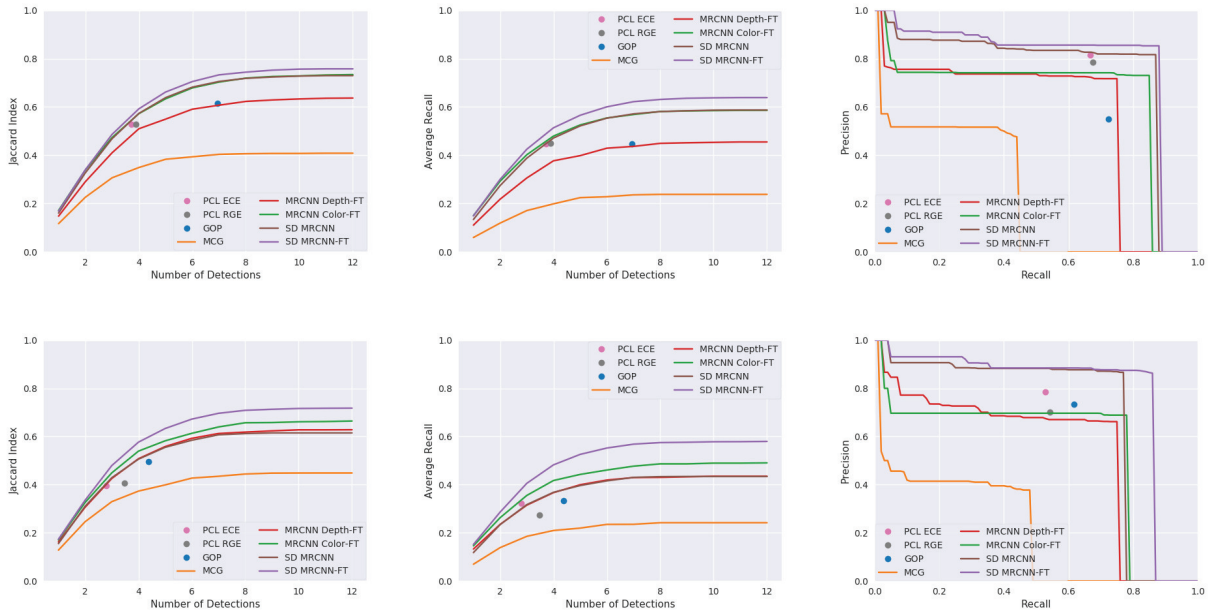


Figure B.2: Average Jaccard Index, Average Recall, and Precision-Recall (at IoU = 0.5) curves for each method and the high-res (top row) and low-res (bottom row) dataset, using segmentation metrics. The fine-tuned SD Mask R-CNN implementation outperforms all baselines on both sensors in the WISDOM-real dataset. The precision-recall curves suggest that the dataset contains some hard instances that are unable to be recalled by any method. These instances are likely heavily occluded objects whose masks get merged with the adjacent mask or flat objects that cannot be distinguished from the bottom of the bin.

Method	**AP**	**AR**
Euclidean Clustering	0.161	0.252
Region Growing	0.172	0.274
SD Mask R-CNN	**0.664**	**0.748**

Table B.1: Average precision and average recall (as defined by COCO benchmarks) on the WISDOM-Sim dataset for the PCL baselines SD Mask R-CNN.

Appendix C

Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter

This appendix contains extended results and implementation details for Chapter 7.

C.1 Extended Results

Tables C.1, C.2, C.3, and C.4 give a detailed breakdown of each policies selected actions and success rate over the 1000 simulated trials and 50 trials on the physical robot for each policy on 15 object heaps. In simulation, pushing actions result in higher success rates. On the physical system, the human policy selects pushing actions much more frequently to clear multiple occluding objects from the target object. We will explore this discrepancy further in future work.

Simulation Policy	**Success Rate**	**Mean Actions**
Random	88.8%	11.26 ± 0.15
Preempted Random	89.7%	8.55 ± 0.16
Preempted Random + Pushing	94.3%	8.87 ± 0.15
Largest-First	90.3%	6.35 ± 0.14
Largest-First + Pushing	93.3%	6.51 ± 0.14

Table C.1: Success rate and mean number of actions (with standard error of the mean) for extraction for 1000 trials of each policy tested in simulation. The largest-first policies extract the target most efficiently, and pushing shows ability to increase overall success rate.

Simulation Policy	Suction	Parallel-Jaw	Push
Random	7015	4467	0
Preempted Random	6168	2870	0
Preempted Random + Pushing	6180	2825	274
Largest-First	5266	1718	0
Largest-First + Pushing	5116	1706	259

Table C.2: Breakdown of action selection for each policy in simulation over 1000 trials. Policies typically attempt many more suction grasps due to better accessibility in clutter, and only attempt pushes a small fraction of the time.

Physical Policy	Success Rate	Mean Actions
Random	92%	10.87 ± 0.66
Preempted Random	90%	6.71 ± 0.61
Preempted Random + Pushing	98%	6.31 ± 0.63
Largest-First	94%	4.85 ± 0.51
Largest-First + Pushing	96%	6.00 ± 0.63
Human	98%	3.06 ± 0.32

Table C.3: Success rate and mean number of actions (with standard error of the mean) for extraction for 50 trials of each policy tested on the physical system. All policies achieve success rates of over 90% due to effective low-level grasping policies, but the human outperforms the best policy by 37% in terms of mean number of actions, suggesting that there is considerable room for action selection policy improvement.

