Efficient Policy Learning for Robust Robot Grasping

by

Jeffrey Brian Mahler

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor John Canny
Assistant Professor Paul Grigas

Summer 2018

Efficient Policy Learning for Robust Robot Grasping

Abstract

Efficient Policy Learning for Robust Robot Grasping

by

Jeffrey Brian Mahler

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

While humans can grasp and manipulate novel objects with ease, rapid and reliable robot grasping of a wide variety of objects is highly challenging due to sensor noise, partial observability, imprecise control, and hardware limitations. Analytic approaches to robot grasping use models from physics to predict grasp success but require precise knowledge of the robot and objects in the environment, making them well-suited for controlled industrial applications but difficult to scale to many objects. On the other hand, deep neural networks trained on large datasets of grasps labeled with empirical successes and failures can rapidly plan grasps across a diverse set of objects, but data collection is tedious, robot-specific, and prone to mislabeling.

To improve the efficiency of learning deep grasping policies, we propose a hybrid method to automate dataset collection by generating millions of synthetic 3D point clouds, robot grasps, and success metrics using analytic models of contact, collision geometry, and image formation. We present the Dexterity-Network (Dex-Net), a framework for generating training datasets by analyzing mechanical models of contact forces and torques under stochastic perturbations across thousands of 3D object CAD models. We describe dataset generation models for training policies to lift and transport novel objects from a tabletop or cluttered bin using a 3D depth sensor and a parallel-jaw (two-finger) or suction cup gripper. To study the effects of learning from massive amounts of training data, we generate datasets containing millions of training examples using distributed Cloud computing, simulations, and parallel GPU processing. We use these datasets to train robust grasping policies based on Grasp Quality Convolutional Neural Networks (GQ-CNNs) that take as input a depth image and a candidate grasp with up to five degrees of freedom and predict the probability of grasp success on an object in the image. To transfer from simulation to reality, we develop novel analytic grasp success metrics based on resisting disturbing forces and torques under stochastic perturbations and bounding an object's mobility under an energy field such as gravity. In addition, we study techniques in algorithmic supervision to guide dataset collection using full knowledge of the object geometry and pose in simulation. We explore extensions to learning policies that sequentially pick novel objects from dense clutter in a bin and that can rapidly decide which gripper hardware is best in a particular scenario.

To substantiate the method, we describe thousands of experimental trials on a physical robot which suggest that deep learning on synthetic Dex-Net datasets can be used to rapidly and reliably plan grasps across a diverse set of novel objects for a variety of depth sensors, robot grippers, and robot arms. Results suggest that policies trained on Dex-Net datasets can achieve up to 95% success in picking novel objects from densely cluttered bins at a rate of over 310 mean picks per hour with no additional training or tuning on the physical system. Code, datasets, videos, and supplemental material for research related to this thesis can be found at `https://berkeleyautomation.github.io/dex-net`.

To my parents, Cheryl Thalmann and Ken Mahler

Contents

List of Figures

List of Tables

Acknowledgments

This thesis has been shaped by interactions with an incredible group of people. First and foremost, I must thank my advisor Ken Goldberg for encouraging me to pursue the topic of robot grasping and for many influential discussions on research direction and presentation. Ken gave me countless advice on research and career, encouraging me to aim high and continue to improve my communication skills. Thanks to Pieter Abbeel for serving as a research mentor to me early in my graduate studies, inspiring me to study machine learning for robotics and sharing some great track workouts and basketball games. My interest in analytic models was greatly influenced by Ruzena Bajcsy, who pushed me to be a better teacher and included me in the design of a new undergraduate course on the subject, EE106B: Robot Manipulation and Interaction. Ruzena also always provided me with valuable guidance and countless stories on the history of robotics and science over the years. I am also grateful for discussions with Sergey Levine, who inspired several key ideas in this thesis such as algorithmic supervision, architectures, and data collection systems.

I also received valuable advice from a number of other faculty members at Berkeley. I owe thanks to my dissertation committee members John Canny and Paul Grigas, who provided fresh insights and ideas about network analysis and problem formulation later in my graduate studies. Anca Dragan and Shankar Sastry influenced me to think of connections with imitation learning, motion planning, and geometric modeling. Ron Fearing, Jonathan Schewchuk, and Trevor Darrell also provided helpful discussions related to this work.

For the first three years of my thesis I also worked part time for my startup company on 3D scanning, Lynx Laboratories, and its acquirer, Occipital Inc. I am extremely grateful to have worked with Chris Slaughter, Dustin Hopper, and Nick Shelton who inspired me to find problems on the line between research and real-world applications. I must also thank my undergraduate advisor and co-founder Sriram Vishawanath, who provided endless career advice to me over the years. Jeff Powers supervised my work with Occipital and influenced me to find applications of Convolutional Neural Networks trained on depth images.

Developing the software and hardware for experiments in this thesis depended on relationships with several industrial partners who provided useful insights and resources. Thanks to Jan Ziska of Photoneo, Peter Puchwein and the team at Knapp, Juan Aparicio Ojea of Siemens, Eric Tondelli and Cristina Cristialli of Loccioni, Inc., Tom Fuhlbrigge, Greg Cole, and the team at ABB, Tye Brady of Amazon, Nuttapong Chentanez of NVIDIA, Mike Haley of Autodesk, and James Kuffner and Kai Kohlhoff of Google (at the time).

The National Defense Science and Engineering (NDSEG) fellowship supported me in the last three years of my research. I am deeply appreciative of the NDSEG program, as the fellowship gave me the freedom to explore and consider high-risk, high-reward research on deep learning from synthetic training data. I am also grateful for receiving a one year fellowship from the EECS department at UC Berkeley during my graduate studies.

My research was also influenced by discussions with many members of the international robotics community that I was fortunate to meet over the years. While naming everyone would be tremendously difficult, I wish to acknowledge several individuals in particular who provided useful insights and feedback on my research: Alberto Rodriguez, Russ Tedrake, Matei Ciocarlie, Rob Platt, Frank van der Stappen, Lerrel Pinto, Andreas ten Pas, Marcus Gualtieri, Maria Bauza, Francois Hogan, Nikhil Chavan Dafle, Nima Fazeli, Lucas Manuelli, Pete Florence, Doug Morrison, Jiaji Zhou, Anastasiia Varava, Ed Johns, Marek Cygan, and Jingyi Xu.

I was extremely lucky to work with several highly talented undergraduate researchers that contributed to the research in this thesis over the years: Sidd Sen, Adithya Murali, Brian Hou, Melrose Roderick, Raul Puri, Sahanna Suri, Sherdil Niyaz, Jacky Liang, Brenton Chu, Richard Doan, Xinyu Liu, Alan (Pusong) Li, Vishal Satish, Chris Correa, Andrew Lee, and Andrew Li. Their excitement and passion for research inspired me to be a better teacher, mentor, and communicator.

Interactions with other brilliant researchers from the Berkeley robotics, machine learning, and computer vision communities influenced this thesis: John Schulman, Kevin Jameison, Phillip Moritz, Saurabh Gupta, Pulkit Agarwal, and Mathieu Aubry.

I owe a huge thanks to Shirley Salanio of the EECS department for guiding me through the Berkeley degree requirements and providing helpful advice along the way.

My colleagues at the Berkeley AUTOLAB had tremendous influence on my graduate experience. Thanks to Sachin Patil, Florian Pokorny, and Ben Kehoe for their mentorship early in my graduate studies. I owe a huge thanks to Michael Laskey, Sanjay Krishnan, Animesh Garg, and Zoe McCarthy for sharing countless debates, big dreams, and unforgettable experiences. Many of the experiments in this thesis would not have been possible without collaboration with Stephen McKinley, David Gealy, and Menglong Guo, who helped develop the robot hardware systems. I appreciate their inspiration to take risks, find adventure, and build incredible systems. Similarly, the software systems required to implement this thesis were only possible with the help of Matthew Matl and my undergraduate mentees above, who pushed my research to new heights, providing key computational insights and making it possible to share with the broader research community. I am also grateful to have worked alongside Lauren Miller, Daniel Seita, Sammy Staszak, Carolyn Chen, Aimee Goncalves, Mike Danielczuk, and Bill DeRose.

Thanks to my friends for support and fun throughout my graduate experience: Anish Mittal, Robert Nishihara, Esther Rolf, Alyssa Morrow, Lucy Stephenson, Jonathan McKinley, Mark Nyman, Jessica Lee, Trevor Crum, Lee Barnett, the EECS Jazz Group, the IHT softball team, and the BAIR ballers. Thanks in particular to Yelena and the Kulik family for encouraging me to pursue a PhD.

Last but not least, I must acknowledge the love, support, and encouragement of my family. I'm especially grateful for my brother Tim Mahler for being there through the good times and bad, my brother Ian Pelkey for reminding me that anything is possible, and my parents Cheryl Thalmann and Ken Mahler for instilling in me a belief that I can always achieve something greater.

Part I

Introduction and Background

In general, we're least aware of what our minds do best.
We're more aware of simple processes that don't work well
than of complex ones that work flawlessly.

MARVIN MINSKY

Chapter 1

Introduction

1.1 Robot Grasping

Humans can grasp and manipulate novel objects with ease, quickly moving dishes from a sink to a dishwasher, picking items off of a grocery shelf, and packing objects into bags or boxes. Robots with a similar ability to reliably grasp a wide variety of objects can benefit applications such as warehousing, logistics, manufacturing, medicine, retail, and assistance for the elderly and disabled, where the set of possible objects is constantly growing and changing. Nonetheless, grasping is highly challenging for robots due to limitations in perception and control. Sensor noise and occlusions obscure the exact shape and position of objects in the environment. Quantities such as contact forces, center of mass, and friction cannot be directly observed and must be inferred based on interaction. Imprecise actuation and calibration make it difficult to reach target arm configurations exactly. Robot grippers have either limited degrees of freedom, control imprecision, or short lifetimes. As a consequence, a robot's policy for planning a grasping strategy based on sensor data such as images cannot assume precise knowledge or control of its arm, grippers, sensors, or objects in the environment. Despite over 40 years of research, reliable robot grasp planning across objects spanning a diverse range of shapes, sizes, and material properties remains unsolved, and current industry applications of robot grasping are limited to highly controlled environments where target objects are exactly known a priori.

1.2 Grasp Analysis

Historically, the primary focus of robot grasping research has been methods for planning and evaluating grasps using model-based analyses based on physics and geometry. Analytic methods have primarily considered the ability of a grasp to achieve a mechanical quasi-static equilibrium or high mechanical advantage in the presence of disturbing forces and torques (e.g. due to gravity) [136], the ability to bound an object's mobility geometrically [144], or the ability to achieve dynamic stability evaluated through Lyapunov analysis [154] or simulation [82]. Given an analytic measure of grasp success, analytic grasp planners aim to find a configuration (e.g. joint angles, gripper pose) to maximize the measure.

These approaches assume precise knowledge of properties such as object shape, pose, and friction, and therefore require a separate perception system that can estimate the state of objects and the gripper. To plan grasps for a physical robot, a common approach is to precompute a database of known 3D objects labeled with grasps and analytic grasp quality metrics such as GraspIt! [51] or OpenGRASP [97]. Pre-computed grasps are indexed using a perception system based on instance recognition and point cloud registration: geometrically matching point clouds to known 3D object models in the database using visual and geometric similarity [14, 24, 50, 63, 186]. Then, the highest quality grasp for the estimated object instance is executed on the physical robot. To model errors in the perception system and imprecision in control, a number of robust analytic grasp planning methods have been developed to analyze grasps under stochastic perturbations by using statistical sampling over known distributions on object shape [35], object pose [81, 82, 178], or gripper pose [94].

Analytic methods can be used to efficiently generate large datasets of labeled grasps across thousands of 3D objects [50] using recent advances in distributed Cloud Computing [81] and parallel GPU processing. However, developing reliable perception systems to register sensor data to known objects and grasps in the dataset remains an open problem [52, 186]. In practice, registration-based perception systems are prone to errors [5], may not generalize well to new objects, and can be slow to match point clouds to known models during execution [50].

1.3 Deep Learning

A promising recent approach to grasp planning is to formulate it as a machine learning problem, where the goal is to train a function to approximate a desired set of outputs from a desired set of inputs based on a dataset of examples [11]. Deep neural networks are a class of particularly expressive functions that have been shown to achieve state-of-the-art performance on benchmarks for image classification [88], object detection [47], speech recognition [56], and machine translation [165] when trained on massive datasets such as ImageNet [31], outperforming decades of previous research. Since deep learning can, in principle, be applied to any dataset defining desired inputs and outputs, grasping can be also posed as a deep learning problem: train the weights of a neural network to map directly from robot sensor data to a grasp configuration for the robot arm. Thus, recent research has explored the question: can deep learning be used for grasp planning that generalizes to a diverse set of novel objects?

In contrast to vision and speech, where large-scale benchmarks such as ImageNet have been developed over decades [31], robot grasping does not have standardized training datasets. Thus, a research on deep learning for robot grasping has largely explored methods for collecting massive labeled training datasets containing millions of example images and grasps. A number of empirical results suggest that deep learning from large datasets of human grasp labels [96] or physical grasp outcomes [98, 131] can be used to plan grasps directly from images or point clouds that are successful across a wide variety of objects.

Research on human-labeled grasping has largely focused on defining interfaces and systems for humans to label graspable regions in RGB-D images [96] or point clouds [32, 62, 77]. While human labels offer the benefit of high empirical correlation with physical success [5] without requiring execution on a physical robot, labels may be expensive to acquire for large datasets and human labeling errors may reduce performance.

Another approach is to use self-supervision, in which a physical robot automatically collects datasets by attempting grasps on a set of training objects and labeling grasps with the success or failure of each attempt [98, 131]. This approach is particularly appealing because the data is unbiased: labels reflect performance on the same physical robot system that will be used for testing. Research on self-supervision has largely focused on methods for dataset collection that iteratively concentrate sampling on more promising grasps. One class of techniques uses dataset aggregation, in which batches of data are collected on continuously operating robot arms using the network trained on all of the data collected so far [98, 131]. Another class of techniques has focused on active grasp acquisition with Multi-Armed Bandits [89, 119, 125] or Reinforcement Learning [74]. Empirical results suggest that deep neural networks can generalize well to novel objects after approximately one robot year of continuous data collection.

While research on deep learning from empirical grasping datasets is promising, it has highlighted one property of robot grasping that makes it difficult to apply standard deep learning techniques: collecting massive empirical training datasets is extremely expensive. Dataset collection requires tedious hand-labeling [77, 96] or months of continuous execution on a physical system [98, 131] and performance appears to plateau as the dataset size grows [98]. The datasets may contain mislabeled data due to human errors, hardware failures, drifting calibration, imprecision in sensors used to automatically detect grasp successes and failures. Futhermore, training datasets are specific to the robot arm, gripper, cameras, and environment conditions such as lighting and may need to be re-collected for each new environment.

1.4 Grasp Planning in Practice

Due to the limitations of both analytic methods and empirical deep learning methods, practical applications of robot grasping either operate under highly controlled conditions where exact object shape and pose [55] are known or use hand-coded heuristics to plan grasps for point cloud and image data [40]. One application of grasp planning is in industrial picking systems, where the goal is to iteratively grasp and transport a known object from a bin or conveyor to a receptacle. Current systems handle one known part at a time, using exact CAD models of the known object to accurately estimate the object shape and pose to index a pre-computed grasp to execute [55, 66]. In less structured applications such as home decluttering, grasp planning for unknown objects is often based on geometric heuristics such as aligning the gripper with the object principal axis [5, 24].

Recently, the Amazon Robotics Challenge (ARC) highlighted the gap between grasp planning theory and practice for robot picking and stowing for warehouse automation. In all three years, the winning team's approach was based on planning grasps for a custom high-flow vacuum suction gripper using point cloud heuristics [40, 60, 120] such as targeting planar surfaces along the inward pointing surface normal. Several teams attempted to build 3D object registration systems which required complicated systems to collect datasets of known objects in known poses on cluttered shelves [141, 186]. In 2016 and 2017, a number of teams began to explore the use of deep neural networks for rapidly predicting grasp success from images [187] by tediously hand-labeling training datasets.

While considerable progress has been made over the past several years, applications such as the ARC have highlighted a clear gap between theory and practice. Empirical dataset collection for training deep grasping policies is impractical due to the high cost of data, and analytic methods do not scale to many diverse objects.

1.5 Thesis Goals and Contributions

The goal of this thesis is to explore algorithmic approaches to robot grasp planning that combine the scalability and interpretability of analytic methods with the generalization ability of empirical deep learning methods. Building upon recent advances in deep learning, parallel computing, high-resolution 3D sensing, and theory from geometric modeling, grasp mechanics, noise models, and stochastic sampling, we propose a novel hybrid approach to grasp planning that uses analytic models to generate massive synthetic training datasets for learning deep grasping policies that plan grasps directly from images.

Key to our hybrid approach is constructing an end-to-end probabilistic generative model for synthetic training data that is computationally efficient and transfers well from simulation to reality. In this thesis, we develop generative models of training data for grasping a wide variety of objects with parallel-jaw and suction cup grippers. To increase learning efficiency, we introduce algorithmic supervision techniques that use the fully known state of objects in simulation to guide data collection toward more promising grasps. We implement the hybrid approach in the Dexterity Network (Dex-Net), a research project including code, datasets, and algorithms for generating datasets of synthetic point clouds, robot grasps, and grasp quality metrics by analyzing models of geometry, physics, and optics across thousands of 3D object models. To evaluate the method, we perform tens of thousands of experimental trials on a physical robot which suggest that deep learning from massive synthetic training datasets generated with Dex-Net over several days of computation can be used to rapidly and reliably plan grasps across a wide variety of novel objects, outperforming existing heuristics and analytic methods. Code, datasets, videos, and supplemental material for research related to this thesis can be found at `https://berkeleyautomation.github.io/dex-net`.

The primary contributions of this thesis are:

- We formalize the robust grasp planning problem in a framework that relates analytic, empirical, and hybrid approaches.
- We develop a common structure for dataset generation distributions in terms of a state distribution, reward distribution, observation function, and algorithmic supervisor.
- We present a hybrid approach to grasp planning that aims to automate the generation of training datasets for learning deep grasping policies that plan grasps directly from image by using computer implementations of end-to-end analytic models for grasp success and synthetic depth images.
- We propose models for generating datasets to train policies to lift and transport objects from a tabletop or cluttered bin using a parallel-jaw (two-finger) or suction cup gripper.
- We derive new robust grasp quality metrics based on resisting target forces and torques under perturbations and bounding the mobility of an object under an energy field such as gravity.
- We develop algorithms for rapid grasp analysis in the Cloud.
- We introduce Grasp Quality Convolutional Neural Network (GQ-CNN) architectures for learning to predict the quality of candidate grasps from depth images.
- We present the Dexterity Network (Dex-Net), a system for generating datasets by analyzing mechanical models of contact forces and torques under stochastic perturbations across thousands of 3D object CAD models, and present five version of massive training datasets generated with Dex-Net.
- We explore methods for learning robust policies that transfer from simulation to reality and that can decide between a set of hardware gripper alternatives to use for grasping a particular object.
- We present the results of tens of thousands of experimental trials on physical robots benchmarking analytic, empirical, heuristic, and hybrid grasp planning methods on a diverse set of objects suggesting that this approach can achieve a grasp success rate of over 95% for lifting and transporting common novel objects from dense clutter in a bin.

1.6 Thesis Outline

This thesis is organized in four parts. Part I provides background on the robust grasp planning problem and introduces the hybrid approach based on synthetic dataset generation.

- Chapter 2 formalizes the robust grasp planning problem, categorizing methods based on their solution approach.

Part II develops an end-to-end model for for generating datasets to train a parallel-jaw grasping policy to lift, transport, and hold an object under perturbations (e.g. shaking) based on geometric 3D point cloud data of the object of interest.

- Chapter 3 introduces Dex-Net 1.0, develops a generative model for grasps and rewards computed across thousands of 3D objects, explores its application to rapid grasp computation for novel 3D objects in the Cloud by learning for prior data, and evaluates quality metrics proposed by the model on a physical robot with a millimeter-accurate registration system. This contains research previously published as [104].
- Chapter 4 extends this generative model to include synthetic point clouds generated by rendering depth images of 3D object models in randomized poses on a tabletop, and demonstrates that the model can be used to train a Grasp Quality Convolutional Neural Network (GQ-CNN) that can rapidly plan grasps for a diverse set of objects on a physical robot. This contains research previously published as [105].
- Chapter 5 further extends the model to a time sequence of grasps for iteratively removing a novel object from a clutter bin to a receptacle, and introduces an imitation learning method based on sampling grasps from and algorithmic grasping supervisor that indexes pre-computed grasps using the known state of objects in simulation. This contains research previously published as [103].

Part III explores extensions of the hybrid method to new grippers and tasks by developing novel physics-based reward functions and algorithmic supervisors that guide data collection by planning optimal grasps given full knowledge of the state of the environment:

- Chapter 6 considers the problem of bounding the mobility of an object under energy fields such as gravity or friction during constant-velocity pushing, and develops an algorithmic supervisor that can compute energy-bounded caging configurations using techniques from computational topology. This contains research previously published as [110].
- Chapter 7 extends the dataset generation distribution for parallel-jaw grippers to vacuum suction grippers, developing the suction ring contact model, a new model of compliant contact for suction cup grippers and wrench resistance, a new reward function for assessing the ability of a grasp to lift and transport objects. This contains research previously published as [106].
- Chapter 8 explores learning composite policies that decide between a parallel-jaw and suction gripper for bin picking based on sub-policies for each gripper, and develops a gripper-agnostic reward model and efficient algorithmic supervisor for large scale dataset generation. Research in this section is previously unpublished.

Part IV discusses limitations of the method uncovered by the work in this thesis and highlights opportunities for future research on efficient learning of robust robot grasping policies.

Chapter 2

Robust Grasp Planning

Robot grasp planning considers the problem of determining a set of control parameters to grasp an object based on imprecise sensor data such as images. Approaches to this problem span a wide spectrum from analysis of physics-based models, to machine learning on empirically-collected training datasets, to design of specialized robot gripper hardware. In this chapter, we formalize a unified objective for robust grasp planning and highlight the problem of generalization across objects, the core topic of this thesis. We classify past research into two approaches to the robust grasp planning problem: analytic and empirical methods. Finally, we introduce a hybrid dataset generation approach for synthesizing massive training datasets of images, grasps, and labels for the success and failure of each grasp.

We wish to emphasize that the goal of formalism in this chapter is not to provide theoretical guarantees. In fact, theoretical guarantees are rarely considered in this thesis due to the difficulty of relating theory to physical robot systems. The aim of this chapter is to provide a common language for describing algorithmic approaches to grasp planning. Through the abstractions discussed in Section 2.1, we identify key relationships between variables in order to understand the tradeoffs of using different methods. This formalism has been crucial to the design of experiments in this thesis and we hope that it can inform comparisons between grasp planning methods in future benchmarks.

2.1 Problem Statement

We formalize robust grasp planning as a Partially Observable Markov Decision Process (POMDP) [72] in which a robot plans grasps for objects in the environment in order to maximize expected reward (e.g. probability of a successful grasp) given imperfect observations of the state of the environment.

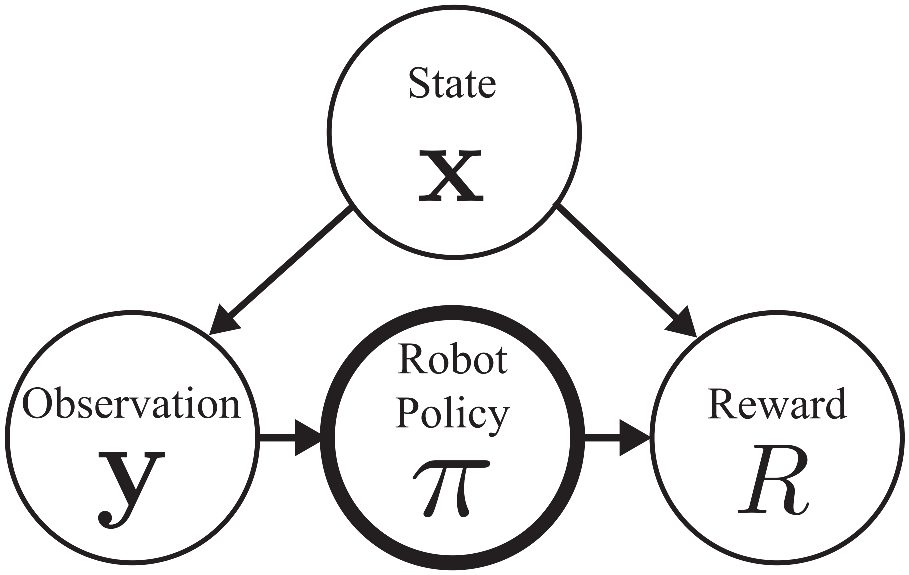


Figure 2.1: Graphical model illustrating the robust grasp planning problem. A robot policy π plans a grasp action $\mathbf{u}$ based on a noisy observation of a set of objects in state $\mathbf{x}$ and receives a reward R based on the result of executing the grasp. The goal is to learn a policy that achieves high expected reward on a distribution of states, observations, and rewards (e.g. a particular set of objects).

2.1.1 Definitions

The robust grasping problem, illustrated in Fig. 2.1 is defined by:

- **States.** Let $\mathbf{x} \in \mathcal{X}$ be the *state* of a set objects and sensors in the environment. The state of each object $\mathcal{O}$ includes object geometry, pose and other variables (e.g. frictional properties, and center of mass). The state of each sensor $\mathcal{C}$ includes the pose and parameters of the sensor (e.g. focal length of a camera).
- **Observations.** Let $\mathbf{y} \in \mathcal{Y}$ be a sensor *observation* (e.g. an image) that the robot acquires with a sensor in a given state. An observation typically depends on the state by the observation function $\mathbf{y} = f(\mathbf{x})$. In general the function f is not invertible due to occlusions.
- **Grasp Actions.** Let $\mathbf{u} \in \mathcal{U}$ define a robot *grasp* action (e.g. gripper pose) from a set of possible grasps. In general, a grasp consists of a minimal set of control parameters to achieve a desired motion. Throughout this thesis, a grasp $\mathbf{u}$ is defined as a robot gripper $\mathcal{G}$ consisting of the geometry and physical parameters (e.g. friction) of the gripper, and a pose for the robot gripper relative to a world reference frame $\mathbf{T}_g$ consisting of a 3D rotation and translation $\mathbf{T}_g = (\mathbf{R}_g, \mathbf{t}_g) \in SE(3)$, unless otherwise specified.

- **Reward.** Let $R(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ be the binary *reward* for a given state and grasp, based on whether grasp $\mathbf{u}$ successfully establishes contact with one or more objects in $\mathbf{x}$ to accomplish a given goal, such as lifting and transporting the object to another location.
- **Robot Policy.** Let $\pi : \mathcal{Y} \to \mathcal{U}$ be a robot grasping *policy* that takes as input an observation and returns a grasp for the robot to execute: $\mathbf{u}_\pi = \pi(\mathbf{y})$.
- **State Distribution.** Let $p(\mathbf{x})$ be a distribution over possible states. For example, this could be a uniform distribution over a set of possible 3D objects and their poses resting on a tabletop or a Gaussian Process Implicit Surface distributio over possible object states (Appendix A)
- **Observation Distribution.** Let $p(\mathbf{y} \mid \mathbf{x})$ be a distribution over observations given a state modeling sensor noise.
- **Reward Distribution.** Let $p(R \mid \mathbf{x}, \mathbf{u})$ be a distribution over rewards for a given state and action due to imprecision in control.

Definition 1 (Environment)**.** *A grasping environment is a joint distribution on states, observations, and rewards given a grasp action:*

$$p(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u}) = p(\mathbf{x}) p(R \mid \mathbf{x}, \mathbf{u}) p(\mathbf{y} \mid \mathbf{x})$$

Example: Grasping a Single Object from a Depth Image

For concreteness, consider the problem of grasping a single object in a randomized pose on a tabletop based on images from a depth camera as illustrated in Fig. 2.2. The state $\mathbf{x}$ consists of the object state $\mathcal{O}$ (geometry and pose) and camera state $\mathcal{C}$ (pose). Sampling the initial state distribution $p(\mathbf{x})$ places the camera directly above the table with a randomized view angle and places the object in a random stable resting pose on the tabletop. Observations $\mathbf{y}$ consist of depth images from the camera viewpoint. The policy plans a grasp action $\mathbf{u}$ consisting of the pose of a parallel-jaw gripper and is rewarded for successfully lifting the object from the tabletop. We consider this environment in detail in Chapter 4.

2.1.2 Objective

The objective is to learn a grasping policy π to maximize the expected reward, or *success rate*, for a given environment:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}\left[R(\mathbf{x}, \pi(\mathbf{y}))\right] \tag{2.1.1}$$

where the expectation is taken with respect to the environment $p(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ and Π is the set of candidate policies.

Many approaches to robust grasping focus on computing a grasp quality function to rank grasp candidates by their expected reward.

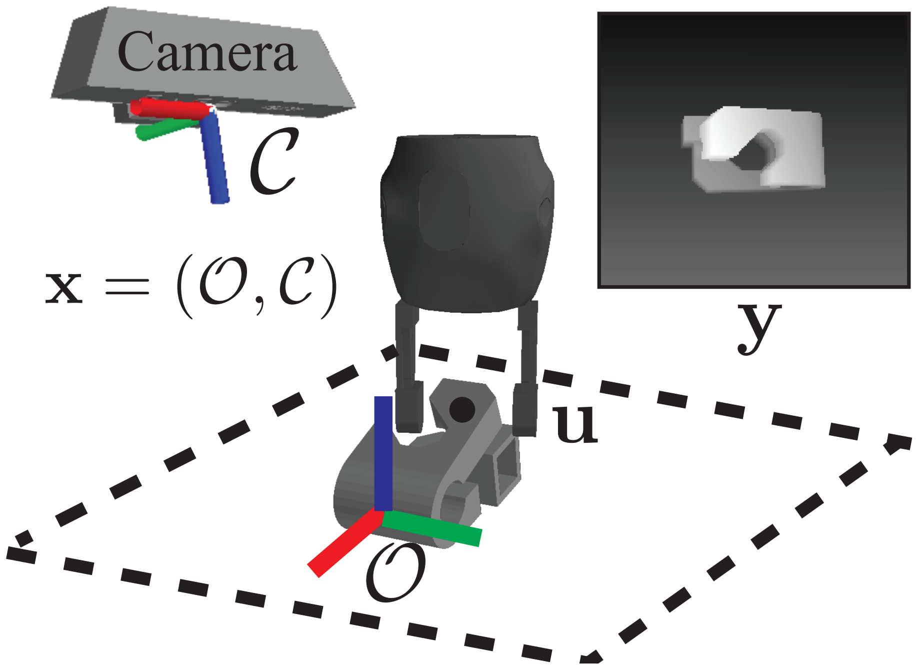


Figure 2.2: Grasping environment for lifting a single object in a randomized pose on a tabletop with a single depth camera.

Definition 2 (Grasp Quality)**.** *Let Q be the quality of a grasp given an observation:*

$$Q(\mathbf{y}, \mathbf{u}) = \mathbb{E}_{R,\mathbf{x}}\left[R(\mathbf{x}, \mathbf{u}) \mid \mathbf{y}\right]$$

Note that unlike R which is a binary function, Q is a continuous function with values in the range $[0, 1]$.

An optimal robust grasping policy computes the grasp that maximizes grasp quality given an observation:

$$\pi^*(\mathbf{y}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmax}}\ Q(\mathbf{y}, \mathbf{u})$$

This follows from the equivalence of the problem statement to a single timestep Partially Observable Markov Decision Process (POMDP) [17, 156].

The success rate for a grasping policy over a distribution of states, observations, and rewards can alternatively be defined as:

$$\Phi(\pi) = \mathbb{E}_{\mathbf{y}}[Q(\mathbf{y}, \pi(\mathbf{y}))]. \tag{2.1.2}$$

This follows from the law of iterated expectations. Using this definition, the grasp planning problem can be written concisely as:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\ \Phi(\pi) \tag{2.1.3}$$

2.1.3 Extensions

In practice, a robot grasping policy is part of a larger system to perform a task that involves a sequence of actions as quickly as possible. For example, robots fulfilling customer orders in warehouses may grasp products from bins or shelves as a first step to packing them in a shipping box. We briefly discuss two extensions to put the robust grasp planning into a broader context: sequential grasp planning and industrial picking.

Sequential Grasping

The goal of sequential grasping is to maximize the success rate of a policy over a sequence of grasp attempts. This is an instance of a Partially Observable Markov Decision Process (POMDP) [72]. In addition to the definitions of Section 2.1.1, sequential grasp planning includes:

- **Horizon.** Let T be an integer representing the maximum allowed number of attempts.
- **Transition Function.** Let $p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$ be a distribution over next states given the current state and action modeling the physics of the problem. For example, a transition function could model the probability that a grasp successfully moves an object to a new position or removes the object from a heap.

Furthermore, in the sequential setting a reward function $R(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ is also a function of the next state. The graphical model for sequential grasp planning is illustrated in Fig. 2.3

The objective of sequential robust grasp planning is to maximize the success rate over a series of grasp attempts:

$$\pi^* = \underset{\pi \in \Pi}{\text{argmax}} \, \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} R(\mathbf{x}_t, \pi(\mathbf{y}_t), \mathbf{x}_{t+1})\right] \tag{2.1.4}$$

where the expectation is taken with respect to the joint distribution

$$p(R_0, \mathbf{x}_0, \mathbf{y}_0, ..., \mathbf{x}_T, \mathbf{y}_T \mid \pi) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} p(\mathbf{y}_t \mid \mathbf{x}_t) p(R_t \mid \mathbf{x}_t, \pi(\mathbf{y}_t), \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \pi(\mathbf{y}_t))$$

This joint distribution specified the grasp environment for a sequential grasp planning problem.

An optimal robust grasping policy for the sequential grasping problem depends on the history of all past actions and observations up to time t: $\mathbf{h}_t = (\mathbf{y}_0, \mathbf{u}_0 ..., \mathbf{y}_{t-1}, \mathbf{u}_{t-1})$ where $\mathbf{h}_0 = \varnothing$ [156]. This is because the sequence of observations is non-Markovian – the probability of the current state of objects in the environment is informed by all past actions and observations.

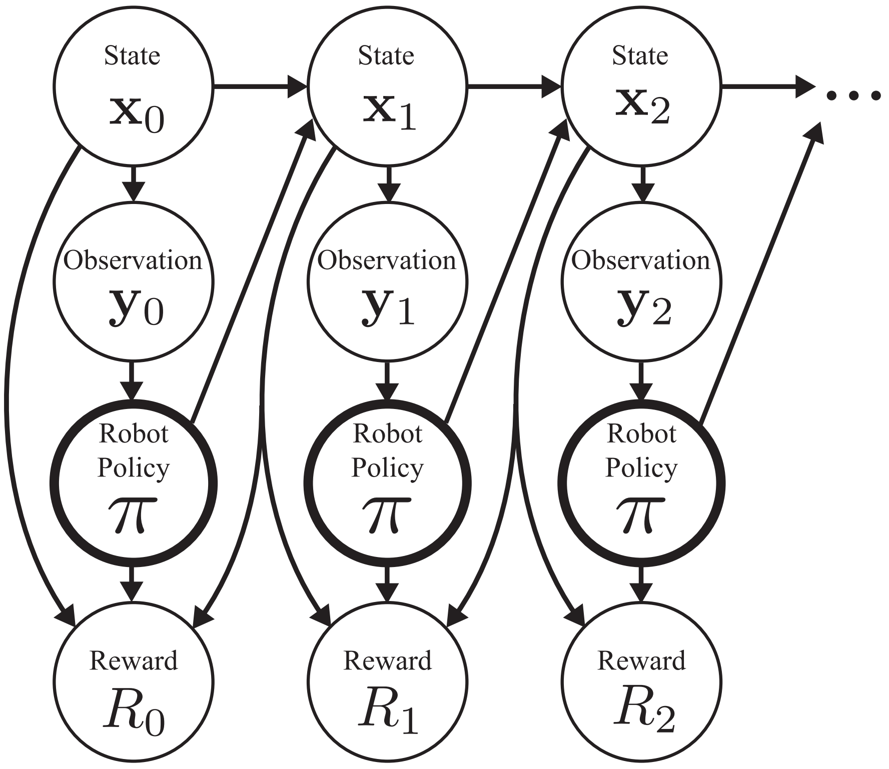


Figure 2.3: Graphical model illustrating the sequential robust grasp planning problem. An initial state $\mathbf{x}_0$ is sampled from the initial state distribution. On each timestep robot policy π plans a grasp action $\mathbf{u}_t$ based on a noisy observation of a set of objects in state $\mathbf{x}_t$ and receives a reward R_t based on the result of executing the grasp. The goal is to learn a policy that achieves high expected reward for up to T consecutive grasp attempts.

We can define an optimal grasping policy in terms of grasp quality by including the policy, history, and future rewards in the definition:

$$Q_\pi(\mathbf{y}_t, \mathbf{u}_t, \mathbf{h}_t) = \mathbb{E}\left[\sum_{s=t}^{T-1} R(\mathbf{x}_s, \mathbf{u}_s, \mathbf{x}_{s+1}) \mid \mathbf{y}_t, \mathbf{h}_t, \pi\right]$$

An optimal policy is given by:

$$\pi^*(\mathbf{y}_t) = \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmax}}\ Q^*(\mathbf{y}_t, \mathbf{u}_t, \mathbf{h}_t)$$

where Q^* is the quality function that satisfies the Bellman equation [166].

Computing Q^* is challenging due to the sample complexity of Q-function estimation [73] and the large number of possible states. We discuss practical approximations in Chapters 5 and 8.

Industrial Picking

The goal of industrial robot picking is to find a robot policy that maximizes Mean Picks Per Hour (MPPH), or the number of objects that are successfully grasped per hour. This has applications in bin picking, where the goal is to grasp and transport a set of objects from a bin to a container as quickly as possible.

Definition 3 (Mean Picks Per Hour)**.** *The mean picks per hour (MPPH) of a policy π is defined by:*

$$\rho(\pi) = \mathbb{E}\left[\frac{\sum_{t=0}^{T} R(\mathbf{x}_t, \pi(\mathbf{y}_t))}{\sum_{t=0}^{T} \Delta(\pi(\mathbf{y}_t))}\right] \tag{2.1.5}$$

where T is the number of grasp attempts and $\Delta(\mathbf{u})$ is the duration of grasp attempt $\mathbf{u}$ as a fraction of an hour. This expectation is with respect to the joint distributions of states $p(\mathbf{x})$, observations $p(\mathbf{y} \mid \mathbf{x})$, and rewards $p(R \mid \mathbf{x}, \mathbf{u})$.

Note that ρ is a problem-dependent quantity and therefore values of ρ cannot be directly compared across different sets of objects or different sensors.

Note that the time duration per grasp includes the time required for sensing, computation, and physical motion of the robot. We can divide Δ into elements to reflect the mean time per grasp required for sensing t_s, computation t_c, and physical motion of the robot t_r:

$$\Delta = t_s + t_c + t_r,$$

where times are expressed as fractions of an hour.

If we consider the time per grasp to be a constant Δ for all t, then we can define the mean *grasp rate* (grasps attempted per hour) as:

$$\nu = 1/\Delta. \tag{2.1.6}$$

Under these assumptions, MPPH is equivalent to:

$$\rho(\pi) = \nu\Phi. \tag{2.1.7}$$

where Φ is defined in Equation 2.1.2.

This suggests that in industrial picking, there is a fundamental tradeoff between achieving a high success rate and executing grasps faster. In practice it is common to first select robot and sensor hardware that meets a predefined computational budget and then find a robust grasping policy (Equation 3.1.1) under these constraints to achieve a higher success rate.

2.2 Solution Approaches

In this section, we formalize each approach to the robust grasping problem. We characterize methods in terms of two phases:

1. **Training Phase.** Learn an estimated quality function $\hat{Q}$ over a dataset of states, observations, grasps, and rewards defining the success or failure of each grasp.
2. **Test Phase.** Compute a grasp given an observation by evaluating a robust grasping policy based on the estimated quality function:

$$\hat{\mathbf{u}} = \hat{\pi}(\mathbf{y}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmax}} \; \hat{Q}(\mathbf{y}, \mathbf{u}). \tag{2.2.1}$$

We first discuss hardware-centric and heuristic methods before describing algorithmic approaches in detail.

2.2.1 Gripper Hardware Design

A large body of research in grasping has focused on the design of robot grippers to facilitate grasping. Research has considered a number of designs, including underactuated grippers [23, 126], compliant fingertips [27, 53], high friction fingertips [58], tactile sensing [71, 159], and soft hands [15, 30, 146]. In the robust grasp planning framework, hardware-based methods can be interpreted as optimizing the reward distribution $p(R \mid \mathbf{x}, \mathbf{u})$ by expanding the set of grasp actions that lead to high reward.