C.2 Siamese Network Implementation Details

The Siamese network architecture involves first passing each input 512×512 RGB image through a ResNet-50 architecture pretrained on ImageNet. During training of the Siamese network, these weights remained fixed. The featurizations of the input images are then concatenated and passed through two dense, fully connected layers: the first with 1024 neurons and ReLU activations, and the second with a single output neuron and sigmoid activation, whose output represents the probability that the two input images are of the same object. The motivation for this architecture is to allow the Siamese network to learn a distance metric over the ResNet-50 featurizations. The training dataset for the Siamese network consists of 5 views of each of the objects used in physical experiments. For each view, we generated a total of 10 additional images: 5 randomly rotated versions of the original image as well as 5 rotated versions that are partially occluded. To simulate occlusions, we

Physical Policy	Suction	Parallel-Jaw	Push
Random	275	300	0
Preempted Random	282	76	0
Preempted Random + Pushing	250	58	19
Largest-First	222	47	0
Largest-First + Pushing	244	59	18
Human	99	27	44

Table C.4: Breakdown of action selection for each policy in 50 physical trials. The human pushes much more frequently than the other policies, especially to clear multiple occluding objects at the beginning of the trial.

took randomly rotated and scaled binary masks of a dataset of synthetic objects, and overlay the masks on the original object. We only used occlusions that covered at least 20% and at most 80% of the original pixels of the object. For training, we sampled 10,000 positive and 10,000 negative image pairs, where a positive image pair consists of an original image of an object and one of the 10 augmented images and a negative image pair consists of an original image of an object and one of the 10 augmented images of an entirely different object. The network is then trained with a contrastive loss function for 10 epochs using a batch size of 64 and the Adam optimizer with a learning rate of 0.0001. For physical experiments, a recognition confidence threshold $t_r = 0.9$ was used.

C.3 Simulated Heap Generation

Simulated heaps are generated by sampling: 1) N objects from a dataset of over 1600 3D models, 2) a heap center around the center of the bin, and 3) planar pose offsets for each object around the heap center. Then, using the Bullet Physics Engine, sampled objects are dropped one by one into the bin from a fixed height at their pose offset from the heap center, and all objects are allowed to come to rest (i.e. all velocities of all objects go to zero). Once all objects have been added to the heap, the modal and amodal segmentation masks for each object are rendered from the camera's perspective. The modal segmentation mask of an object is a segmentation mask of the portion of the object visible from the perspective of the camera (accounting for occlusions), while the amodal segmentation mask of an object is a segmentation mask of the object's exact position in the scene given ground truth information from the simulation environment. Using these masks, the target object is chosen to be the object with the smallest ratio between modal and amodal segmentation mask area (i.e., the least visible object in the bin). This metric is used as a proxy for finding the most occluded object.

C.4 Simulation Policy Parameters

Grasp confidence thresholds of $t_{\text{thresh}} = 0.15$ and $t_{\text{high}} = 0.3$ are used in simulation for the high-level action selector to determine whether to execute grasp actions from the low level grasp policies. In experimental trials, these values were found to provide a balance between avoiding grasp failures and quickly clearing objects from the bin as soon as sufficiently good grasps become available.

C.5 Physical Policy Parameters

In physical experiments, grasp confidence thresholds of $t_{\text{thresh}} = 0.1$ and $t_{\text{high}} = 0.3$ were used for action selection to determine whether to execute grasp action plans from the low-level grasp action policies. These values are similar to those used in simulation, but t_{thresh} is made slightly lower for physical experiments since it we observed that low confidence grasps succeeded more often in physical experiments than in simulation, which was designed to be conservative to encourage good transfer to reality. Additionally, $t_{\text{high}} = 0.5$ was used for policies that included low-level pushing action policies, so that pushing would be further encouraged over low-quality grasp actions.

Bibliography

[1] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. `https://github.com/matterport/Mask_RCNN/`. 2017.

[2] Wisdom C Agboh and Mehmet R Dogar. "Pushing Fast and Slow: Task-Adaptive MPC for Pushing Manipulation Under Uncertainty". In: *Workshop on the Algorithmic Foundation of Robotics (WAFR)*. 2018.

[3] Pulkit Agrawal et al. "Learning to poke by poking: Experiential learning of intuitive physics". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2016, pp. 5074–5082.

[4] Srinivas Akella and Matthew T Mason. "Posing polygonal objects in the plane by pushing". In: *Int. Journal of Robotics Research (IJRR)* 17.1 (1998), pp. 70–88.

[5] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2189–2202.

[6] Pablo Arbeláez et al. "Multiscale combinatorial grouping". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 328–335.

[7] Alper Aydemir et al. "Search in the real world: Active visual object search based on spatial relations". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2011, pp. 2818–2824.

[8] Behnam Bahr, Yingjie Li, and Mahmoud Najafi. "Design and suction cup analysis of a wall climbing robot". In: *Computers & electrical engineering* 22.3 (1996), pp. 193–209.

[9] Chandrajit L Bajaj, Fausto Bernardini, and Guoliang Xu. "Automatic reconstruction of surfaces and scalar fields from 3D scans". In: *Proc. Conf. on Computer graphics and interactive techniques*. 1995, pp. 109–118.

[10] Ruzena Bajcsy. "Active perception". In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[11] Devin J Balkcom and Jeffrey C Trinkle. "Computing wrench cones for planar rigid body contact tasks". In: *Int. Journal of Robotics Research (IJRR)* 21.12 (2002), pp. 1053–1066.

[12] Federico Barbagli et al. "Simulating human fingers: a soft finger proxy model and algorithm". In: *Proc. IEEE Int. S. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. 2004.

[13] Dhruv Batra et al. "Rearrangement: A challenge for embodied ai". In: *arXiv preprint arXiv:2011.01975* (2020).

[14] Patrick Beeson and Barrett Ames. "TRAC-IK: An open-source library for improved solving of generic inverse kinematics". In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2015, pp. 928–935.

[15] Wissam Bejjani et al. "Occlusion-Aware Search for Object Retrieval in Clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2021.

[16] Wissam Bejjani et al. "Planning with a receding horizon for manipulation in clutter using a learned value function". In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2018, pp. 1–9.

[17] Dmitry Berenson and Siddhartha S Srinivasa. "Grasp synthesis in cluttered environments for dexterous hands". In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2008, pp. 189–196.

[18] Matthew Berger et al. "A survey of surface reconstruction from point clouds". In: *Computer Graphics Forum*. Vol. 36. 1. 2017, pp. 301–329.

[19] Antonio Bicchi. "On the closure properties of robotic grasping". In: *Int. Journal of Robotics Research (IJRR)* 14.4 (1995), pp. 319–334.

[20] Antonio Bicchi and Vijay Kumar. "Robotic grasping and contact: A review". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2000.

[21] Jeannette Bohg et al. "Data-driven grasp synthesis – a survey". In: *IEEE Trans. Robotics* 30.2 (2014).

[22] Jeannette Bohg et al. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Trans. Robotics* 33.6 (2017), pp. 1273–1291.

[23] Ch Borst, Max Fischer, and Gerd Hirzinger. "Grasp planning: How to choose a suitable task wrench space". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Vol. 1. 2004, pp. 319–325.

[24] Abdeslam Boularias, James Andrew Bagnell, and Anthony Stentz. "Learning to Manipulate Unknown Objects in Clutter by Reinforcement." In: *Proc. AAAI Conf. on Artificial Intelligence*. 2015, pp. 1336–1342.

[25] Konstantinos Bousmalis et al. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 4243–4250.

[26] Randy C Brost. "Automatic grasp planning in the presence of uncertainty". In: *Int. Journal of Robotics Research (IJRR)* 7.1 (1988), pp. 3–17.

[27] Francisc Bungiu et al. "Efficient computation of visibility polygons". In: *arXiv preprint arXiv:1403.3905* (2014).

[28] Berk Calli et al. "The ycb object and model set: Towards common benchmarks for manipulation research". In: *Proc. IEEE Int. Conf. Advanced Robotics (ICAR)*. 2015, pp. 510–517.

[29] Hanwen Cao et al. "Suctionnet-1billion: A large-scale benchmark for suction grasping". In: *IEEE Robotics & Automation Letters* 6.4 (2021), pp. 8718–8725.

[30] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. "Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics". In: *Proc. Robotics: Science and Systems (RSS)*. 2015.

[31] Justin Carpentier and Nicolas Mansard. "Multicontact locomotion of legged robots". In: *IEEE Trans. Robotics* 34.6 (2018), pp. 1441–1460.

[32] Joao Carreira et al. "Semantic segmentation with second-order pooling". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2012, pp. 430–443.

[33] Rohan Chabra et al. "Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2020.

[34] Lillian Chang, Joshua R Smith, and Dieter Fox. "Interactive singulation of objects from a pile". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012, pp. 3875–3882.

[35] Krzysztof Charusta et al. "Independent contact regions based on a patch contact model". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012.

[36] Yi-Ting Chen, Xiaokai Liu, and Ming-Hsuan Yang. "Multi-instance object segmentation with occlusion handling". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3470–3478.

[37] Xiaozhi Chen et al. "3d object proposals using stereo imagery for accurate object class detection". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 40.5 (2018), pp. 1259–1272.

[38] Zhiqin Chen and Hao Zhang. "Learning implicit fields for generative shape modeling". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5939–5948.

[39] Özgün Çiçek et al. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *Int. Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2016, pp. 424–432.

[40] Matei Ciocarlie, Claire Lackner, and Peter Allen. "Soft finger model with adaptive contact geometry for grasping and manipulation tasks". In: *Proc. IEEE Eurohaptics Conf. and S. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. 2007.

[41] Matei Ciocarlie, Andrew Miller, and Peter Allen. "Grasp analysis using deformable fingers". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2005.

[42] Matei Ciocarlie et al. "Functional analysis of finger contact locations during grasping". In: *Proc. IEEE Eurohaptics Conf. and S. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. 2009.

[43] Matei Ciocarlie et al. "Towards reliable grasping and manipulation in household environments". In: *Int. S. Experimental Robotics (ISER)*. 2014, pp. 241–252.

[44] Enric Corona, Kaustav Kundu, and Sanja Fidler. "Pose estimation for objects with rotational symmetry". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2018, pp. 7215–7222.

[45] Akansel Cosgun et al. "Push planning for object placement on cluttered table surfaces". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2011, pp. 4627–4632.

[46] Erwin Coumans and Yunfei Bai. *pybullet, a Python module for physics simulation, games, robotics and machine learning*. `http://pybullet.org/`. 2017.

[47] Angela Dai and Matthias Nießner. "Scan2mesh: From unstructured range scans to 3d meshes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5574–5583.

[48] Michael Danielczuk et al. "Linear Push Policies to Increase Grasp Access for Robot Bin Picking". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2018.

[49] Michael Danielczuk et al. "Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 1614–1621.

[50] Michael Danielczuk et al. "REACH: Reducing False Negatives in Robot Grasp Planning with a Robust Efficient Area Contact Hypothesis Model". In: *Int. S. Robotics Research (ISRR)*. 2019.

[51] Michael Danielczuk et al. "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 7283–7290.

[52] Michael Danielczuk et al. "X-Ray: Mechanical Search for an Occluded Object by Minimizing Support of Learned Occupancy Distributions". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[53] Nikhil Das and Michael Yip. "Learning-based proxy collision detection for robot motion planning applications". In: *IEEE Trans. Robotics* (2020).

[54] M Dogar et al. "Physics-based grasp planning through clutter". In: *Proc. Robotics: Science and Systems (RSS)*. 2012.

[55] Mehmet R Dogar and Siddhartha S Srinivasa. "Push-grasping with dexterous hands: Mechanics and a method". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2010, pp. 2123–2130.

[56] Aaron M Dollar and Robert D Howe. "The highly adaptive SDM hand: Design and performance evaluation". In: *Int. Journal of Robotics Research (IJRR)* 29.5 (2010).

[57] Yukiyasu Domae et al. "Fast graspability evaluation on single depth maps for bin picking with general grippers". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014, pp. 1997–2004.

[58] Yan Duan et al. "One-shot imitation learning". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)* 30 (2017).

[59] Herbert Edelsbrunner and Ernst P Mücke. "Three-dimensional alpha shapes". In: *ACM Trans. Graphics (TOG)* 13.1 (1994), pp. 43–72.

[60] Andreas Eitel, Nico Hauff, and Wolfram Burgard. "Learning to Singulate Objects using a Push Proposal Network". In: *Int. S. Robotics Research (ISRR)*. 2017.

[61] Ian Endres and Derek Hoiem. "Category independent object proposals". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2010, pp. 575–588.

[62] Clemens Eppner, Arsalan Mousavian, and Fox Dieter. "ACRONYM: A Large-Scale Grasp Dataset Based on Simulation". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)* (2021).

[63] Clemens Eppner et al. "Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems." In: *Proc. Robotics: Science and Systems (RSS)*. 2016.