While advances in gripper design are promising, hardware-design is not a complete approach to grasp planning – hardware-centric methods still require a policy that plans robust grasps for a given gripper. Since soft and underactuated grippers are difficult to model, heuristic grasp planning methods such as aligning with the object principal axis, are common in practice.

2.2.2 Heuristic Methods

A number of grasp planning methods in practice are based on geometric heuristics. Most methods are based on ranking a set of candidate grasps by hand-designed grasp quality functions.

Training Phase

In the training phase, a person typically tunes a hand-coded grasp quality metric $\hat{Q}$ until a sufficient level of performance is reached. Heuristic quality functions typically differ based on the gripper. For vacuum suction cup grippers, common heuristic quality metrics include alignment with the inward-pointing surface normal [120], targeting planar surfaces [184], and targeting surfaces near the object centroid [60]. For parallel-jaw and multifinger grippers, the most common heuristic is to align the gripper with object principal axes after segmenting the object from the background [5].

Test Phase

To deploy a heuristic grasping policy, a set of grasp candidates are sampled based on the sensor data and scored using the heuristic grasp quality function. The robot then executes the highest-quality grasp that is kinematically feasible and collision-free.

Discussion

Heuristic grasp planning methods may work well in practice and have been used by the winning teams of the Amazon Robotics Challenge in all three years [40, 60, 120] However, these methods have several significant shortcomings. First, they are often tuned for a particular object set, camera, or environment and require expert knowledge to tune for a new scenario. Second, heuristics cannot be related to the robust grasp planning objective of Equation 3.1.1, making it difficult to analyze performance and understand the limitations of when they can or cannot be used. Third, many heuristic require an approximate segmentation of the target object in images, which is very difficult in unstructured environments such as cluttered bins.

2.2.3 Analytic Methods

The majority of research in robot grasping over the past 40 years [136] is based on analytic methods. Analytic methods assume a separate perception system that provides a state estimate $\hat{\mathbf{x}}$ for a set of objects. The estimate used to index pre-computed quality metrics for a set of pre-computed grasps on CAD models of similar objects which can then be executed directly on the robot.

Analytic approaches fall into two categories based on the training method: *classical methods* that evaluate $\hat{Q}$ deterministically assuming perfect state knowledge and control precision and *stochastic methods* that evaluate $\hat{Q}$ via statistical sampling using a distribution over possible states and grasps due to imprecision in perception and actuation. In both cases the estimated quality function $\hat{Q}$ is associated with a pre-computed grasps defined relative to the object, so that in the test phase grasps can be executed by indexing the grasp and pose from a database [116].

Training Phase: Classical Analytic Methods

Classical analytic grasp planning methods assume perfect, fully-observed state information (e.g. $\hat{\mathbf{x}} = \mathbf{x}$) from a perception system and that the reward for a given grasp can be determined using an analytic formula based on physics or geometry. To plan a grasp, classical methods find a grasp (not necessarily unique) to maximize the analytic grasp reward metric using exact knowledge of the object and contact locations:

$$\mathbf{u}^* = \pi(\mathbf{x}) \in \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmax}}\ R(\mathbf{x}, \mathbf{u}). \tag{2.2.2}$$

An extensive body of research explores methods to evaluate $R(\mathbf{x}, \mathbf{u})$. Reward functions are typically based on three types of criteria: mechanical analysis of quasi-static equilibria [164], geometric constraints on object mobility [144], and analysis of dynamic stability [154].

Methods based on mechanical analysis typically consider the feasibility of a quasi-static equilibrium between a contact forces exerted by the robot fingers and a set of disturbing wrenches (forces and torques) on the object e.g. due to gravity. A grasp is in *force closure* if a quasi-static equilibrium exists for any possible disturbing wrench, assuming that the gripper can exert infinite contact forces. When a quasi-static equilibrium exists, several metrics have been proposed to provide a relative ranking between grasps. One class of metrics measures the ratio of the magnitude of actuated contact forces to the magnitude of a disturbing wrench that the grasp is resisting. The most widely known metric is the epsilon grasp quality [41], which computes the worst-case ratio over all possible disturbing wrenches. Another class of metrics measures the ability of the grasp to exert task-specific wrenches, often modeled as an ellipsoid in wrench space [100].

Reward functions based on constraining object mobility analyze whether or not the object mobility is bounded by rigid placement of the robot gripper. A grasp that completely immobilizes the object is in *form closure* [136]. When the object can move but the grasp constrains the object motion to a bounded region of the state space, then the grasp *cages* the object [142]. These conditions are usually evaluated by analyzing the configuration space between the object and gripper.

A third class of analytic reward functions consider whether or not a grasp is dynamically stable under the presence of disturbing wrenches. Stability-based methods typically model each finger contact as a spring-mass damper system attached to the object and use Lyapunov analysis to determie stability [154]. Several methods have also been proposed to construct stable compliance matrices for fingertips of a force closure grasp [123]. These analysis methods are relatively uncommon, perhaps due to the computational complexity of stability analysis when contact modes can change (e.g. from sticking to slipping).

Training Phase: Robust Analytic Methods

In practice, perfect state information is not available, nor is it possible to perfectly control a robot to achieve a desired grasp. Robust analytic methods evaluate classical analytic reward functions under stochastic perturbations in the object state and grasp action to model imprecision in sensing and control.

Robust methods define a state with a random variable $\tilde{\mathbf{x}} = \hat{\mathbf{x}} + \epsilon$, where $\hat{\mathbf{x}} = g(\mathbf{y})$ is a nominal state estimate from a perception system g and ϵ models errors in state estimation due to sensor imprecision. For example, ϵ might model errors in a 6-DOF pose estimate for a known rigid object from a geometric alignment algorithm such as Iterated Closest Point (ICP). Often this distribution models either Gaussian error in object pose [82, 178] or shape [35]. Similarly, grasp commands may be defined by a random variable $\tilde{\mathbf{u}} = \mathbf{u} + \delta$ where $\mathbf{u}$ is the command sent to the robot and δ models imprecision in control [94]. Typically the uncertainty ϵ and δ are modeled with zero-mean Gaussian distributions, although in practice parametric models may not be appropriate due to asymmetries in sensing and control (eg, cable hysteresis).

Robust analytic grasp planning methods maximize the expected reward under the modeled distributions on state and action:

$$\begin{aligned}\pi(\mathbf{y}) &= \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmax}}\ Q(\mathbf{y},\mathbf{u}) \\ &= \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmax}}\ \mathbb{E}\left[R(\tilde{\mathbf{x}},\tilde{\mathbf{u}}) \mid \hat{\mathbf{x}},\mathbf{u}\right] \\ &= \underset{\mathbf{u}\in U}{\operatorname{argmax}}\ \int R(\tilde{\mathbf{x}},\tilde{\mathbf{u}})\ p(\tilde{\mathbf{x}} \mid \hat{\mathbf{x}})\ p(\tilde{\mathbf{u}} \mid \mathbf{u})\ d\tilde{\mathbf{x}}\ d\tilde{\mathbf{u}}\end{aligned} \tag{2.2.3}$$

To estimate the robustness Q for an object and grasp, a common approach is to use Monte-Carlo integration with N samples of object state and grasp perturbation samples drawn from $p(\tilde{\mathbf{x}} \mid \hat{\mathbf{x}})$ and $p(\tilde{\mathbf{u}} \mid \mathbf{u})$:

$$\hat{Q}(\hat{\mathbf{x}},\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} R(\mathbf{x}_i,\mathbf{u}_i) \tag{2.2.4}$$

This estimate can then be associated with an object in a database. The Monte-Carlo approach is popular because it is straightforward to implement and highly parallelizable. However, this approach may require a large number of samples to evaluate quality for each grasp on each object.

An alternative approach is to use adaptive sampling methods to allocate samples on more promising grasps. Several approaches have been proposed based on Bayesian Optimization [89] and Multi-armed Bandits [94]. These approaches may significantly reduce the number of samples required to identify the highest quality grasp on an object by an order of magnitude. However, one drawback is that lower-quality grasps will have poor quality estimates because they have been allocated fewer samples. This may be problematic when grasping an object from clutter, where only a small number of grasps may be available.

Test Phase

To deploy an analytic grasping policy in practice, the state estimate $\hat{\mathbf{x}}$ from perception is used to index a database of precomputed objects, grasps, and quality estimates. The robot executes the highest-quality grasp that is kinematically feasible and collision-free.

Discussion

Analytic methods can be used to rapidly evaluate the quality of a large number of known grasps and objects and may be useful on a physical robot for grasping a small number of known objects in a controlled environment. However, in practice it is often not realistic to assume a perception system that can accurately estimate state up to a known error distribution for a wide variety of possible objects in varied environments. Object registration systems are prone to errors [5], may not generalize well to new objects, and can be slow to match point clouds to known models during execution [50]. Furthermore, object instance recognition remains a highly challenging problem in computer vision [52] and in practice it is difficult to reliably categorize and estimate the pose for a large number of novel objects. In Chapter 3 we discuss experimental results that suggest that these approaches do not scale well to a large number of objects.

2.2.4 Empirical (Data-Driven) Methods

Empirical methods are a more recent approach to grasping developed in the computer vision and machine learning communities. The fundamental idea behind empirical methods is that the quality function can be approximated using machine learning over a large dataset of images, grasps, and rewards. Since the quality function does not depend on a state estimates for known objects, it is straightforward to evaluate the learned quality function on novel objects. Furthermore, if the training dataset is collected from humans or physical experiments, then the dataset would, in theory, reflect ground truth grasp success on a physical system. Optimizing performance on these large and unbiased datasets would therefore directly optimize for performance on a physical robot.

Training Phase: Dataset Collection

Empirical methods attempt to learn a robustness function directly from observations using a training dataset:

$$\mathcal{D} = \{(\mathbf{y}_i, \mathbf{u}_i, R_i)\}_{i=1}^N \tag{2.2.5}$$

with N triples including an observation $\mathbf{y}_i$, grasp $\mathbf{u}_i$, and binary success label R_i for each obtained from humans [96] or physically executing each grasp on a robot and recording the result [98]. The first step of empirical methods is to collect such a dataset.

Human-labeled datasets are popular due to empirical correlation with physical success [5]. Research on collecting datasets from humans has largely focused on associating human labels with graspable regions in color images [150], RGB-D images [96] or point clouds [32, 62, 77]. Notably, Lenz et al. [96] created a dataset of over 1k RGB-D images with human labels of successful and unsuccessful grasping regions, which has been used to train fast CNN-based detection models [140]. However, it may be expensive to collect hand-labeled examples.

Self-supervised techniques use the outcomes of physical trials to label training datasets. Due to the high cost of collecting training datasets on a physical robot, early research in this area studied active methods for acquiring grasping experiences such as Multi-Armed Bandits using Correlated Beta Processes [119] or Prior Confidence Bounds [124]. Recent research has scaled up dataset collection by continuously running one or more robot arms and iteratively aggregating training datasets [98, 131].

Training Phase: Policy Learning

The second step is to learn a quality function by optimizing performance on the training dataset. The quality function Q_θ is defined by a set of parameters $\theta \in \Theta$ such as the weights of a neural network. Early empirical methods parameterized Q_θ using weights for a set of predefined features such as the outputs of a filter bank [150]. Recently it has become popular to use hyperparametric function approximators such as Deep Neural Networks to learn an "end-to-end" policy directly from observations (depth maps or pixels) to grasp actions (or torques) [98].

The most common approach to training is supervised learning, in which the quality function parameters are optimized to minimize a loss function penalizing deviations from the reward label R_i:

$$\theta^* = \min_{\theta \in \Theta} \sum_{i=1}^{N} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)) \tag{2.2.6}$$

This objective is motivated by that fact that for certain choices of loss function (e.g. binary cross entropy), $Q_{\theta^*} = Q$ for all possible grasps and images as long as there exists some $\theta \in \Theta$ such that $Q_\theta = Q$ [117].

Another approach to training is reinforcement learning [166], which attempts to directly optimize:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\ \mathbb{E}\left[R(\mathbf{x}, \pi(\mathbf{y}))\right]$$

to continuously learn from grasp attempts on a physical system Solving this objective directly requires a tradeoff between exploration and exploitation since actions proposed by the policy affect the distribution of rewards that the policy receives.

Empirical reinforcement learning techniques for grasping were first considered by Salgonicoff et al. [148] who attempted to use algorithms based on Multi-Armed Bandits [3] to actively acquire grasp experiences on a physical system. To generalize to novel objects, several extensions of the Multi-Armed Bandit approach have been proposed such as Correlated Beta Processes [119] or Prior Confidence Bounds [124]. Reinforcement learning to reason about grasp sequences has only recently been attempted at significant scale on physical robots. Initial results show promise for picking a wide variety of objects [74], but policies may not generalize to other robots without months of additional data collection [98]. While reinforcement learning techniques mitigate bias in the dataset sampling distribution, they may require a very large number of samples to learn as initial exploratory grasp attempts may receive little reward.

Test Phase

Once function Q_θ is learned, empirical methods plan the grasp that maximizes the learned quality function:

$$\pi(\mathbf{y}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmax}}\ Q_\theta(\mathbf{y}, \mathbf{u}). \tag{2.2.7}$$

Note that Q_θ can be computed very rapidly for many machine learning models as it typically requires only basic mathematical operations such as matrix multiplication.

One approach is to sample a fixed set of grasp candidates $\mathcal{U}$ and rank them according to Q_{θ^*}. Another method is to iteratively re-sample and re-rank the list of grasps using a derivative-free optimization technique such the Cross Entropy Method (CEM) [145].

Discussion

Given sufficient data, empirical methods can work well in practice and can yield $\Phi \approx 90\%$ with relatively fast compute and execution time ν. However, they require a very time-consuming dataset collection on a physical robot and the system is specific to the sensor and robot configuration it was trained on. Training datasets may need to be re-collected whenever there are changes to the robot, objects, sensors, calibration, or environment (e.g. lighting, background objects).

An additional complication is that empirical dataset collection may produce corrupt training examples. Human labelers can make errors, hardware can break during dataset collection, and sensors can drift or become mis-calibrated. A separate system for data cleaning [87] in post-processing may be required to achieve desired learning performance.

A third challenge with empirical methods is the difficulty of diagnosing failures. For example, when an empirically-trained policy fails to grasp a novel object it is difficult to understand whether the failure is due to (a) a corrupt reward value for similar example from the training dataset, (b) changes to the environment (e.g. lighting, hardware calibration) from the training setting, or (c) too few training examples. While in theory failures may be alleviated by reinforcement learning with additional data collection, current reinforcement learning methods have high sample complexity and may require millions of additional examples to learn [73].

2.2.5 Hybrid Methods: Model-Based Dataset Generation

In this thesis we propose a third, hybrid approach to robust grasp planning that aims to combine the scalability and interpretability of analytic methods with the generalization ability of empirical methods. The key idea is to automate training dataset collection by constructing a probabilistic generative model that can efficiently sample observations, grasps, and rewards using analytic models from physics, geometry, and optics. A quality function can be trained on a dataset sampled from a computer implementation of the generative model in identical fashion to empirical policy learning, in which a function approximator is trained with supervised or reinforcement learning.

Training Phase: Dataset Generation

Hybrid methods require the definition of a dataset generation environment $q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ based on analytic models that can be used to sample datasets of observations and rewards based on a set of candidate grasps. The dataset generation environment can be implemented on a computer and used to rapidly sample millions of examples.

The environment distribution factors according to the graphical model defining the robust grasping problem (Fig. 2.1):

$$q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u}) = \underbrace{q(\mathbf{x})}_{\text{states}} \underbrace{q(R \mid \mathbf{x}, \mathbf{u})}_{\text{rewards}} \underbrace{q(\mathbf{y} \mid \mathbf{x})}_{\text{observations}}$$

The factors model the following quantities:

1. **States** $q(\mathbf{x})$**:** Models variations on objects and sensors. Object variations consist of geometries (e.g. 3D CAD models), poses, frictional properties, center of masses, etc., and may include static objects in the scene such as bins or shelves. Sensor variations may consist of perturbations in the position, orientation, lighting, and camera optical parameters.

2. **Rewards** $q(R \mid \mathbf{x}, \mathbf{u})$**:** Models the results of executing a grasp action using models of contact and wrench mechanics with uncertainty in the grasp outcome due to imprecision in control.

3. **Observations** $q(\mathbf{y} \mid \mathbf{x})$**:** Models a observations for a given state under sensing noise based on rendering (e.g. noisy synthetic point clouds).

To produce a set of candidate grasps to evaluate, hybrid methods also define an explicit *action candidate distribution* $q(\mathbf{u} \mid \mathbf{x}, \mathbf{y})$. This can be a uniform distribution over the action set $\mathcal{U}$ or concentrated on more promising grasps using the current trained policy or an algorithmic supervisor that pre-computes robust grasps:

Definition 4 (Algorithmic Supervisor)**.** *An algorithmic supervisor is a function* $\Omega : \mathcal{X} \to \mathcal{U}$ *that takes as input a fully observed state and returns a grasp for the robot to execute in a synthetic environment:* $\mathbf{u}_\Omega = \Omega(\mathbf{x})$.

An algorithmic supervisor can act as an oracle that produces high-quality grasps with full state knowledge. This is particularly useful in environments where successful grasps are rare, such as grasping objects from cluttered heaps.

Early approaches to synthetic dataset generation for grasping have considered labeling a dataset of point clouds collected from the real robot using antipodality [129], a geometric condition that indicates that a parallel-jaw grasp is in force closure, and using dynamic simulation and depth image rendering to evaluate grasps for a parallel-jaw [70] or multi-finger [175] gripper. The dataset generation distribution often includes significant variation in parameters of physics, such as friction and mass, parameters of sensing, such as lighting and camera intrinsic parameters, and configurations of objects to reflect uncertainty about physics, sensing, and control. This technique is sometimes referred to as *Domain Randomization* for simulation to reality transfer, and it has been shown to improve performance on physical systems for learning grasping policies [70, 129, 169] and navigation policies [147, 191] from synthetic data.

Training Phase: Policy Learning

Once a generative model for datasets has been defined, a quality function can be learned with the same techniques used by empirical methods. Recent research suggesting that synthetic training datasets can be used to train policies that perform well on physical robots for grasping [129, 70] and navigation [147, 191].

Supervised learning techniques may be used to minimize the loss between the reward and predicted quality for a fixed dataset sampled from the dataset generation distribution:

$$\theta^* = \min_{\theta \in \Theta} \sum_{i=1}^{N} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)) \qquad\qquad R_i, \mathbf{x}_i, \mathbf{y}_i, \mathbf{u}_i \sim q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$$

where $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u}) = q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u}) q(\mathbf{u} \mid \mathbf{x}, \mathbf{y})$ is the *dataset generation distribution*, the product of the dataset generation environment and candidate action distribution. Reinforcement learning techniques can be used to iteratively optimize a policy on its own distribution of actions by adaptively sampling from the dataset generation distribution:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \ \mathbb{E}_q \left[R(\mathbf{x}, \pi(\mathbf{y})) \right]$$

where we have specifically noted that the expectation is with respect to the dataset generation distribution q. See Section 2.2.4 for a more detailed description of learning techniques.

Test Phase

Similar to empirical methods, once a quality function Q_θ is learned, hybrid methods plan the grasp that maximizes the learned quality function:

$$\pi(\mathbf{y}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmax}} \ Q_\theta(\mathbf{y}, \mathbf{u}).$$

Discussion

Hybrid grasp planning methods offer several potential benefits. First, synthetic dataset generation can be scalable. Training datasets containing millions of examples can be sampled from efficient computer-implemented generative models in a matter of hours instead of months or years. Second, synthetic datasets can be guaranteed to have clean data. Analytic models can be validated with unit tests to ensure that every datapoint is correct with respect to the model. Third, dataset generation is interpretable. Generated observations, grasps, and rewards that are inconsistent with reality can be traced to a factor of the generative model that makes an incorrect assumption, and the model can be updated accordingly to achieve a desired behavior.

Nonetheless, a key question remains: given a grasp planning problem, how do we design a dataset generation distribution q such that policies learned from q perform well on a physical robot system? Is it even possible to design such a distribution for arbitrary grasp planning problems? Certainly it seems unreasonable to expect that we can construct dataset generation distributions that exactly reflect performance on a physical system.

This thesis explores these questions by focusing on five properties of dataset generation distributions that are important for using hybrid methods in practical applications:

- **Diversity:** Randomization over a wide variety of objects, sensors, and grasps.
- **Efficiency:** The ability to rapidly sample millions of examples.
- **Precision:** Rewarding only grasps that would be successful when executed on a physical robot.
- **Robustness:** Modeling noise and perturbations to be robust to differences between simulation and reality.
- **Flexibility:** The ability to exchange objects, sensors, and grippers without having to re-define the model from scratch.

To explore these topics, this thesis formalizes the hybrid approach to robust grasp planning and explores its application to specific grasping problems including grasping with a parallel-jaw and suction cup gripper, industrial bin picking, and constant-velocity planar pushing. In each chapter, we modify the dataset generation distribution to extend the method to new sensors, objects, grippers, and grasp reward functions and highlight modifications to the policy learning method to utilize the generative model. The methods developed in each chapter are evaluated through large-scale simulated and physical robot experiments to assess performance and provide understanding of the limitations of the method.

The remainder of this thesis details the approach. Part II develops a full sequential dataset generation model q for parallel-jaw grasping to lift, transport, and hold and object under perturbations (e.g. shaking). Part III explores adapting the hybrid method to new problem settings by developing novel reward distributions $q(R \mid \mathbf{x}, \mathbf{u})$ and action candidate distributions $q(\mathbf{u} \mid \mathbf{x}, \mathbf{y})$ based on algorithmic supervisors that guide dataset collection by planing optimal grasps given full knowledge of the state of objects and sensors in the environment. Finally, Part IV discusses the strengths and weaknesses of the approach and highlights opportunities for future research on the topic.

Part II

End-to-End Large Scale Dataset Generation

There is no need to ask the question "Is the model true?".
If "truth" is to be the "whole truth" the answer must be "No".
The only question of interest is:
"Is the model illuminating and useful?"

GEORGE BOX

Chapter 3

Robust Analytic Grasp Planning for Large Datasets of 3D Objects

In this part of the thesis, we develop an stochastic end-to-end generative model to automatically synthesize training datasets for learning robust grasping policies for a parallel-jaw gripper. We first consider the *fully-observed* setting, in which the robot has knowledge of physical parameters (e.g. friction) and object state, such as industrial CAD model registration systems. We then analyze the *partially-observed* setting, in which the robot must plan grasps for unknown objects based on sensor data, and use deep learning to train a robust grasping policy on depth images. Finally, we extend the model to the *sequential* setting, in which the robot must plan a series of grasps over time, and use imitation learning to learn a robust grasping policy for bin picking. In each chapter, we detail the unique attributes of the graphical model and then present a learning-based method for robust grasp planning based on a training dataset sampled from the model.

We begin by presenting the model for computing stochastic analytic grasp quality metrics (rewards) assuming a separate perception system that estimates distributions on state variables such as object shape, pose, and friction from sensor data. This research in this chapter is motivated by stochastic analytic grasp planning using Cloud-based Robotics and Automation systems that exchange grasp data and perform computation via networks instead of operating in isolation with limited computation and memory. Potential advantages to using the Cloud include Big Data: access to updated libraries of images, maps, and object/product data; and Parallel Computation: access to grid computing for statistical analysis, machine learning, and planning [80]. We explore these benefits in the context of machine learning for rapidly planning robust grasps using stochastic analytic methods by developing a "network" measuring similarity between objects and grasps based on deep learned features.

In this chapter we derive a model for computing the quality of parallel-jaw grasps under imprecision in sensing and control over a distribution over possible object shapes. We also develop a Multi-Armed Bandit (MAB) algorithm that models correlated rewards between similar grasps on similar objects, where similarity is defined by a novel distance in feature space based on Multi-View Convolutional Neural Networks (MV-CNNs). We use the model and learning method to generate the Dexterity Network (Dex-Net) 1.0, a dataset containing over ten thousand 3D object models and 2.5 million parallel-jaw grasps with associated robust grasp quality metrics. We implement the Dex-Net algorithm in the Cloud and present experiments suggesting that the MAB algorithm can accelerate robust grasp planning for novel 3D object models by leveraging correlations with pre-computed grasps and objects in Dex-Net 1.0. In particular, we examine the effects of using larger, more diverse object datasets on the number of samples required for the Dex-Net 1.0 algorithm to plan the most robust grasp and find that orders of magnitudes of additional data can lead to a $2\times$ reduction in the number of samples required for grasp planning. This suggests that large datasets of 3D object geometries can aid in robust grasp planning across a diverse set of objects.

The model developed in this chapter may be relevant for robot grasp planning in constrained settings with a small number of possible objects with exactly known CAD models, and is an important building block of the models we use for training dataset generation later in this thesis.

3.1 Problem Statement

We consider accelerating the Training Phase of stochastic analytic grasp planning methods described in Section 2.2.3). In particular, we focus on the problem of pre-computing a large set of robust grasps for each object in a massive dataset of 3D CAD models. When the object is encountered on a physical robot, the set of grasps can be downloaded such that at least one grasp is achievable in the presence of clutter and occlusions.

Formally, our goal is to plan a parallel-jaw grasp that maximizes expected reward for a given 3D object model in as few samples as possible under perturbations in object pose, gripper pose, and friction coefficient.

Assumptions

We assume the exact object shape is given as a triangular 3D mesh. We assume the object is specified in units of meters with given center of mass $\mathbf{z} \in \mathbb{R}^3$. Furthermore, we assume soft-finger point contacts with a Coulomb friction model [190]. We also assume that the gripper jaws are always opened to their maximal width $w \in \mathbb{R}$ before closing.

3.1.1 Definitions

This section uses the following definitions:

- **States.** Let $\mathbf{x} \in \mathcal{X}$ consist of the state of a single object $\mathcal{O} = \{\mathcal{M}, \mathbf{T}_o, \mu, \mathbf{z}\}$ where $\mathcal{M}$ is a 3D triangular mesh representing the geometry of the object, $\mathbf{T}_o$ is the pose of the object geometry, μ is the Coulomb friction coefficient, and $\mathbf{z}$ is the 3D center of mass. An alternative representation of the geometry if a Gaussian Process Implicit Surface (GPIS). See Appendix A for a detailed description.

- **Grasp Actions.** Let $\mathbf{u} = (\mathbf{p}, \mathbf{v}) \in \mathcal{U}$ consist of a *grasp center* in 3D space $\mathbf{p} \in \mathbb{R}^3$ and a *grasp axis* defining the line between the fingertips in 3D space $\mathbf{v} \in \mathcal{S}^2$. This leave one extra degree of freedom, the approach angle defining rotation about the grasp axis, to decide when executing grasps in 3D space. The approach angle could be resolved by evaluating kinematic feasibility or collisions. The nominal grasp pose $\mathbf{T}_g$ is set defined by the grasp center and axis with the approach angle equal to zero. The grasp parameters are illustrated in Fig. 3.1.

- **Reward.** Let $R(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ be a binary grasp reward. Let $F(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ denote force closure, or the ability of a grasp to resist arbitrary wrenches. Let $P_F = \mathbb{E}\left[F \mid \mathbf{x}, \mathbf{u}\right]$ be the probability of force closure under uncertainty in object pose, gripper pose, and friction. We refer to the expected reward as the probability of success, $P_S(\mathbf{x}, \mathbf{u}) = \mathbb{E}[R(\mathbf{x}, \mathbf{u})]$.

- **State Distribution.** The state is sampled by selecting an object geometry from a large dataset of 3D CAD models and setting the pose to a reference frame centered at the object center of mass $\mathbf{z}$ and oriented along the principal axes of $\mathcal{S}$, setting the center of mass to the center of the mesh bounding box, and setting the friction coefficient to the nominal value $\gamma = 0.5$.

Reward Distribution

In this chapter we evaluate grasp quality using the probability of force closure (P_F), or the ability to resist external force and torques in arbitrary directions [108]. P_F allows us to study the effects of large amounts of data on approximate solutions to the robust grasp planning objective of Equation because it is relatively inexpensive to evaluate, and P_F has also shown promise in physical experiments [83, 178].

Let $F \in \{0, 1\}$ denote the occurrence of force closure. For a grasp $\mathbf{u}$ on an object in state $\mathbf{x}$ under uncertainty in gripper pose $\mathbf{T}_g$, object pose $\mathbf{T}_o$, and friction coefficient γ, the probability of force closure $P_F(\mathbf{x}, \mathbf{u}) = \mathbb{P}\left(F = 1 \mid \mathbf{x}, \mu, \mathbf{T}_g, \mathbf{T}_o, \gamma\right)$. To compute force closure for a grasp $\mathbf{u}$ on an object in state $\mathbf{x}$, we compute a set of possible contact wrenches $\mathcal{W}$ using a soft finger contact model [190] based on the sampled values of the grasp pose, object pose, and friction coefficient. A grasp is in force closure ($F = 1$) if $\mathbf{0}$ is in the convex hull of $\mathcal{W}$ [178].

Let $\mathbf{c}_i \in \mathbb{R}^3$ denote the 3D contact location between the i-th jaw and surface as shown in Fig. 3.1. Let $\mathbf{n}_i$ denote the surface normal of $\mathcal{O}$ at $\mathbf{c}_i$ and let $\mathbf{t}_{i,1}, \mathbf{t}_{i,2} \in \mathbb{S}^2$ be its tangent vectors. To compute the forces that each soft contact can apply to the object for friction coefficient $\hat{\mu}$, we discretize the friction cone at $\mathbf{c}_i$ [135] into a set of l facets with vertices:

$$\mathcal{F}_i = \left\{ \mathbf{n}_i + \hat{\gamma} \cos\left(\frac{2\pi j}{l}\right) \mathbf{t}_{i,1} + \hat{\gamma} \sin\left(\frac{2\pi j}{l}\right) \mathbf{t}_{i,2} \Big| j = 1, ..., l \right\}$$

Thus the set of wrenches that $\mathbf{u}$ can apply to $\mathcal{O}$ is:

$$\mathcal{W} = \left\{ \mathbf{w}_{i,j} = (\mathbf{f}_{i,j}, \rho_i \times \mathbf{f}_{i,j} \times) \Big| i = 1, 2 \text{ and } \mathbf{f}_{i,j} \in \mathcal{F}_i \right\}$$

where $\rho_i = (\mathbf{c}_i - \mathbf{z})$ is the moment arm at $\mathbf{c}_i$.

When computing grasp quality $P_F(\mathbf{x}, \mathbf{u})$ we assume Gaussian distributions on the grasp pose, object pose, and friction coefficient to model errors in registration, robot calibration, or classification of material properties. Let $\upsilon \sim \mathcal{N}(\mathbf{0}, \Sigma_\upsilon)$ denote a zero-mean Gaussian on $\mathbb{R}^6$ modeling imprecision in control. We define the grasp pose random variable $\tilde{\mathbf{T}}_g = \exp(\upsilon^\wedge)\,\mathbf{T}_g$, where the $\wedge$ operator maps from $\mathbb{R}^6$ to the Lie algebra $\mathfrak{se}(3)$ [6]. Let $\nu \sim \mathcal{N}(\mathbf{0}, \Sigma_\nu)$ denote a zero-mean Gaussian modeling object pose uncertainty due to registration errors. We define the object pose random variable $\tilde{\mathbf{T}}_o = \exp(\upsilon^\wedge)\,\mathbf{T}_o$ Let $\epsilon_\gamma \sim \mathcal{N}(0, \Sigma_\gamma)$ denote a Gaussian distribution on the friction coefficient and let $\tilde{\gamma} = \gamma + \epsilon_\gamma$. We denote by $\hat{\mathbf{x}}$ and $\hat{\mathbf{u}}$ samples of the object state and grasp action based on samples of the pose and friction random variables.

We can evaluate grasp quality by generating M samples of grasp poses, object poses, and friction coefficients and taking the average reward:

$$\hat{P}_F = \hat{Q}(\mathbf{x}, \mathbf{u}) = \frac{1}{M}\sum_{i=1}^{M} R(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i).$$

3.1.2 Objective

The objective of the Dex-Net 1.0 algorithm is, for a given object state $\mathbf{x}$, to find a grasp $\mathbf{u}^*$ that maximizes an expected binary grasp quality metric $R(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ such as force closure [83, 94, 108, 178] subject to uncertainty in the state of the object, environment, or robot. Since sampling in high-dimensional spaces can be computationally expensive, we attempt to solve for $\mathbf{u}^*$ within T samples by maximizing over the sum of $P_S(\mathbf{x}, \mathbf{u}_t)$ for grasps sampled at times $t = 1, ..., T$ [94, 160]:

$$\underset{\mathbf{u}_1,..,\mathbf{u}_T \in \mathcal{U}}{\text{maximize}} \; \textstyle\sum_{t=1}^{T} P_S(\mathbf{x}, \mathbf{u}_t). \tag{3.1.1}$$

Past research has solved this objective by evaluating and ranking a discrete set of K candidate grasps $\Gamma = \{\mathbf{u}_1, ..., \mathbf{u}_K\}$ using Monte-Carlo integration [79, 178] or Multi-Armed Bandits (MAB) [94]. In this chapter, we extend the 2D MAB model of [94] to leverage similarities between prior grasps and 3D objects in Dex-Net to reduce the number of samples [64].

3.2 The Dexterity Network 1.0 Dataset

The Dexterity Network (Dex-Net) 1.0 dataset includes over 10,000 unique 3D object models annotated with 2.5 million parallel-jaw grasps.

3.2.1 Object Mesh Data

Dex-Net 1.0 contains 13,252 3D mesh models: 8,987 from the SHREC 2014 challenge dataset [99], 2,539 from ModelNet40 [181], 1,371 from 3DNet [180], 129 from the KIT object

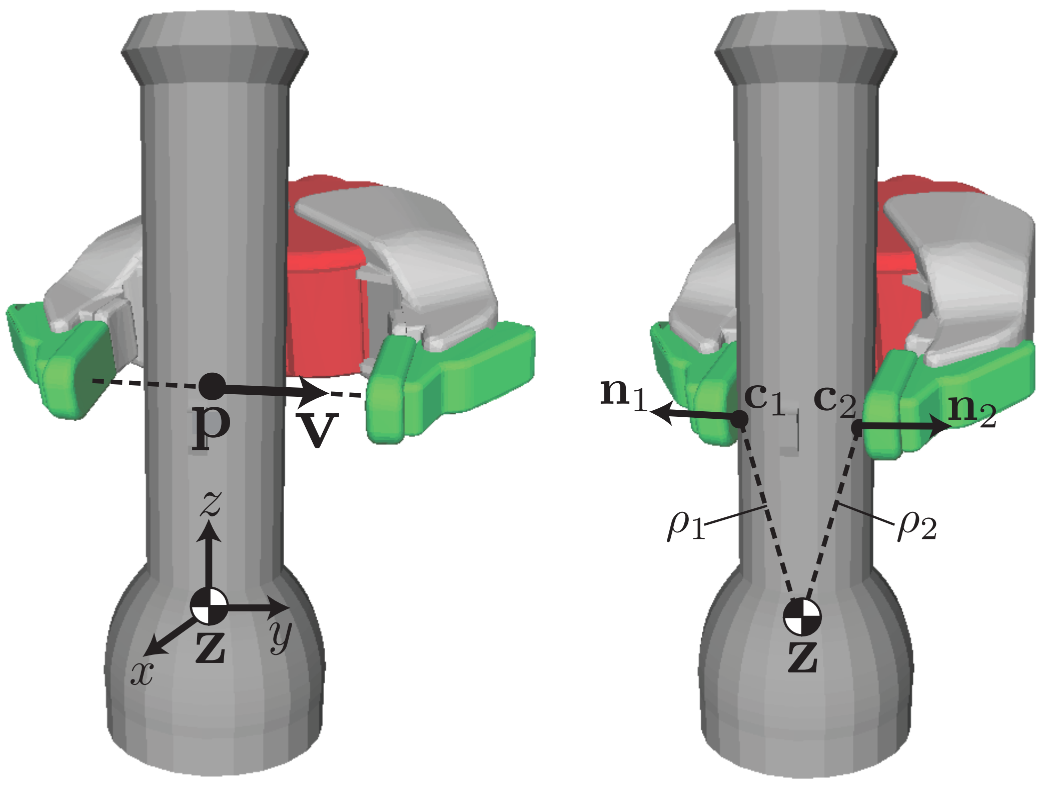


Figure 3.1: Grasp parameterization and contact model. (Left) We parameterize parallel-jaw grasps by the centroid of the jaws $\mathbf{p} \in \mathbb{R}^3$ and approach direction, or direction along which the jaws close, $\mathbf{v} \in \mathbb{S}^2$. The parameters $\mathbf{x}$ and $\mathbf{v}$ are specified with respect to a coordinate frame at the object center of mass $\mathbf{z}$ and oriented along the principal directions of the object. (Right) The jaws are closed until contacting the object surface at locations $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^3$, at which the surface has normals $\mathbf{n}_1, \mathbf{n}_2 \in \mathbb{S}^2$. The contacts are used to compute the moment arms $\rho_i = \mathbf{c}_i - \mathbf{z}$.

database* [78], 120 from BigBIRD* [155], 80 from the Yale-CMU-Berkeley dataset* [16], and 26 from the Amazon Picking Challenge* scans (* indicates laser-scanner data). We preprocess each mesh by removing unreferenced vertices, computing a reference frame with Principal Component Analysis (PCA) on the mesh vertices, setting the mesh center of mass $\mathbf{z}$ to the center of the mesh bounding box, and rescaling the synthetic meshes to fit the smallest dimension of the bounding box within $w = 0.1m$. To resolve orientation ambiguity in the reference frame, we orient the positive z-axis toward the side of the xy plane with more vertices. We also convert each mesh to an SDF using SDFGen [7].

3.2.2 Grasp Sampling

Each 3D object $\mathcal{O}_i$ in Dex-Net is labeled with up to 250 parallel-jaw grasps and their P_F. We generate K grasps for each object using a modification of the 2D algorithm presented in Smith et al. [157] to concentrate samples on grasps that are antipodal [108]. To sample a

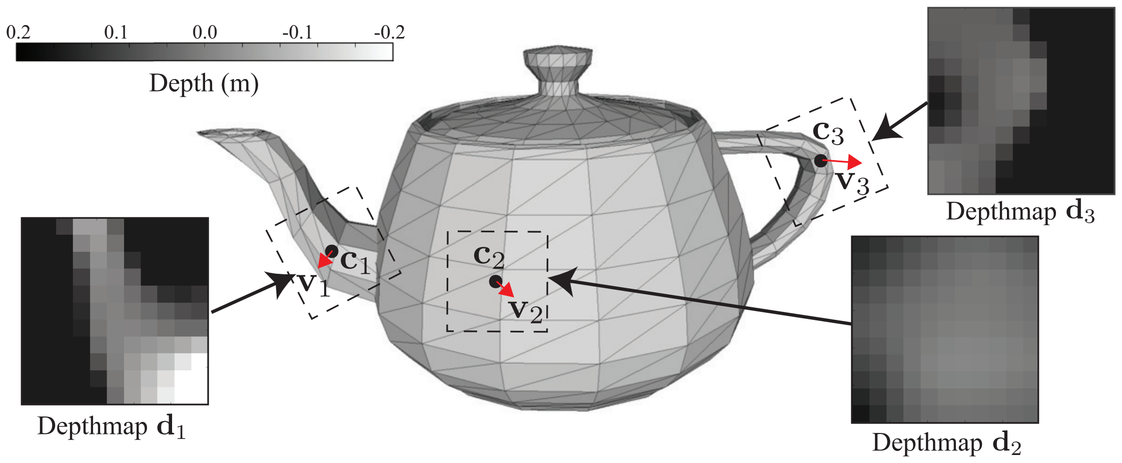


Figure 3.2: Local surface depth map features for measuring grasp similarity for three grasp contact locations on a teapot. Each depthmap is "rendered" along the grasp axis $\mathbf{v}_i$ at contact $\mathbf{c}_i$ and oriented by the directions of maximum variation in the depthmap. We use gradients of the depthmaps for similarity between grasps in Dex-Net.

single grasp, we generate a contact point $\mathbf{c}_1$ by sampling uniformly from the object surface $\mathcal{S}$, sampling a direction $\mathbf{v} \in \mathbb{S}^2$ uniformly at random from the friction cone, and finding an antipodal contact $\mathbf{c}_2$ on the line $\mathbf{c}_1 + t\mathbf{v}$ where $t \geqslant 0$. We add the grasp $\mathbf{u}_{i,k} = (0.5(\mathbf{c}_1 + \mathbf{c}_2), \mathbf{v})$ to the candidate set if the contacts are antipodal [108]. We evaluated $P_F(\mathbf{u}_{i,k})$ using Monte-Carlo integration [79, 178] by sampling the object pose, gripper pose, and friction random variables $N = 500$ times and recording $Z_{i,k}$, the number of samples for which $\mathbf{u}_{i,k}$ achieved force closure ($F = 1$).