[64] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *Int. journal of computer vision* 88.2 (2010), pp. 303–338.

[65] Kuan Fang et al. "Learning Task-Oriented Grasping for Tool Manipulation from Simulated Self-Supervision". In: *Proc. Robotics: Science and Systems (RSS)*. 2018.

[66] Carlo Ferrari and John Canny. "Planning optimal grasps". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1992.

[67] Mauro Figueiredo et al. "Collision Detection for Point Cloud Models With Bounding Spheres Hierarchies". In: *Int. Journal of Virtual Reality* 11.2 (2012), pp. 37–43.

[68] Richard E Fikes and Nils J Nilsson. "STRIPS: A new approach to the application of theorem proving to problem solving". In: *Artificial intelligence* 2.3-4 (1971), pp. 189–208.

[69] Clement Fuji Tsang et al. *Kaolin: A Pytorch Library for Accelerating 3D Deep Learning Research*. `https://github.com/NVIDIAGameWorks/kaolin`. 2022.

[70] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[71] Abdul Ghafoor, Jian S Dai, and Joseph Duffy. "Stiffness modeling of the soft-finger contact in robotic grasping". In: *Journal of mechanical design* 126.4 (2004).

[72] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203.

[73] Ken Goldberg et al. "Part pose statistics: Estimators and experiments". In: *IEEE Trans. Robotics and Automation* 15.5 (1999).

[74] Kenneth Y Goldberg. "Orienting polygonal parts without sensors". In: *Algorithmica* 10.2-4 (1993), pp. 201–225.

[75] Kenneth Yigael Goldberg. "Stochastic plans for robotic manipulation". PhD thesis. Carnegie Mellon University, 1991.

[76] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. "Planar sliding with dry friction part 1. limit surface and moment function". In: *Wear* 143.2 (1991).

[77] Thibault Groueix et al. "A papier-mâché approach to learning 3d surface generation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 216–224.

[78] Marcus Gualtieri and Robert Platt. "Learning 6-dof grasping and pick-place using attention focus". In: *Conf. on Robot Learning (CoRL)*. 2018, pp. 477–486.

[79] Marcus Gualtieri et al. "High precision grasp pose detection in dense clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 598–605.

[80] Menglong Guo et al. "Design of parallel-jaw gripper tip surfaces for robust grasping". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017.

[81] Megha Gupta et al. "Interactive environment exploration in clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2013, pp. 5265–5272.

[82] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. "Perceptual organization and recognition of indoor scenes from RGB-D images". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 564–571.

[83] Saurabh Gupta et al. "Cognitive mapping and planning for visual navigation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2616–2625.

[84] Saurabh Gupta et al. "Learning rich features from RGB-D images for object detection and segmentation". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2014, pp. 345–360.

[85] Kensuke Harada et al. "Stability of soft-finger grasp under gravity". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014, pp. 883–888.

[86] Bharath Hariharan et al. "Hypercolumns for object segmentation and fine-grained localization". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 447–456.

[87] Bharath Hariharan et al. "Simultaneous detection and segmentation". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2014, pp. 297–312.

[88] Robert Haschke et al. "Task-oriented quality measures for dextrous grasping." In: *Cira*. 2005, pp. 689–694.

[89] Joshua A Haustein et al. "Learning Manipulation States and Actions for Efficient Non-prehensile Rearrangement Planning". In: *arXiv preprint arXiv:1901.03557* (2019).

[90] Kaiming He et al. "Deep residual learning for image recognition". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[91] Kaiming He et al. "Mask r-cnn". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2017, pp. 2961–2969.

[92] Tucker Hermans, James M Rehg, and Aaron Bobick. "Guided pushing for object singulation". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 4783–4790.

[93] Carlos Hernandez et al. "Team delft's robot winner of the amazon picking challenge 2016". In: *Robot World Cup*. 2016, pp. 613–624.

[94] Heinrich Rudolf Hertz. "Uber die Beruhrung fester elastischer Korper und Uber die Harte". In: *Verhandlung des Vereins zur Beforderung des GewerbefleiBes, Berlin* (1882).

[95] Stefan Hinterstoisser et al. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes". In: *Proc. Asian Conf. on Computer Vision (ACCV)*. 2012, pp. 548–562.

[96] Robert D Howe and Mark R Cutkosky. "Practical force-motion models for sliding manipulation". In: *Int. Journal of Robotics Research (IJRR)* 15.6 (1996), pp. 557–572.

[97] Robert D Howe, Imin Kao, and Mark R Cutkosky. "The sliding of robot fingers under combined torsion and shear loading". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1988, pp. 103–105.

[98] Eric Huang, Zhenzhong Jia, and Matthew T Mason. "Large-scale multi-object rearrangement". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 211–218.

[99] Huang Huang et al. "Mechanical Search on Shelves using Lateral Access X-RAY". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2021.

[100] Isabella Huang et al. "DefGraspSim: Physics-based simulation of grasp outcomes for 3D deformable objects". In: *IEEE Robotics & Automation Letters* (2022).

[101] Philip M Hubbard. "Approximating polyhedra with spheres for time-critical collision detection". In: *ACM Trans. Graphics (TOG)* 15.3 (1996), pp. 179–210.

[102] Kao-Shing Hwang, JL Ling, and Wei-Han Wang. "Adaptive reinforcement learning in box-pushing robots". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2014, pp. 1182–1187.

[103] Takahiro Inoue and Shinichi Hirai. "Elastic model of deformable fingertip for soft-fingered manipulation". In: *IEEE Trans. Robotics* 22.6 (2006).

[104] Eric Jang et al. "End-to-End Learning of Semantic Grasping". In: *Conf. on Robot Learning (CoRL)* (2017).

[105] Chiyu Jiang et al. "Local implicit grid representations for 3d scenes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 6001–6010.

[106] Edward Johns, Stefan Leutenegger, and Andrew J Davison. "Deep learning a grasp function for grasping under gripper pose uncertainty". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 4461–4468.

[107] Matthew Johnson-Roberson et al. "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 746–753.

[108] Rico Jonschkowski et al. "Probabilistic multi-class segmentation for the amazon picking challenge". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 1–7.

[109] Krishnanand N Kaipa et al. "Enhancing robotic unstructured bin-picking performance by enabling remote human interventions in challenging perception scenarios". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2016, pp. 639–645.

[110] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *Conf. on Robot Learning (CoRL)*. 2018.

[111] Dmitry Kalashnikov et al. "Scalable deep reinforcement learning for vision-based robotic manipulation". In: *Conf. on Robot Learning (CoRL)*. 2018, pp. 651–673.

[112] Gaetano Kanizsa. *Organization in vision: Essays on Gestalt perception*. 1979.

[113] Imin Kao and Mark R Cutkosky. "Quasistatic manipulation with compliance and sliding". In: *Int. Journal of Robotics Research (IJRR)* 11.1 (1992), pp. 20–40.

[114] Imin Kao, Kevin Lynch, and Joel W Burdick. "Contact modeling and manipulation". In: *Springer Handbook of Robotics*. 2008.

[115] Imin Kao and Fuqian Yang. "Stiffness and contact mechanics for soft fingers in grasping and manipulation". In: *IEEE Trans. Robotics and Automation* 20.1 (2004), pp. 132–135.

[116] Dov Katz et al. "Clearing a pile of unknown objects using interactive perception". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2013, pp. 154–161.

[117] Dov Katz et al. "Perceiving, learning, and exploiting object affordances for autonomous pile manipulation". In: *Autonomous Robots* 37.4 (2014), pp. 369–382.

[118] Wadim Kehl et al. "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2017, pp. 1521–1529.

[119] Ben Kehoe, Dmitry Berenson, and Ken Goldberg. "Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012, pp. 576–583.

[120] Ben Kehoe et al. "A survey of research on cloud robotics and automation". In: *IEEE Trans. Automation Science and Engineering* 12.2 (2015), pp. 398–409.

[121] Jacqueline Kenney, Thomas Buckley, and Oliver Brock. "Interactive segmentation for manipulation in unstructured environments". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2009, pp. 1377–1382.

[122] J. Chase Kew et al. "Neural Collision Clearance Estimator for Fast Robot Motion Planning". In: *arXiv preprint arXiv:1910.05917* (2019).

[123] Chung Min Kim et al. "Simulation of Parallel-Jaw Grasping using Incremental Potential Contact Models". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2022.

[124] Jennifer E King, Marco Cognetti, and Siddhartha S Srinivasa. "Rearrangement planning using object-centric and robot-centric action spaces". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2016, pp. 3940–3947.

[125] Jan Klein and Gabriel Zachmann. "Point cloud collision detection". In: *Computer Graphics Forum*. Vol. 23. 3. 2004, pp. 567–576.

[126] Jens Kober, J Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey". In: *Int. Journal of Robotics Research (IJRR)* 32.11 (2013), pp. 1238–1274.

[127] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition". In: *ICML Deep Learning Workshop*. Vol. 2. 2015.

[128] Simon Kohl et al. "A probabilistic u-net for segmentation of ambiguous images". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2018, pp. 6965–6975.

[129] Ilya Kostrikov, Dumitru Erhan, and Sergey Levine. "End to end active perception". In: (2016).

[130] Philipp Krähenbühl and Vladlen Koltun. "Geodesic object proposals". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2014, pp. 725–739.

[131] Senka Krivic, Emre Ugur, and Justus Piater. "A robust pushing skill for object delivery between obstacles". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2016, pp. 1184–1189.

[132] Robert Krug, Yasemin Bekiroglu, and Máximo A Roa. "Grasp quality evaluation done right: How assumed contact force bounds affect wrench-based quality metrics". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017.

[133] Sumit Kumar, Shushman Choudhary, and Siddhartha Srinivasa. "Learning Configuration Space Belief Model from Collision Checks for Motion Planning". In: *arXiv preprint arXiv:1901.07646* (2019).

[134] Weicheng Kuo, Bharath Hariharan, and Jitendra Malik. "Deepbox: Learning objectness with convolutional networks". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2015, pp. 2479–2487.

[135] Weicheng Kuo et al. "Shapemask: Learning to segment novel objects by refining shape priors". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2019, pp. 9207–9216.

[136] Andrey Kurenkov et al. "Semantic and geometric modeling with neural message passing in 3d scene graphs for hierarchical mechanical search". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021, pp. 11227–11233.

[137] Andrey Kurenkov et al. "Visuomotor mechanical search: Learning to retrieve target objects in clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020, pp. 8408–8414.

[138] Michael Laskey et al. "Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 358–365.

[139] Michael Laskey et al. "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2016, pp. 827–834.

[140] Jinhwi Lee et al. "Efficient obstacle rearrangement for object manipulation tasks in cluttered environments". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 183–189.

[141] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *Int. Journal of Robotics Research (IJRR)* 34.4-5 (2015), pp. 705–724.

[142] Mark Levi. *The mathematical mechanic: using physical reasoning to solve problems*. 2009.

[143] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection". In: *Int. Journal of Robotics Research (IJRR)* 37.4-5 (2018), pp. 421–436.

[144] Sergey Levine et al. "Learning hand-eye coordination for robotic grasping with large-scale data collection". In: *Int. S. Experimental Robotics (ISER)*. 2016, pp. 173–184.

[145] Jue Kun Li, David Hsu, and Wee Sun Lee. "Act to see and see to act: POMDP planning for objects search in clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 5701–5707.

[146] Wai Ho Li and Lindsay Kleeman. "Autonomous segmentation of near-symmetric objects through vision and robotic nudging". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2008, pp. 3604–3609.

[147] Yanmei Li and Imin Kao. "A review of modeling of soft-contact fingers and stiffness control for dextrous manipulation in robotics". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Vol. 3. 2001, pp. 3055–3060.

[148] Yi Li et al. "Deepim: Deep iterative matching for 6d pose estimation". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2018, pp. 683–698.

[149] Zexiang Li and S Shankar Sastry. "Task-oriented optimal grasping by multifingered robot hands". In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 32–44.

[150] Jacky Liang et al. "GPU-accelerated robotic simulation for distributed reinforcement learning". In: *Conf. on Robot Learning (CoRL)*. 2018.