3.3 Features for Grasp and Object Similarity

3.3.1 Depthmap Gradient Features for Grasp Similarity

To measure grasp similarity in the Dex-Net 1.0 algorithm, we embed each grasp $\mathbf{u}$ of object $\mathcal{O}$ in Dex-Net in a feature space based on a 2D map of the local surface orientation at the contacts, inspired by grasp heightmaps [62, 77]. We generate a depthmap $\mathbf{d}_i$ for contact $\mathbf{c}_i$ by orthogonally projecting the local object surface onto an $m \times m$ grid centered at $\mathbf{c}_i$ and oriented along the line to the object center of mass, $\mathbf{a}_i = \mathbf{z} - \mathbf{c}_i$. Since F only depends on $\mathbf{c}_i$ and its surface normal, rotations of $\mathbf{d}_i$ about $\mathbf{a}_i$ correspond to grasps of equivalent quality. We therefore make each $\mathbf{d}_i$ rotation-invariant by orienting its axes along the eigenvectors of a weighted covariance matrix of the 3D surface points that generate $\mathbf{d}_i$ as described in [149]. Fig. 3.2 illustrates local surface patches extracted by this procedure. We finally take the x- and y-image gradients of $\mathbf{d}_i$ to form depthmap gradients $\nabla\mathbf{d}_i = (\nabla_x\mathbf{d}_i, \nabla_y\mathbf{d}_i)$, motivated by the dependence of F on surface normals [135], and we store each in Dex-Net 1.0.

3.3.2 Multi-View Convolutional Neural Networks for Object Similarity

We use Multi-View Convolutional Neural Networks (MV-CNNs) [162] to efficiently index prior 3D object and grasp data from Dex-Net by embedding each object in a vector space where distance represents object similarity, as shown in Fig. 3.3. We first render every object on a white background in a total of $C = 50$ virtual camera views oriented toward the object center and spaced on a grid of angle increments $\delta_\theta = \frac{2\pi}{5}$ and $\delta_\varphi = \frac{2\pi}{5}$ on a viewing sphere with radii $r = R, 2R$, where R is the maximum dimension of the object bounding box. Then we train a CNN with the architecture of AlexNet [88] to predict the 3D object class label for the rendered images on a training set of models. We initialize the weights of the network with the weights learned on ImageNet by Krizhevsky et al. [88] and optimize using Stochastic Gradient Descent (SGD). Next, we pass each of the C views of each object through the optimized CNN and max-pool the output of the fc7 layer, the highest layer of the network before the class label prediction. Finally, we use Principal Component Analysis (PCA) to reduce the max-pooled output from 4,096 dimensions to a 100 dimensional feature vector $\psi(\mathcal{O})$.

Given the MV-CNN object representation, we measure the dissimilarity between two objects $\mathcal{O}_i$ and $\mathcal{O}_j$ by the Euclidean distance $\|\psi(\mathcal{O}_i) - \psi(\mathcal{O}_j)\|_2$. For efficient lookups of similar objects, Dex-Net contains a KD-Tree nearest neighbor query structure with the feature vectors of all prior objects. In our implementation, we trained the MV-CNN using the Caffe library [69] on rendered images from a training set of approximately 6,000 3D models sampled from the SHREC 2014 [99] portion of Dex-Net, which has 171 unique categories, for 500,000 iterations of SGD. To validate the implementation, we tested on the SHREC 2014 challenge dataset and achieved a 1-NN accuracy of 86.7%, compared to 86.8% achieved by the winner of SHREC 2014 [99]. See Fig. 3.4 for a detailed analysis of performance.

3.4 Correlated Multi-Armed Bandit Algorithm

The Dex-Net 1.0 algorithm (see pseudocode below) optimizes the probability of success P_S (Equation 3.1.1) for a binary quality metric such as force closure over a discrete set of candidate grasps Γ on an object $\mathcal{O}$ using a Bayesian Multi-Armed Bandit (MAB) model [94, 160] with correlated rewards [64] and priors computed from Dex-Net 1.0. We first generate the set of K candidate grasps using the antipodal grasp sampling described in Section 3.2.2 and treat the grasps as “arms” in the MAB model. Next, we predict P_S for each grasp using the M most similar objects from the Dex-Net 1.0 dataset and estimate a Bayesian posterior distribution on our prediction. Then, for iterations $t = 1, ..., T$ we use Thompson sampling [94, 124] to select a grasp $\mathbf{u}_{t,k} \in \Gamma$ to evaluate, sample the force closure reward $R(\mathbf{x}, \mathbf{u}_{t,k})$, and update a posterior belief distribution on P_S for each grasp. Finally, we rank Γ by the q-lower confidence bound on P_S for each grasp and store the ranking in the database.

To illustrate convergence of the algorithm, we use force closure [190] as our binary quality

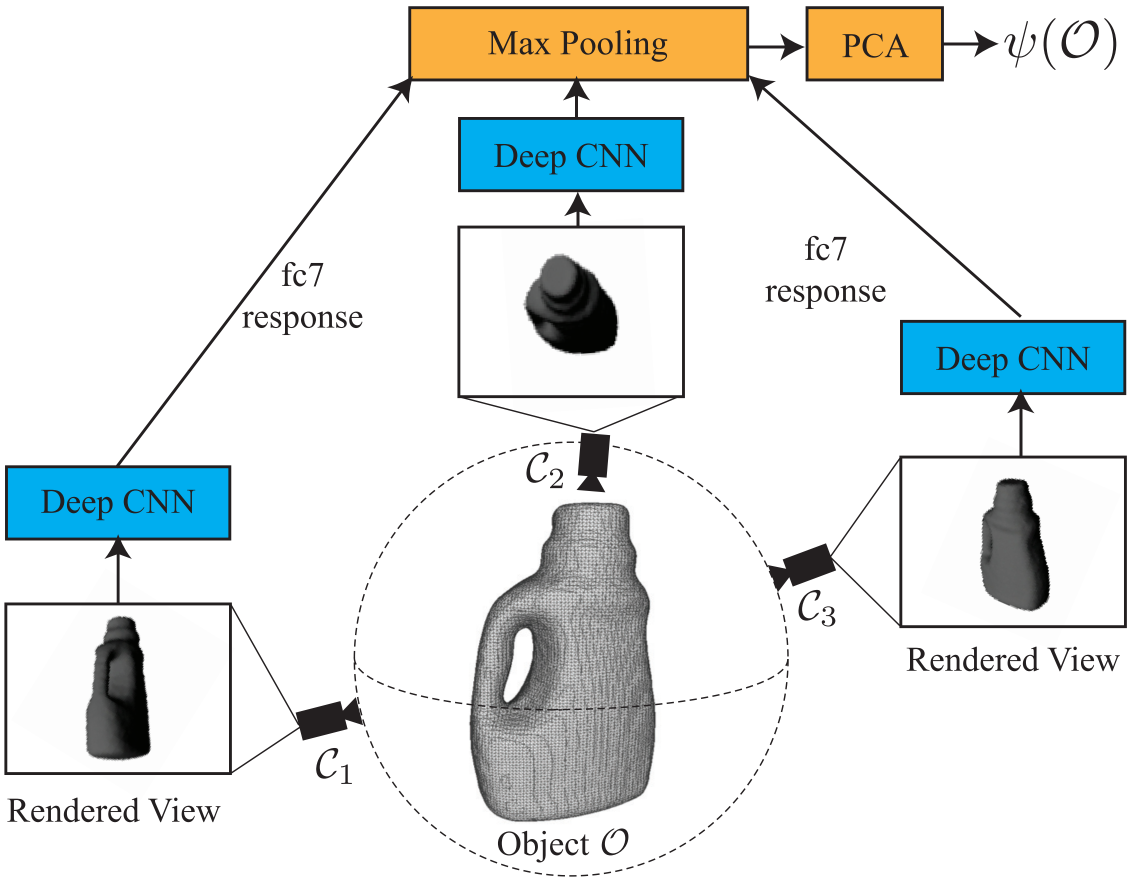


Figure 3.3: Multi-View Convolutional Neural Network (MV-CNN) deep learning architecture for embedding 3D object models in a Euclidean vector space to compute global shape similarity. We pass a set of 50 virtually rendered camera viewpoints discretized around a sphere through a deep Convolutional Neural Network (CNN) with the AlexNet [88] architecture. Finally, we take the maximum fc7 response across each of the 50 views for each dimension and run PCA to reduce dimensionality.

metric. We plan to study other quality metrics such as success on physical trials [89, 119] and alternate MAB methods based on upper confidence bounds [89, 124] or Gittins indices [94] in future work.

3.4.1 Model of Correlated Rewards

Let $R_j \sim p(R \mid \mathbf{x}, \mathbf{u}_j) \in \{0, 1\}$ be a random binary quality metric evaluated on grasp $\mathbf{u}_j \in \Gamma$. For example, R_j might model force closure under uncertainty in object pose, gripper pose, or friction. Each R_j is a Bernoulli random variable with probability of success $\theta_j = P_S(\mathbf{u}_j)$.

We use Continuous Correlated Beta Processes (CCBPs) [48, 119] to model a joint posterior belief distribution over the θ_j for all grasps in Dex-Net, which enables us to predict θ_j from prior grasp and object data in Dex-Net 1.0 using a closed-form posterior update. The joint distribution models pairwise correlations of θ between grasp-object pairs $\mathcal{P} = (\mathbf{u}, \mathcal{O})$

Metric	Best of 2014 SHREC Challenge	Our CNN-based Method
Nearest Neighbor	**0.868**	0.867
First Tier	0.528	**0.542**
Second Tier	0.661	**0.682**
E-Measure	0.255	**0.260**
Discounted Cumulated Gain	0.823	**0.837**

Figure 3.4: Comparison of MV-CNN with the winner of the SHREC 2014 challenge on the SHREC 2014 benchmark.

(points in a Grasp Moduli Space [133]) measured using a normalized kernel function $k(\mathcal{P}_i, \mathcal{P}_j)$ that approaches 1 as the arguments become increasingly similar and approaches 0 as the arguments become dissimilar.

Dex-Net 1.0 measures similarity using a set of feature maps $\varphi_m \in \mathbb{R}^{d_m}$ for $m = 1, ..., 3$, where d_m is the dimension of the feature space for each. The first feature map $\varphi_1(\mathcal{P}) = (\mathbf{x}, \mathbf{v}, \|\rho_1\|_2, \|\rho_2\|_2)$ captures similarity in the grasp parameters, where $\mathbf{x} \in \mathbb{R}^3$ is the grasp center, $\mathbf{v} \in \mathbb{S}^2$ is the approach axis, and $\rho_i \in \mathbb{R}^3$ is the i-th moment arm. The second feature map $\varphi_2(\mathcal{P}) = (\nabla \mathbf{d}_1, \nabla \mathbf{d}_2)$ uses the depthmap gradients described in Section 3.3.1. Our third feature map $\varphi_3(\mathcal{P}) = \psi(\mathcal{O})$ is the object similarity map described in Section 3.3.2 to capture global shape similarity.

Given the feature maps, we use the squared exponential kernel

$$k(\mathcal{P}_p, \mathcal{P}_q) = \exp\left(-\frac{1}{2} \sum_{m=1}^{3} \|\varphi_m(\mathcal{P}_p) - \varphi_m(\mathcal{P}_q)\|_{C_m}^2 \right).$$

where $C_m \in \mathbb{R}^{d_m \times d_m}$ is the bandwidth for φ_m and $\|\mathbf{y}\|_{C_m} = \mathbf{y}^T C_m^{-1} \mathbf{y}$. The bandwidths are set by maximizing the log-likelihood [48] of the true θ on a set of training data.

3.4.2 Predicting Grasp Quality Using Prior Data

Before evaluating any grasps in Γ, the Dex-Net 1.0 algorithm predicts θ_j for each candidate grasp $\mathbf{u}_j$ based on its kernel similarity to all grasps and objects from the Dex-Net 1.0 dataset $\mathcal{D}$. In particular, we estimate a Bayesian posterior density $p(\theta_j)$ by treating $\mathcal{D}$ as prior observations and using the closed form posterior update for CCBPs [48]:

$$p(\theta_j \mid \alpha_{j,0}, \beta_{j,0}) \propto \theta_j^{\alpha_{j,0}-1}(1-\theta_j)^{\beta_{j,0}-1} \tag{3.4.1}$$

$$\alpha_{j,0} = \alpha_0 + \sum_{i=1}^{|\mathcal{D}|}\sum_{k=1}^{K} k(\mathcal{P}_j, \mathcal{P}_{i,k}) Z_{i,k} \tag{3.4.2}$$

$$\beta_{j,0} = \beta_0 + \sum_{i=1}^{|\mathcal{D}|}\sum_{k=1}^{K} k(\mathcal{P}_j, \mathcal{P}_{i,k})(N - Z_{i,k}) \tag{3.4.3}$$

where α_0 and β_0 are prior parameters for the Beta distribution [94], N is the number of times each grasp $\mathbf{g}_{i,k} \in \mathcal{D}$ was sampled to estimate θ_i, and $Z_{i,k}$ is the number of observed successes for $\mathbf{u}_{i,k}$. Intuitively, the prior dataset contributes fractional observations of successes and failures for the grasp candidates Γ proportional to the kernel similarity. We estimate the above sums using the M nearest neighbors to the object in the object similarity KD-Tree described in Section 3.3.2.

3.4.3 Grasp Selection Policy

On iteration t we select the next grasp to sample $\mathbf{u}_j \in \Gamma$ using Thompson Sampling. In Thompson Sampling we draw a sample $\hat{\theta}_\ell \sim p(\theta_\ell \mid \alpha_{\ell,t}, \beta_{\ell,t})$ for each grasp $\mathbf{u}_\ell \in \Gamma$, then choose the grasp $\mathbf{u}_j$ with the highest $\hat{\theta}_j$ [94]. After observing R_j, we update the belief for all grasps $\mathbf{u}_\ell \in \Gamma$ by updating a running count of the fractional successes and failures [48]:

$$\alpha_{\ell,t} = \alpha_{\ell,t-1} + k(\mathcal{P}_\ell, \mathcal{P}_j) R_j \tag{3.4.4}$$

$$\beta_{\ell,t} = \beta_{\ell,t-1} + k(\mathcal{P}_\ell, \mathcal{P}_j)(1 - R_j). \tag{3.4.5}$$

3.5 Experiments on Large-Scale Grasp Analysis

We evaluate the performance of the Dex-Net 1.0 algorithm on robust grasp planning for varying sizes of prior data used from Dex-Net using force closure as our binary quality metric, and we explore the sensitivity of the convergence rate to object shape, the similarity kernel bandwidths, and uncertainty. We created two training sets of 1,000, and 10,000 objects by uniformly sampling objects from Dex-Net. We uniformly sampled a set of 300 validation objects for selecting algorithm hyperparameters and selected a set of 45 test objects from the remaining objects. We ran the algorithm for $T = 2,000$ iterations with $M = 10$ nearest neighbors, $\alpha_0 = \beta_0 = 1$ [94], and a lower confidence bound containing $q = 75\%$ of the belief distribution. We used isotropic Gaussian uncertainty with object and gripper translation variance $\sigma_t = 0.005$, object and gripper rotation variance $\sigma_r = 0.1$, and friction variance $\sigma_\gamma = 0.1$. For each experiment we compare the Dex-Net algorithm to Thompson sampling without priors (TS) [94], a state-of-the-art method for robust grasp planning, and uniform

```
1  Input: Object $\mathcal{O}$, Number of Candidate Grasps $K$, Number of Nearest Neighbors $M$,
     Dex-Net 1.0 Dataset $\mathcal{D}$, Feature maps $\varphi$, Maximum Iterations $T$, Prior beta shape $\alpha_0$, $\beta_0$,
     Lower Bound Confidence $q$, Reward Function $R$
   Result: Estimate of the grasp with highest $P_F$, $\hat{\mathbf{g}}^*$
   // Generate candidate grasps and priors
2  $\Gamma = \text{AntipodalGraspSample}(\mathcal{O}, K)$ ;
3  $\mathcal{A}_0 = \varnothing, \mathcal{B}_0 = \varnothing$;
4  for $\mathbf{g}_k \in \Gamma$ do
       // Equations 3.4.2 and  3.4.3
5      $\alpha_{k,0}, \beta_{k,0} = \text{ComputePriors}(\mathcal{O}, \mathbf{u}_k, \mathcal{D}, M, \varphi, \alpha_0, \beta_0)$;
6      $\mathcal{A}_0 = \mathcal{A}_0 \cup \{\alpha_{k,0}\}, \mathcal{B}_0 = \mathcal{B}_0 \cup \{\beta_{k,0}\}$;
7  end
   // Run MAB to Evaluate Grasps
8  for $t = 1, .., T$ do
9      $j = \text{ThompsonSample}(\mathcal{A}_{t-1}, \mathcal{B}_{t-1})$;
10     $S_j = \text{SampleQuality}(\mathbf{u}_j, \mathcal{O}, S)$;
       // Equations 3.4.4 and  3.4.5
11     $\mathcal{A}_t, \mathcal{B}_t = \text{UpdateBeta}(j, R_j, \Gamma)$;
12     $\mathbf{g}_t^* = \text{MaxLowerConfidence}(q, \mathcal{A}_t, \mathcal{B}_t)$;
13 end
14 return $\mathbf{u}_T^*$;
15 Dex-Net 1.0 Algorithm: Robust Grasp Planning Using Multi-Armed Bandits with
     Correlated Rewards
```

allocation (UA), a widely-used method for robust grasp planning that selects the next grasp to evaluate uniformly at random [79, 83, 178].

The inverse kernel bandwidths were selected by maximizing the log-likelihood of the true P_F under the CCBP model [48] on the validation set using a grid search over hyperparameters. The inverse bandwidths of the similarity kernel were $C_g = diag(0, 0, 3 \times 10^{-5}, 3 \times 10^{-5})$ for the grasp parameter features, an isotropic Gaussian mask C_d with mean $\mu_d = 500.0$ and $\sigma_d = 0.33$ for the differential depthmaps, and $C_s = 10^6 * I$ for the shape similarity features.

To scale experiments, we developed a Cloud-based system on top of Google Cloud Platform. We used Google Compute Engine (GCE) to construct the Dex-Net 1.0 dataset and to distribute subsets of objects to virtual machines for MAB experiments, and we used Google Cloud Storage to store Dex-Net. The system launched up to 1,500 GCE virtual instances at once for experiments, reducing the runtime by an estimated three orders of magnitude to approximately 315 seconds per object for both loading the dataset and running the Dex-Net 1.0 algorithm. Each virtual instance ran Ubuntu 12.04 on a single core with 3.75 GB of RAM.

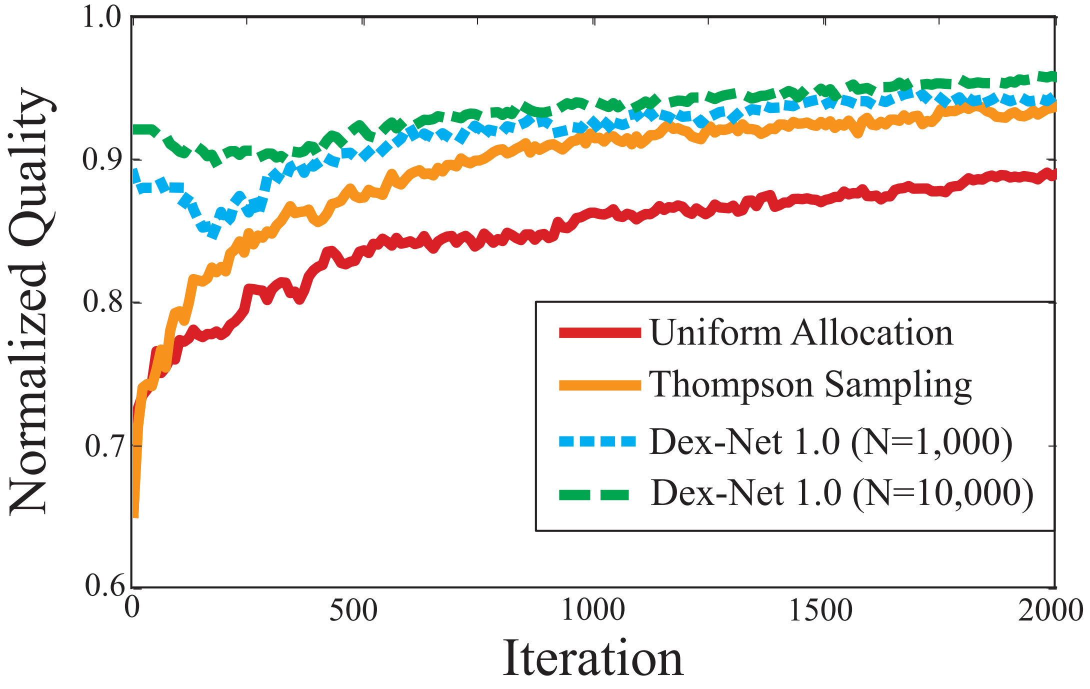


Figure 3.5: Average normalized grasp quality versus iteration over 45 test objects and 25 trials per object for the Dex-Net 1.0 algorithm with 1,000 and 10,000 prior 3D objects from Dex-Net. We measure quality by the P_F for the best grasp predicted by the algorithm on each iteration and compare with Thompson sampling without priors and uniform allocation. The algorithm converges faster with 10,000 models, never dropping below approximately 90% of the grasp with highest P_F from a set of 250 candidate grasps.

3.5.1 Scaling of Average Convergence Rate

To examine the effects of orders of magnitude of prior data on convergence to a grasp with high P_F, we ran the Dex-Net 1.0 algorithm on the test objects with priors computed from 1,000 and 10,000 objects from Dex-Net. Fig. 3.5 shows the normalized P_F (the ratio of the P_F for the best grasp predicted by the algorithm to the highest P_F of the candidate grasps) versus iteration averaged over 25 trials for each of the 45 test objects to facilitate comparison across objects. The average runtime per iteration was 16 ms for UA, 17 ms for TS, and 22 ms for Dex-Net 1.0. The algorithm with 10,000 objects takes approximately $2\times$ fewer iterations to reach the maximum normalized P_F value reached by TS, which is particularly promising for binary success metrics that are expensive to evaluate such as detailed physics simulations, human labels, or physical grasping trials. Furthermore, the 10,000 object curve does not fall below approximately 90% of the best grasp in the set across all iterations, suggesting that a grasp with high P_F is found using prior data alone. The maximum standard error of the mean over all iterations was 2×10^{-3} for UA, 2×10^{-3} for TS, and 1×10^{-3} for Dex-Net 1.0 with 1,000 and 10,000 objects.

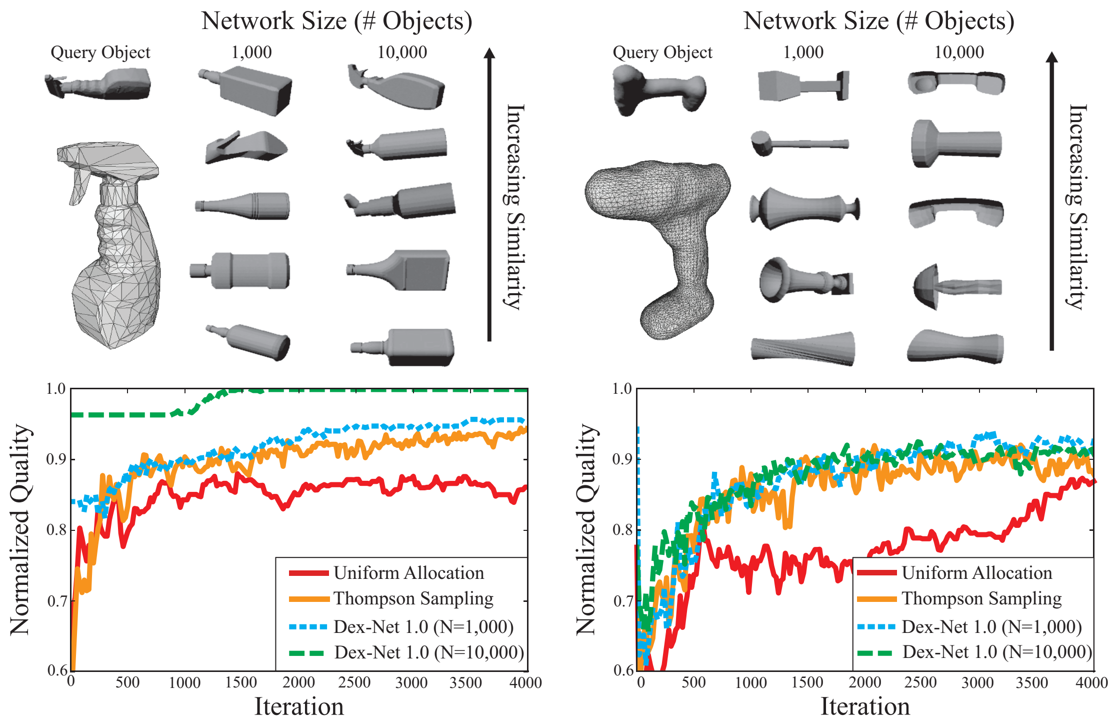


Figure 3.6: Average normalized grasp quality versus iteration for 25 trials for the Dex-Net 1.0 Algorithm with 1,000 and 10,000 prior 3D objects from Dex-Net (bottom) and illustrations of five nearest neighbors in Dex-Net (top) for a spray bottle (left) and drill (right). We measure quality by the probability of force closure of the best grasp predicted by the algorithm on each iteration and compare with Thompson sampling without priors [94] and uniform allocation [79, 178]. (Top) The spray bottle has no similar neighbors with 1,000 objects, but two other spray bottles are found by the MV-CNN in the 10,000 object set. The drill, which is relatively rare in the dataset, has no geometrically similar neighbors even with 10,000 objects. (Bottom) For the spray bottle the Dex-Net 1.0 algorithm quickly converges to the optimal grasp with 10,000 prior objects, but for the drill the lack of similar objects leads to no significant performance increase over Thompson sampling without priors.

3.5.2 Sensitivity to Object Shape

To understand the behavior of the Dex-Net 1.0 algorithm on individual 3D objects, we examined performance on a drill and spray bottle from the test set, both uncommon object categories in Dex-Net 1.0. Fig. 3.6 show the normalized P_F versus iteration averaged over 25 trials for 2,000 iterations on the spray bottle and drill, respectively. We see that the spray bottle converges very quickly when using a prior dataset of 10,000 objects, finding the optimal grasp in the set in about 1,500 iterations. This convergence may be explained by the two similar spray bottles retrieved by the MV-CNN from the 10,000 object dataset. Fig. 3.7 illustrates the grasps predicted to have the highest P_F on the spray bottle by the different algorithms after 100 iterations. On the other hand, performance on the drill does not improve using either 1,000 or 10,000 objects, perhaps because the closest model in Dex-Net according to the similarity metric is a phone.

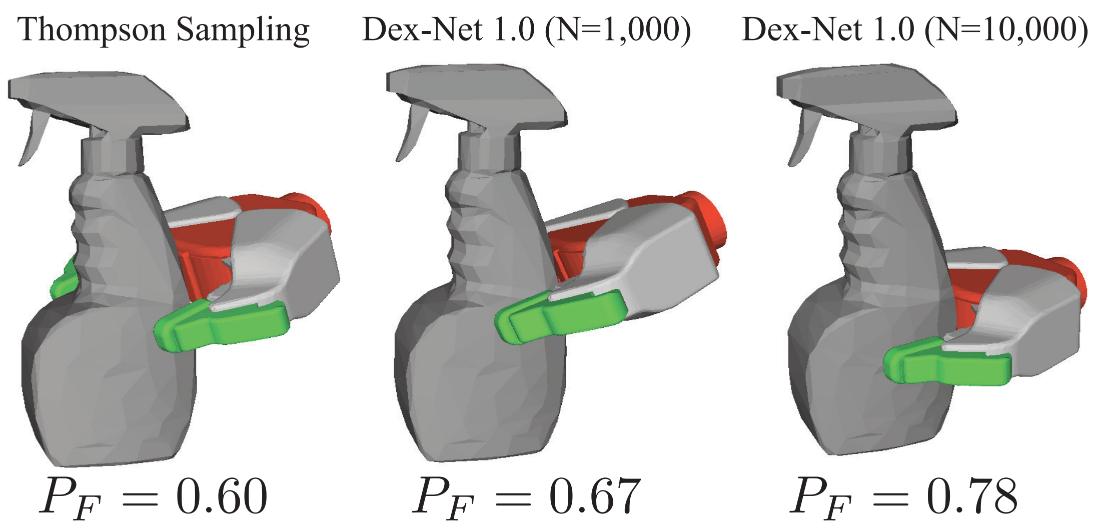


Figure 3.7: Example grasps predicted to have the highest P_F on the spray bottle after only 100 iterations by Thompson sampling without priors and the Dex-Net 1.0 algorithm with 1,000 and 10,000 prior objects. Thompson sampling without priors chooses a grasp near the edge of the object, while the Dex-Net algorithm selects grasps closer to the object center-of-mass. For reference, the highest quality grasp for the spray bottle was $P_F = 0.81$.

3.5.3 Sensitivity to Similarity and Uncertainty

We also studied the sensitivity of the Dex-Net algorithm to the similarity kernel bandwidths described in Section 3.4 and the levels of pose and friction uncertainty for the test object. We varied the inverse bandwidths of the kernel for the grasp parameters and depthmap gradients to the lower values $C_g = diag(0, 0, 15, 15)$, $\mu_d = 350.0$, and $\sigma_d = 3.0$ as well as the higher values $C_g = diag(0, 0, 300, 300)$, $\mu_d = 750.0$, and $\sigma_d = 1.75$. We also tested low uncertainty with variances $(\sigma_t, \sigma_r, \sigma_\gamma) = (0.0025, 0.05, 0.05)$ and high uncertainty with variances $(\sigma_t, \sigma_r, \sigma_\gamma) = (0.01, 0.2, 0.2)$. Fig. 3.8 shows the normalized P_F versus iteration averaged over 25 trials for 2,000 iterations on the 45 test objects. The results suggest that a conservative setting of similarity kernel bandwidth is important for convergence and that the algorithm is not sensitive to uncertainty levels.

3.6 Experiments on a Physical Robot

We performed addition experiments to compare the predictions of robust grasp quality metrics with the results of executing grasps on a physical system.

3.6.1 Experimental Setup

Grasping trials were run on a Zymark Zymate 2 robot with 4 degrees of freedom plus gripper control and a rotating turntable for 5 total controllable degrees of freedom (Fig. 3.9). The

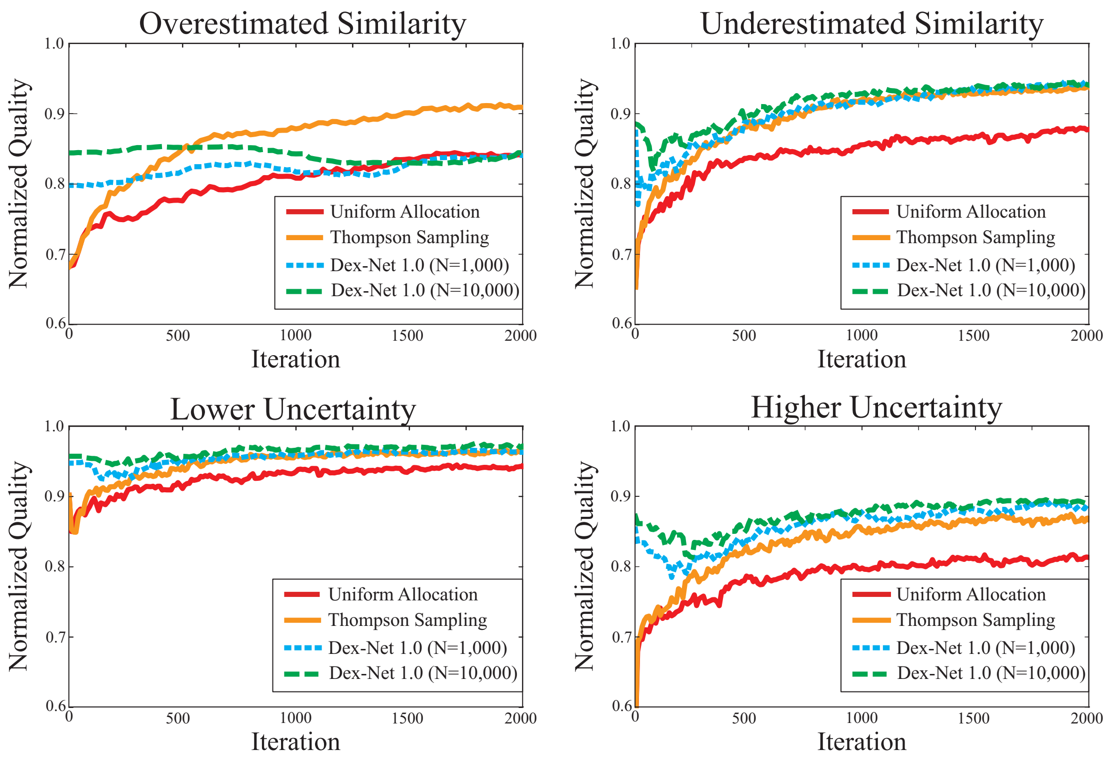


Figure 3.8: Sensitivity to similarity kernel (top) and pose and friction uncertainty (bottom) for the normalized grasp quality versus iteration averaged over 25 trials per object for the Dex-Net algorithm with 1,000 and 10,000 prior 3D objects. (Top-left) Using a higher inverse bandwidth causes the algorithm to measure false similarities between grasps, leading to performance on par with uniform allocation. (Top-right) A lower inverse bandwith decreases the convergence rate, but on average the Dex-Net algorithm still selects a grasp within approximately 85% of the grasp with highest P_F for all iterations. (Bottom-left) Lower uncertainty increases the quality for all methods, (bottom-right) higher uncertainty decreases the quality for all methods, and the Dex-Net algorithm with 10,000 prior objects still converges approximately 2$\times$ faster than Thompson sampling without priors.

parallel-jaw gripper consisted of two pincer fingers with rubber fingertips.

We evaluated performance across 13 unique 3D printed objects pictured in Fig. 3.10, and attempted 12 unique grasps for each object. The objects were chosen to be difficult to grasp with a parallel-jaw gripper due to smooth, curved surfaces, and challenging collision geometries.

On each grasp attempt, a single 3D printed object was placed in the center of the table in a known stable resting pose using an automatic hardware reset mechanism, which lifted the object using a pulley and damper system and slowly lowering the object to rest on the table. Then a grasp was executed by registering a depth image from a Primesense Carmine RGB-D camera to a known 3D CAD model of the object and indexing a pre-computed grasp from Dex-Net 1.0.

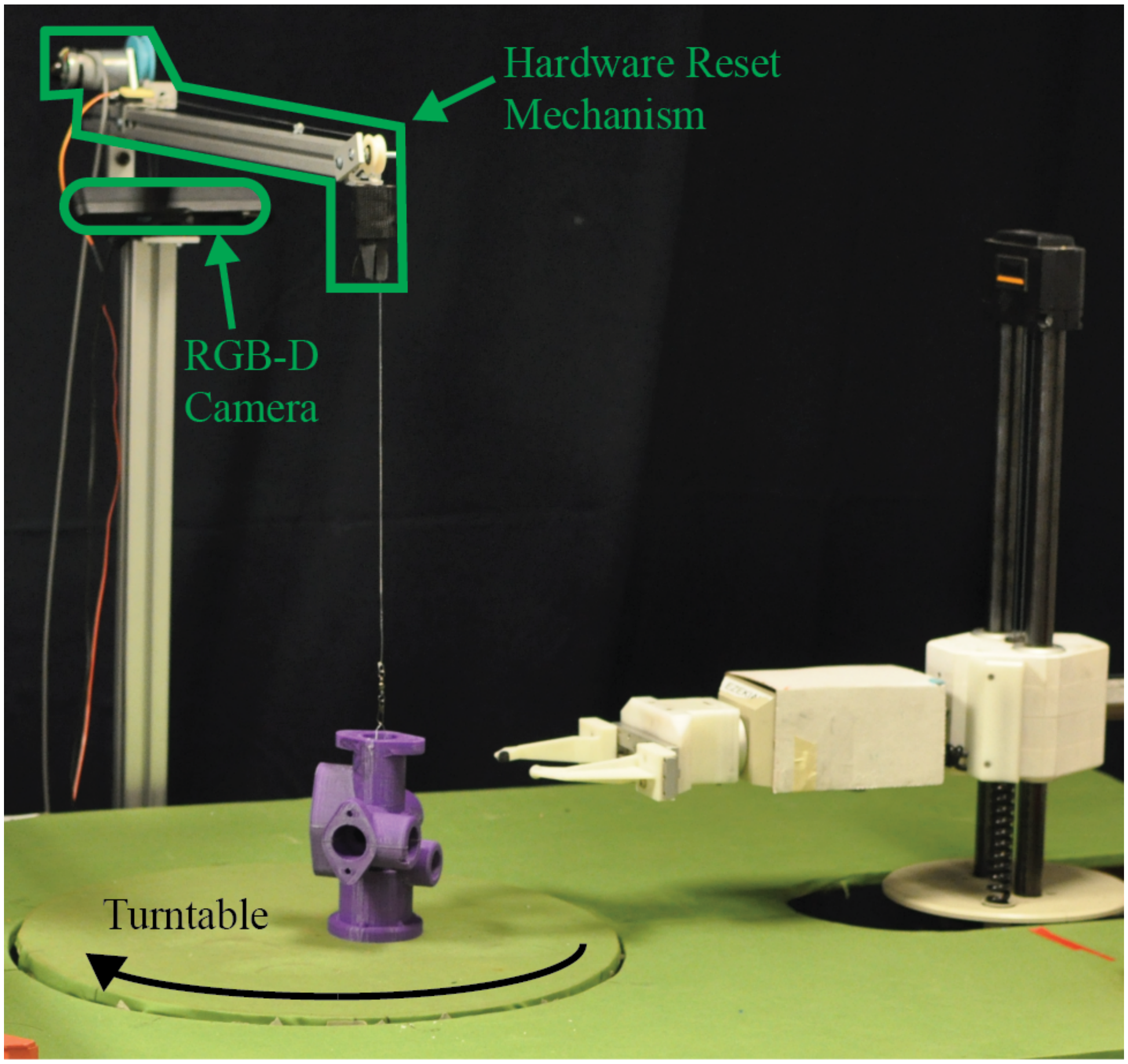


Figure 3.9: Experimental setup used to evaluate the correlation between robust analytic grasp quality metrics and the empirical outcomes of grasp attempts on a physical system. The robot consisted of a Zymark Zymate 2 arm (right) with four degrees of freedom and a parallel-jaw pincer gripper as well as a rotating turntable to augment the system for five controllable degrees of freedom. On each grasp attempt, a single 3D printed object was placed in the center of the table in a known stable resting pose using an automatic hardware reset mechanism. Then a grasp was executed by registering a depth image from a Primesense Carmine RGB-D camera to a known 3D CAD model of the object and indexing a pre-computed grasp from Dex-Net 1.0.

3.6.2 Perception System

In order to execute pre-computed grasps on the physical robot, we designed and implemented a novel perception system for registering the global 3D pose of a known 3D object based on Multi-View Convolutional Neural Networks (MV-CNNs). The perception system, illustrated in Fig. 3.11, consisted of two phases: object pose recognition and geometric alignment. First a database of known images for each object was created by rendering a set of a synthetic

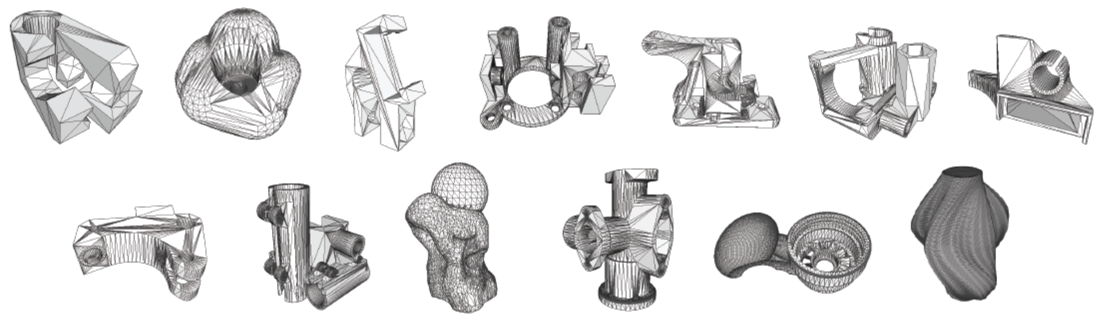


Figure 3.10: Set of 13 adversarial 3D printed objects used to evaluate robust analytic grasp quality metrics on a physical robot.

segmentation masks for each object in various orientations. Each synthetic segmentation mask was featurized using the fc7 output of the trained MV-CNN of Section 3.3.2 and the feature vectorse were stored in a database At runtime, the object was segmented from the background in a depth image using the known pose of the RGB-D camera and featurized using the MV-CNN. Then the features were matched to the database of pre-computed features for the known object and the pose corresponding to the closest feature match was used as an initial estimate of the object pose. Finally, the pose was refined by geometrically aligning the segmented object point cloud with the known 3D CAD model of the object using the point-to-plane Iterated Closest Point algorithm [122].

To validate our registration system, we ran over 1,000 trials of registration over the 13 3D printed test objects in known poses and compared the estimated pose to ground truth. First, we placed each object in a known pose on the turntable using templates. Then, the system estimated the object pose for ten unique images and updated the object pose by rotating the turntable by 30°. After rotating a full 360°, the object was changed.

Fig. 3.12 displays the pose error histograms for the x and y position and orientation of the object on the turntable. The median absolute translation error was 4.5mm and the mean absolute orientation error was 3.5°.

3.6.3 Evaluation of Robust Analytic Grasp Metrics

To evaluate the robust analytic grasp quality metrics on the physical robot, we iteratively executed grasps for each of the 3D printed test objects and compared the predictions with the outcomes on the physical system. For each object, we hand-selected 12 unique grasps to attempt that covered the surface of the object. Each grasp was evaluated over 10 trials. In each trial, the object was placed on the turntable in a known stable resting pose using the hardware reset mechanism, the perception system estimated the object pose, the pre-computed grasp was converted to a 3D pose for the physical robot gripper using the pose estimate, and the robot executed the grasp by moving linearly to the grasp pose and lifting

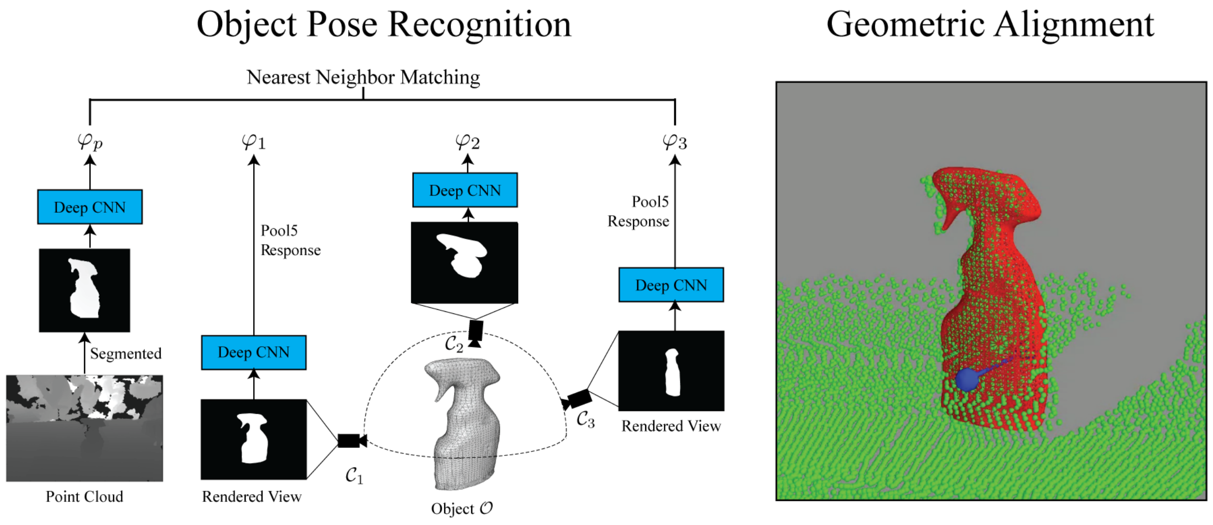


Figure 3.11: Perception system used to estimate the 3D pose of known objects from depth images in experiments on a physical robot. In the object pose recognition phase, a segmentation mask for the object was matched to a set of pre-rendered segmentation masks of the object CAD model in various orientations using CNN features. Then the pose of the CAD model corresponding to the most similar pre-rendered segmentation mask was used to seed the geometric alignment phase, in which the CAD model was iteratively aligned to the segmented point cloud using point-to-plane Iterated Closest Point.

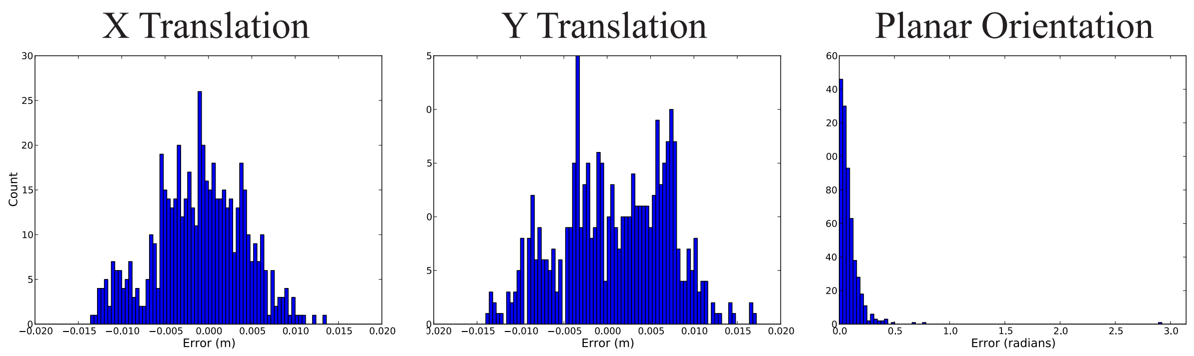
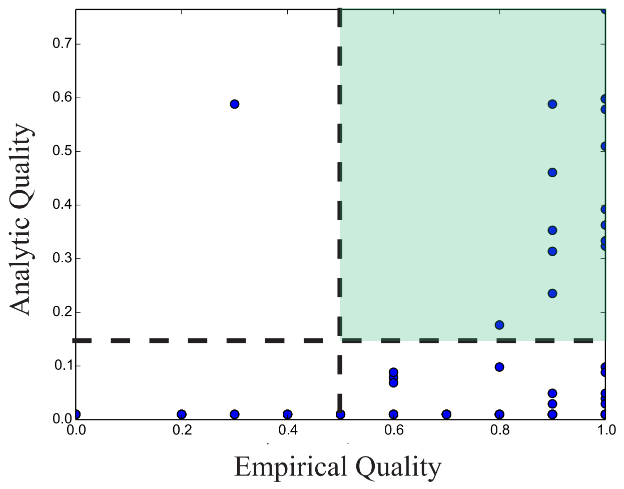


Figure 3.12: Planar pose error histograms for the known object pose registration system used to evaluate robust analytic grasp quality metrics.

the object. This resulted in 1,560 total grasp attempts.

Grasps were labeled with binary success of failure by a human operator. A grasp was considered successful if it was lifted above the table surface and was held firmly by the robot fingertips (not other parts of the gripper geometry). The successes for each grasp were converted to an empirical quality estimate by taking the fraction of successes over the number of total attempts.

We computed a number of robust analytic grasp quality metrics for each grasp to understand performance over a range of parameter values. The metrics considered were: proba-



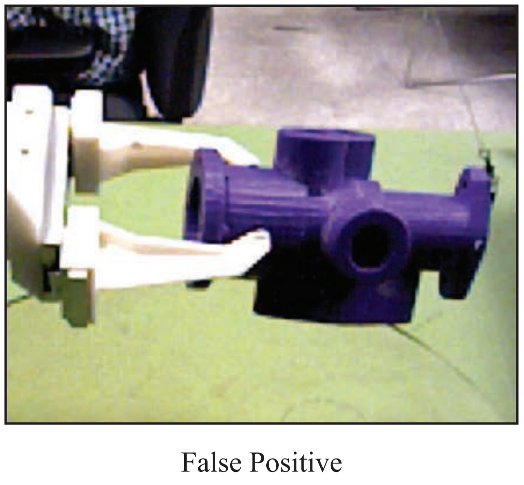


Figure 3.13: Performance of probability of partial closure, the highest precision robust analytic grasp quality metric, on predicting the outcomes of grasp attempts on a physical robot (empirical quality). (Left) A scatter plot of the predicted versus empirical quality for a set of 156 grasps suggests that the analytic quality metric has few false positives. The majority of classification errors are due to false negatives, suggesting that robust analytic metrics are overly conservative in predicting the outcomes of empirical grasps. This may be due to unmodeled affordances such as dynamically pushing the object into alignment. (Right) The lone false positive is a grasp for which the object dynamically rotates out of the gripper due to gravity.

bility of force closure [178], expected epsilon quality [77], and probability of partial closure with respect to forces and torques due to gravity [91]. Each metric was considered over a range of uncertainty parameters: none, low, medium, and high. We then evaluated the predictive value of the metrics by considering the binary classification performance of the robust analytic grasp quality metrics on the empirical quality thresholded by 50% success.

None of the robust analytic metrics performed well in terms of classification accuracy. The highest performing metric was the expected epsilon quality with an accuracy of 63%. However, several of the metrics had over 90% precision (percent of predicted successful grasps that were actually successful). This is important because it indicates that grasps with high robust analytic quality are highly likely to succeed on a physical robot, and in practice a robot could plan grasps by maximizing robust analytic quality.

Fig. 3.13 displays the results for the probability of partial closure with medium uncertainty, the best performing metric with 93% precision. We see that the classification accuracy is low because there are a large number of grasps that are predicted to fail by the robust analytic metrics that actually succeed on the physical system. In practice, these grasps succeed due to dynamic effects that are unmodeled by the metrics such as pushing the object into alignment. The single false positive is a grasp for which the object dynamically rotates out of the jaws due to gravity.

3.7 Discussion

In this chapter, we presented the Dexterity Network (Dex-Net) 1.0, a new dataset and associated algorithm to study the scaling effects of Big Data and Cloud Computation on stochastic analytic grasp planning with binary grasp rewards. The algorithm uses a Multi-Armed Bandit model with correlated rewards to leverage prior grasps and 3D object models. Experiments using the Google Cloud Platform suggest that prior data can speed robust grasp planning by a factor of 2 and that average grasp quality increases with the number of similar objects in the dataset.

Experiments using Dex-Net 1.0 on a physical robot uncovered several benefits and drawbacks of the approach presented in this chapter. First, experiments suggest that robust grasp quality metrics defined by the reward and state distribution in this chapter may have high precision on a physical robot. Given appropriate parameter settings, robust grasp quality metrics may underestimate the expected reward for grasping on a physical system. When a grasp has high quality according to these metrics, the grasp is likely to succeed on the physical system. However, experiments also revealed that force closure is not a particularly useful metric – more complex grasp metrics based on disturbing wrenches are necessary to accurately predict outcomes on a physical system. Furthermore, the instance recognition and pose-registration perception system necessary for estimating object identity and pose does not scale beyond a very small number of objects. This raises the question: can robust grasps according to analytic quality metrics be identified directly from images without estimating exact object shape and pose?

Chapter 4

Learning to Plan Grasps from Synthetic Point Clouds and Analytic Metrics

Reliable grasping across a wide variety of objects is challenging due to imprecision in sensing and actuation, which makes it difficult to estimate object shape, pose, material properties, and mass. Empirical results suggest that deep neural networks trained on large datasets of human grasp labels [96] or physical grasp outcomes [131] can be used to plan grasps that are successful across a wide variety of objects directly from images or point clouds, similar to generalization results in computer vision [88]. However, data collection requires either tedious human labeling [77] or months of execution time on a physical system [98].

An alternative approach is to plan grasps using physics-based analyses such as caging [144], wrench space analysis [136], robust wrench space analysis [178], or simulation [77], which can be rapidly computed using Cloud Computing [81]. However, these methods assume a separate perception system that estimates state perfectly [136] or according to known Gaussian distributions [104]. This is prone to errors [5], may not generalize well to new objects, and can be slow to match point clouds to known models during execution [50].

In this chapter, we build upon the state and reward model of Dex-Net 1.0 to develop an end-to-end generative model for training datasets of synthetic point clouds, grasps, and analytic rewards using analytic models of robust grasping and image formation [57, 111]. We learn a grasp quality function to predict the value of analytic grasp metrics directly from depth images by training a deep Convolutional Neural Network (CNN) on a massive synthetic training dataset, building upon recent research on classifying force closure grasps [129, 152] and the outcomes of dynamic grasping simulations [70, 77, 175].

The contributions of this chapter are:

1. The Dexterity Network (Dex-Net) 2.0, a dataset associating 6.7 million point clouds and analytic grasp quality metrics with parallel-jaw grasps planned using robust quasi-static GWS analysis on a dataset of 1,500 3D object models.

2. A Grasp Quality Convolutional Neural Network (GQ-CNN) model trained to classify robust grasps in depth images using expected epsilon quality as supervision, where each grasp is specified as a planar pose and depth relative to a camera.

3. A grasp planning method that samples antipodal grasp candidates and ranks them with a GQ-CNN.

In over 1,000 physical trials of grasping single objects on a tabletop with an ABB YuMi robot, we compare Dex-Net 2.0 to image-based grasp heuristics, a random forest [152], an SVM [129], and a baseline that recognizes objects, registers their 3D pose [50], and indexes Dex-Net 1.0 [104] for the most robust grasp to execute. We find that the Dex-Net 2.0 grasp planner is 3$\times$ faster than the registration-based method, 93% successful on objects seen in training (the highest of learning-based methods), and is the best performing method on novel objects, achieving 99% precision on a dataset of 40 household objects despite being trained entirely on synthetic data.

4.1 Problem Statement

We consider the problem of planning a robust planar parallel-jaw grasp for a singulated rigid object resting on a table based on point clouds from a depth camera. We learn a quality function that takes as input a candidate grasp and a depth image and outputs an estimate of robust grasp quality [81, 178], or the expected reward of a grasp under uncertainty in sensing and control.

Assumptions

We assume a parallel-jaw gripper, rigid objects singulated on a planar worksurface, and single-view (2.5D) point clouds taken with a depth camera. For generating datasets, we assume a known gripper geometry and a single overhead depth camera with known intrinsics.

4.1.1 Definitions

Fig. 4.1 illustrates our variables and the dataset generation distribution for Dex-Net 2.0. We use the following definitions:

- **States.** Let $\mathbf{x} = (\mathcal{O}, \mathcal{C})$ denote a state describing the variable properties of environment consisting of a single overhead depth camera and a single object in stable resting pose on a tabletop. The object state $\mathcal{O}$ specifies the geometry $\mathcal{M}$, pose $\mathbf{T}_o$, friction coefficient γ, and center of mass $\mathbf{z}$. The camera state $\mathcal{C}$ specifies the intrinsic parameters $\mathcal{I}$ and pose $\mathbf{T}_c$.

- **Grasp Actions.** Let $\mathbf{u} = (\mathbf{p}, \varphi) \in \mathbb{R}^3 \times \mathcal{S}^1$ denote a parallel-jaw grasp in 3D space specified by a center $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and an angle in the table plane $\varphi \in \mathcal{S}^1$.

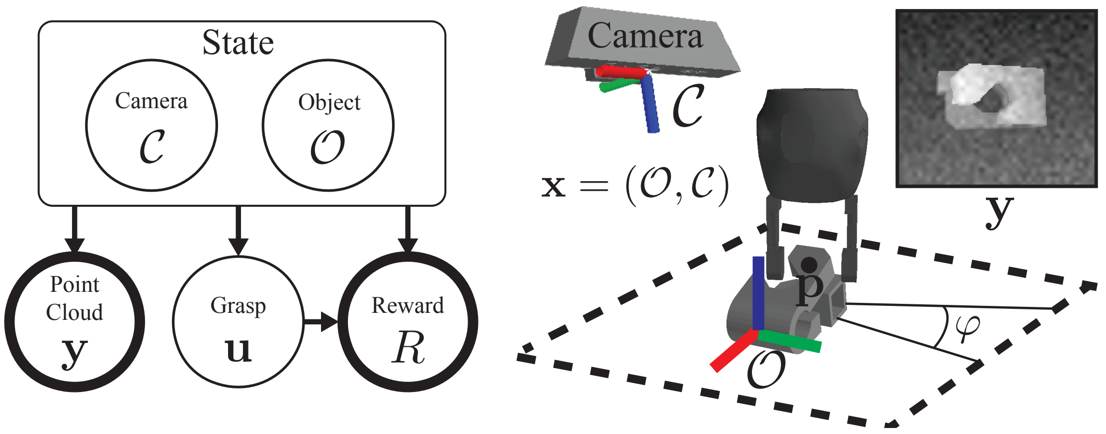


Figure 4.1: Graphical model for robust parallel-jaw grasping of objects on a table surface based on point clouds. Object shapes $\mathcal{O}$ are uniformly distributed over a discrete set of object models and object poses $\mathbf{T}_o$ are distributed over the object's stable poses and a bounded region of a planar surface. Grasps $\mathbf{u} = (\mathbf{p}, \varphi)$ are sampled uniformly from the object surface using antipodality constraints. Given a coefficient of friction γ, we evaluate an analytic reward metric R for a grasp on an object. A synthetic 2.5D point cloud $\mathbf{y}$ is generated from 3D meshes based on the camera $\mathcal{C}$ in pose $\mathbf{T}_c$ and is corrupted with multiplicative and Gaussian Process noise.

- **Point Clouds.** Let $\mathbf{y} = \mathbb{R}_+^{H \times W}$ be a 2.5D point cloud represented as a depth image with height H and width W taken by a camera with known intrinsics [57].
- **Reward Distribution.** Let $R(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ be a binary-valued grasp reward metric, such as force closure or physical lifting. Let $q(R \mid \mathbf{x}, \mathbf{u})$ model probabilistic grasp outcomes due to a distribution over contact locations resulting from control imprecision.

Let $q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ be the dataset generation environment defining a distribution on rewards, states, and point clouds modeling imprecision in sensing and control. For example, q could be defined by noisy sensor readings of a known set of industrial parts coming down a conveyor belt in arbitrary poses. Let the *robust grasp quality* of a grasp given an observation [14, 178] be the expected value of the metric, or probability of success under uncertainty in sensing and control: $Q(\mathbf{y}, \mathbf{u}) = \mathbb{E}\left[R \mid \mathbf{y}, \mathbf{u},\right]$.

Details of Graphical Model

Our graphical model is illustrated in Fig. 4.1 and models $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ as the product of a state distribution $q(\mathbf{x})$, an observation model $q(\mathbf{y}|\mathbf{x})$, a grasp candidate model $q(\mathbf{u}|\mathbf{x})$, and a reward distribution based on analytic grasp metrics $q(R|\mathbf{x}, \mathbf{u})$.

We model the state distribution as

$$q(\mathbf{x}) = q(\gamma)q(\mathcal{M})q(\mathbf{T}_o|\mathcal{M})q(\mathbf{T}_c)$$

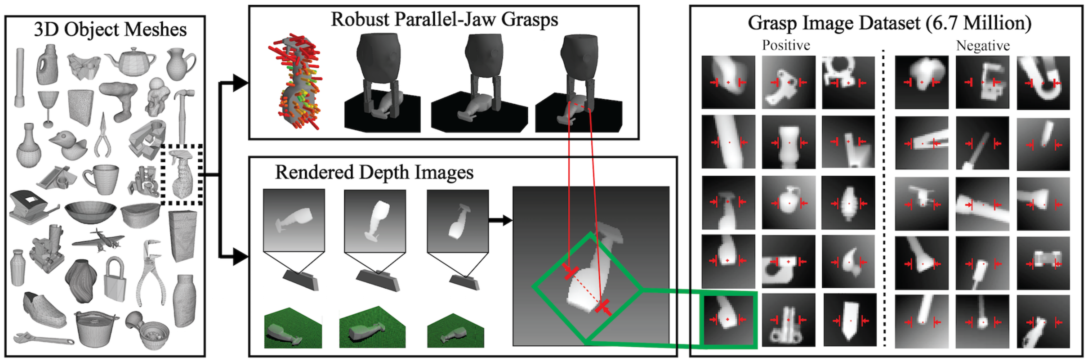


Figure 4.2: Dex-Net 2.0 pipeline for training dataset generation. (Left) The database contains 1,500 3D object mesh models. (Top) For each object, we sample hundreds of parallel-jaw grasps to cover the surface and evaluate robust analytic grasp metrics using sampling. For each stable pose of the object we associate a set of grasps that are perpendicular to the table and collision-free for a given gripper model. (Bottom) We also render point clouds of each object in each stable pose, with the planar object pose and camera pose sampled uniformly at random. Every grasp for a given stable pose is associated with a pixel location and orientation in the rendered image. (Right) Each image is rotated, translated, cropped, and scaled to align the grasp pixel location with the image center and the grasp axis with the middle row of the image, creating a 32×32 grasp image. The full dataset contains over 6.7 million grasp images.

Distribution	Description
$q(\gamma)$	truncated Gaussian distribution over friction coefficients
$q(\mathcal{O})$	discrete uniform distribution over 3D object geometries (triangular meshes)
$q(\mathbf{T}_o \vert \mathcal{M})$	continuous uniform distribution over the discrete set of object stable poses and planar poses on the table surface
$q(\mathbf{T}_c)$	continuous uniform distribution over spherical coordinates for radial bounds $[r_\ell, r_u]$ and polar angle in $[0, \delta]$

Table 4.1: Details of the distributions used in the Dex-Net 2.0 graphical model for generating the Dex-Net training dataset.

where the distributions are detailed in Table 4.1. The grasp candidate model $q(\mathbf{u} \mid \mathbf{x})$ is a uniform distribution over pairs of antipodal contact points on the object surface that form a grasp axis parallel to the table plane. The observation model is $\mathbf{y} = \alpha\hat{\mathbf{y}} + \epsilon$ where $\hat{\mathbf{y}}$ is a rendered depth image for a given object in a given pose, α is a Gamma random variable modeling depth-proportional noise, and ϵ is zero-mean Gaussian Process noise over pixel coordinates with bandwidth ℓ and measurement noise σ modeling additive noise [111]. The grasp candidate model samples antipodal grasps [20] from the object mesh uniformly at random. We model grasp reward as:

$$R(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 & E_Q > \delta \text{ and } collfree(\mathbf{x}, \mathbf{u}) \\ 0 & \quad\quad otherwise \end{cases}$$

where E_Q is the robust epsilon quality defined in [152], a variant of the pose error robust metric [178] that includes uncertainty in friction and gripper pose, and $collfree(\mathbf{x}, \mathbf{u})$ indicates that the gripper does not collide with the object or table.

4.1.2 Objective

Our goal is to learn a robustness function $Q_{\theta^*}(\mathbf{y}, \mathbf{u}) \in [0, 1]$ over many possible grasps, objects, and images that classifies grasps according to the binary success metric:

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \ \mathbb{E}_{q(R,\mathbf{x},\mathbf{y},\mathbf{u})} \left[\mathcal{L}(R, Q_\theta(\mathbf{y}, \mathbf{u}))\right] \tag{4.1.1}$$

where $\mathcal{L}$ is the cross-entropy loss function and Θ defines the parameters of the Grasp Quality Convolutional Network (GQ-CNN) described in Section 4.2.3. The estimated robustness function can be used in a grasping policy that maximizes Q_{θ^*} over a set of candidate grasps: $\pi_\theta(\mathbf{y}) = \operatorname{argmax}_{\mathbf{u} \in \mathcal{U}} Q_\theta(\mathbf{y}, \mathbf{u})$, where $\mathcal{U}$ specifies constraints on the set of available grasps, such as collisions or kinematic feasibility. Learning Q rather than directly learning the policy allows us to enforce task-specific constraints without having to update the learned model.

4.2 Learning a Grasp Quality Function

Solving for the grasp robustness function in objective 7.1.1 is challenging for several reasons. First, we may need a huge number of samples to approximate the expectation over a large number of possible objects. We address this by generating Dex-Net 2.0, a training dataset of 6.7 million synthetic point clouds, parallel-jaw grasps, and robust analytic grasp metrics across 1,500 3D models sampled from the graphical model in Fig. 4.1. Second, the relationship between point clouds, grasps, and metrics over a large dataset of objects may be complex and difficult to learn with linear or kernelized models. Consequently, we develop a Grasp Quality Convolutional Neural Network (GQ-CNN) model that classifies robust grasp poses in depth images and train the model on data from Dex-Net 2.0.

4.2.1 Supervised Learning

We estimate Q_{θ^*} using a sample approximation [44] of the objective in Equation 7.1.1 using i.i.d samples $(R_1, \mathbf{x}_1, \mathbf{y}_1, \mathbf{u}_1), ..., (R_N, \mathbf{x}_N, \mathbf{y}_N, \mathbf{u}_N) \sim q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ from our generative graphical model for images, grasps, and rewards:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)).$$

4.2.2 Dataset Generation

Dex-Net 2.0 contains 6.7 million datapoints generated using the pipeline of Fig. 4.2.

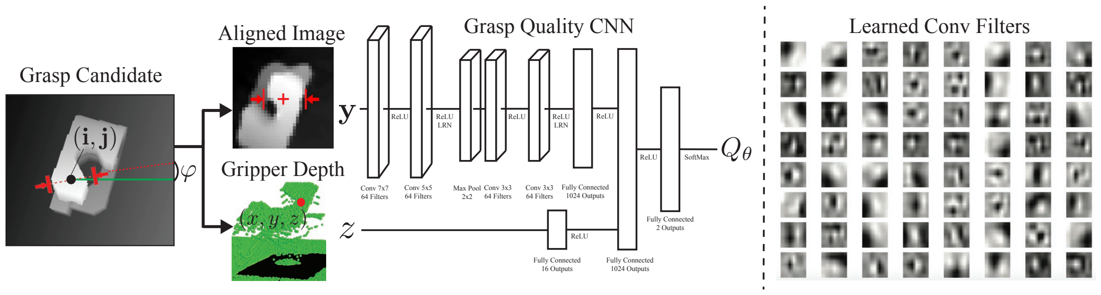


Figure 4.3: Architecture of the Grasp Quality Convolutional Neural Network (GQ-CNN). (Left) Planar grasp candidates $\mathbf{u} = (i, j, \varphi, z)$ are generated from a depth image and transformed to align the image with the grasp center pixel (i, j) and orientation φ. The architecture contains four convolutional layers in pairs of two separated by ReLU nonlinearities followed by 3 fully connected layers and a separate input layer for the z, the distance of the gripper from the camera. The use of convolutional layers was motivated by the relevance of depth edges as features for learning in previous research [10, 96, 104] and the use of ReLUs was motivated by image classification results [88]. The network estimates the probability of grasp success (robustness) $Q_\theta \in [0, 1]$, which can be used to rank grasp candidates. (Right) The first layer of convolutional filters learned by the GQ-CNN on Dex-Net 2.0. The filters appear to compute oriented image gradients at various scales, which may be useful for inferring contact normals and collisions between the gripper and object.

3D Models. The dataset contains a subset of 1,500 mesh models from Dex-Net 1.0: 1,371 synthetic models from 3DNet [180] and 129 laser scans from the KIT object database [78]. Each mesh is aligned to a standard frame of reference using the principal axes, rescaled to fit within a gripper width of $5.0cm$ (the opening width of an ABB YuMi gripper), and assigned a mass of $1.0kg$ centered in the object bounding box since some meshes are non-closed. For each object we also compute a set of stable poses [49] and store all stable poses with probability of occurrence above a threshold.

Parallel-Jaw Grasps. Each object is labeled with a set of up to 100 parallel-jaw grasps. The grasps are sampled using the rejection sampling method for antipodal point pairs developed in Dex-Net 1.0 [104] with constraints to ensure coverage of the object surface [109]. For each grasp we evaluate the expected epsilon quality E_Q [134] under object pose, gripper pose, and friction coefficient uncertainty using Monte-Carlo sampling [152],

Rendered Point Clouds. Every object is also paired with a set of 2.5D point clouds (depth images) for each object stable pose, with camera poses and planar object poses sampled according to the graphical model described in Section 4.1.1. Images are rendered using a pinhole camera model and perspective projection with known camera intrinsics, and each rendered image is centered on the object of interest using pixel transformations. Noise is added to the images during training as described in Section 4.2.3.

4.2.3 Grasp Quality Convolutional Neural Network

Architecture

The Grasp Quality Convolutional Neural Network (GQ-CNN) architecture, illustrated in Fig. 4.3 and detailed in the caption, defines the set of parameters Θ used to represent the grasp robustness function Q_θ. The GQ-CNN takes as input the gripper depth from the camera z and a depth image centered on the grasp center pixel $\mathbf{v} = (i, j)$ and aligned with the grasp axis orientation φ. The image-gripper alignment removes the need to learn rotational invariances that can be modeled by known, computationally-efficient image transformations (similar to spatial transformer networks [67]) and allows the network to evaluate any grasp orientation in the image rather than a predefined discrete set as in [70, 131]. Following standard preprocessing conventions, we normalize the input data by subtracting the mean and dividing by the standard deviation of the training data and then pass the image and gripper depth through the network to estimate grasp robustness. The GQ-CNN has approximately 18 million parameters.

Training Dataset

GQ-CNN training datasets are generated by associating grasps with a pixel $\mathbf{v}$, orientation φ, and depth z relative to rendered depth images as illustrated in Fig. 4.2. We compute these parameters by transforming grasps into the camera frame of reference using the camera pose T_c and projecting the 3D grasp position and orientation onto the imaging plane of the camera [57]. We then transform all pairs of images and grasp configurations to a single image centered on $\mathbf{v}$ and oriented along φ (see the left panel of Fig. 4.3 for an illustration). The Dex-Net 2.0 training dataset contains 6.7 million datapoints and approximately 21.2% positive examples for the thresholded robust epsilon quality with threshold $\delta = 0.002$ [77] and a custom YuMi gripper.

Optimization

We optimize the parameters of the GQ-CNN using backpropagation with stochastic gradient descent and momentum [88]. We initialize the weights of the model by sampling from a zero mean Gaussian with variance $\frac{2}{n_i}$, where n_i is the number of inputs to the i-th network layer [59]. To augment the dataset, we reflect the image about its vertical and horizontal axes and rotate each image by 180° since these lead to equivalent grasps. We also adaptively sample image noise from our noise model (see Section 4.1.1) before computing the batch gradient for new samples during training to model imaging noise without explicitly storing multiple versions of each image. To speed up noise sampling we approximate the Gaussian Process noise by upsampling an array of uncorrelated zero-mean Gaussian noise using bilinear interpolation. We set hyperparameters based on the performance on a randomize synthetic validation set as described in Section 4.4.3.

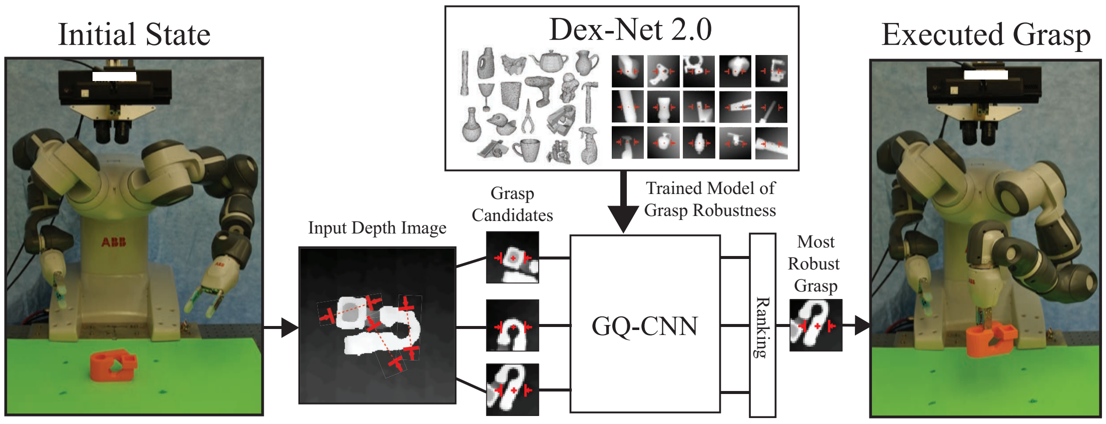


Figure 4.4: Dex-Net 2.0 Architecture. (Center) The Grasp Quality Convolutional Neural Network (GQ-CNN) is trained offline to predict the robustness candidate grasps from depth images using a dataset of 6.7 million synthetic point clouds, grasps, and associated robust grasp metrics computed with Dex-Net 1.0. (Left) When an object is presented to the robot, a depth camera returns a 3D point cloud, where pairs of antipodal points identify a set of several hundred grasp candidates. (Right) The GQ-CNN rapidly determines the most robust grasp candidate, which is executed with the ABB YuMi robot.

4.3 Grasp Planning

The Dex-Net 2.0 grasp planner uses the robust grasping policy $\pi_\theta(\mathbf{y}) = \text{argmax}_{\mathbf{u}\in\mathcal{C}(\mathbf{y})} Q_\theta(\mathbf{u}, \mathbf{y})$ illustrated in Fig. 4.4. The set $\mathcal{C}(\mathbf{y})$ is a discrete set of antipodal candidate grasps [20] sampled uniformly at random in image space for surface normals defined by the depth image gradients. Each grasp candidate is evaluated by the GQ-CNN, and the most robust grasp that is (a) kinematically reachable and (b) not in collision with the table is executed. We explore two implementations of the robust grasping policy: (1) sampling a large, fixed set of antipodal grasps and choosing the most robust one and (2) optimizing for the most robust grasp using derivative free optimization.

4.3.1 Antipodal Grasp Candidate Generation

The antipodal grasp sampling method used in the paper is designed to sample antipodal grasps specified as a planar pose, angle, and height with respect to a table. The algorithm is detailed in Algorithm 1. We first threshold the depth image to find areas of high gradient. Then, we use rejection sampling over pairs of pixels to generate a set of candidate antipodal grasps, incrementally increasing the friction coefficient until a desired number of grasps is reached in case the desired number cannot be achieved with a smaller friction coefficient. We convert antipodal grasps in image space to 3D by assigning discretizing the gripper height between the height of the grasp center pixel relative and the height of the table surface itself.

This grasp sampling method is used for all image based grasp planners in the paper. We used $M = 1000$, K set to the intrinsics of a Primesense Carmine 1.08, T_c determined by chessboard registration, $g = 0.0025m$, $\mu_\ell = 0.4$, $\delta_\mu = 0.2$, $N = 1000$, and $\delta_h = 0.01m$.

```
1  **Input:** Depth image $\mathbf{y}$, Number of grasps $M$, Camera Intrinsics Matrix $K$, Camera pose $T_c$,
     Depth gradient threshold $g$, Min friction coef $\gamma_\ell$, Friction coef increment $\delta_\gamma$, Max samples
     per friction coef $N$, Gripper height resolution $\delta_h$
   **Result:** $\mathcal{G}$, set of candidate grasps
   // Compute depth edges
2  $G_x = \nabla_x \mathbf{y}, G_y = \nabla_y \mathbf{y}$;
3  $\mathcal{E} = \{\mathbf{u} \in \mathbb{R}^2 : G_x(\mathbf{u})^2 + G_y(\mathbf{u})^2 > g\}$;
   // Find antipodal pairs
4  $\mathcal{G} = \{\}$, $\gamma = \gamma_\ell$, $i = 0$, $j = 0$;
5  while $|\mathcal{G}| < M$ and $\gamma <= 1.0$ do
6      $\mathbf{u}, \mathbf{v} =$UniformRandom$(\mathcal{E}, 2)$;
7      if Antipodal($\mathbf{u}, \mathbf{v}, \mu$) then
           // Compute point in world coordinates
8          $\mathbf{c} = 0.5 * (\mathbf{u} + \mathbf{v})$;
9          $\mathbf{p}_c =$Deproject$(K, \mathbf{y}, \mathbf{c})$;
10         $\mathbf{p} = T_c * \mathbf{p}_c$;
11         $h = \mathbf{p}.z$;
           // Add all heights
12         while $h > 0$ do
13             $\mathcal{G} = \mathcal{G} \cup \{\mathbf{g}(\mathbf{u}, \mathbf{v}, h)\}$;
14             $h = h - \delta_h$;
15         end
16     end
17     $i = i + 1$, $j = j + 1$;
       // Update friction coef
18     if $j >= N$ then
19         $\gamma = \gamma + \delta_\gamma$;
20         $j = 0$;
21     end
22 end
23 return $\mathcal{G}$;
```

Algorithm 1: Antipodal Grasp Sampling from a Depth Image

4.3.2 Derivative Free Optimization

One problem with choosing a grasp from a fixed set of candidates is that the set of candidates may all have a low probability of success. This can be difficult when an object can only be grasped in a small set of precise configurations, such as the example in Fig. 4.5. Some of these failures can be seen in the right panel of the failure modes figure in the original paper.

In our second generalization study we addressed this problem using the cross entropy method (CEM) [98, 145], a form of derivative-free optimization, to optimize for the most robust grasp by iteratively resampling grasps from a learned distribution over robust grasps and updating the distribution. The method, illustrated in Algorithm 2, models the distribution on promising grasps using a Gaussian Mixture Model (GMM) and seeds the initial set of grasps with antipodal point pairs using Algorithm 1 with no iterative friction coefficient updates. The algorithm takes as input the number of CEM iterations m, the number of initial grasps to sample n, the number of grasps to resample from the model c, the number of GMM mixture components k, a friction coefficient μ, and elite percentage γ, and the GQ-CNN Q_θ, and returns an estimate of the most robust grasp $\mathbf{u}$. In our generalization experiment we used $m = 3$, $n = 100$, $c = 50$, $\mu = 0.8$, $k = 3$, and $\gamma = 25\%$. The qualitative performance of our method on several examples from our experiments is illustrated in Fig. 4.6.

```
1 Input: Num rounds $m$, Num initial samples $n$, Num CEM samples $c$, Num GMM mixture
    $k$, Friction coef $\gamma$, Elite percentage $\nu$, Robustness function $Q_\theta$
  Result: $\mathbf{u}$, most robust grasp
2 $\mathcal{U} \leftarrow$uniform set of $n$ antipodal grasps with friction coef $\gamma$;
3 for $i$ = 1, ..., $m$ do
4     $\mathcal{E} \leftarrow$top $\nu-$percentile of grasps ranked by $Q_\theta$;
5     $M \leftarrow$GMM fit to $\mathcal{E}$ with $k$ mixtures;
6     $G \leftarrow c$ iid samples from $M$;
7 end
8 return $\operatorname{argmax}_{\mathbf{u}\in\mathcal{U}} Q_\theta(\mathbf{u}, \mathbf{y})$;
```

Algorithm 2: Robust Grasping Policy using the Cross Entropy Method on a Learned GQ-CNN

4.4 Experiments

We evaluated classification performance on both real and synthetic data and performed extensive physical evaluations on an ABB YuMi with custom silicone gripper tips designed by Guo et al. [53] to benchmark the performance of grasping a single object. All experiments ran on a Desktop running Ubuntu 16.04 with a 3.4 GHz Intel Core i7-6700 Quad-Core CPU and an NVIDIA GeForce 980, and we used an NVIDIA GeForce GTX 1080 for training large models.

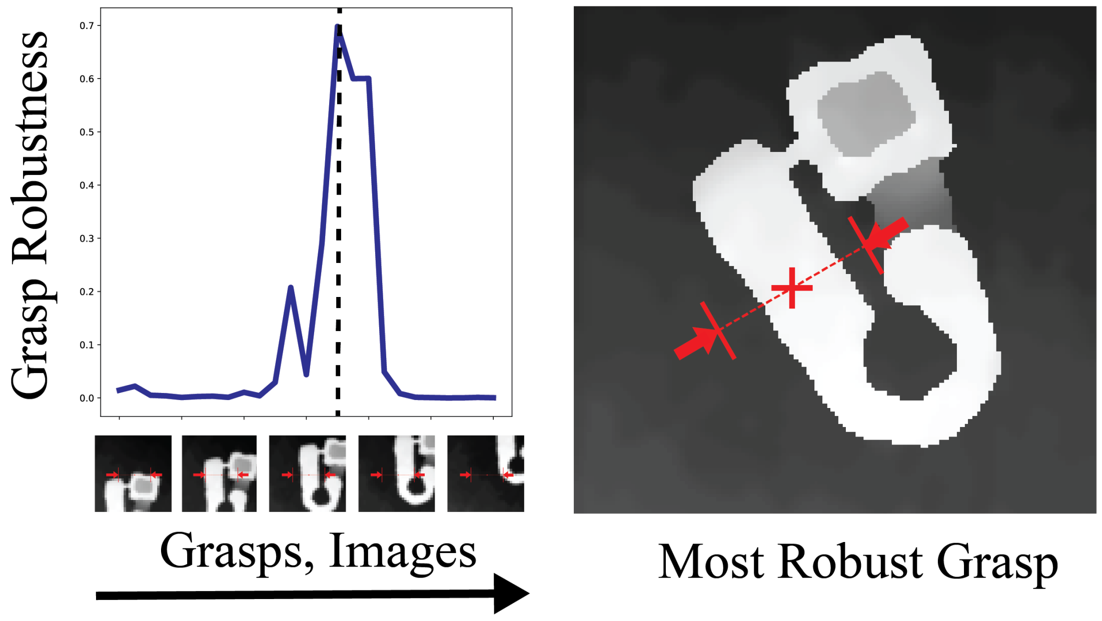


Figure 4.5: Grasp robustness predicted by a Grasp Quality Convolutional Neural Network (GQ-CNN) trained with Dex-Net 2.0 over the space of depth images and grasps for a single point cloud collected with a Primesense Carmine. (Left) As the center of the gripper moves from the top to the bottom of the image the GQ-CNN prediction stays near zero and spikes on the most robust grasp (Right), for which the gripper fits into a small opening on the object surface. This suggests that the GQ-CNN has learned a detailed representation of the collision space between the object and gripper. Furthermore, the sharp spike suggests that it may be difficult to plan robust grasps by randomly sampling grasps in image space. We consider planning the most robust grasp using the cross-entropy method on the GQ-CNN response.