[151] Guosheng Lin et al. "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[152] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2014, pp. 740–755.

[153] Yun Lin and Yu Sun. "Grasp planning to maximize task coverage". In: *Int. Journal of Robotics Research (IJRR)* 34.9 (2015), pp. 1195–1210.

[154] Zhijian Liu et al. "Point-Voxel CNN for efficient 3D deep learning". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2019, pp. 965–975.

[155] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[156] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987), pp. 163–169.

[157] Xibai Lou, Yang Yang, and Changhyun Choi. "Collision-Aware Target-Driven Object Grasping in Constrained Environments". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021.

[158] Kevin M Lynch. "The mechanics of fine manipulation by pushing". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1992, pp. 2269–2276.

[159] Kevin M Lynch and Matthew T Mason. "Controllability of pushing". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. Vol. 1. 1995, pp. 112–119.

[160] Kevin M Lynch and Matthew T Mason. "Stable pushing: Mechanics, controllability, and planning". In: *Int. Journal of Robotics Research (IJRR)* 15.6 (1996), pp. 533–556.

[161] Miles Macklin et al. "Non-Smooth Newton Methods for Deformable Multi-Body Dynamics". In: *ACM Trans. Graphics (TOG)* 38.5 (2019).

[162] Jeffrey Mahler and Ken Goldberg. "Learning deep policies for robot bin picking by simulating robust grasping sequences". In: *Conf. on Robot Learning (CoRL)*. 2017, pp. 515–524.

[163] Jeffrey Mahler et al. "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2016, pp. 1957–1964.

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

[165] Jeffrey Mahler et al. "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 5620–5627.

[166] Jeffrey Mahler et al. "Learning ambidextrous robot grasping policies". In: *Science Robotics* 4.26 (2019), eaau4984.

[167] Fabian Manhardt et al. "Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2019.

[168] Duane W Marhefka and David E Orin. "A compliant contact model with nonlinear damping for simulation of robotic systems". In: *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans* 29.6 (1999).

[169] Pat Marion et al. "LabelFusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018.

[170] Jeremy A Marvel et al. "Technology readiness levels for randomized bin picking". In: *Proc. Workshop on Performance Metrics for Intelligent Systems*. 2012, pp. 109–113.

[171] Matthew T Mason. *Mechanics of robotic manipulation*. 2001.

[172] Matthew T Mason and J Kenneth Salisbury Jr. *Robot hands and the mechanics of manipulation*. 1985.

[173] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. "Super 4pcs fast global pointcloud registration via smart indexing". In: *Computer Graphics Forum*. Vol. 33. 5. 2014.

[174] Lars Mescheder et al. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4460–4470.

[175] Claudio Michaelis, Matthias Bethge, and Alexander Ecker. "One-shot segmentation in clutter". In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2018, pp. 3549–3558.

[176] Anton Milan et al. "Semantic segmentation from limited training data". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 1908–1915.

[177] Bhubaneswar Mishra, Jacob T Schwartz, and Micha Sharir. "On the existence and synthesis of multifinger positive grips". In: *Algorithmica* 2.1-4 (1987), pp. 541–558.

[178] Mark Moll, Lydia Kavraki, Jan Rosell, et al. "Randomized physics-based motion planning for grasping in cluttered and uncertain environments". In: *IEEE Robotics & Automation Letters* 3.2 (2017), pp. 712–719.

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

[180] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Learning robust, real-time, reactive robotic grasping". In: *Int. Journal of Robotics Research (IJRR)* 39.2-3 (2020), pp. 183–201.

[181] Douglas Morrison et al. "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 7757–7764.

[182] Tomohiro Motoda et al. "Bimanual Shelf Picking Planner Based on Collapse Prediction". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. 2021, pp. 510–515.

[183] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-dof graspnet: Variational grasp generation for object manipulation". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2019, pp. 2901–2910.

[184] Arsalan Mousavian et al. "Visual representations for semantic target driven navigation". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 8846–8852.

[185] Adithyavairavan Murali et al. "6-DOF Grasping for Target-driven Object Manipulation in Clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 6232–6238.

[186] Richard M Murray. *A mathematical introduction to robotic manipulation*. 2017.

[187] Van-Duc Nguyen. "Constructing force-closure grasps". In: *Int. Journal of Robotics Research (IJRR)* 7.3 (1988), pp. 3–16.

[188] Tonci Novkovic et al. "Object finding in cluttered scenes using interactive perception". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 8338–8344.

[189] Manuel M Oliveira et al. "Fast digital image inpainting". In: *Proc. Int. Conf. on Visualization, Imaging and Image Processing (VIIP)*. 2001, pp. 106–107.

[190] Damir Omrčen et al. "Autonomous acquisition of pushing actions to support object grasping with a humanoid robot". In: *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*. 2009, pp. 277–283.

[191] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A general purpose library for collision and proximity queries". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012, pp. 3859–3866.

[192] Jia Pan, Sachin Chitta, and Dinesh Manocha. "Probabilistic collision detection between noisy point clouds using robust classification". In: *Int. S. Robotics Research (ISRR)*. 2011, pp. 77–94.

[193] Jia Pan and Dinesh Manocha. "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing". In: *Int. Journal of Robotics Research (IJRR)* 35.12 (2016), pp. 1477–1496.

[194] Jia Pan and Dinesh Manocha. "GPU-based parallel collision detection for fast motion planning". In: *Int. Journal of Robotics Research (IJRR)* 31.2 (2012), pp. 187–200.

[195] Jeong Joon Park et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 165–174.

[196] Andreas ten Pas and Robert Platt. "Using geometry to detect grasp poses in 3d point clouds". In: *Proc. Robotics: Science and Systems (RSS)*. 2018, pp. 307–324.

[197] Andreas ten Pas et al. "Grasp pose detection in point clouds". In: *Int. Journal of Robotics Research (IJRR)* 36.13-14 (2017), pp. 1455–1473.

[198] Chris Paxton et al. "Do what i want, not what i did: Imitation of skills by planning sequences of actions". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016.

[199] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollár. "Learning to segment object candidates". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2015, pp. 1990–1998.

[200] Pedro O Pinheiro et al. "Learning to refine object segments". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2016, pp. 75–91.