4.4.1 Physical Benchmark Description

We created a benchmark for grasping single objects on a tabletop to compare grasp planning methods. The setup is illustrated in Fig. 4.7 and the experimental procedure is described in the caption and shown in the supplemental video[1]. Each grasp planner received as input a color image, depth image, bounding box containing the object, and camera intrinsics, and output a target grasping pose for the gripper. A human operator was required to reset the object in the workspace on each trial, and therefore blinded operators from which grasp planning method was being tested in order to remove bias.

We compared performance on this benchmark with the following metrics:

1. **Success Rate:** The percentage of grasps that were able to lift, transport, and hold a desired object after shaking.

2. **Precision:** The success rate on grasps that are have an estimated robustness higher than 50%. This measures performance when the robot can decide not to grasp an

[1]https://youtu.be/9eqAxk95I3Y

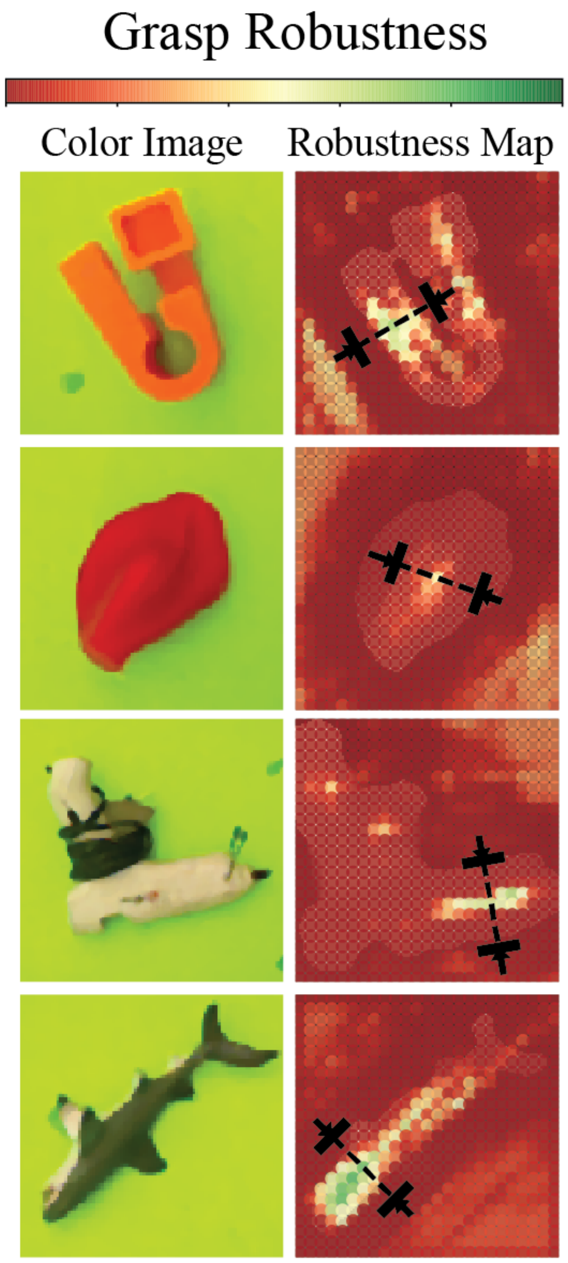


Figure 4.6: Example input color images and maps of the grasp robust estimated by the GQ-CNN over grasp centers for a constant grasp axis angle in image space and height above the table, with the grasp planned by our CEM-based robust grasping policy shown in black. CEM is able to find precise robust grasping locations encoded by the GQ-CNN that are very close to the global maximum for the given grasp axis and height. The GQ-CNN also appears to assign non-zero robustness to several grasps that completely miss the object. This is likely because no such grasps are in the training set, and future work could augment the training dataset to avoid these grasps.

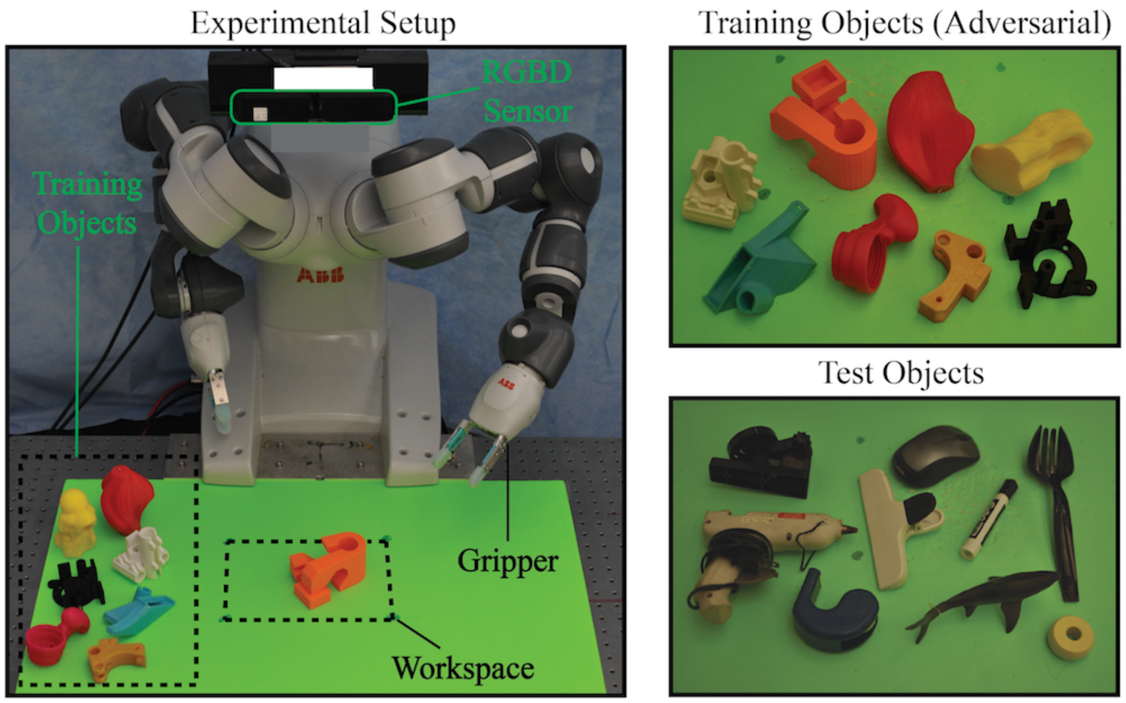


Figure 4.7: Experimental setup for benchmarking grasping with the ABB YuMi. (Left) In each trial a human operator sampled an object pose by shaking the object in a box and placing it upside down in the workspace. Then RGB-D image was taken with a Primsense Carmine 1.08, the image was processed using inpainting [70], and the object was segmented using color background subtraction. The grasp planner under evaluation then planned a gripper pose and the YuMi executed the grasp. Grasps were considered successful if the gripper held the object after lifting, transporting, and shaking the object. (Top-Right) The training set of 8 objects with adversarial geometric features such as smooth curved surfaces and narrow openings for grasping known objects. (Bottom-Right) The test set of 10 household objects not seen during training. The dataset was selected to test performance on challenging objects of varying material, geometry, and surface reflectance properties.

object, which could be useful when the robot has other actions (e.g. pushing) available.

3. **Robust Grasp Rate:** The percentage of planned grasps with an estimated robustness higher than 50%.

4. **Planning Time:** The time in seconds between receiving an image and returning a planned grasp.

4.4.2 Datasets

Fig. 4.7 illustrates the physical object datasets used in the benchmark:

1. **Train:** A validation set of 8 3D-printed objects with adversarial geometric features such as smooth, curved surfaces. This is used to set model parameters and to evaluate performance on known objects.

2. **Test:** A set of 10 household objects similar to models in Dex-Net 2.0 with various material, geometric, and specular properties. This is used to evaluate generalization to unknown objects.

We chose objects based on geometric features under three constraints: (a) small enough to fit within the workspace, (b) weight less than $0.25kg$, the payload of the YuMi, and (c) height from the table greater than $1.0cm$ due to a limitation of the silicone gripper fingertips.

We used four different GQ-CNN training datasets to study the effect on performance, each with a 80-20 image-wise training and validation split:

1. **Adv-Synth:** Synthetic images and grasps for the adversarial objects in Train (189k datapoints).

2. **Emp:** Outcomes of executing random antipodal grasps with random gripper height and friction coefficient of $\mu = 0.5$ in 50 physical trials per object in Train (400 datapoints).

3. **Dex-Net-Small:** A subset of data from 150 models sampled uniformly from Dex-Net 2.0 (670k datapoints).

4. **Dex-Net-Large:** Data from all 1500 models in Dex-Net 2.0 (6.7m datapoints).

4.4.3 Grasp Planning Methods Used for Comparison

We compared a number of grasp planning methods on simulated and real data. We tuned the parameters of each method based on synthetic classification performance and physical performance on the training objects. All methods other than point cloud registration used the antipodal grasp sampling method described in Section 4.3 with the same set of parameters to generate candidate grasps, and each planner executes the highest-ranked grasp according to the method. Additional details on the methods and their parameters can be found in the supplemental file.

Image-based Grasp Quality Metrics (IGQ). We sampled a set of force closure grasp candidates by finding antipodal points on the object boundary [20] using edge detection and ranked grasps by the distance from the center of the jaws to the centroid of the object segmentation mask. We set the gripper depth using a fixed offset from the depth of the grasp center pixel.

Point-Cloud Registration (REG). We also compared with grasp planning based on point cloud registration, a state-of-the-art method for using precomputed grasps [50, 60]. We first coarsely estimated the object instance and pose based on the top 3 most similar synthetic images from Dex-Net 2.0, where similarity is measured as distance between AlexNet conv5 features [50, 104]. After coarse matching, we fine-tuned the pose of the object in the table plane using Iterated Closest Point [54, 81] with a point-to-plane cost. Finally, we retrieved the most robust gripper pose from Dex-Net 2.0 for the estimated object. We constrained gripper poses to the same four degrees of freedom as the learning-based methods. The system had a median translational error of $4.5mm$ a median rotational error of 3.5° in the table plane for known objects.

Alternative Machine Learning Models (ML). We also compared the performance of a Random Forest with 200 trees of depth up to 10 (ML-RF) motivated by the results of [152] and a Support Vector Machine with the RBF kernel and a regularization parameter of 1 (ML-SVM) motivated by the results of [10, 150, 129]. For the RF we used the raw transformed images and gripper depths normalized by the mean and standard deviation across all pixels as features. For the SVM we used a Histogram of Oriented Gradients (HOG) [28] feature representation. Both methods were trained using scikit-learn on the Adv-Synth dataset.

Grasp Quality CNNs (GQ). We trained the GQ-CNN (abbrev. GQ) using the thresholded robust epsilon metric with $\delta = 0.002$ [77] for 5 epochs on Dex-Net-Large (all of Dex-Net 2.0) using Gaussian process image noise with standard deviation $\sigma = 0.005$. We used TensorFlow [1] with a batch size of 128, a momentum term of 0.9, and an exponentially decaying learning rate with step size 0.95. Training took approximately 48 hours on an NVIDIA GeForce 1080. The first layer of 7×7 convolution filters are shown in the right panel of Fig. 4.3, and suggest that the network learned fine-grained vertical edge detectors and coarse oriented gradients. We hypothesize that vertical filters help to detect antipodal contact normals and the coarse oriented gradients estimate collisions.

To benchmark the architecture outside of our datasets, we trained on the Cornell Grasping Dataset [96] (containing 8,019 examples) and achieved a 93.0% recognition rate using grayscale images and an $80 - 20$ imagewise training-validation split compared to 93.7% on RGB-D images in the original paper. We also trained several variants to evaluate sensitivity to several parameters:

Dataset Size. We trained a GQ-CNN on Dex-Net-Small for 15 epochs (GQ-S).

Amount of Pre-training We trained three GQ-CNNs on the synthetic dataset of adversarial training objects (Adv-Synth) to study the effect of pre-training with Dex-Net for a new, known set of objects. The model GQ-Adv was trained on only Adv-Synth for 25 epochs. The models GQ-L-Adv and GQ-S-Adv were initialized with the weights of GQ and GQ-S, respectively, and fine-tuned for 5 epochs on Adv-Synth.

Success Metric. We trained a GQ-CNN using probability of force closure thresholded at 25% (GQ-Adv-FC), which is a robust version of the antipodality metric of [129], and and labels for 400 random grasp attempts on the Train objects using a physical robot [98, 131] (GQ-Emp).

Noise Levels. We trained a GQ-CNN with zero noise $\sigma = 0$ (GQ-Adv-LowU) and high noise with $\sigma = 0.01$ (GQ-Adv-HighU).

4.4.4 Classification of Synthetic Data

The GQ-CNN trained on all of Dex-Net 2.0 had an accuracy of 85.7% on a held out validation set of approximately 1.3 million datapoints. Due to the memory and time requirements of training SVMs, we compared synthetic classification performance across methods on the smaller Adv-Synth dataset. Fig. 4.8 shows the receiver operating characteristic curve comparing the performance of GQ-L-Adv, GQ-S-Adv, GQ-Adv, ML-SVM, and ML-RF on a held-out validation set and Table 4.2 details the classification accuracy for the various methods. The GQ-CNNs outperformed ML-RF and ML-SVM, achieving near-perfect validation accuracy.

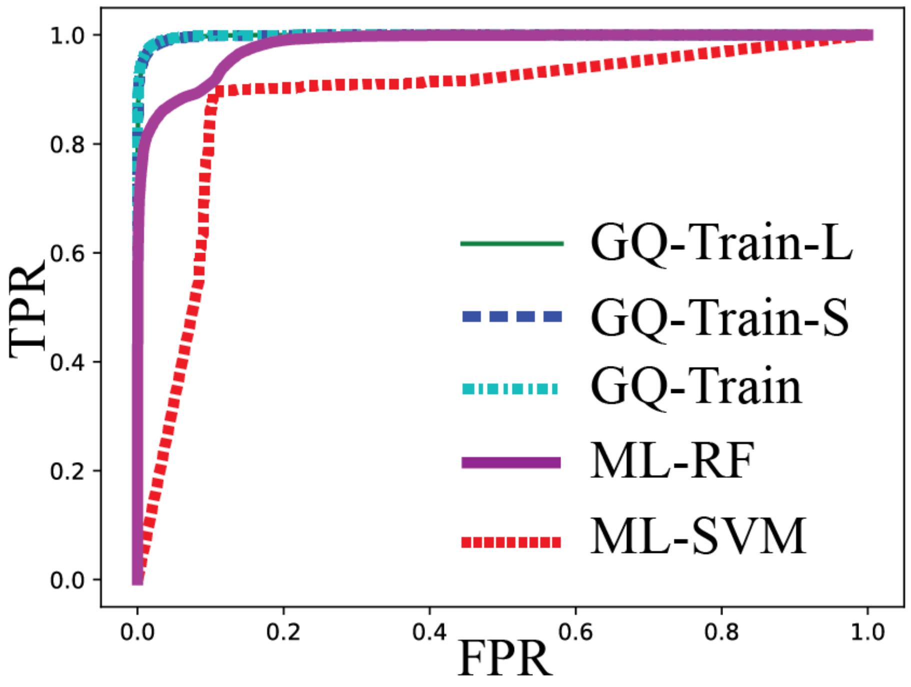


Figure 4.8: Receiver operating characteristic comparing the performance of learning models on Adv-Synth. The GQ-CNN models all perform similarly and have a significantly higher true positive rate when compared to ML-RF and ML-SVM.

	Comparisons of Methods						GQ-CNN Parameter Sensitivity					
	Random	**IGQ**	**ML-RF**	**ML-SVM**	**REG**	**GQ-L-Adv**	**GQ-S-Adv**	**GQ-Adv**	**GQ-Emp**	**GQ-Adv-FC**	**GQ-Adv-LowU**	**GQ-Adv-HighU**
Success Rate (%)	58±11	70±10	75±9	80±9	95±5	**93±6**	85±8	83±8	80±9	83±8	78±9	86±8
Precision (%)	N/A	N/A	100	100	N/A	**94**	90	91	80	89	90	92
Robust Grasp Rate (%)	N/A	N/A	5	0	N/A	43	60	44	100	89	53	64
Planning Time (sec)	N/A	1.9	0.8	0.9	2.6	**0.8**	0.9	0.8	0.8	0.7	0.8	0.9

Table 4.3: Performance of grasp planning methods on the Train dataset with 95% confidence intervals for the success rate. Each method was tested for 80 trials (10 trials per object). Details on the methods used for comparison can be found in Section 4.4.3. We see that REG (point cloud registration) has the highest success rate at 95% but the GQ-L-Adv performs comparably at 93% and is 3× faster. Performance of the GQ-CNN drops to 80% when trained on the Empirical dataset (GQ-Emp), likely due to the small number of training examples, and drops to 78% when no noise is added to the images during training (GQ-Adv-LowU).

4.4.5 Performance Comparison on Known Objects

We evaluated the performance of the grasp planning methods on known objects from Train. Each grasp planner had 80 trials (10 per object). The left half of Table 4.3 compares the performance with other grasp planning methods and the right half compares the performance of the GQ-CNN varations. We found that GQ planned grasps 3× faster than REG and achieved a high 93% success rate and 94% precision. The results also suggest that training on the full Dex-Net 2.0 dataset was necessary to achieve higher than 90% success.

	IGQ	REG	GQ-Emp	GQ-Adv	GQ-S	GQ
Success Rate (%)	60±13	52±14	68±13	74±12	72±12	**80±11**
Precision (%)	N/A	N/A	68	87	92	**100**
Robust Grasp Rate (%)	N/A	N/A	100	30	48	**58**
Planning Time (sec)	1.8	3.4	0.7	0.7	0.8	0.8

Table 4.4: Performance of grasp planning methods on our grasping benchmark with the test dataset of 10 household objects with 95% confidence intervals for the success rate. Each method was tested for 50 trials, and details on the methods used for comparison can be found in Section 4.4.3. GQ performs best in terms of success rate and precision, with 100% precision (zero false positives among 29 positive classifications). Performance decreases with smaller training datasets, but the GQ-CNN methods outperform the image-based grasp quality metrics (IGQ) and point cloud registration (REG).

4.4.6 Performance Comparison on Novel Objects

We also compared the performance of the methods on the ten novel test objects from Test to evaluate generalization to novel objects. Each method was run for 50 trials (5 per object). The parameters of each method were set based on Train object performance without knowledge of the performance on Test. Table 4.4 details the results. GQ performed best with an 80% success rate and 100% precision (zero false positives over 29 grasps classified as robust).

Model	**Accuracy (%)**
ML-SVM	89.7
ML-RF	90.5
GQ-S-Adv	97.8
GQ-L-Adv	97.8
GQ-Adv	98.1

Table 4.2: Classification accuracy of models on Adv-Synth. The GQ-CNNs have less than 2.5% test error while ML-RF and ML-SVM are closer to 10% error. Pre-training does significantly affect performance.

4.4.7 Generalization Ability of the Dex-Net 2.0 Grasp Planner

We evaluated the generalization performance of GQ in 100 grasping trials on the 40 object test set illustrated in Fig. 4.9, which contains articulated (e.g. can opener) and deformable (e.g. washcloth) objects. We used the cross entropy method (CEM) [98], which iteratively samples a set of candidate grasps and re-fits the candidate grasp distribution to the grasps with the highest predicted robustness, in order to find better maxima of the robust grasping policy. More details can be found in the supplemental file. The CEM-augmented Dex-Net 2.0 grasp planner achieved 94% success and 99% precision (68 successes out of 69 grasps classified as robust), and it took an average of 2.5s to plan grasps.

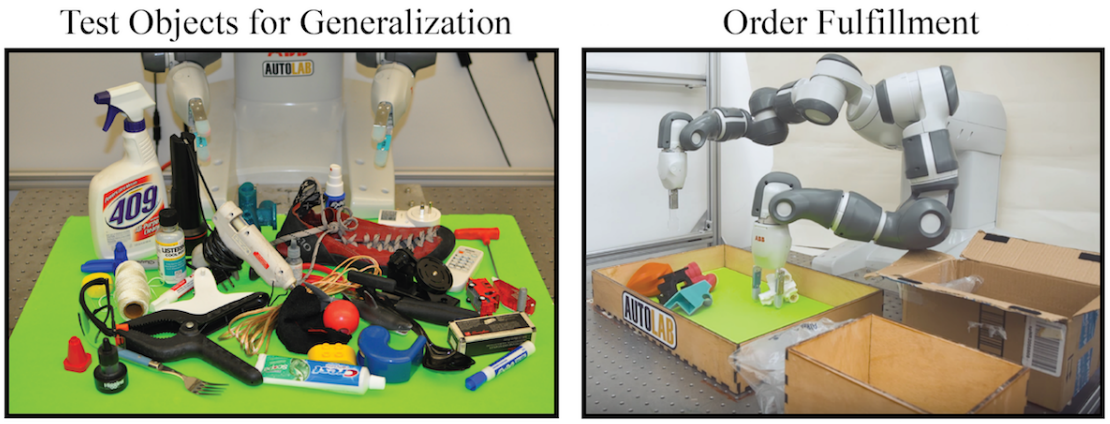


Figure 4.9: Experimental setup for evaluating the Dex-Net 2.0 in novel scenarios. (Left) The test set of 40 household objects used for evaluating the generalization performance of the Dex-Net 2.0 grasp planner. The dataset contains rigid, articulated, and deformable objects. (Right) The experimental setup for order fulfillment with the ABB YuMi. The goal is to grasp and transport three target objects to a shipping container (box on right).

4.4.8 Application: Order Fulfillment

To demonstrate the modularity of the Dex-Net 2.0 grasp planner, we used it in an order fulfillment application with the ABB YuMi. The goal was to grasp and transport a set of three target objects to a shipping box in the presence of three distractor objects when starting with the objects in a pile on a planar worksurface, illustrated in Fig. 4.9. Since the Dex-Net 2.0 grasp planner assumes singulated objects, the YuMi first separated the objects using a policy learned from human demonstrations mapping binary images to push locations [92]. When the robot detected an object with sufficient clearance from the pile, it identified the object based on color and used GQ-L-Adv to plan a robust grasp. The robot then transported the object to either the shipping box or a reject box, depending on whether or not the object was a distractor. The system successfully placed the correct objects in the box on 4 out of 5 attempts and was successful in grasping on 93% of 27 total attempts.

4.4.9 Failure Modes

Fig. 4.10 displays some common failures of the GQ-CNN grasp planner. One failure mode occured when the RGB-D sensor failed to measure thin parts of the object geometry, making these regions seem accessible. A second type of failure occured due to collisions with the object. It appears that the network was not able to fully distinguish collision-free grasps in narrow parts of the object geometry. This suggests that performance could be improved with more accurate depth sensing and using analytic methods to prune grasps in collision.

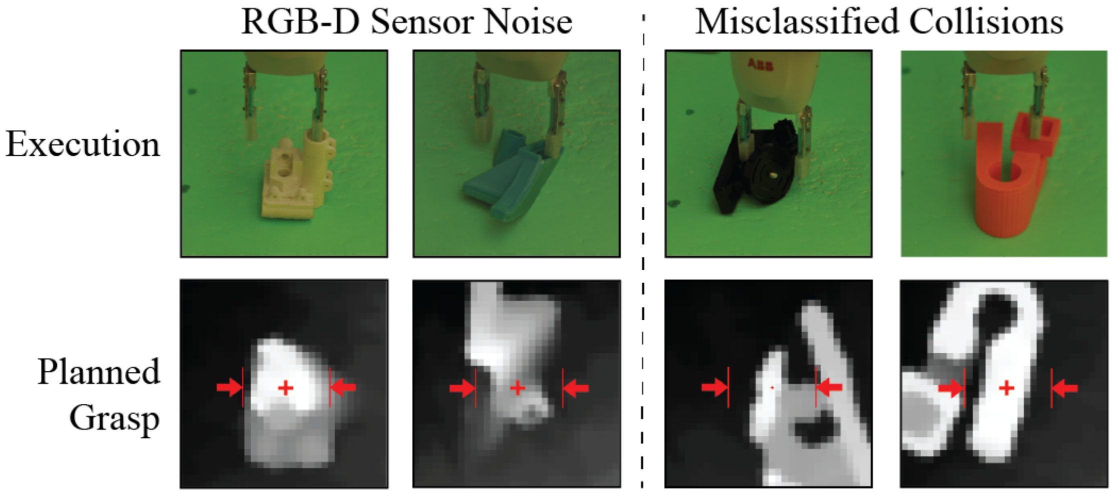


Figure 4.10: Examples of failed grasps planned using the GQ-CNN from Dex-Net 2.0. The most common failure modes were related to: (left) missing sensor data for an important part of the object geometry, such as thin parts of the object surface, and (right) collisions with the object that are misclassified as robust.

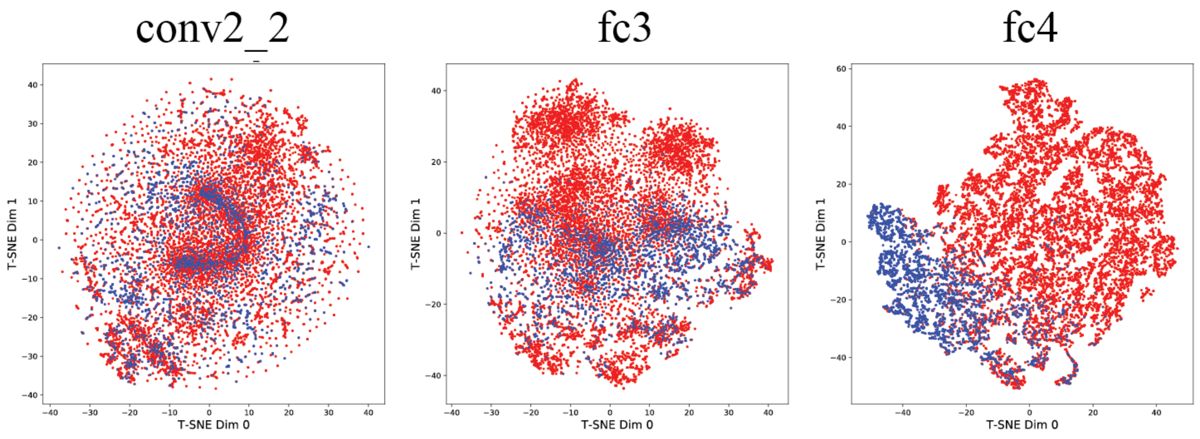


Figure 4.11: Visualization of t-SNE for the GQ-CNN on the Dex-Net 2.0 validation set illustrating the separation of positive (blue) and negative (red) examples. The network appears to start separating the positive and negative grasps and images in the fc4 layer.

4.4.10 Feature Analysis

To better understand the representation learned by the Dex-Net 2.0 GQ-CNN, we analyzed the outputs of individual network layers to varying sets of input grasps and depth images.

Fig. 4.11 displays a t-Stochastic Neighbor Embedding (t-SNE) [102] for the outputs of the conv2_2, fc3, and fc4 layers of the GQ-CNN trained on Dex-Net 2.0 for the set of synthetic validation datapoints. We see that the positive (blue) and negative (red) examples are not well-separated until the fc4 layer. This suggests that the GQ-CNN does may not discriminate

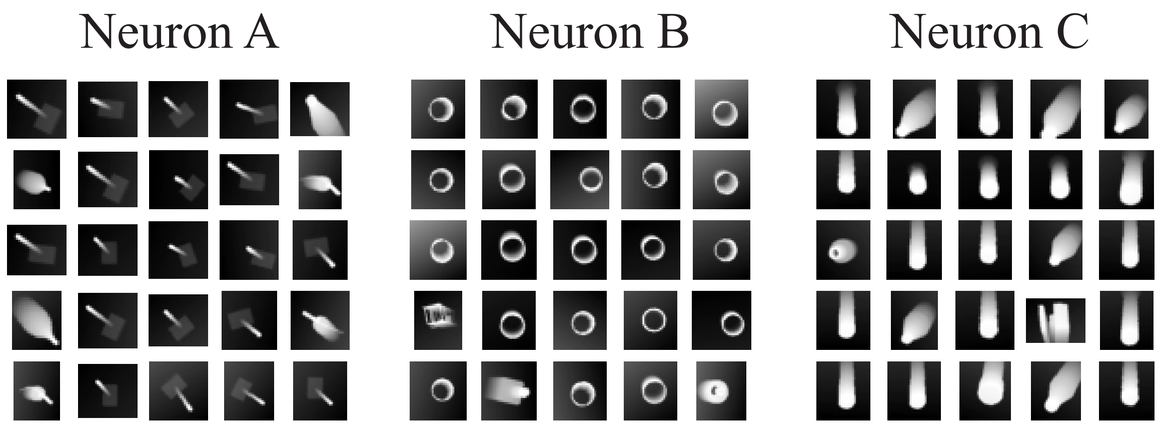


Figure 4.13: Visualization of the maximum activations from the Dex-Net 2.0 validation set for a selection of three neurons from the conv2_2 layer of GQ-CNN. The neurons appear to respond to oriented parallel lines and circular patterns.

between positive and negative examples for the majority of layers in the network.

In order to evaluate whether or not the network discriminates between simulated and real data, we computed a t-SNE for the fc4 response on a set of 200 datapoints subsampled randomly from the set of grasps and images collected in experiments and 200 datapoints from the Dex-Net 2.0 validation set. Fig. 4.12 displays the embedding. We see that there is no clear separation between the simulated and real datapoints, suggesting that the GQ-CNN has learned a representation that is robust to the differences between synthetic and real depth images. This may explain the high success rate observed when using GQ-CNN directly on a physical system with no explicit transfer learning.

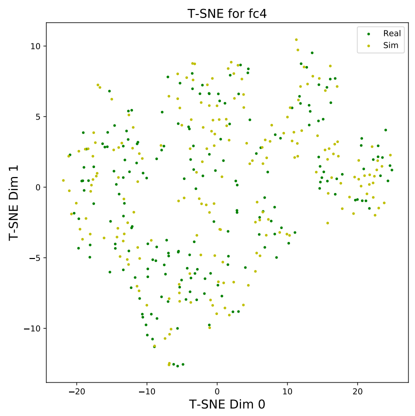


Figure 4.12: Visualization of t-SNE for the fc4 response of GQ-CNN to a set of 200 synthetic datapoints (yellow) and 200 datapoints collected from a physical robot system (green).

We also analyze the synthetic validation datapoints that maximize activations of neurons in the conv2_2 layer using the method of [47]. Fig. 4.13 displays a set of the activations for three neurons. Interestingly, some neurons appear to respond strongly to primitive shapes such as oriented bars and circles. This may be because oriented parallel vertical lines correlate well with high grasp quality due to the input image rotation and translation.

4.5 Discussion

In this chapter, we introduced a generative model for synthesizing massive training datasets of point clouds, parallel-jaw grasps, and robust analytic grasp metrics and presented Dex-Net 2.0, a dataset of 6.7 million examples generated from the model. We developed a Grasp Quality Convolutional Neural Network (GQ-CNN) architecture that predicts grasp robustness from a point cloud and trained it on Dex-Net 2.0, a dataset containing 6.7 million point clouds, parallel-jaw grasps, and robust grasp metrics. In over 1,000 physical evaluations, we found that the Dex-Net 2.0 grasp planner is as reliable and 3× faster a method based on point cloud registration, and had 99% precision on a test set of 40 novel objects.

The results of Dex-Net 2.0 suggest that is possible to train a robust grasping policy that generalizes to a wide variety of novel objects entirely on synthetic data, without any explicit sim-to-real transfer. However, the results also raise an important question: does this generalize beyond the problem setting considered in this chapter? Variations in the robot (e.g. different grippers), sensing (e.g. different modalities such as color), and object configurations (e.g. clutter) may affect performance. The subsequent chapters study a subset of these variations in order to evaluate the scope of problems for which the hybrid grasp planning method is applicable.

Chapter 5

Sequential Grasp Planning for Bin Picking

The results of the previous chapter suggest that deep learning from massive synthetic training datasets of point clouds, grasps, and analytic metrics can lift and transport a wide variety of rigid objects on a physical robot when objects are singulated (sufficiently clear from obstacles). However, objects are often in disorganized heaps in applications such as industrial bin picking, which is challenging due to sensor noise, obstructions, and occlusions that make it difficult to infer object shapes and poses from point clouds [34]. A robot must consider collisions with adjacent objects and cannot assume a finite set of stable resting poses for each object [49]. Furthermore, the task of clearing all objects from the bin is sequential: grasp attempts affect the state of objects in the bin for future grasp attempts.

Recent research suggests that it is possible to grasp a diverse set of objects from disorganized heaps using deep Convolutional Neural Networks (CNNs) trained using empirical dataset collection on a physical robot [12, 98]. However, the time cost of collecting physical data makes it difficult to collect clean and sufficiently large datasets to train different robots in different environments. An alternative is to train on synthetic datasets of grasps and point clouds labeled using geometric conditions related to grasp stability such as antipodality [129], but current methods require dense 3D point clouds from multiple viewpoints to mitigate occlusions [129, 176].

In this chapter, we extend the dataset generation model for learning to grasp a single object from a tabletop to grasping a sequence of objects from clutter in order to learn a robust policy for rapid bin picking from a single viewpoint. We formulate a discrete-time Partially Observed Markov Decision Process (POMDP) modeling bin picking as a sequence of 3D object poses in a heap with noisy point cloud observations and rewards for removing objects. Due to the difficulty of training POMDPs with continuous states and observations [153], we use imitation learning based on an algorithmic supervisor that synthesizes robust collision-free grasps using robust wrench space analysis and full knowledge of object shapes and poses in the environment [136].

This chapter makes four contributions:

1. Formulating bin picking as a Partially Observable Markov Decision Process (POMDP) modeling the process of iteratively grasping and removing objects from a heap based on point clouds. This extends the single-object non-sequential robust grasping model from Dex-Net 2.0.

2. Dex-Net 2.1: A dataset of 10,000 rollouts in an implementation of the POMDP collected using noise injection on an algorithmic robust grasping supervisor that plans robust grasps with full state knowledge.

3. A study of transfer learning to learn a bin picking policy from pre-trained weights of a GQ-CNN policy for grasping singulated objects.

4. Experiments evaluating performance of the bin picking policies on heaps of up to 20 novel objects on an ABB YuMi robot.

Experiments suggest that a greedy bin picking policy trained synthetic data from Dex-Net 2.1 can achieve up to 416 successful picks per hour with 96% average precision (very few false positives). This suggests that modeling the sequential structure may not be necessary to achieve high success rates in bin picking.

5.1 Problem Statement

We consider the problem of bin picking: clearing a heap of objects on a table by iteratively grasping a single object from the heap with a parallel-jaw gripper and transporting each object to a receptacle. The goal of this chapter is to learn a policy that takes as input point clouds from an overhead depth camera and outputs a robust grasp, or gripper pose to remove an object from the heap, along with a confidence value for the grasp.

Assumptions

The model assumes quasi-static physics, where inertial effects are negligible, to compute grasp robustness. The model also assumes a parallel-jaw gripper, rigid objects, a depth sensor with bounds on the camera intrinsic parameters and pose relative to the robot, and bounds on friction across objects and their surfaces. These assumptions are common in industrial robotics [55]. We make the additional simplifying assumption that only one object is be grasped at a time. The model also does not consider object identity when grasping.

5.1.1 Definitions

Due to the cost of learning a policy directly from data on a physical robot, we learn a policy in simulation using a model of iteratively grasping objects from a heap on an infinite planar worksurface based on models of quasi-static contact, image formation, and sensor noise. Specifically, we model the task of bin picking as a Partially Observable Markov

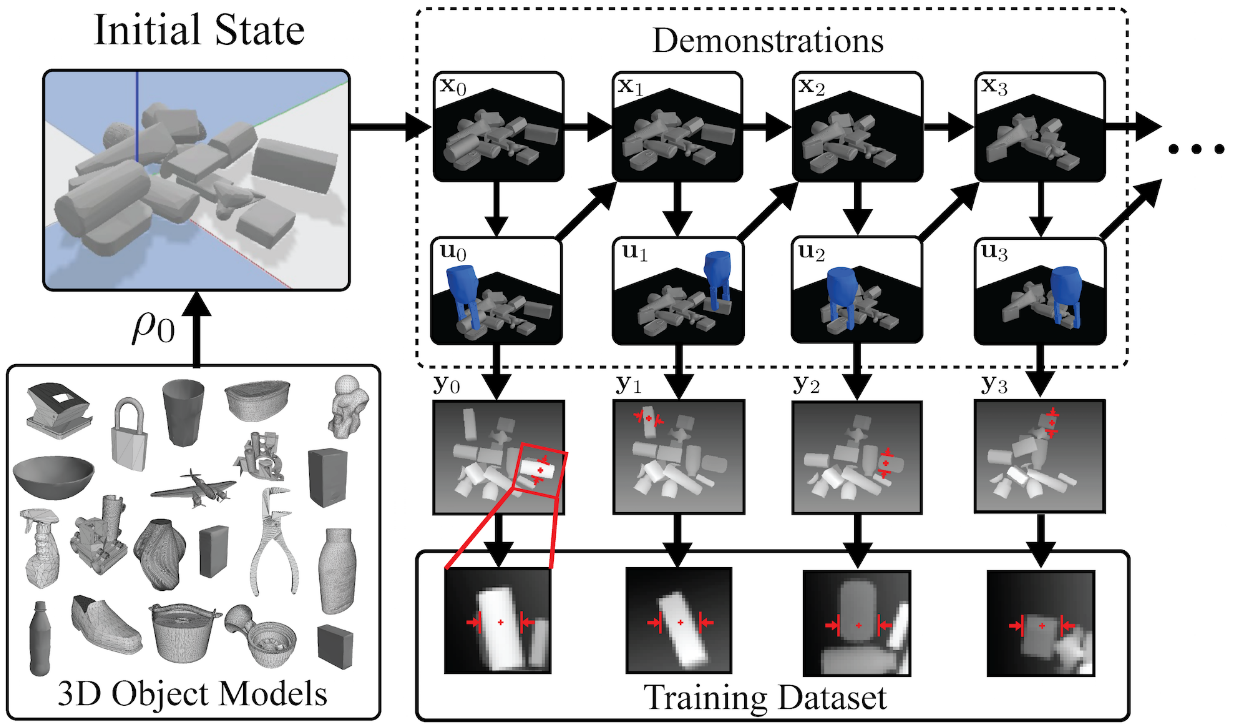


Figure 5.1: Overview of the Dex-Net 2.1 POMDP model and simulator. We sample from the initial state distribution ρ_0 by uniformly sampling m 3D CAD object models from a dataset and dropping them in random poses in the pybullet dynamic simulator [26] to form a heap. The state $\mathbf{x}_t$ includes object shapes and poses in the heap. We generate demonstrations of robot grasping using an algorithmic supervisor Ω from Dex-Net 2.0 [105] that indexes the most robust collision-free parallel-jaw grasp $\mathbf{u}_t$ from a pre-planned grasp database using knowledge of the full state. We aggregate synthetic point cloud observations $\mathbf{y}_t$ and collected rewards R_t to form a labeled dataset for training a policy that classifies the supervisor's actions on the partial observations using imitation learning. We preprocess training data by transforming the point clouds to align the grasp center and axis with the center pixel and middle row to improve GQ-CNN classification performance [96, 105].

Decision Process (POMDP) (see Fig. 5.1) specified as a tuple $(\mathcal{X}, \mathcal{U}, \mathcal{Y}, R, q(\mathbf{x}_0), q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t), q(\mathbf{y} \mid \mathbf{x}))$ consisting of a set of states $\mathcal{X}$ (object shapes and poses), a set of actions $\mathcal{U}$ (gripper poses), a set of observations $\mathcal{Y}$ (point clouds), a reward function R, an initial state distribution $q(\mathbf{x}_0)$ (object heaps), a transition distribution $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$, and a sensor noise distribution $q(\mathbf{y}_t \mid \mathbf{x}_t)$ [168]. Our POMDP uses a fixed maximum time horizon T. See Section 5.3.5 for numeric values of parameters for each distribution.

Initial State Distribution ($q(\mathbf{x}_0)$). The initial state distribution $q(\mathbf{x}_0)$ models the position and shape of objects in a heap as well as the parameters of the camera and friction which stay constant over an episode. We model $q(\mathbf{x}_0)$ as the product of distributions on:

1. *Object Count (m):* Poisson distribution with mean λ.