[201] Lerrel Pinto and Abhinav Gupta. "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2016, pp. 3406–3413.

[202] Mamy Pouliquen et al. "Real-time finite element finger pinch grasp simulation". In: *Proc. IEEE Eurohaptics Conf. and S. on Haptic Interfaces for Virtual Environment and Teleoperator Systems*. 2005.

[203] Domenico Prattichizzo and Jeffrey C Trinkle. "Grasping". In: *Springer handbook of robotics*. 2008, pp. 671–700.

[204] Andrew Price, Linyi Jin, and Dmitry Berenson. "Inferring occluded geometry improves performance when retrieving an object from dense clutter". In: *Int. S. Robotics Research (ISRR)*. 2019.

[205] The CGAL Project. *CGAL User and Reference Manual*. 5.3. 2021. URL: `https://doc.cgal.org/5.3/Manual/packages.html`.

[206] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. "Deep directional statistics: Pose estimation with uncertainty quantification". In: *Proc. European Conf. on Computer Vision (ICCV)*. 2018, pp. 534–551.

[207] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, pp. 652–660.

[208] Charles Ruizhongtai Qi et al. "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2017, pp. 5099–5108.

[209] Zhapeng Qiu et al. "Human motions analysis and simulation based on a general criterion of stability". In: *International Symposium on Digital Human Modeling*. 2011, pp. 1–8.

[210] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. "Segmentation of point clouds using smoothness constraint". In: *International archives of photogrammetry, remote sensing and spatial information sciences* 36.5 (2006), pp. 248–253.

[211] Mahdi Rad and Vincent Lepetit. "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2017, pp. 3828–3836.

[212] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. "RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion". In: *Proc. Robotics: Science and Systems (RSS)*. 2018, pp. 26–30.

[213] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[214] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2015, pp. 91–99.

[215] Elon Rimon and Joel Burdick. *The Mechanics of Robot Grasping*. 2019.

[216] Máximo A Roa and Raúl Suárez. "Grasp quality measures: review and performance". In: *Autonomous robots* 38.1 (2015), pp. 65–88.

[217] German Ros et al. "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3234–3243.

[218] Christian Rupprecht et al. "Learning in an uncertain world: Representing ambiguity through multiple hypotheses". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2017, pp. 3591–3600.

[219] Radu Bogdan Rusu. "Semantic 3D object maps for everyday manipulation in human living environments". In: *KI-Künstliche Intelligenz* 24.4 (2010), pp. 345–348.

[220] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2011, pp. 1–4.

[221] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PLOS One* 10.3 (2015).

[222] J Kenneth Salisbury and B Roth. "Kinematic and force analysis of articulated mechanical hands". In: *Journal of Mechanisms, Transmissions, and Automation in Design* 105.1 (1983).

[223] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks". In: *IEEE Robotics & Automation Letters* 4.2 (2019), pp. 1357–1364.

[224] Ashutosh Saxena, Lawson LS Wong, and Andrew Y Ng. "Learning Grasp Strategies with Partial Shape Information". In: *Proc. AAAI Conf. on Artificial Intelligence*. Vol. 3. 2. 2008, pp. 1491–1494.

[225] Max Schwarz et al. "NimbRo Picking: Versatile part handling for warehouse automation". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 3032–3039.

[226] Max Schwarz et al. "RGB-D object detection and semantic segmentation for autonomous manipulation in clutter". In: *Int. Journal of Robotics Research (IJRR)* (2016), p. 0278364917713117.

[227] Amirreza Shaban et al. "One-shot learning for semantic segmentation". In: *British Machine Vision Conference (BMVC)*. 2017.

[228] Lin Shao, Ye Tian, and Jeannette Bohg. "ClusterNet: Instance Segmentation in RGB-D Images". In: *arXiv preprint arXiv:1807.08894* (2018).

[229] Rahul Shome et al. "Fast, high-quality dual-arm rearrangement in synchronous, monotone tabletop setups". In: *Workshop on the Algorithmic Foundation of Robotics (WAFR)*. 2018, pp. 778–795.

[230] Jamie Shotton et al. "Real-time human pose recognition in parts from single depth images". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 1297–1304.

[231] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[232] Pramath R Sinha and Jacob M Abel. "A contact stress model for multifingered grasps of rough objects". In: *IEEE Trans. Robotics and Automation* 8.1 (1992).

[233] Shuran Song et al. "Grasping in the Wild: Learning 6DoF Closed-Loop Grasping from Low-Cost Demonstrations". In: *IEEE Robotics & Automation Letters* (2020).

[234] Shuran Song et al. "Semantic scene completion from a single depth image". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1746–1754.

[235] Siddharth Srivastava et al. "Combined task and motion planning through an extensible planner-independent interface layer". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014, pp. 639–646.

[236] Stefan Stevšić, Sammy Christen, and Otmar Hilliges. "Learning to assemble: Estimating 6D poses for robotic object-object manipulation". In: *IEEE Robotics & Automation Letters* 5.2 (2020), pp. 1159–1166.

[237] Morten Strandberg. "A grasp evaluation procedure based on disturbance forces". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 2. 2002, pp. 1699–1704.

[238] Hannah S Stuart et al. "Suction helps in a pinch: Improving underwater manipulation with gentle suction flow". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2015, pp. 2279–2284.

[239] Hao Su et al. "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2015, pp. 2686–2694.

[240] Martin Sundermeyer et al. "Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)* (2021).

[241] Jaeyong Sung, Bart Selman, and Ashutosh Saxena. "Learning sequences of controllers for complex manipulation tasks". In: *Proc. Int. Conf. on Machine Learning (ICML)*. 2013.

[242] Richard S Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2 (1999), pp. 181–211.

[243] Zhou Teng and Jing Xiao. "Surface-Based Detection and 6-DoF Pose Estimation of 3-D Objects in Cluttered Scenes". In: *IEEE Trans. Robotics* 32.6 (2016), pp. 1347–1361.

[244] Demetri Terzopoulos and Manuela Vasilescu. "Sampling and reconstruction with adaptive meshes." In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Vol. 91. 1991, pp. 70–75.