2. *Object Heap ($\mathcal{O}$):* Uniform distribution over a discrete set of m 3D triangular meshes $\{\mathcal{M}_0, ...\mathcal{M}_{m-1}\}$ and the pose from which each mesh is dropped into the heap.

3. *Depth Camera ($\mathcal{C}$):* Uniform distribution over the camera pose and intrinsic parameters.

4. *Coulomb Friction (α):* Truncated Gaussian constrained to $[0, 1]$.

The initial state is sampled by (1) sampling an object count m and a set of m 3D CAD models, (2) sampling a planar pose for the heap center and planar pose offsets from the pile center for each of the objects, and (3) dropping the objects one by one from a fixed height h_0 above the table and running dynamic simulation until all objects come to rest (all velocities are zero). Any objects that roll beyond a distance W from the world center are removed.

States ($\mathcal{X}$). The state $\mathbf{x}_t$ at time t consists of the current set of 3D object meshes $\mathcal{O}_t$ and their poses.

Grasp Actions ($\mathcal{U}$). The robot can attempt to grasp and remove an object from the environment by executing an action $\mathbf{u}_t$ specified as a 4-DOF gripper pose $(\mathbf{p}, \theta, d)$ where $\mathbf{p}$ is the grasp center pixel, θ is the orientation of the gripper in image space, and d is the grasp depth, or distance of the 3D grasp center from the image plane [105]. The action is related to a grasp center in 3D space $\mathbf{c}$ by the formula $\mathbf{c} = (1/d)K^{-1}(\mathbf{p}_x, \mathbf{p}_y, 1)^T$ [57]. The robot executes an action by moving to the target 3D gripper pose along a linear approach trajectory, closing the jaws with constant force, and lifting upwards.

Observations ($\mathcal{Y}$). The robot observes a point cloud $\mathbf{y}_t$ specified as real-valued $H \times W \times 3$ matrix representing a set of 3D points imaged with a depth camera with $H \times W$ resolution.

Rewards. Binary rewards occur on transitions that remove a single object from the heap. Let $m_t = |\mathcal{O}_t|$ be the number of objects remaining in the heap. Then $R(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}) = 1$ if $m_{t+1} < m_t$.

Transition Distribution ($q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$). We use mechanical wrench space analysis to determine whether or not an object can be lifted from the heap under quasi-static conditions [136, 129], and we use multibody dynamic simulation with a velocity-based complementarity formulation implemented in pybullet [26] to determine the next state of the objects after an object is lifted.

Let $\mathcal{M}_i \in \mathbf{x}_t$ be the first object to be contacted by the gripper jaws when executing action $\mathbf{u}_t$. Then we measure grasp success with a binary-valued metric $S(\mathbf{x}_t, \mathbf{u}_t) \in \{0, 1\}$ that measures whether or not $\mathbf{u}_t$ is collision-free and can resist external wrenches on object $\mathcal{M}_i$ under state perturbations using point contact models [178]. Specifically, $S = 1$ if the robust epsilon metric is greater than a threshold $\delta = 0.002$ [105] and the gripper does not collide with the table or object along a linear approach trajectory [41, 178]. If $S(\mathbf{x}_t, \mathbf{u}_t) = 1$, then $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, i)$ where f returns the set of object meshes and poses resulting from a dynamic simulation of object heap $\mathcal{O}$ as object $\mathcal{M}_i$ is lifted until the remaining objects come to rest. Otherwise the state remains unchanged. If an object rolls beyond a distance W from the world center then it is re-dropped in the pile.

Observation Distribution ($q(\mathbf{y}_t \mid \mathbf{x}_t)$. We model depth-proportional noise with a Gamma distribution modeling depth-proportional noise due to errors in disparity computation and a Gaussian Process to model correlated zero-mean noise in pixel space [105, 111].

5.1.2 Policy

A policy maps point cloud observations to actions $\pi_\theta(\mathbf{y}_t) = \mathbf{u}_t$, where policies are parametrized by a vector of neural network weights θ. We consider policies of the form:

$$\pi_\theta(\mathbf{y}) = \text{argmax }_{\mathbf{u}\in\mathcal{U}} Q_\theta(\mathbf{y}, \mathbf{u})$$

where $\mathcal{U}$ specifies constraints on the set of available grasps such as collisions and $Q_\theta(\mathbf{y}, \mathbf{u}) \in [0, 1]$ is a scoring function for actions given observations [70, 98, 105, 131]. A policy induces a distribution over trajectories given the initial state, next state, and perceptual distributions of the POMDP: $p(\tau \mid \theta) = \rho_0 \prod_{t=0}^{T-1} p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \pi_\theta(\mathbf{y}_t))q(\mathbf{y}_t \mid \mathbf{x}_t)$, where τ is a trajectory of length $T + 1$ defined as a vector of states, actions, and observations: $\tau = (\mathbf{x}_0, \mathbf{u}_0, \mathbf{y}_0, ...\mathbf{x}_T, \mathbf{u}_T, \mathbf{y}_T)$.

5.1.3 Objective

The objective is to learn a policy π_θ that maximizes the sum of undiscounted rewards (Section 2.1.3):

$$\theta^* = \underset{\theta\in\Theta}{\text{argmax}}\ \mathbb{E}_{p(\tau|\theta)} \left[\sum_{t=0}^{T-1} R(\mathbf{x}_t, \pi_\theta(\mathbf{y}_t), \mathbf{x}_{t+1})\right].$$

In this POMDP, the objective corresponds to maximizing the number of objects removed from the heap.

5.2 Imitation Learning from an Algorithmic Supervisor

Optimal solutions to POMDPs are known to be computationally intractable [128]. Many approximate solution methods exist, however many assume closed-form dynamics [72], discrete or low-dimensional state spaces [153], or Gaussian distributions [132].

We explore the use of imitation learning (IL) [2] to learn the actions of an algorithmic supervisor that computes actions based on full state knowledge from the simulator. IL has been used to approximately solve POMDPs when there exists an algorithmic supervisor with access to full state information [21]. We first compute an algorithmic supervisor using the singulated object robust grasp planner from Dex-Net 2.0 [105] which computes grasps using mechanical wrench space analysis given known object shape and pose. We then collect a dataset of point clouds, actions, and rewards by rolling out the algorithmic supervisor with noise injection [93] to balance the distribution of positive and negative examples. Finally, we learn a CNN to classify the supervisor's actions and use the trained CNN as action scoring function [85, 139].

5.2.1 Robust Grasping Algorithmic Supervisor

The algorithmic supervisor Ω pre-computes a set of robust grasps for each 3D object in a dataset using full state knowledge. For computational efficiency, the supervisor is implemented by pre-computing a database of robust grasps (such as Dex-Net) for each 3D object by evaluating the grasp success metric $R(\mathbf{x}, \mathbf{u})$ using Monte-Carlo sampling [105]. Given a state $\mathbf{x}_t$, Ω plans a robust grasp by pruning the set of possible actions for every object in the heap using collision checking and returning an action uniformly at random from the remaining set of robust grasps, if one exists. We note that Ω maximizes reward for the only current timestep and may not be optimal for the full time horizon.

5.2.2 Learning a Bin Picking Policy

We learn a bin picking policy by learning a classifier for the supervisor's actions [85, 139] on rollouts of Ω with noise injection [93]. Noise injection balances the distribution of positive and negative examples for the classifier, as rolling out the algorithmic supervisor results in all positive examples.

Our policy learning algorithm consists of two steps: (1) collect demonstrations by executing Ω with probability $1-\epsilon$ and a random action from the grasp database with probability ϵ, and (2) use supervised learning to classify actions taken by the supervisor with a Grasp Quality CNN (GQ-CNN). Specifically, given K demonstrations from the noise-injected supervisor we optimize:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{j=1}^{K} \sum_{t=1}^{T} \mathcal{L}\left(Q_\theta(\mathbf{y}_{j,t}, \mathbf{u}_{j,t}), R_{j,t}\right)$$

where $R_{j,t} = 1$ if the supervisor agrees with the action on timestep t in rollout j and $\mathcal{L}$ is the cross entropy classification loss. Given a point cloud, we use the robust grasping policy of Dex-Net 2.0 [105] that samples and ranks a set of antipodal grasp candidates according to $Q_{\hat{\theta}}$ using the Cross Entropy Method.

Transfer Learning

Research in computer vision suggests that features from deep CNNs performs well as generic features when classifying images in new domains [22]. Since simulating object heaps is slower than simulating singulated objects due to object interactions [26], we explore optimizing the neural network weights of the bin picking policy by transfer learning from features of a GQ-CNN trained to grasp singulated objects on millions of examples from Dex-Net 2.0 [105]. Specifically, we fine-tune features from the Dex-Net 2.0 GQ-CNN by using the weights as an initialization for optimization with SGD and only updating the fully connected layers, leaving the conv layers fixed. We update the network for 10 epochs using SGD with momentum of 0.9, a base learning rate of 0.01, and a staircase exponential learning rate decay with a decrease of 5% on each epoch.

5.3 Experiments

We evaluated classification performance on synthetic data from the simulator and performed extensive physical evaluations on an ABB YuMi with a Primesense Carmine 1.08 depth sensor and custom silicone gripper tips designed by Guo et al. [53]. All experiments ran on a Desktop running Ubuntu 16.04 with a 3.4 GHz Intel Core i7-6700 Quad-Core CPU and an NVIDIA GeForce 980, and we used an NVIDIA GeForce GTX 1080 for training large models.

5.3.1 Synthetic Training

We generated three versions of the Dex-Net 2.1 training dataset with noise levels $\epsilon = \{0.1, 0.5, 0.9\}$ using 10,000 rollouts of the noisy supervisor policy in our POMDP with an average of $\lambda = 5$ objects per heap sampled from the 1,500 object models of Dex-Net 2.0 [105]. The dataset sizes were approximately: $\epsilon = 0.1$: 16.5k, $\epsilon = 0.5$: 34.8k, $\epsilon = 0.9$: 102.6k.

We trained the following models with a 80-20 image-wise split on each of the $\epsilon = 0.1$, $\epsilon = 0.5$, and $\epsilon = 0.9$ datasets:

- **SVM.** A bagging classifier composed of 50 SVMs trained on the first 100 principal components of the GQ-CNN fc4 feature space [105].
- **Random Forest (RF).** A set of 50 trees of max depth 10 trained on the first 100 principal components of the GQ-CNN fc4 feature space [105].
- **Dex-Net 2.1 (Scratch).** A GQ-CNN [105] trained only on Dex-Net 2.1 for 25 epochs.
- **Dex-Net 2.1 (Fine-tuned).** A GQ-CNN [105] initialized with pre-trained weights from Dex-Net 2.0 and fine-tuned on Dex-Net 2.1 for 10 epochs with fixed conv layers.

Fig. 5.2 shows the precision-recall curve for the methods. The GQ-CNN-based policy trained with transfer learning performs significantly better than the other methods with an Average Precision of 64% compared to 59% for the next best method. The SVM used the output of conv2_2 layer of the Dex-Net 2.0 GQ-CNN as input and the Random Forest used the output of the fc4 layer of the Dex-Net 2.0 GQ-CNN as input.

Table 5.1 compares the classification accuracy, AP, and percentage of objects cleared (successful grasps) from rolling out bin picking policies with various levels of noise injection in the simulator. We see that the policy with high noise ($\epsilon = 0.9$) has the highest classification accuracy and the policy with low noise ($\epsilon = 0.1$) level clears the most objects. This suggests

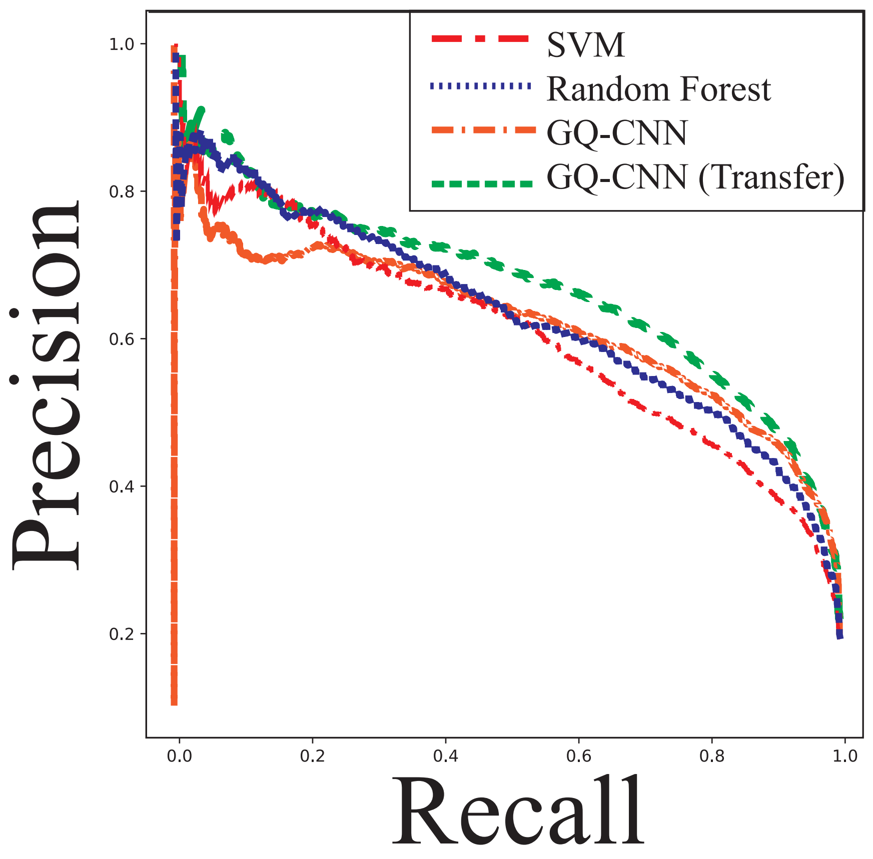


Figure 5.2: Precision-recall curve for the top four machine learning models on a fixed validation subset of the Dex-Net 2.1 $\epsilon = 0.9$ dataset containing approximately 20k datapoints.

Noise (%)	Acc.(%)	AP	(%) Cleared
0.1	5.4	**0.8**	**62**
0.5	5.3	0.3	43
0.9	**13.8**	**0.8**	44

Table 5.1: Performance of bin picking policies on rollouts in the simulator as a function of the level of noise injection.

that the policy trained with low noise levels may be overly optimistic while the policy with high noise tends to be pessimistic and fails to find successful grasps on many timesteps.

5.3.2 Bin Picking on an ABB YuMi

To study performance on a physical robot, we designed a bin picking benchmark where the robot was presented a subset of objects from a dataset of 50 test objects in a bin and the goal was to iteratively move objects from the bin to a receptacle as illustrated in Fig. 5.3. First,

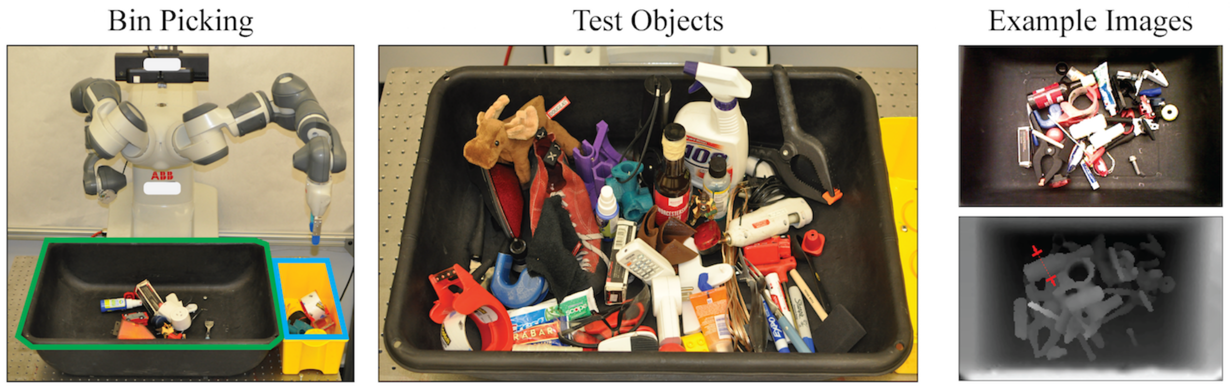


Figure 5.3: Experimental setup for benchmarking bin picking policies. (Left) For each experiment, a subset of N validation objects are randomly dropped into a bin (green rim, center), at which point the YuMi iteratively plans grasps from point clouds and attempts to lift and transport the objects to a packing box (blue rim, right side).(Middle) A set of 50 test objects with various shapes, sizes, and material properties. A subset of 25 are rigid and opaque, and 25 others have transparency (e.g. goggles), moving parts (e.g. can opener), or deformable material (e.g. cloth). (Right) Example color and depth images from the physical setup with example grasp planned with the Dex-Net 2.1 $\epsilon = 0.9$ policy.

we sampled N objects from the validation set sampled uniformly at random. Then, each of the N objects was placed in a box and the box was shaken and placed upside-down in the center of the bin to randomize object poses. On each timestep the grasping policy received as input a depth image, bounding box containing the object, and camera intrinsics, and output a target pose of the gripper in the robot's coordinate frame. The robot then approached the target grasp along a linear approach trajectory and closed the jaws. Grasp success was defined by whether or not the grasp transported the target object to the receptacle. The system iterated until either (a) no objects remain or (b) the robot has 5 consecutive failed grasps on the same object.

Performance Metrics

We compared performance on this benchmark with the following metrics:

1. **Success Rate:** The percentage of grasp attempts that moved an object to the packing box.

2. **Average Precision (AP):** The area under the precision-recall curve. In some applications a robot can decide whether or not to execute grasps based on a threshold for the classifier confidence. AP measure the average success rate over all possible thresholds for these scenarios.

3. **Percent Cleared:** The fraction of objects that were moved to the receptacle.

4. **Picks per Hour (PPH):** The estimated number of bin picks per hour of runtime computed by multiplying the average number of grasp attempts per hour by the success rate. Human performance is approximately 600 PPH.

Datasets

Fig. 5.3 illustrates the test set of physical objects used in the benchmark, which includes 50 objects of various sizes, shapes, and materials. The objects all satisfy three criteria to be graspable by the YuMi: (1) the jaws can fit around the object in at least one configuration, (2) the object weighs less than $0.25kg$ due to the YuMi payload, (3) some part of the object is opaque and non-specular (can be sensed with a depth camera), and (4) the min diameter of the object is greater than $1cm$.

We break the test set up into two partitions:

- **Basic.** A subset of 25 test objects that are rigid, weigh less than $0.25kg$ (the payload of the YuMi), and are fully visible with a depth camera, which tests generalization to novel shapes when assumptions of the simulator are satisfied.
- **Typical.** The entire 50 object dataset, which additionally tests generalization to novel object properties (e.g. transparency, deformation, moving parts).

Model Performance

Table 5.2 compares the performance of five policies on the bin picking benchmark with $N = \{5, 10\}$ test objects from the Basic subset for 20 and 10 trials, respectively. This measures performance on a heap size and set of objects that match the assumptions of the simulator. We compared the following policies:

1. **Image-Based Force Closure.** Executes a random planar force grasp with friction coefficient $\mu = 0.8$ computed from edge detection in depth images inspired by [129].
2. **Dex-Net 2.0.** Ranks grasps using the GQ-CNN model of [105] trained on Dex-Net-Large.
3. **Dex-Net 2.1 ($\epsilon = 0.1, \epsilon = 0.5, \epsilon = 0.9$).** Ranks grasps using the Dex-Net 2.1 (Fine-tuned) classifier for varying levels of noise injection in the training dataset.

The Dex-Net 2.1 ($\epsilon = 0.9$) variant performed best across all metrics. The increase in performance over the other noise levels may be because the training dataset was heavily skewed toward negative examples, encouraging the learned policy to predict grasp failure when uncertain.

	5 Objects				10 Objects			
Policy	**Success (%)**	**AP (%)**	**% Cleared**	**PPH**	**Success (%)**	**AP (%)**	**% Cleared**	**PPH**
Force Closure	54	N/A	97	271	55	N/A	92	276
Dex-Net 2.0	92	96	**100**	407	83	84	98	367
Dex-Net 2.1 ($\epsilon = 0.1$)	91	91	**100**	402	86	89	99	380
Dex-Net 2.1 ($\epsilon = 0.5$)	85	89	98	376	66	69	96	292
Dex-Net 2.1 ($\epsilon = 0.9$)	**94**	**97**	**100**	**416**	**89**	**93**	**100**	**394**

Table 5.2: Performance of grasping policies on the Basic dataset containing 25 opaque and rigid test objects with heaps of size $N = \{5, 10\}$ averaged over 20 and 10 independent trials, respectively. Human performance is approximately 600 PPH.

	10 Objects				20 Objects			
Policy	**Success (%)**	**AP (%)**	**% Cleared**	**PPH**	**Success (%)**	**AP (%)**	**% Cleared**	**PPH**
Force Closure	64	N/A	98	321	50	N/A	77	251
Dex-Net 2.0	81	88	98	358	70	79	**97**	310
Dex-Net 2.1 ($\epsilon = 0.9$)	**85**	**93**	**100**	**376**	**78**	**86**	**97**	**345**

Table 5.3: Generalization performance of grasping policies on the Typical dataset containing 50 test objects with hinged parts, deformability, and some material transparency on heaps of size 10 and 20 with 5 independent trials of each.

Generalization

We also evaluated the Dex-Net 2.1 $\epsilon = 0.9$ policy on larger heaps of size $N = \{10, 20\}$ with all 50 test objects for 5 independent trials each to evaluate generalization to large piles and different object properties that were not encountered in the simulator. The results are detailed in Table 5.3. While performance decreases across all categories, the $\epsilon = 0.9$ policy outperforms the Dex-Net 2.0 and antipodal baseline across all metrics. The performance appears to be more significantly affected by the heap size than the addition of deformable objects. Qualitative failure modes of the Dex-Net 2.1 policy included collisions where the gripper pressed into another object in the heap and an inability to find robust grasps on thin, curved objects such as the measuring spoon and scissors.

Fig. 5.4 illustrates grasp planning with the $\epsilon = 0.9$ policy on a heap of novel objects from the test set.

Failure Modes

Common failure modes included collisions due to complex geometry, failure to sense object geometry with the depth camera due to object material properties, thin objects that were very close to the table surface, and grasps attempted on multiple objects at once.

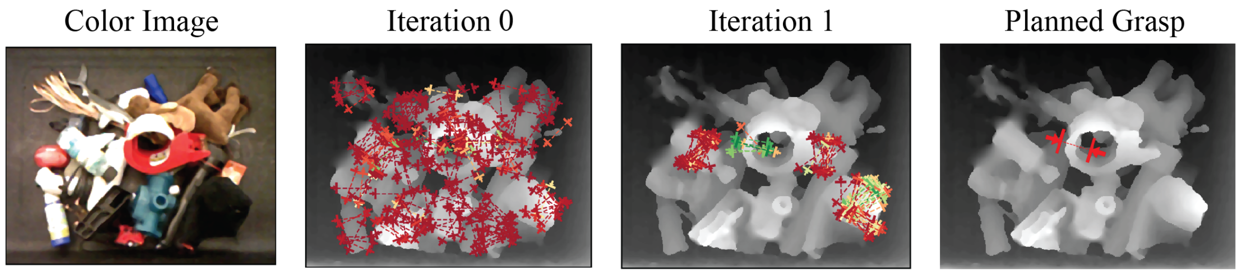


Figure 5.4: Example of grasp planning with the Dex-Net 2.1 $\epsilon = 0.9$ robust grasping policy on a heap of novel objects from the test set. The iterations (left to right) show the set of grasps sampled during the progression of the Cross Entropy Method for grasp optimization.

5.3.3 t-SNE Embedding

To gain insight into the feature space of the GQ-CNN fine-tuned on Dex-Net 2.0, we visualized a t-distributed Stochastic Neighbor (t-SNE) embedding [102] of the fc4 features. Specifically, we projected the validation subset of the Dex-Net 2.1 ($\epsilon = 0.9$) dataset on the 100 principal components of the fc4 features for the training subset of the data, using the fc4 features of both the Dex-Net 2.0 GQ-CNN and Dex-Net 2.1 ($\epsilon = 0.9$) GQ-CNN. Then we ran the Barnes-Hut version of t-SNE with tree-based acceleration [174] to compute a 2-dimensional embedding.

Fig. 5.5 compares the t-SNE embedding for the Dex-Net 2.0 GQ-CNN trained on singulated objects and the Dex-Net 2.1 ($\epsilon = 0.9$) GQ-CNN trained for bin picking. For each datapoint we plot the transformed grasp image colored by whether or not grasps are from the supervisor's action set (green) or from the set failed grasps from the 90% random actions (red). We see that in the original feature space some of the supervisor's grasps are mixed in with negative examples, and many of these examples have some level of clutter around the target object. This suggests that some robust grasps in clutter cannot be discriminated from failed grasps based on the GQ-CNN weights. There are also examples of false positives for the Dex-Net 2.0 GQ-CNN, such as the top-leftmost red image which is not collision-free for the gripper. However, after fine-tuning [183] we see that the supervisor's actions are more tightly clustered, with fewer negative examples mixed it.

5.3.4 Qualitative Differences Between Policies

We also looked into the failure modes across all methods to gain insights into the performance gains for the Dex-Net 2.1 bin picking policy. Two examples are illustrated in Fig. 5.6.

The first demonstrates collision avoidance in piles. The Dex-Net 2.0 GQ-CNN predicts grasp success for a grasp that will put the hand in collision with the pile. We hypothesize that this is due to the GQ-CNN picking up on the edge between the objects as a graspable area without considering collisions, as it has only seen single objects in training. However, the Dex-Net 2.1 GQ-CNN plans a grasp that avoids the other objects in the pile, picking

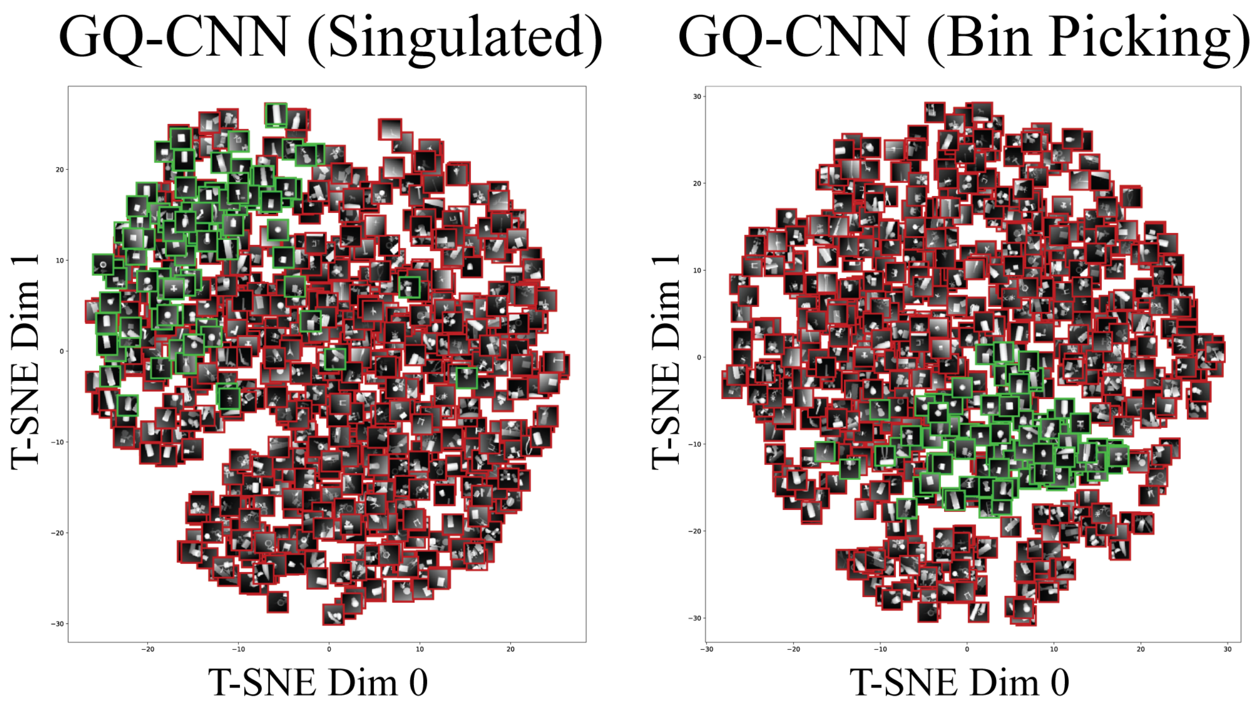


Figure 5.5: t-SNE embedding of the first 100 principal components of fc4 features for the validation subset of the Dex-Net 2.1 ($\epsilon = 0.9$) dataset (best viewed in color). (Left) The embedding for features from the original Dex-Net 2.0 GQ-CNN. (Right) The GQ-CNN fine-tuned on Dex-Net 2.1 ($\epsilon = 0.9$). Each datapoint shows the rotated and translated depth image that is input to the to GQ-CNN (see [105]). Images corresponding to actions taken by the algorithmic supervisor are outlined in green while images corresponding to failed random actions are outlined in red.

out a single one.

The second example demonstrates that the Dex-Net 2.1 policy can also find narrow openings to place the fingers in a dense heap in order to achieve a robust grasp. The Dex-Net 2.0 GQ-CNN falsely predicts that a grasp of the tip of the glue gun will be successful, perhaps because it has seen no similar example in training. On the other hand, the Dex-Net 2.1 policy finds the frame of the goggles on the top left of the pile, where it can safely fit a finger between the goggles and bottle of sauce.

5.3.5 Details of POMDP Parameters

For replicability, this section lists the numeric values of parameters used in our Partially Observable Markov Decision Process (POMDP) model of bin picking. The maximum time horizon used was 50 timesteps. If the heap was cleared before this time, then the rollout ended immediately.

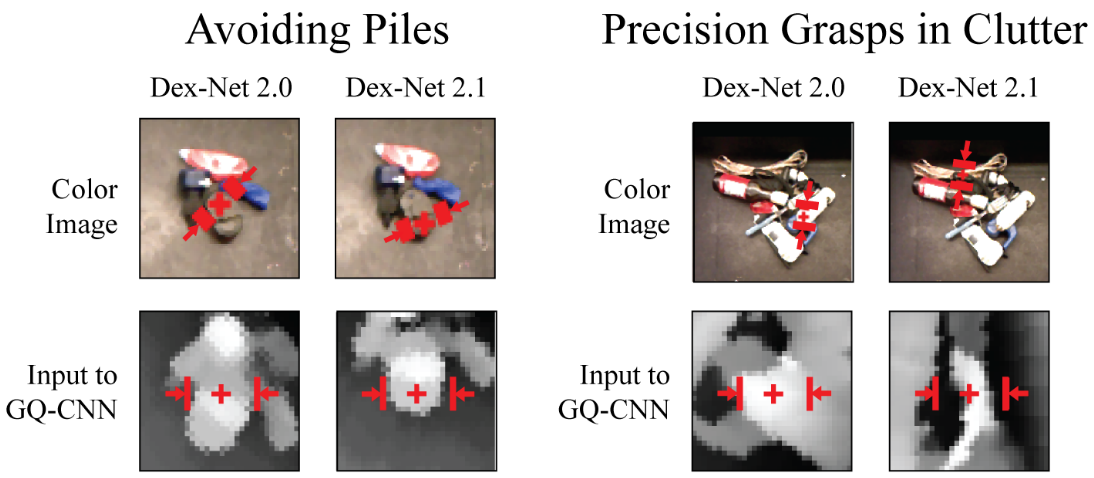


Figure 5.6: Comparison of grasps planned by the Dex-Net 2.0 (singulated object) and Dex-Net 2.1 (bin picking) grasping policies.

Initial State Distribution $q(\mathbf{x}_0)$

The initial state distribution is a product of the following distributions:

1. *Object Count (m):* Poisson distribution with mean $\lambda = 5$.

2. *Object Heap ($\mathcal{O}$):* Discrete uniform distribution over the $1,500$ 3D triangular meshes $\{\mathcal{M}_0, ...\mathcal{M}_{m-1}\}$ from the KIT [78] and 3DNet [180] datasets, as well as the pose from which each mesh is dropped into the heap. The pose distribution consists of a fixed drop height of $0.2m$, a random planar translation above the table in $[-0.1, 0.1] \times [-0.1, 0.1]$ (units of meters), and a random rotation sampled by first uniformly sampling spherical coordinates, then sampling an angle in $[0, 2pi)$ radians for the orientation in the table plane.

3. *Depth Camera ($\mathcal{C}$):* Uniform distribution over the camera pose using spherical coordinates $r, \theta, \varphi \sim \mathcal{U}([0.65, 0.75] \times [0, 2\pi) \times [1°, 10°])$, where the camera optical axis always intersects the center of the table. The camera focal length was sampled uniformly from $[520, 530]$ pixels.

4. *Coulomb Friction (α):* Truncated Gaussian with mean 0.5 and variance 0.1, constrained to $[0, 1]$.

The maximum distance that an object could translate or roll from the world center before removal was $W = 0.2$.

Next State Distribution $q(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t)$

The next state distribution used the parameters of [105] for the robust epsilon metric, which consists of

1. **Gripper Pose:** An isotropic Gaussian with mean on the target gripper pose, a translational variance of $1.0mm$, and a rotational variance of 0.001 in Lie Algebra coordinates.

2. **Object Pose:** An isotropic Gaussian with mean on the true object pose, a translational variance of $5.0mm$, and a rotational variance of 0.01 in Lie Algebra coordinates.

We used the wrench space norm of $\|\mathbf{w}| = \sqrt{\|\mathbf{f}\|_2^2 + \nu\|\tau\|_2^2}$ [41] with $\nu = 1000$. If an object translated or rolled from the world center a distance greater than $W = 0.2$, then it was re-dropped in the pile using the initial object pose distribution of ρ.

Observation Distribution $q(\mathbf{y}_t \mid \mathbf{x}_t)$

Our observation distribution follows [105]. We model images as $\mathbf{y} = \alpha * \hat{\mathbf{y}} + \epsilon$ where $\hat{\mathbf{y}}$ is a rendered depth image created using OSMesa offscreen rendering. We model α as a Gamma random variable with shape= 1000.0 and scale=0.001. We model ϵ as Gaussian Process noise drawn with measurement noise $\sigma = 0.005$ and kernel bandwidth $\ell = \sqrt{2}px$.

Parameters of Graphical Model

The graphical model specifies $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ as the product of a state distribution $q(\mathbf{x})$, an observation model $q(\mathbf{y}|\mathbf{x})$, an algorithmic supervisor distribution $q(\mathbf{u}|\mathbf{x})$, and a grasp reward distribution $q(R|\mathbf{g}, \mathbf{x})$.

We model the state distribution as $q(\mathbf{x}) = q(\gamma)q(\mathcal{O})q(\mathbf{T}_o|\mathcal{O})q(T_c)$. We model $q(\gamma)$ as a Gaussian distribution $\mathcal{N}(0.5, 0.1)$ truncated to $[0, 1]$. We model $q(\mathcal{O})$ as a discrete uniform distribution over 3D objects in a given dataset. We model $q(\mathbf{T}_o|\mathcal{O}) = p(\mathbf{T}_o|\mathbf{T}_s)p(\mathbf{T}_s|\mathcal{O})$, where is $q(\mathbf{T}_s|\mathcal{O})$ is a discrete uniform distribution over object stable poses and $q(\mathbf{T}_o|\mathbf{T}_s)$ is uniform distribution over 2D poses: $\mathcal{U}([-0.1, 0.1] \times [-0.1, 0.1] \times [0, 2\pi))$. We compute stable poses using the quasi-static algorithm given by Goldberg et al. [49]. We model $q(\mathbf{T}_c)$ as a uniform distribution on spherical coordinates $r, \theta, \varphi \sim \mathcal{U}([0.65, 0.75] \times [0, 2\pi) \times [0.05\pi, 0.1\pi])$, where the camera optical axis always intersects the center of the table.

Our distribution over grasps is a uniform distribution over pairs of antipodal points on the object surface that are parallel to the table plane. We sample from this distribution for a fixed coefficient of friction $\mu = 0.6$ and reject samples outside the friction cone or non-parallel to the surface.

We model images as $\mathbf{y} = \alpha * \hat{\mathbf{y}} + \epsilon$ where $\hat{\mathbf{y}}$ is a rendered depth image created using OSMesa offscreen rendering. We model α as a Gamma random variable with shape= 1000.0 and scale=0.001. We model ϵ as Gaussian Process noise drawn with measurement noise $\sigma = 0.005$ and kernel bandwidth $\ell = \sqrt{2}px$.

We compute grasp robustness metrics using the graphical model and noise parameters of [152].

5.4 Discussion

In this chapter, we formulated bin picking as a POMDP and trained a GQ-CNN to predict grasps with high reward using a simulator of robust grasping and dynamic object interactions in a heap. We used the model to sample the Dex-Net 2.1 dataset of tens of thousands of demonstrations across 1,500 3D object models from an algorithmic supervisor with noise injection that used full state information to index precomputed robust grasps for 3D models in the simulation.

Experiments suggest that a policy trained using behavior cloning with high levels of noise injection (90% probability of selecting a random action) has the highest performance across all metrics when transferred to a physical robot, suggesting that conservative policies which favor false negatives over false positives may transfer better from simulation to reality. Furthermore, the high average precision score of GQ-CNNs from Dex-Net 2.1 suggest that performance could be improved by introducing alternative actions, such as probing or using a suction cup [106], when the model does not have high confidence. The high success rate of the greedy policy also suggests that the sequential structure is relatively weak in bin picking: policies may not need to be trained with labels reflecting the long-term affects of actions.

Experiments also uncovered several shortcomings of the method presented in this chapter. First, the policies exhibit repeated failures, which we believe may be a symptom of using a greedy policy rather than a policy that uses history as suggested by the POMDP theory (Section 2.1.3). Furthermore, fine-tuning appears to lead to peculiar failure modes such as grasping at smooth angled surfaces, which may be the result of catastrophic forgetting [84]. Additionally, the current method is limited to grasping only the set of objects that can be picked from clutter with a parallel-jaw gripper, which is a relatively small range due to dense clutter and size constraints from the parallel-jaw gripper.

Part III

Physics-Based Reward Design

Whenever the Creator gives you something, don't hesitate. Grab it.

NATIVE AMERICAN ELDER

Chapter 6

Synthesis of Energy-Bounded Caging Grasps with Persistent Homology

This part of the thesis considers extending the dataset generation distribution developed in Part II to new grasp reward functions, new grippers, and policies composed of multiple grasping sub-policies for different grippers. Chapter 6 considers a novel class of a reward functions based on energy-bounded caging: using a gripper to geometrically constrain the motion of an object under an energy field such as gravity. We develop an algorithmic supervisor that can synthesize energy-bounded caging grasps given a polygonal gripper and a novel polygonal object using techniques from computational topology. Chapter 7 explores extending the model of Chapter 4 to a vacuum suction cup gripper, proposes a new compliant contact model for suction cup grippers, and presents experiments suggesting that training datasets generated using the novel contact model can train grasping policies that generalize to a wide variety of objects on a physical robot. Chapter 8 studies dataset generation models for training composite policies, policies that select an action from a set of a sub-policies for different action types, and presents experiments suggesting that a gripper-agnostic reward model can be used to train policies that can clear bins of up to 50 novel objects with 90% on a physical robot by deciding between a suction cup and parallel-jaw gripper. Along the way we refine the generative model structure to facilitate extensions to additional tasks, grippers, or composite policies.

In this chapter, we focus on non-prehensile grasping, in which a robot can transport an object without immobilizing the object. This is motivated by applications in manufacturing where parts must be reliably grasped and moved without precise constraints on object pose (for example in kitting or logistics). Caging configurations, in which an object's mobility is bounded by a set of obstacles, can provide robustness to perturbations in object pose.