[245] Paolo Tiezzi and Imin Kao. "Modeling of viscoelastic contacts and evolution of limit surface for robotic contact interface". In: *IEEE Trans. Robotics* 23.2 (2007), pp. 206–217.

[246] Josh Tobin et al. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 23–30.

[247] Tuan Tran, Jory Denny, and Chinwe Ekenna. "Predicting Sample Collision with Neural Networks". In: *arXiv preprint arXiv:2006.16868* (2020).

[248] Tokuo Tsuji et al. "Grasp planning for constricted parts of objects approximated with quadric surfaces". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2014.

[249] Angel J Valencia et al. "A 3D vision based approach for optimal grasp of vacuum grippers". In: *IEEE Int. Workshop of electronics, control, measurement, signals and their application to mechatronics (ECMSM)*. 2017, pp. 1–6.

[250] Koen EA Van de Sande et al. "Segmentation as selective search for object recognition". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2011, pp. 1879–1886.

[251] Mark Van der Merwe et al. "Learning continuous 3d reconstructions for geometrically aware grasping". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 11516–11522.

[252] Jacob Varley et al. "Shape completion enabled robotic grasping". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 2442–2447.

[253] Ulrich Viereck et al. "Learning a visuomotor controller for real world robotic grasping using simulated depth images". In: *Conf. on Robot Learning (CoRL)*. 2017, pp. 291–300.

[254] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Proc. Conf. on Neural Information Processing Systems (NeurIPS)*. 2016, pp. 3630–3638.

[255] Anh-Vu Vo et al. "Octree-based region growing for point cloud segmentation". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 104 (2015), pp. 88–100.

[256] Rui Wang, Yinglong Miao, and Kostas E Bekris. "Efficient and High-quality Prehensile Rearrangement in Cluttered and Confined Spaces". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2022.

[257] Weiyue Wang et al. "SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[258] *Weiss robotics tactile sensor*. URL: `https://www.weiss-robotics.com/en/produkte/unkategorisiert/wts-en` (visited on 08/26/2019).

[259] Eric W Weisstein. "Barycentric coordinates". In: *MathWorld – A Wolfram Web Resource* (2003).

[260] Jonathan Weisz and Peter K Allen. "Pose error robust grasping from contact wrench space metrics". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012.

[261] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. "Model predictive path integral control: From theory to parallel computation". In: *Journal of Guidance, Control, and Dynamics* 40.2 (2017), pp. 344–357.

[262] Jason Wolfe, Bhaskara Marthi, and Stuart J Russell. "Combined Task and Motion Planning for Mobile Manipulation". In: *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*. 2010.

[263] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920.

[264] Yu Xiang et al. "Learning RGB-D Feature Embeddings for Unseen Object Instance Segmentation". In: *Conf. on Robot Learning (CoRL)*. 2020.

[265] Yu Xiang et al. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes". In: *Proc. Robotics: Science and Systems (RSS)*. 2018.

[266] Yuchen Xiao et al. "Online Planning for Target Object Search in Clutter under Partial Observability". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 8241–8247.

[267] Chris Xie et al. "Rice: Refining instance masks in cluttered environments with graph neural networks". In: *Conf. on Robot Learning (CoRL)*. 2022, pp. 1655–1665.

[268] Christopher Xie et al. "Unseen object instance segmentation for robotic environments". In: *IEEE Trans. Robotics* 37.5 (2021), pp. 1343–1359.

[269] Danfei Xu et al. "Neural task programming: Learning to generalize across hierarchical tasks". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018.

[270] Jingyi Xu et al. "6DLS: Modeling Nonplanar Frictional Surface Contacts for Grasping Using 6-D Limit Surfaces". In: *IEEE Trans. Robotics* 37.6 (2021), pp. 2099–2116.

[271] Jingyi Xu et al. "Grasping posture estimation for a two-finger parallel gripper with soft material jaws using a curved contact area friction model." In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 2253–2260.

[272] Jingyi Xu et al. "Minimal work: A grasp quality metric for deformable hollow objects". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 1546–1552.

[273] Nicholas Xydas, Milind Bhagavat, and Imin Kao. "Study of soft-finger contact mechanics using finite elements analysis and experiments". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2000.

[274] Nicholas Xydas and Imin Kao. "Modeling of contact mechanics and friction limit surfaces for soft fingers in robotics, with experimental results". In: *Int. Journal of Robotics Research (IJRR)* 18.9 (1999).

[275] Gengshan Yang, Peiyun Hu, and Deva Ramanan. "Inferring Distributions Over Depth from a Single Image". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2019.

[276] Yang Yang, Hengyue Liang, and Changhyun Choi. "A deep learning approach to grasping the invisible". In: *IEEE Robotics & Automation Letters* 5.2 (2020), pp. 2232–2239.

[277] Linwei Ye, Zhi Liu, and Yang Wang. "Depth-aware object instance segmentation". In: *IEEE Int. Conf. on Image Processing (ICIP)*. 2017, pp. 325–329.

[278] Kuan-Ting Yu et al. "A Summary of Team MIT's Approach to the Amazon Picking Challenge 2015". In: *arXiv preprint arXiv:1604.03639* (2016).

[279] Kuan-Ting Yu et al. "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 30–37.

[280] Weihao Yuan et al. "Rearrangement with nonprehensile manipulation using deep reinforcement learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 270–277.

[281] Kevin Zakka et al. "Form2fit: Learning shape priors for generalizable assembly from disassembly". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 9404–9410.

[282] Andy Zeng et al. "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2018, pp. 4238–4245.

[283] Andy Zeng et al. "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 1386–1383.

[284] Andy Zeng et al. "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2018, pp. 1–8.

[285] Andy Zeng et al. "Transporter Networks: Rearranging the Visual World for Robotic Manipulation". In: *Conf. on Robot Learning (CoRL)*. 2020.

[286] Zhong-Qiu Zhao et al. "Object detection with deep learning: A review". In: *IEEE Trans. Neural Networks and Learning Systems* 30.11 (2019), pp. 3212–3232.

[287] Yin Zhou and Oncel Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4490–4499.

[288] Yuke Zhu et al. "Target-driven visual navigation in indoor scenes using deep reinforcement learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2017, pp. 3357–3364.