The standard model of caging (complete caging) considers whether a set of obstacles can be placed in a configuration such that the object cannot escape because its mobility is restricted to a bounded set in the free configuration space $\mathcal{F}$ [143, 171] as illustrated in the left part of Fig. 6.1. This chapter extends caging theory by defining *energy-bounded cages* under a potential energy field $U : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ such as gravity based on the minimum energy

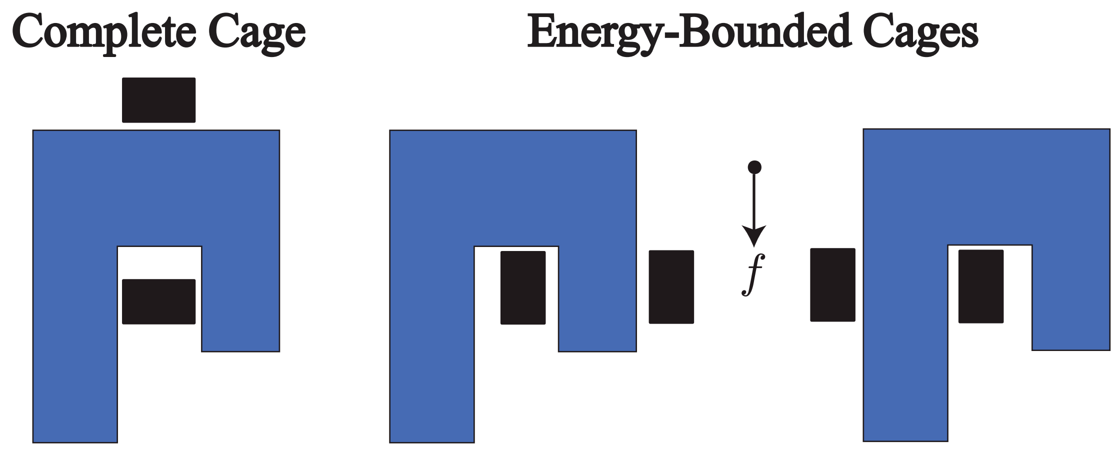


Figure 6.1: Complete and energy-bounded cages. Left: a complete cage. The blue object is constrained to a bounded component of the free configuration space by the rigid arrangement of the two gripper fingers (black). Middle and right figure: Two energy-bounded cages with respect to a force direction f e.g. from gravity or constant velocity pushing with Coulomb friction. The blue object is constrained by both the gripper and the force field. The rightmost configuration requires more energy to escape than the middle configuration.

required to escape. The minimum escape energy can be used as a reward function for caging grasps.

This chapter also presents Energy-Bounded-Cage-Synthesis-2D (EBCS-2D), a sampling-based algorithm for synthesis of energy-bounded cages given a polygonal object and a rigid configuration of polygonal obstacles under a concave energy field defined over object translations, such as gravity or planar pushing. EBCS-2D synthesizes an list of energy-bounded cages ranked by escape energy using persistent homology, a tool from computational topology that efficiently computes representatives for bounded components of the free configuration over varying escape energy thresholds. EBCS-2D constructs a weighted α-shape from samples of object poses and a lower-bound on their penetration depth [107], finds a set of candidate energy bounded cages using persistence, and prunes the candidates based on collisions and energy level. The escape energies returned by EBCS-2D provably lower bound the true minimum escape energy for each returned cage. If the returned escape energy is infinite then the object is completely caged.

We implement EBCS-2D using the Persistent Homology Algorithms Toolbox (PHAT) [8] to efficiently identify the most robust energy bounded cages. We evaluate EBCS-2D on a set of seven polygonal parts with parallel-jaw grippers using a push energy field and use it to synthesize optimal push directions. In each case, RRT* optimal path planning was unable to find an escape path with lower energy than the estimated lower bound within 120 seconds. We also use EBCS-2D for planar pushing on a physical Zymark Zymate robot and ABB YuMi with parallel-jaw grippers and confirm that configurations synthesized by EBCS-2D successfully push objects on a planar worksurface.

6.1 Problem Statement

Given a rigid polygonal object $\mathcal{O}$, a rigid configuration of obstacles $\mathcal{G}$ on a planar worksurface, and an energy function U, we consider the problem of finding and ranking the set of energy-bounded cages of $\mathcal{O}$ by $\mathcal{G}$.

6.1.1 Complete Caging and Energy-Bounded Caging

We consider a planar configuration space $\mathcal{C} \subseteq SE(2)$ of a compact polygonal planar object $\mathcal{O} \subset \mathbb{R}^2$ placed in a planar workspace with obstacles defined by fixed positions of a set of k polygons $\mathcal{G} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k \subset \mathbb{R}^2$, such as the jaws of a robot gripper. We assume the center of mass is known for both the object and obstacles. We denote the object polygon in pose $\mathbf{q} = (x, y, \theta) \in SE(2) = \mathbb{R}^2 \times \mathbb{S}^1$ relative to a reference pose $\mathbf{q}_0$ by $\mathcal{O}(\mathbf{q})$. We define the *collision space* of $\mathcal{O}$ relative to $\mathcal{G}$ by $\mathcal{Z} = \{\mathbf{q} \in SE(2) : int(\mathcal{O}(\mathbf{q})) \cap \mathcal{G} \neq \emptyset\}$ and denote by $\mathcal{F} = SE(2) - \mathcal{Z}$ the *free configuration space*.

We define the energy required to move the object between poses by an *energy function* $U : SE(2) \times SE(2) \to \mathbb{R}$ satisfying $U(\mathbf{q}, \mathbf{q}) = 0, \forall \mathbf{q} \in SE(2)$. This is consistent with [107], in which the reference pose was implicit in the energy function. For a fixed threshold $u \in \mathbb{R}$ and reference $\mathbf{q}_0 \in SE(2)$ define the *u-energy forbidden space* by $\mathcal{Z}_u(\mathbf{q}_0) = \mathcal{Z} \cup \{\mathbf{q} \in \mathcal{C} : U(\mathbf{q}, \mathbf{q}_0) > u\}$ and the *u-energy admissible space* $\mathcal{F}_u(\mathbf{q}_0) = SE(2) - \mathcal{Z}_u(\mathbf{q}_0)$. In this work we use the following definitions of caging [107] (see Fig. 6.2):

Definition 5 (Complete and Energy-bounded caging). *A configuration* $\mathbf{q}_0 \in \mathcal{F}$ *is completely caged if* $\mathbf{q}_0$ *lies in a bounded path-component of* $\mathcal{F}$*. We call* $\mathbf{q}_0$ *a* u-energy-bounded cage of $\mathcal{O}$ with respect to U *if* $\mathbf{q}_0$ *lies in a bounded path-component of* $\mathcal{F}_u(\mathbf{q}_0)$*. Furthermore, the* minimum escape energy*,* u^**, for an object* $\mathcal{O}$ *and obstacle configuration* $\mathcal{G}$*, is the infimum over values of* u *such that* q *is not a* u*-energy-bounded cage of* $\mathcal{O}$*, if a finite such* u^* *exists. Otherwise, we define* $u^* = \infty$.

While energy-bounded cages can be defined for any energy function U, finding bounded components of $\mathcal{C}$ for all possible pairs of poses in the energy function may be computationally expensive. Thus, for synthesis, we require that the energy function can be derived from a univariate *potential function* $P : SE(2) \to \mathbb{R}$: $U(\mathbf{q}_i, \mathbf{q}_j) = P(\mathbf{q}_i) - P(\mathbf{q}_j)$. In this work, we further assume that P depends only on the translational component $\mathbb{R}^2$ of $SE(2)$ and that P is concave on that space, which guarantees that the point of minumum potential within any convex set is on the boundary of the set. Given such an energy field U, the objective is to synthesize all energy-bounded cages $\mathbf{q}_i \in SE(2)$ with nonzero minimum escape energy.

6.1.2 Energy Functions

We now derive energy functions for gravity in the vertical plane and constant force pushing in the horizontal plane. We develop such functions based on the energy (mechanical work)

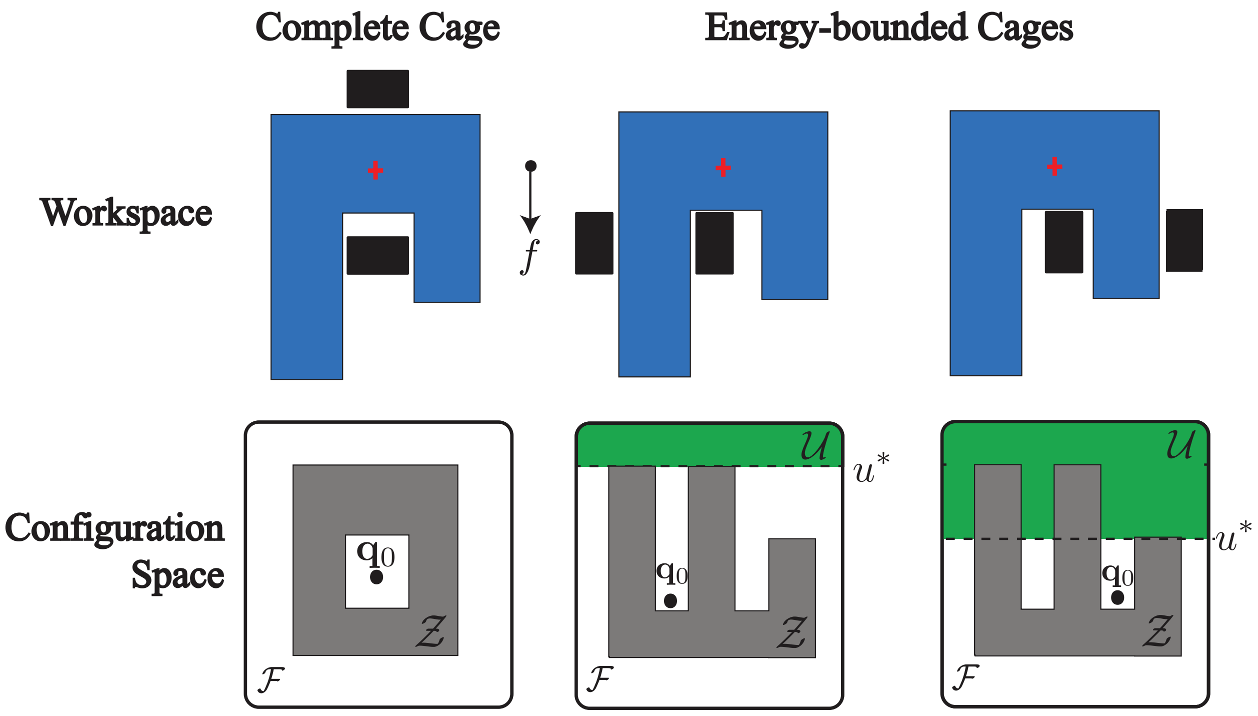


Figure 6.2: Definition of caging and energy-bounded caging. The top row depicts gripper jaws $\mathcal{G}$ (in black) and an object $\mathcal{O}$ (in blue) in three configurations. The bottom row illustrates conceptually the corresponding point $\mathbf{q}_0 \in SE(2)$ in configuration space. While a complete cage corresponds to an initial pose $\mathbf{q}_0$ completely enclosed by forbidden space $\mathcal{Z}$, the energy-bounded cage on the right instead correpsonds to a case where $\mathbf{q}_0$ is enclosed by $\mathcal{Z}_u = \mathcal{Z} \cup \mathcal{U}(\mathbf{q}_0, u)$ where $\mathcal{U}(\mathbf{q}_0, u) = \{\mathbf{q} \in \mathcal{C} : U(\mathbf{q}, \mathbf{q}_0) > u\}$ for U that is strictly increasing with increasing vertical coordinate. The smallest value of u such that $\mathbf{q}_0$ is not enclosed is called the minimum escape energy, u^*.

that wrenches must exert to transport the object between two poses under a nominal wrench resulting from pushing or gravity.

Gravity in the Vertical Plane. Let m denote the mass of the object. Then the energy required to move the object from a reference configuration $\mathbf{q}_i$ to configuration $\mathbf{q}_j$ is $U(\mathbf{q}_j, \mathbf{q}_i) = mg(y_j - y_i)$, where $g = 9.81 m/s^2$ is the acceleration due to gravity in the y-direction [46, 107]. This corresponds to the potential $P(\mathbf{q}) = mgy$.

Constant-Velocity Linear Pushing in the Horizontal Plane. Consider an object being pushed along a fixed direction $\hat{\mathbf{v}} \in \mathcal{S}^1$ by a gripper with a constant velocity on a horizontal worksurface under quasi-static conditions and Coulomb friction with uniform coefficient of friction μ [114, 113, 130]. Then the energy function $U(\mathbf{q}_j, \mathbf{q}_i) = F_p \hat{\mathbf{v}} \cdot (x_j - x_i, y_j - y_i) - \kappa$ is a lower bound on the energy required to move the object from pose $\mathbf{q}_i$ to $\mathbf{q}_j$ relative to $\mathcal{G}$, where $F_p \in \mathbb{R}$ is a bound on the possible resultant force due to contact between the object and gripper and $\kappa \in \mathbb{R}$ is a bound on the possible contact torques and frictional wrenches depending on the object geometry $\mathcal{O}$, the gripper geometry $\mathcal{G}$, and

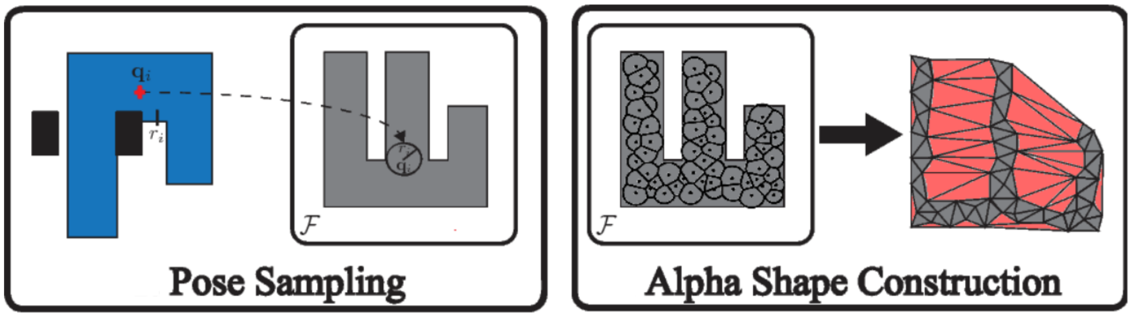


Figure 6.3: Simplicial complex approximation of configuration space. (Left) We sample a set of poses Q and their penetration depth. (Right) An approximation of the forbidden space $\mathcal{Z} \subset SE(2)$ from Fig. 6.2 by unions of balls around sampled points Q results in an α-shape simplicial complex $A(X)$ (gray triangles) that is a subset of $\mathcal{Z}$. The triangles of the weighted Delaunay triangulation $D(X)$ that are not in $A(X)$ approximate the free space (red triangles).

the friction coefficient μ. A justification is given in Section 6.5. We use the linear potential $P(\mathbf{q}) = F_p\hat{\mathbf{v}} \cdot (x, y)$ to lower bound the minimum energy required for the object to escape under the nominal push wrench.

6.1.3 Configuration Spaces and α-Complexes

We utilize a family of simplicial complexes called α-complexes [37] to approximate the collision space $\mathcal{Z}$ and u-energy forbidden space [107]. For this purpose, we first uniformly sample a collection of s poses $Q = \{\mathbf{q}_1, \dots, \mathbf{q}_s\}$, $\mathbf{q}_i = (x_i, y_i, \theta_i)$ in $\mathcal{Z}$ and determine the radius $r(\mathbf{q}_i) > 0$ for each $\mathbf{q}_i$, such that the metric ball $\mathbb{B}(\mathbf{q}_i) = \{\mathbf{q} \in SE(2) : d(\mathbf{q}, \mathbf{q}_i) \leqslant r(\mathbf{q}_i)\}$ is completely contained in $\mathcal{Z}$. These radii are computed using algorithms to lower bound the penetration depth [189] using the standard metric d on $SE(2)$; details can be found in [107]. The union of these balls $B(Q) = \cup_{i=1}^{s} \mathbb{B}(\mathbf{q}_i)$ forms a subset of the collision space that approximates $\mathcal{Z}$. See the left part of Fig. **??** for a conceptual illustration.

We can construct a cell-based approximation to $\mathcal{Z}$ using weighted α-shapes to guarantee that the cells are a subset of $\mathcal{Z}$. First, we follow the approach of [107] to lift samples from Q to a set $X \subset \mathbb{R}^3$ for computational reasons (see [107] for details). We then construct a weighted α-shape representation of $B(X)$ [37, 38] since the shape of the union of balls is difficult to analyze computationally.

Weighted α-shapes are a type of *simplicial complex*. A geometric k-simplex $\sigma = [\mathbf{v}_0, \dots, \mathbf{v}_k]$ in $\mathbb{R}^d$ is a convex hull of $k+1$ ordered affinely independent elements $\mathbf{v}_0, \dots, \mathbf{v}_k \in \mathbb{R}^d$ and a convex hull of an ordered subset of these elements is called a face τ of σ, indicated by $\tau \leqslant \sigma$. A finite simplicial complex $\mathcal{K}$ is a non-empty set of simplices such that if $\sigma \in \mathcal{K}$ and $\tau \leqslant \sigma$, then $\tau \in \mathcal{K}$ and if $\sigma, \sigma' \in \mathcal{K}$ then $\sigma \cap \sigma'$ is empty or an element of $\mathcal{K}$. In dimension 3, a simplicial complex $\mathcal{K}$ is a union of points, line-segments, triangles and tetrahedra whose intersections are either empty or another simplex in $\mathcal{K}$, thus generalizing the idea of both a graph and a triangulation in $\mathbb{R}^3$. The α-shape simplicial complex $A(X)$ corresponding to

$B(X)$ lies strictly inside $B(X)$ and is homotopy-equivalent to $B(X)$, meaning that topological properties of $B(X)$ can be computed directly from $A(X)$ [37]. Additionally, all simplices in $A(X)$ are contained in $D(X)$, the weighted Delaunay triangulation of X, a data structure that triangulates the convex hull of X. Fig. 6.3 provides a conceptual illustration.

6.1.4 Persistent Second Homology

Persistent Homology [36] studies the topological features (e.g. holes, voids) that are created and destroyed over one parameter families of simplicial complexes called *filtrations*. Fig. 6.4 provides a conceptual visualization of 2D slices of "voids" found by persistence for a 3D filtration and a qualitative persistence diagram. A simplex-wise filtration of a simplicial complex $K = \cup_{i=1}^{n} \sigma_i$ is a collection of simplicial complexes K_i such that $K_i = \cup_{i=1}^{i} \sigma_i$, so that K_{i+1} is the result of adding a single simplex σ_{i+1} to K_i. We call i the filtration index. Such a filtration can arise naturally when a function $f : K \to \mathbb{R}$ is defined on the set simplices of K and simplices are ordered in decreasing values of f: $f(\sigma_i) \geqslant f(\sigma_j)$ for all $i \leqslant j$. Thus persistence finds the topological features that emerge as the simplices are added in order of decreasing f. Here, $f(\sigma_i)$ is called the filtration value corresponding to filtration index i. The j-th persistence diagram measures the dimension of the j-th homology group $H_j(K_i)$ that corresponds to a vector space (with finite field coefficients). The dimension of each of these spaces is a topological invariant that does not vary under continuous deformations of the underlying simplicial complex $H(K_i)$. In this work, we are interested in sub-complexes K_i of the weighted Delaunay triangulation $D(X) \subset \mathbb{R}^3$ and the *second homology group* $H_2(K_i)$. Regions of space that are completely enclosed by K_i correspond to components of $H_2(K_i)$ [39]. These voids in K_i can appear as we add new simplices with increasing i, or they can disappear as voids are filled in. The persistent second homology diagram enables us to visualize these topological changes. Each point (x, y) in the diagram corresponds to a pair of filtration indices (i, j) recording the fact that a void has "appeared" at index i and disappeared at index j. For a geometric simplicial complex, these index pairs (i, j) correspond to simplices (σ_i, σ_j) where σ_j is a *tetrahedron* (a 3-simplex) which destroys or "fills in" a void, while σ_i corresponds to a *triangle* (2-simplex) that corresponds to the last complex needed to first create a fully enclosed void. The set of (i, j) pairs can be displayed in the (index)-persistence diagram, or alternatively, when the filtration arises from a function f, we may display the set of points $(f(\sigma_i), f(\sigma_j))$. By considering the vertical distance $|f(\sigma_i) - f(\sigma_j)|$ from the diagonal, we can read off the parameter range of f during which a void exists in the evolution of the filtration.

6.2 The EBCS-2D Synthesis Algorithm

EBCS-2D (Algorithm 3) takes as input a polygonal object $\mathcal{O}$, obstacle configuration $\mathcal{G}$, and continuous concave potential function P, and outputs a set of energy-bounded cages that require nonzero energy to escape.

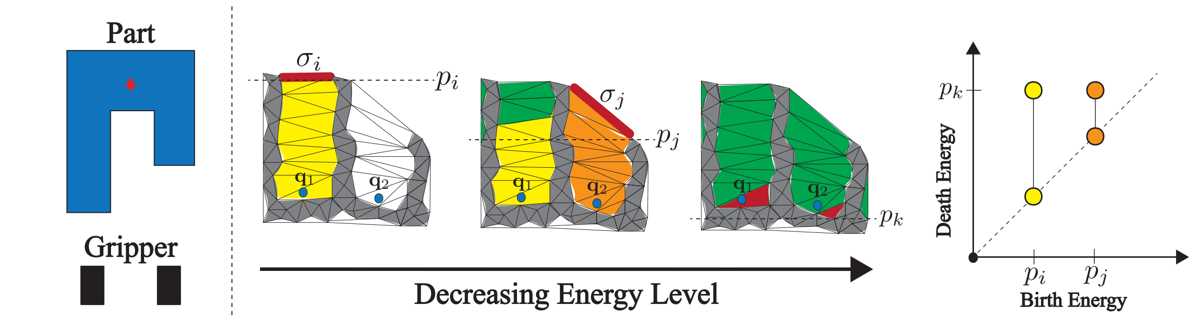


Figure 6.4: Persistence diagram for ranking energy-bounded cages. Left: polygonal part and gripper polygons serve as input. We sample object poses X in collision and generate an α-shape representation (shown in gray in the three middle figures). Given an energy potential, we insert simplices in $D(X) - A(X)$ in decreasing order of energy potential, creating a filtration of simplicial complexes. Voids (yellow and orange) are born with the addition of edges σ_i and σ_j (red) at threshold potential levels p_i and p_j respectively, and die with the additions of the last triangle in each void at potential p_k (red). The associated second persistence diagram reveals voids corresponding to energy-bounded cages. In particular, configuration $\mathbf{q}_1$ is persistent for a larger energy difference than configuration $\mathbf{q}_2$ (right figure). The escape energy of each configuration is equal to the difference in potentials: $u_1 = p_k - p_i$ and $u_2 = p_k - p_j$, and by the filtration ordering this implies that $\mathbf{q}_1$ has higher escape energy than $\mathbf{q}_2$.

Using uniform sampling, the algorithm first generates s object poses in collision $Q = \{\mathbf{q}_1, ..., \mathbf{q}_s\}$ and their corresponding penetration depths $R = \{r_1, ..., r_s\}$. We lift the poses to $\mathbb{R}^3$ and construct an α-shape approximation to the configuration space. Next, we construct a filtration by sorting all simplices in the free space in order of decreasing energy level and use persistent homology to identify path components that are bounded by the u-energy forbidden space. Finally, we examine the simplices within each bounded component in order of increasing energy to check for a collision-free object pose, and return the poses extracted from each component. Fig. 6.4 illustrates the use of persistence in our algorithm.

6.2.1 Filtrations and Persistence from Energy Functions

To synthesize energy-bounded cages with persistence, we first order the simplices of the α-shape approximation by decreasing energy level. We assumed that the potential $P : SE(2) \to \mathbb{R}^3$ depends only on the translational component $\mathbb{R}^2$ of $SE(2)$ and is concave on that space. In this case, for any k-simplex $\sigma = \text{Conv}(\mathbf{v}_0, \ldots, \mathbf{v}_k) \in D(X) - A(X)$ the maximum principle of convex optimization [13] implies that the minimum occurs on the boundary of the simplex, which is a vertex of σ since $\mathbb{R}^2$ is unbounded:

$$\min_{\mathbf{x}\in\sigma} P(\pi(\mathbf{x})) = \min\{P(\pi(\mathbf{v}_0)), \ldots, P(\pi(\mathbf{v}_k))\}$$

where $\pi : \mathbb{R}^3 \to SE(2)$ denotes the projection to $SE(2)$. Using this fact, we construct a function $D(X) \to \mathbb{R}$:

$$f(\sigma) = \begin{cases} \min_{\mathbf{x}\in\sigma} P(\pi(\mathbf{x})) & \sigma \in D(X) - A(X) \\ \infty & \sigma \in A(X) \end{cases}$$

This gives rise to a filtration $K = K(X, U) : \emptyset = K_0 \subset K_1 \subset \ldots \subset K_n \subset D(X)$ of simplices in $D(X)$ with respect to P as described in Section 6.3.3, which we can use to find bounded path components corresponding to energy-bounded cages.

EBCS-2D finds pairs of simplices σ_i, σ_j corresponding to the birth and death, respectively, of a bounded path component $C(X) \subset D(X)$ in the free configuration space using persistent homology. All collision-free configurations within the bounded path component are energy-bounded cages by definition. Therefore, EBCS-2D next searches for the configuration $\mathbf{q} \in C(X) \cap \mathcal{F}$ with the highest minimum escape energy by iterating over the set of centroids of simplices in $C(X)$. While the set of simplex centroids only approximates $C(X) \cap \mathcal{F}$, in practice the centroids cover the space well due to the large number of samples used to construct the configuration space. The algorithm runs in $O(s^3 + sn^2)$ time where s is the number of samples and n is the total number of object and obstacle vertices, since α-shape construction is $O(s^2 + sn^2)$ [107, 115] and the matrix reduction used in persistent homology is $O(s^3)$ in the worst case [19].

6.2.2 Correctness

EBCS-2D returns energy-bounded cages with a provable lower bound on the minimum escape energy:

Theorem 1. *Let $\hat{Q} = \{(\hat{\mathbf{q}}_1, \hat{u}_1), ...(\hat{\mathbf{q}}_n, \hat{u}_n)\}$ denote the energy bounded cages returned by EBCS-2D. For each $(\hat{\mathbf{q}}_i, \hat{u}_i) \in \hat{Q}$, $\hat{\mathbf{q}}$ is a $\hat{u}$-energy bounded cage of $\mathcal{O}$ with respect to U.*

A detailed proof is given in Section 6.4.

6.2.3 Extension to Pushing

EBCS-2D can be applied to push grasping in the horizontal plane. We use it to find push directions that yield robust energy-bounded cages by running EBCS-2D for a set of sampled push directions using the constant velocity linear push energy of Section 6.1.2. The extension runs EBCS-2D using M push angles uniformly sampled from $[\frac{\pi}{2} - \varphi, \frac{\pi}{2} + \varphi]$ and returns a ranked list of push directions and energy-bounded cages that can be reached by a linear, collision-free path along the push direction. While the potential changes for each such push direction, the simplices only need to be re-sorted and therefore the sampling and α-complex construction only need to be performed once.

```
1  Input: Polygonal robot gripper $\mathcal{G}$, Polygonal object $\mathcal{O}$, Potential function $P$, Number of
     pose samples $s$
   Result: $\hat{Q}$, set of energy-bounded cages with estimated escape energies
   // Sample poses in collision
2  $Q = \varnothing, R = \varnothing, \ell = diam(\mathcal{G}) + diam(\mathcal{O})$;
3  $\mathcal{W} = [-\ell, \ell] \times [-\ell, \ell] \times [0, 2\pi)$;
4  for $i \in \{1, ..., s\}$ do
5      $\mathbf{q}_i$ = RejectionSample($\mathcal{W}$);
6      $r_i$ = LowerBoundPenDepth($\mathbf{q}_i, \mathcal{O}, \mathcal{G}$);
7      if $r_i > 0$ then
8          $Q = Q \cup \{\mathbf{q}_i\}$, $R = R \cup \{r_i\}$;
9  end
10 $X$ = ConvertToEuclideanSpace($Q$);
   // Create alpha shape
11 $D(X, R)$ = WeightedDelaunayTriangulation($X, R$);
12 $A(X, R)$ = WeightedAlphaShape($D(X, R), \alpha = 0$);
   // Run Persistent Homology
13 $K$ = Filtration($D(X, R), A(X, R)$);
14 $\Delta$ = ComputeSecondHomologyPersistencePairs($K$);
   // Find Energy-Bounded Cages
15 for $(i, j) \in \Delta$ do
16     $C(K_i, \sigma_j)$ = PathComponent($\sigma_j$, $K_i$);
17     if Bounded($C(K_i, \sigma_j)$) then
18         for $\sigma \in$ Sorted($C(K_i, \sigma_j)), P$) do
19             $\mathbf{q}$ =Centroid($\sigma$);
20             $u = P(\sigma_i) - P(\mathbf{q})$;
21             if CollisionFree($\mathbf{q}$) and $u > 0$ then
22                 $\hat{Q} = \hat{Q} \cup \{(\mathbf{q}, u)\}$;
23             end
24         end
25     end
26 end
27 return $\hat{Q}$;
```

Algorithm 3: Energy-Bounded-Cage-Synthesis-2D

6.3 Experiments

We implemented EBCS-2D in C++ and evaluated its performance on a set of polygonal objects under both gravitational and pushing energy fields. We used CGAL [167] to compute α-shapes, the GJK-EPA algorithm of libccd [43] to compute penetration depth, and the twist reduction algorithm implemented in PHAT [8] to compute the second persistence diagram. Our dataset consisted of seven polygonal parts created by triangulating the projections of models from YCB [16] and 3DNet [180] onto a plane. All experiments ran on an Intel Core i7-4770K 350GHz processor with 6 cores.

6.3.1 Energy Bounded Cages Under Linear Push Energy

We consider a linear push energy field with a push force bound of $F_p = 1.0$ for the set of parts with four grippers: rectangular parallel jaws, an overhead projection of a Zymark Zymate gripper with parallel jaws [95], an overhead projection of a Barrett hand with a pre-grasping shape inspired by [33], and a four finger disc gripper inspired by [163]. We ran the pushing extension to EBCS-2D for the rectangular parallel jaws, Zymark gripper, and Barrett hand with $s = 200,000$ samples, an angle limit of $\varphi = \frac{\pi}{4}$, and $P = 5$ push directions to sweep from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$ in intervals of $\frac{\pi}{8}$, and pruned all pushes with $\hat{u} < 0.5$ to ensure that our set of pushes was robust. For the four finger gripper, we ran EBCS-2D with a fixed vertical push direction to illustrate the ability of our algorithm to prove complete cages. EBCS-2D took approximately 170 seconds to run on average for a single push direction. Fig. 6.5 illustrates configurations synthesized by EBCS-2D with the estimated minimum escape energy $\hat{u}$, which is the distance against the linear push energy that the object must travel to escape. To evaluate the lower bound of Theorem 1, we also used RRT* to attempt to plan an object escape path over the set of collision-free poses with energy less than $\hat{u}$, which was not able to find an escape path with energy less than $\hat{u}$ in 120 seconds of planning [107].

6.3.2 Sample and Time Complexity

We also studied the sensitivity of the estimated escape energy for the highest energy configurations synthesized by EBCS-2D for a fixed push direction and the algorithm runtime to the number of pose samples s.

The left panel of Fig. 6.6 shows the ratio of $\hat{u}$ for $s \in \{12.5, 25, 50, 100, 200, 400\} \times 10^3$ pose samples to $\hat{u}$ at $s = 400,000$ pose samples for each of the displayed objects and gripper configurations. The top panel shows results using a parallel-jaw gripper and the bottom panel shows results with a Zymark Zymate gripper to illustrate sensitivity to the complexity of the polygonal gripper model. We averaged the ratios over 5 independent trials per value of s. Case A is only within 80% of the value at $s = 400,000$ after $s = 200,000$ samples, possibly because of the long thin portion of the configuration space as observed in [107]. Case B and C both converge to within 95% after about $s = 200,000$ samples. For cases D, E, and F with the complex Zymate gripper, all configurations require more samples to converge, possibly due to the thin portions of the gripper tips. The sample complexity is comparable to analysis of a single, fixed configuration with EBCA-2D.

The right panel of Fig. 6.6 shows the relationship between the runtime of EBCS-2D in seconds versus the number of pose samples s over 5 independent runs of the algorithm for the same objects. We broke down the run time by the section of the algorithm: sampling poses, constructing the α-shape to approximate $\mathcal{C}$, sorting the simplices for the filtration, and computing and pruning candidate energy bounded cages with persistence. The runtime is approximately linear in the number of pose samples, and the largest portion of runtime is the time to sample poses and compute penetration depth. This suggests that the runtime is considerably below the worst case s^3 scaling in practice. The persistence diagram com-

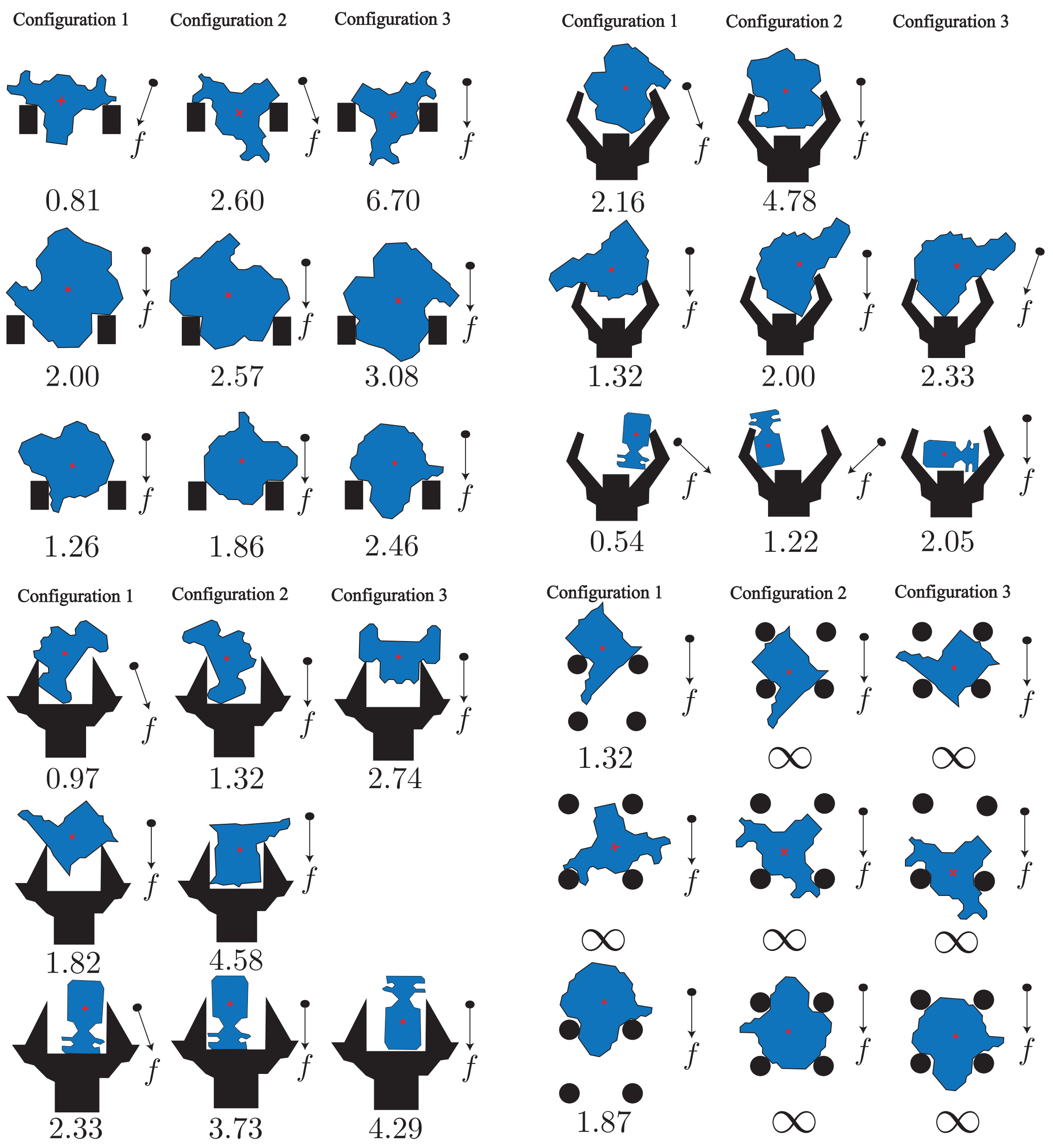


Figure 6.5: Highest energy configurations and push directions synthesized using EBCS-2D ranked from left to right for seven example polygonal objects (blue) and grippers (black) under a linear planar pushing energy field with a push force bound of $F_p = 1.0$. Displayed are three objects for each of the following grippers: (left-to-right, top-to-bottom) parallel-jaw grippers with rectangular jaws, a Barrett hand with fixed preshape, a Zymark Zymate gripper with fixed opening width, and a four finger disc gripper. Below each object the escape energy $\hat{u}$ estimated by EBCS-2D using $s = 200,000$ pose samples, which is the distance the object would have to travel against the pushing direction, and to the right is the synthesized push energy direction f. For each test case we searched over 5 energy directions from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$ and checked push reachability as described in Section 6.2.3 except for the four finger gripper, for which we ran only EBCS-2D to illustrate complete cages. EBCA-2D synthesizes several complete cages for the four finger gripper.

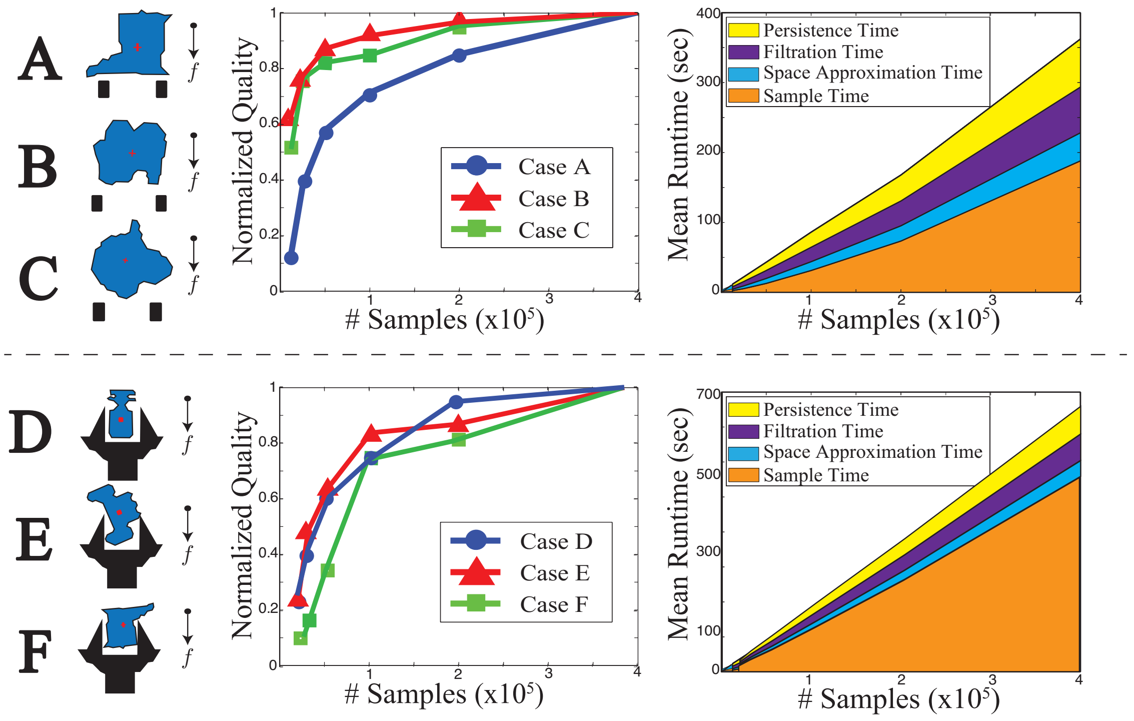


Figure 6.6: Sensitivity analysis of EBCS-2D. (Middle) The sample complexity of EBCS-2D. Plotted is the ratio of the highest minimum escape energy out of the energy-bounded cages synthesized by EBCS-2D, $\hat{u}^*$, for the number of pose samples $s = \{12.5, 25, 50, 100, 200, 400\} \times 10^3$ on the object and gripper test cases displayed on the left. Performance is broken down by the polygonal gripper model used: parallel-jaw grippers (Top) and a Zymark Zymate gripper (Bottom). (Right) The mean runtime of EBCS-2D in seconds is broken down by component of the algorithm for varying numbers of pose samples $s = \{12.5, 25, 50, 100, 200, 400\} \times 10^3$. Each datapoint is averaged over five independent runs for each of the object and gripper configurations on the left. Despite the theoretical worst case s^3 runtime, the algorithm runtime is approximately linear in s, and is dominated by sampling time.

putation in particular has been observed to commonly exhibit sub-quadratic runtime [19] despite its worst-case cubic complexity. Runtime approximately doubles with the Zymate gripper due to an increase in sampling time, consistent with the quadratic time complexity of EBCS-2D with respect to the number of object and obstacle vertices.

6.3.3 Persistence Diagrams

To further illustrate the notion of persistence, we study the persistence diagrams of the second homology group for a single object and the Zymark Zymate gripper in Fig. 6.7. To generate the diagram we constructed the weighted Delauanay triangulation and α-shape using $s = 200,000$ pose samples and examined the list Δ (generated on Line 14 of the EBCS-2D pseudocode in Algorithm 3). We see that the three energy-bounded cages returned by EBCS-2D correspond to the three most persistent pairs, which appear furthest from

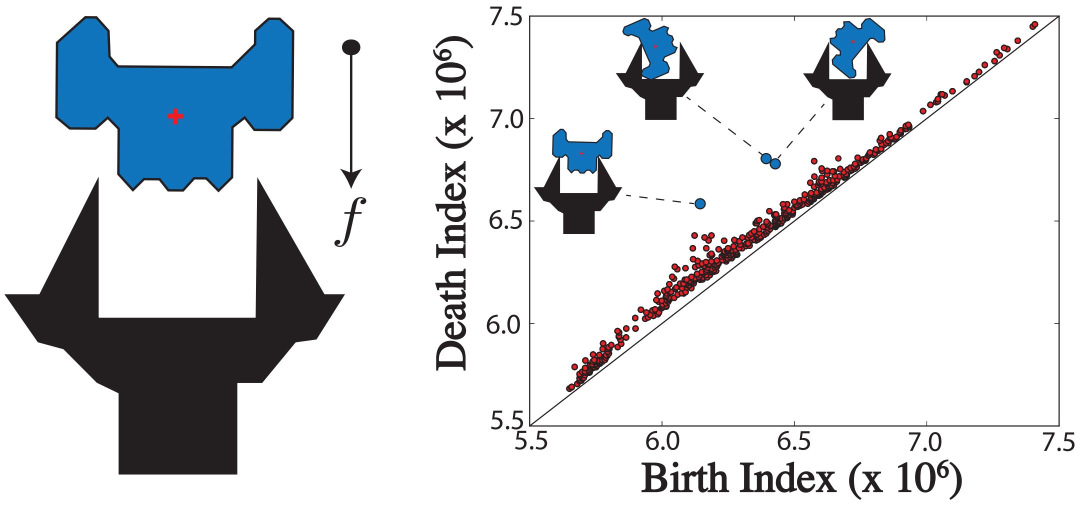


Figure 6.7: Persistence diagram for the second homology persistence pairs (corresponding to "voids") in the filtration K identified during a run of EBCS-2D with $s = 200,000$ pose samples for a part (blue) and gripper configuration (black) with a vertical push force. The (i, j) coordinate for each point corresponds to the birth and death indices of the voids. Points in red were pruned by our algorithm. The three points in blue were identified by EBCS-2D as energy-bounded cages, and their corresponding workspace configurations are illustrated next to the points. Note that the magnitude of differences between indices may not be indicative of the magnitude of energy differences between configurations.

the diagonal. Furthermore, our algorithm correctly rejects the large number of candidate configurations with very low persistence.

6.3.4 Physical Experiments

We evaluated the performance of energy-bounded cages synthesized by EBCS-2D in pushing and grasping planar objects on two physical robots.

Known Object Pose

We evaluated the pushes synthesized by EBCS-2D for the three object configurations on a Zymark Zymate robot with the Zymark gripper illustrated in Fig. 6.5 on a set of extruded fiberboard polygonal parts [95] to evaluate performance when the exact object pose is known, as is common in industrial automation. The top panel of Fig. 6.8 illustrates this experiment. For each configuration, the object was placed in the center of a turntable on a template to register the object pose, rotated to align the push direction with the arm's major axis, and pushed forward while the turntable oscillated with an amplitude of ± 0.1 radians to simulate external wrenches on the object. To test robustness we added zero-mean Gaussian noise with standard deviation of 5mm to the gripper translation and 0.04 radians to the gripper rotation in the plane. We then evaluated whether or not the object was captured and remained within the gripper jaws after being pushed 150mm. Pushes planned by EBCS-2D had a success rate of 100% versus 41% for a baseline of pushes planned by choosing gripper poses uniformly at random from (x, y) in the object bounding box and θ in $[0, 2\pi)$.

Image-Based Pose Registration

We also evaluated planar pushing on a set of six 3D objects using an ABB YuMi with a parallel-jaw gripper using image-based registration to index a planned push from a database of pushes synthesized with EBCS-2D to evaluate performance when the object pose is not known a priori. The bottom panel of Fig. 6.8 illustrates our experimental setup. First, we detected and segmented the each object from the background using color background subtraction with images from an overhead Primesense Carmine 1.08. We extruded and triangulated the segmentation masks and used EBCS-2D to plan energy-bounded push-cages. To execute a planned push, the object was placed in the center of the planar worksurface by a human operator and the object was registered by minimizing the pixel-wise difference between the new and original segmentation mask over all possible orientations. The robot then attempted to push the object 10cm and lift the object by closing the jaws on the object after the attempted push. We added zero-mean Gaussian noise with standard deviation of 2.5mm to the gripper translation in the plane to test robustness to perturbations.

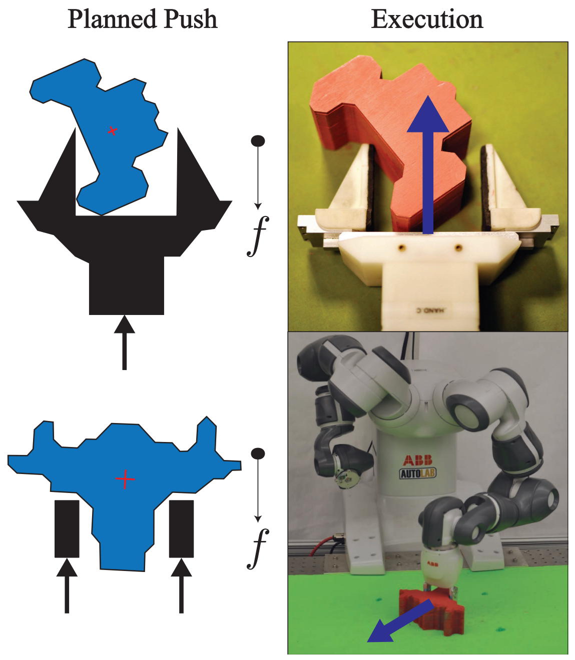


Figure 6.8: Experimental setups for executing energy-bounded cages synthesized with EBCS-2D on a Zymark Zymate robot (Top) and ABB YuMi robot (Bottom). The Zymate was used to test performance when exact object pose was known and the YuMi was used to test performance when planning based on object segmentation masks in images. (Left) The synthesized planar configuration for each manipulator. (Right) The object remains in the gripper as it is pushed along a planar worksurface.

Table 6.1 summarizes the performance of energy-bounded cages performed on the ABB YuMi for capturing the object (keeping it between the jaws), pushing the object 10cm, and grasping and lifting the object. We evaluated each for four trials with the object rotated by $\frac{\pi}{2}$ on each trial, and we compared against the random baseline used in the Zymark Zymate experiments. The most common failure mode occurred when the parallel jaws contacted the object before reaching the target gripper configuration, suggesting that modeling uncertainty in the gripper approach could improve performance.

Method	Captures (%)	Pushes (%)	Grasps (%)
Random	36	66	18
EBCS-2D	**98**	**79**	**86**

Table 6.1: Performance of energy-bounded cages planned by EBCS-2D for capturing, pushing, and grasping six planar test objects with an ABB YuMi with parallel-jaw grippers using image-based registration to determine the object pose. We compare with a random baseline that selects random gripper orientations and target gripper positions uniformly at random from the object bounding box. We executed 14 planned pushes for each method, each for 4 trials.

6.4 Proof of Correctness for the EBCS-2D Algorithm

Energy-Bounded-Cage-Synthesis-2D (EBCS-2D) synthesizes energy-bounded cages of a polygonal object $\mathcal{O}$ by a rigid configuration of polygonal obstacles $\mathcal{G}$ with respect to a continuous energy function $U : SE(2) \times SE(2) \to \mathbb{R}$. We require that the energy function U can be derived from a univariate potential function $P(\mathbf{q}) : SE(2) \to \mathbb{R}$, $U(\mathbf{q}_i, \mathbf{q}_j) = P(\mathbf{q}_i) - P(\mathbf{q}_j)$. We refer to poses as $\mathbf{q} = (x, y, z) \in SE(2)$. See Algorithm 3 for the EBCS-2D pseudocode or Section 6.1 for further definitions.

Theorem 2 (Correctness of EBCS-2D)**.** *Assume the object is specified as a compact polygon $\mathcal{O} \subset \mathbb{R}^2$ and the obstacles are defined a rigid configuration of a set of k polygons $\mathcal{G} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k \subset \mathbb{R}^2$. Furthermore, assume the energy function $U : SE(2) \times SE(2) \to \mathbb{R}$ satisfies the following:*

1. *$U(\mathbf{q}, \mathbf{q}) = 0$ for all $\mathbf{q} \in SE(2)$.*
2. *$U(\mathbf{q}_i, \mathbf{q}_j) = P(\mathbf{q}_i) - P(\mathbf{q}_j)$ for all $\mathbf{q} \in SE(2)$ and a potential function $P : SE(2) \to \mathbb{R}$.*
3. *P is continuous.*
4. *$P((x, y, \cdot)) = c$ for some $c \in \mathbb{R}$ (P does not depend on the orientation).*
5. *P is concave on the translational component $\mathbb{R}^2$.*

Let $\hat{\mathcal{Q}} = \{(\hat{\mathbf{q}}_i, \hat{u}_i)\}_{i=1}^N$ be the list of poses returned by EBCS-2D. For each $(\hat{\mathbf{q}}, \hat{u}) \in \hat{\mathcal{Q}}$, $\hat{\mathbf{q}}$ is a $\hat{u}$-energy bounded cage of $\mathcal{O}$ with respect to U.

Proof. It suffices to show that $(\hat{\mathbf{q}}, \hat{u})$ will only be added to the solution set $\hat{\mathcal{Q}}$ if $\hat{\mathbf{q}}$ is a $\hat{u}$-energy bounded cage of $\mathcal{O}$.

Lemma 1. *Let $D(X, R)$ denote the weighted Delaunay triangulation of the pose samples X and penetration depths R (computed on Line 11). Let $A(X, R) \subset D(X, R)$ denote the weighted alpha shape of X and R at $\alpha = 0$ (computed on Line 12). Let π be the covering map* *defined in Section 4 of [107]. Given any $p \in \mathbb{R}$, let $W_p(X, R) = \{\sigma \in D(X, R) \mid f(\sigma) > p\}$ denote the p-*potential forbidden subcomplex *of X, R, where:*

$$f(\sigma) = \begin{cases} \min_{x \in \sigma} P(\pi(x)) & \sigma \in D(X) - A(X) \\ \infty & \sigma \in A(X) \end{cases}$$

For any pose $\mathbf{q} \in SE(2)$ and $u \in \mathbb{R}$, let $p(\mathbf{q}) = u + P(\mathbf{q})$. Then if $\mathbf{q} \in \mathcal{F}$ is in a bounded path component of $\mathcal{C} - \pi(W_{p(\mathbf{q})}(X, R))$, $\mathbf{q}$ is a u-energy bounded cage of $\mathcal{O}$.

Proof. The p-potential forbidden subcomplex $W_{p(\mathbf{q})}(X, R) \subset V_u(X, R)(\mathbf{q})$ the *u-energy forbidden subcomplex* of X, R with respect to $\mathbf{q}$ defined in [107]. This is because $\forall \sigma \in W_{p(\mathbf{q})}(X, R)$, either $\sigma \in A(X, R) \Rightarrow \sigma \in V_u(X, R)(\mathbf{q})$ or $P(\sigma) > p(\mathbf{q}) \Leftrightarrow P(\mathbf{q}_i) > p(\mathbf{q})$ $\forall \mathbf{q}_i \in \sigma \Leftrightarrow P(\mathbf{q}_i) - P(\mathbf{q}) = U(\mathbf{q}_i) > u$. Recall that $\pi(V_u(X, R)(\mathbf{q})) \subset \mathcal{Z}_u$, the u-energy forbidden space defined in [107], and therefore $\mathcal{C} - \pi(W_{p(\mathbf{q})}(X, R)) \supset \mathcal{F}_u(\mathbf{q})$, the u-energy admissible space defined in [107]. Therefore any path in $\mathcal{C} - \pi(W_{p(\mathbf{q})}(X, R))$ can be restricted to $\mathcal{F}_u(\mathbf{q})$ which implies that $\mathbf{q}$ lies in a bounded-path component of $\mathcal{F}_u(\mathbf{q})$. Thus, by definition $\mathbf{q}$ is a u-energy-bounded cage. □

Now define the filtration $K(X, P) : \emptyset = K_0 \subset K_1 \subset \ldots \subset K_n \subset D(X, R)$ of simplices in $D(X, R)$ with respect to f. Let $\mathcal{I} = \{(i_m, j_m)\}_{m=1}^{M}$ denote the set of k persistence pairs for $K(X, P)$ such that $dim(\sigma_{i_m}) == 2$. Then any pair $(i, j) \in \mathcal{I}$ corresponds to the birth and death of a class of the second homology group H_2. We take this as an indication that $\pi(\sigma_k)$ lies in a bounded path component $\pi(C(K_i, \sigma_j)) \subset \pi(D(X, R) - K_i)$ and additionally verify boundedness using a flood-fill algorithm [115, 107].

To verify that the component is bounded with respect to the $\hat{u}$ energy forbidden space, let $p = P(\sigma_i)$ and $u(\mathbf{q}) = P(\sigma_i) - P(\mathbf{q})$ for any $\mathbf{q} \in C(K_i, \sigma_j) \cap \mathcal{F}$, if such a pose exists. By the definition of the filtration, $K_i = W_p(X, R)$. By Lemma 1 any $\mathbf{q} \in C(K_i, \sigma_j) \cap \mathcal{F}$ is in a bounded path component and is therefore a $u(\mathbf{q})$-energy-bounded cage of $\mathcal{O}$. EBCS-2D only returns $\mathbf{q}, u$ for $u = P(\sigma_i) - P(\mathbf{q})$ (Line 19) if $\mathbf{q}$ is collision free (Line 20) and in the same bounded path component as σ_j (Line 16). Therefore $\hat{\mathbf{q}}$ is a $\hat{u}$-energy-bounded cage of $\mathcal{O}$ with respect to U.

6.5 Energy Function for Constant Velocity Quasi-Static Planar Pushing

Consider a compact polygonal object $\mathcal{O} \subset \mathbb{R}^2$ of mass m_O and a rigid configuration of a set of k polygonal obstacles $\mathcal{G} = \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_k \subset \mathbb{R}^2$ of total mass m_G. Let $m = m_O + m_G$ be the total mass. Denote by $\mathbf{q} \in SE(2)$ the pose of $\mathcal{O}$ relative to the reference frame of $\mathcal{G}$. Assuming quasi-static conditions and a Coulomb friction model, let the object and gripper rest on a horizontal worksurface under gravity with a uniform coefficient of friction between the gripper, object, and surface: $\mu \in \mathbb{R}$. Assume a uniform pressure distribution for both $\mathcal{O}$ and $\mathcal{G}$ and let the center of mass of each be located at the centroid of the respective pressure distributions. Assume that the magnitude of any external wrench $w_e = (f_e, \tau_e)$ on the object is bounded by a constant λ.

Now let $\mathcal{G}$ move along a fixed direction $\hat{v} \in \mathcal{S}^1$ with constant translational velocity of magnitude $\beta \in \mathbb{R}$ and zero angular velocity. Zero net force must be acting on $\mathcal{G}$ and $\mathcal{O}$ to maintain this velocity, and therefore the force due to pushing f_p is equal and opposite of the forces due to friction f_f and forces due to external perturbations f_e. Thus the pushing force is bounded by the maximum force due to friction and maximum magnitude of external wrenches: $\|f_p\|_2 \leqslant \mu M g + \lambda$. This force may be applied to $\mathcal{O}$ through contact with $\mathcal{G}$, and therefore f_p may exert a torque τ_p on $\mathcal{O}$ relative to $\mathcal{G}$ such that $\tau_p \leqslant \rho f_p$, where $\rho \in \mathbb{R}$ is the maximum moment arm of $\mathcal{O}$ [114, 113, 130].

To derive the energy function, consider the amount of energy (mechanical work) that the time-varying external wrench $w_e(t) = (f_e(t), \tau_e(t))$ would have to exert to move $\mathcal{O}$ along a continuous path $\gamma : [0,1] \to SE(2)$ from pose $\mathbf{q}_i$ to $\mathbf{q}_j$ (e.g. $\gamma(0) = \mathbf{q}_i, \gamma(1) = \mathbf{q}_j$) with a constant speed η under the pushing wrench $w_p = (f_p, \tau_p)$ and time-varying wrenches due to friction $w_f(t) = (f_f(t), \tau_f(t))$ [46]:

$$\mathcal{E}(w_e) = \int_0^1 w_e(t) \cdot \dot{\gamma}(t) dt$$

By the constant speed assumption, the kinetic energy of the object does not change. Therefore the net work done on the object over the path γ is zero due to conservation of energy:

$$\begin{aligned} \mathcal{E}(w_e + w_p + w_f) &= \int_0^1 (w_e(t) + w_p + w_f(t)) \cdot \dot{\gamma}(t) dt = 0 \\ &\Rightarrow \mathcal{E}(w_e) = -\mathcal{E}(w_p + w_f) \end{aligned}$$

We can upper bound the amount of work done by the constant pushing wrench w_p and time-dependent frictional wrench w_f using Cauchy-Schwarz:

$$\begin{aligned} \mathcal{E}(w_p + w_f) &= \int_0^1 w_p \cdot \dot{\gamma}(t) dt + \int_0^1 w_f(t) \cdot \dot{\gamma}(t) dt \\ &= w_p \cdot \int_0^1 \dot{\gamma}(t) dt + \int_0^1 w_f(t) \cdot \dot{\gamma}(t) dt \\ &\leqslant (\mu M g + \lambda) \hat{\mathbf{v}} \cdot (\mathbf{x}_j - \mathbf{x}_i) \\ &+ \rho(\mu M g + \lambda)(\theta_j - \theta_i) \\ &+ \mu M g \int_0^1 \|\dot{\gamma}(t)\|_2 dt \text{ (by Cauchy-Schwarz)} \\ &\leqslant (\mu M g + \lambda) \hat{\mathbf{v}} \cdot (\mathbf{x}_j - \mathbf{x}_i) \\ &+ 2\pi\rho(\mu M g + \lambda) + \eta \mu M g \end{aligned}$$

And therefore under our assumptions we can lower bound the energy exerted by any external wrenches by:

$$\begin{aligned}U(\mathbf{q}_j, \mathbf{q}_i) &= F_p\hat{\mathbf{v}} \cdot (\mathbf{x}_j - \mathbf{x}_i) - \kappa(\mathcal{O}, \mathcal{G}, \mu)\\ F_p &= -(\mu M g + \lambda)\\ \kappa(\mathcal{O}, \mathcal{G}, \mu) &= 2\pi\rho(\mu M g + \lambda) + \eta\mu M g\end{aligned}$$

This motivates our use of the linear, univariate potential $P(\mathbf{q}) = F_p\hat{\mathbf{v}} \cdot (\mathbf{x}_j - \mathbf{x}_i)$.

6.6 Numeric Issues in Implementation

In order for EBCS-2D to be correct, the computed penetration depth r_i for a pose $\mathbf{q}_i$ must not be greater than the true 2D generalized penetration depth, $r_i \leqslant p(\mathbf{q}_i)$. This can be theoretically achieved using the lower-bound algorithm given by Zhang et al. [188] by taking the maximum of the exact penetration depth between pairs of convex pieces in a convex decomposition of the object and obstacles, where the exact penetration depth can be computed with the Gilbert-Johnson-Keerthi Expanding Polytope Algorithm (GJK-EPA) [173]. However, in practice GJK-EPA computes the exact penetration depth up to some tolerance $\pm\epsilon$. Thus to avoid mis-identifying a configuration as a complete or energy-bounded cage, in practice we use $r_i = \max(\hat{r}_i - \epsilon, 0)$, where $\hat{r}_i$ is the penetration depth computed using the algorithm of Zhang et al. [188]. To avoid further numeric issues related to imprecision in the convex decomposition, computation of the maximum moment arm, or the triangulation, in practice it may be beneficial to additionally multiply the returned penetration depth by some shrinkage factor $0 < \nu < 1$, $r_i = \nu\max(\hat{r}_i - \epsilon, 0)$.

6.7 Discussion

This chapter developed EBCS-2D, a synthesis algorithm for energy-bounded cages of polygonal objects and rigid configurations of obstacles under a 2D energy field, and use EBCS-2D to synthesize constant velocity planar pushes under Coulomb friction. One shortcoming of this work is that the extension to 3D is nontrivial; α-shape construction is exponential in the dimensionality of the input points. As such, it remains to be seen whether or not caging configurations can be used in practical applications on the physical robot.

In principle, methods developed in this section could be used to construct a dataset generation distribution to sample various polygons and compute energy-bounded cages as supervised training data for learning a caging policy that plans linear pushes from a segmentation mask of objects on a tabletop. We leave this an open problem for future work.

Chapter 7

Computing Vacuum Suction Grasps with Compliant Contact Modeling

While the majority of this thesis has focused on parallel jaw grippers, vacuum-based suction grippers are widely-used for pick-and-place tasks in industry and warehouse order fulfillment. As shown in the Amazon Picking Challenge, suction has an advantage over parallel-jaw or multifinger grasping due to its ability to reach into narrow spaces and pick up objects with a single point of contact. However, while a substantial body of research exists on parallel-jaw and multifinger grasp planning [11], comparatively little research has been published on planning suction grasps.

Grasp planning methods typically search for gripper configurations that maximize a quality metric derived from mechanical wrench space analysis [121], human labels [150], or self-supervised labels [98]. However, in practice suction grasps are often planned directly on point clouds using heuristics such as grasping near the object centroid [60] or at the center of planar surfaces [25, 34]. These heuristics work well for prismatic objects such as boxes and cylinders but may fail on objects with non-planar surfaces near the object centroid, which is common for industrial parts and household objects such as staplers or children's toys. Analytic models of suction cups for grasp planning exist, but they typically assume that a vacuum seal has already been formed and that the state (e.g. shape and pose) of the object is perfectly known [4, 86, 112]. A robot may need to form seals on non-planar surfaces while being robust to external wrenches (e.g. gravity and disturbances), sensor noise, control imprecision, and calibration errors, which are significant factors when planning grasps from point clouds.

In this chapter, we propose a novel compliant suction contact model to measure grasp rewards for rigid, non-porous objects that consists of two components: (1) a test for whether a seal can be formed between a suction cup and a target object surface and (2) an analysis of the ability of the suction contact to resist external wrenches. We use the model to evaluate grasp robustness by analyzing seal formation and wrench resistance under perturbations in object pose, suction tip pose, material properties, and disturbing wrenches using Monte-Carlo sampling similar to that in the Dexterity Network (Dex-Net) 1.0 [104].

This chapter makes four contributions:

1. A compliant suction contact model that quantifies seal formation using a quasi-static spring system and the ability to resist external wrenches (e.g. gravity) using a contact wrench basis derived from the ring of contact between the cup and object surface.

2. Robust wrench resistance: a robust version of the above model under random disturbing wrenches and perturbations in object pose, gripper pose, and friction.

3. Dex-Net 3.0, a dataset of 2.8 million synthetic point clouds annotated with suction grasps and grasp robustness labels generated by analyzing robust wrench resistance for approximately 375k grasps across 1,500 object models.

4. Physical robot experiments measuring the precision of robust wrench resistance both with and without knowledge of the target object's shape and pose.

We perform physical experiments using an ABB YuMi robot with a silicone suction cup tip to compare the precision of a GQ-CNN-based grasping policy trained on Dex-Net 3.0 with several heuristics such as targeting planar surfaces near object centroids. We find that the method achieves success rates of 98%, 82%, and 58% on datasets of Basic (prismatic or cylindrical), Typical (more complex geometry), and Adversarial (with few available suction-grasp points), respectively.

7.1 Problem Statement

Given a point cloud from a depth camera, the goal of this chapter is to find a robust suction grasp (target point and approach direction) for a robot to lift an object in isolation on a planar worksurface and transport it to a receptacle. A robust suction grasp maximizes the probability that the robot can hold the object under gravity and perturbations sampled from a distribution over sensor noise, control imprecision, and random disturbing wrenches.

7.1.1 Assumptions

The stochastic model is based on the following assumptions:

1. Quasi-static physics (e.g. inertial terms are negligible) with Coulomb friction.

2. Objects are rigid and made of non-porous material.

3. Each object is singulated on a planar worksurface in a stable resting pose [49].

4. A single overhead depth sensor with known intrinsics, position, and orientation relative to the robot.

5. A vacuum-based end-effector with known geometry and a single disc-shaped suction cup made of linear-elastic material.

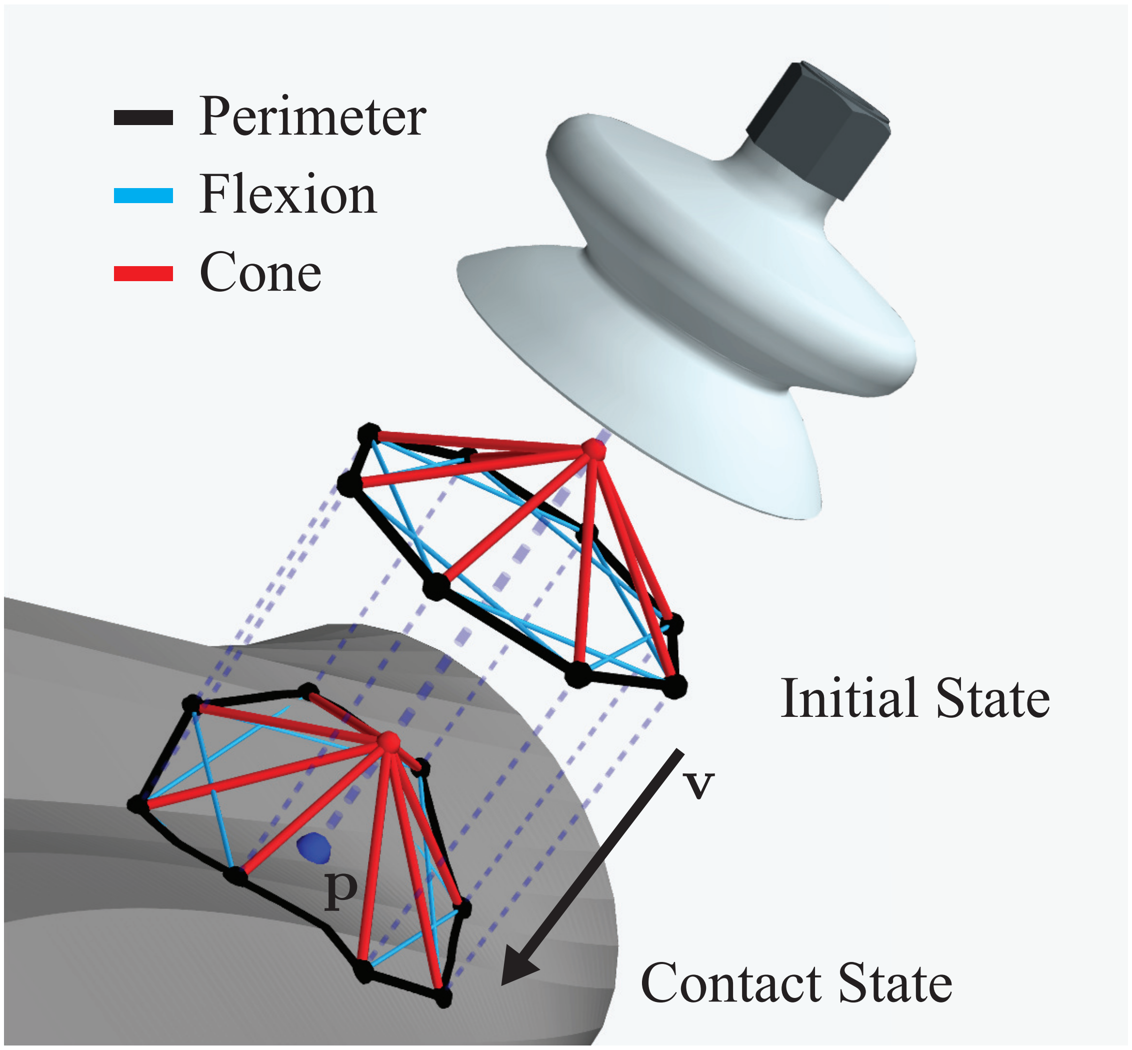


Figure 7.1: Quasi-static spring model used for determining when seal formation is feasible. The model contains three types of springs – perimeter, flexion, and cone springs. An initial state for C is chosen given a target point $\mathbf{p}$ and an approach direction $\mathbf{v}$. Then, a contact state for C is computed so that C's perimeter springs form a complete seal against object mesh M. Seal formation is deemed feasible if the energy required to maintain this contact state is sufficiently low in every spring.

7.1.2 Definitions

Fig. 7.2 illustrates our variables and the dataset generation distribution for Dex-Net 3.0. We use the following definitions:

- **States.** Let $\mathbf{x} = (\mathcal{O}, \mathcal{C}, \mathbf{w})$ denote a state describing the variable properties of environment consisting of a single overhead depth camera and a single object in stable resting pose on a tabletop. The object state $\mathcal{O}$ specifies the geometry $\mathcal{M}$, pose $\mathbf{T}_o$, friction coefficient γ, and center of mass $\mathbf{z}$. The camera state $\mathcal{C}$ specifies the intrinsic

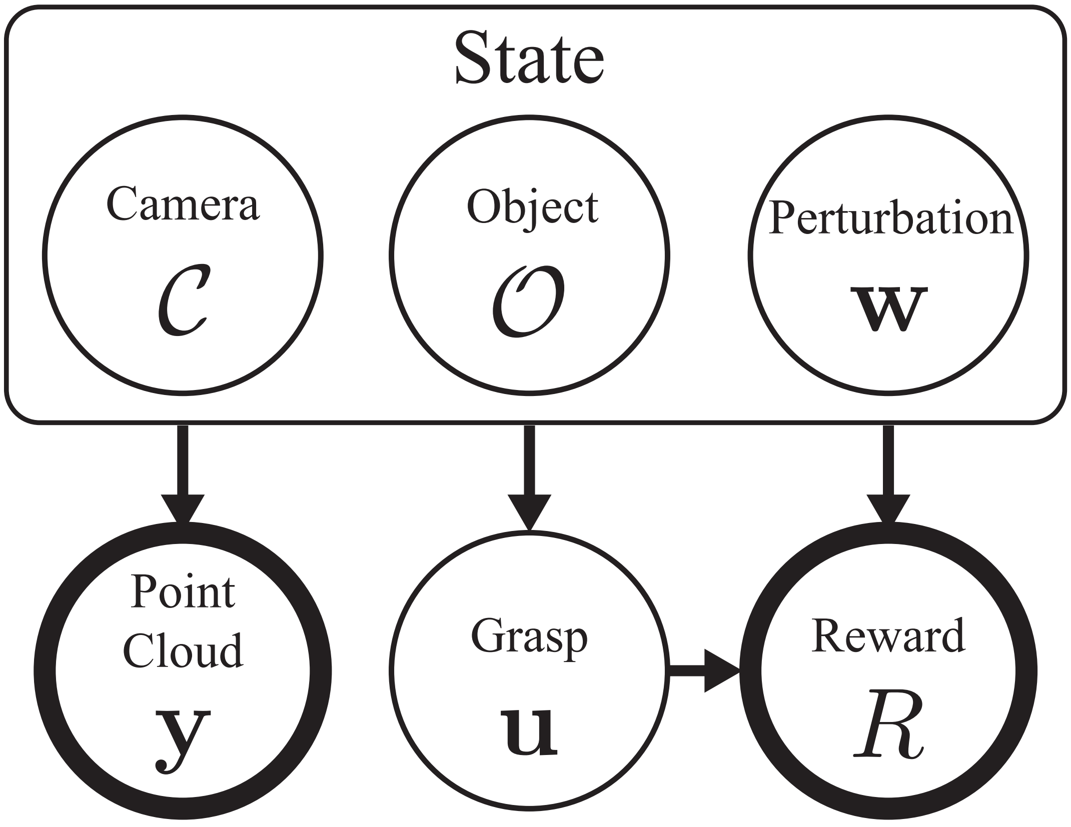


Figure 7.2: Graphical model for robust vacuum suction grasping of objects on a table surface based on point clouds. Object shapes $\mathcal{O}$ are uniformly distributed over a discrete set of object models and object poses $\mathbf{T}_o$ are distributed over the object's stable poses and a bounded region of a planar surface. Grasps $\mathbf{u} = (\mathbf{p}, \mathbf{v})$ are sampled uniformly from the object surface. Given a coefficient of friction γ and a perturbing force and torque vector $\mathbf{w}$ (e.g. due to gravity or inertia), we evaluate an analytic reward metric R based on the ability of the grasp to resist $\mathbf{w}$. A synthetic 2.5D point cloud $\mathbf{y}$ is generated from 3D meshes based on the camera $\mathcal{C}$ in pose $\mathbf{T}_c$.

parameters $\mathcal{I}$ and pose $\mathbf{T}_c$. The perturbation $\mathbf{w}$ specifies a wrench, or force and torque vector, applied to the object through its center of mass such as forces due to gravity.

- **Grasp Actions.** Let $\mathbf{u} = (\mathbf{p}, \mathbf{v}) \in \mathbb{R}^3 \times \mathcal{S}^2$ denote a suction grasp in 3D space specified by a center $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and an approach direction $\mathbf{v} \in \mathcal{S}^2$.

- **Point Clouds.** Let $\mathbf{y} = \mathbb{R}_+^{H \times W}$ be a 2.5D point cloud represented as a depth image with height H and width W taken by a camera with known intrinsics [57].

- **Reward Distribution.** Let $R(\mathbf{x}, \mathbf{u}) \in \{0, 1\}$ be a binary-valued grasp reward metric that indicates the feasibility of a quasi-static equilibrium between the perturbation wrench $\mathbf{w}$ and forces applied actively and passively through contact between the suction cup and object. This reward function is used a proxy for whether or not the grasp can successfully lift and transport the object to a receptacle. Let $q(R \mid \mathbf{x}, \mathbf{u})$ model probabilistic grasp outcomes due to control imprecision and uncertainty in the exact parameters of the suction cup (e.g. radius, material elasticity).

- **Environment Distribution.** Let $q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ be the dataset generation environment defining a distribution on rewards, states, and point clouds modeling imprecision in sensing and control.

Further details are given in Section 7.7.3.

7.1.3 Objective

The goal is to find a grasp that maximizes robustness given a point cloud:

$$\pi^*(\mathbf{y}) = \text{argmax}_{\mathbf{u} \in \mathcal{U}} Q(\mathbf{y}, \mathbf{u})$$

where $\mathcal{U}$ specifies constraints on the set of available grasps, such as collisions or kinematic feasibility. We approximate π^* by optimizing the weights θ of a deep Grasp Quality Convolutional Neural Network (GQ-CNN) $Q_\theta(\mathbf{y}, \mathbf{u})$ on a training dataset $\mathcal{D} = \{(R_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i=1}^N$ consisting of reward values, point clouds, and suction grasps sampled from the dataset generation distribution. The optimization objective is to find weights θ that minimize the cross-entropy loss $\mathcal{L}$ over $\mathcal{D}$:

$$\theta^* = \underset{\theta \in \Theta}{\text{argmin}} \sum_{i=1}^{N} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_i)). \tag{7.1.1}$$

7.2 Compliant Suction Contact Model

To measure rewards for a suction cup gripper, we develop a quasi-static spring model of the suction cup material and a model of contact wrenches that the suction cup can apply to the object through a ring of contact on the suction cup perimeter. Under our model, the reward $R = 1$ if:

1. A seal is formed between the perimeter of the suction cup and the object surface.
2. Given a seal, the suction cup is able to resist an external wrench on the object due to gravity and disturbances.

7.2.1 Seal Formation

A suction cup can lift objects due to an air pressure differential induced across the membrane of the cup by a vacuum generator that forces the object into the cup. If a gap exists between

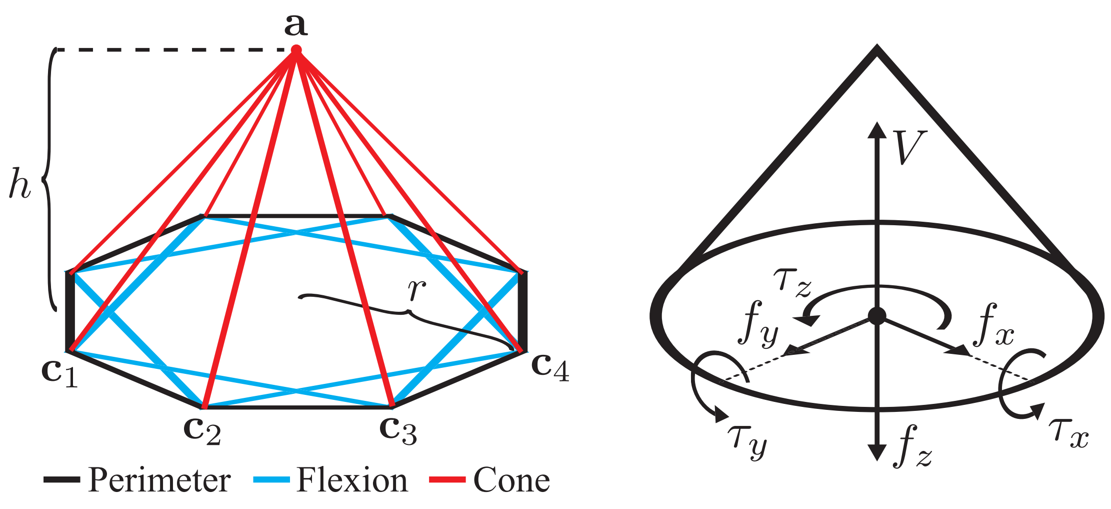


Figure 7.3: Compliant suction contact model. (Left) The quasi-static spring model used in seal formation computations. This suction cup is approximated by $n = 8$ components. Here, r is equal to the radius of the cup and h is equal to the height of the cup. $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$ are the base contact vertices and $\mathbf{a}$ is the apex. (Right) Wrench basis for the suction ring contact model. The contact exerts a constant pulling force on the object of magnitude V and additionally can push or pull the object along the contact z axis with force f_z. The suction cup material exerts a normal force $f_N = f_z + V$ on the object through a linear pressure distribution (force per unit length) on the ring. This pressure distribution induces a friction limit surface bounding the set of possible frictional forces in the tangent plane $f_t = (f_x, f_y)$ and the torsional moment τ_z, and also induces torques τ_x and τ_y about the contact x and y axes due to elastic restoring forces in the suction cup material.

the perimeter of the cup and the object, then air flowing into the gap may reduce the differential and cause the grasp to fail. Therefore, a tight seal between the cup and the target object is important for achieving a successful grasp.

To determine when seal formation is possible, we model circular suction cups as a conical spring system $\mathcal{C}$ parameterized by real numbers (n, r, h), where n is the numer of vertices along the contact ring, r is the radius of the cup, and h is the height of the cup. See see Fig. 7.3 for an illustration.

Rather than performing a computationally expensive dynamic simulation with a spring-mass model to determine when seal formation is feasible, we make simplifying assumptions to evaluate seal formation geometrically. Specifically, we compute a configuration of $\mathcal{C}$ that achieves a seal by projecting $\mathcal{C}$ onto the surface of the target object's triangular mesh $\mathcal{M}$ and evaluate the feasibility of that configuration under quasi-static conditions as a proxy for the dynamic feasibility of seal formation.

In our model, $\mathcal{C}$ has two types of springs – *structural* springs that represent the physical structure of the suction cup and *flexion* springs that do not correspond to physical structures but instead are used to resist bending along the cup's surface. Dynamic spring-mass systems with similar structures have been used in prior work to model stiff sheets of rubber [137]. The undeformed structural springs of $\mathcal{C}$ form a right pyramid with height h and with a base that is a regular n-gon with circumradius r. Let $\mathcal{V} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n, \mathbf{a}\}$ be the set of contact vertices of the undeformed right pyramid, where each $\mathbf{c}_i$ is a base contact vertex and $\mathbf{a}$ is the pyramid's apex. We define the model's set of springs as follows:

- **Perimeter (Structural) Springs** – Springs linking vertex $\mathbf{c}_i$ to vertex $\mathbf{c}_{i+1}$, $\forall i \in \{1, \ldots, n\}$.
- **Cone (Structural) Springs** – Springs linking vertex $\mathbf{c}_i$ to vertex $\mathbf{a}$, $\forall i \in \{1, \ldots, n\}$.
- **Flexion Springs** – Springs linking vertex $\mathbf{c}_i$ to vertex $\mathbf{c}_{i+2}$, $\forall i \in \{1, \ldots, n\}$.

In the model, a complete seal is formed between $\mathcal{C}$ and $\mathcal{M}$ if and only if each of the perimeter springs of $\mathcal{C}$ lies entirely on the surface of $\mathcal{M}$. Given a target mesh $\mathcal{M}$ with a target grasp $\mathbf{u} = (\mathbf{p}, \mathbf{v})$ for the gripper, we choose an initial configuration of $\mathcal{C}$ such that $\mathcal{C}$ is undeformed and the approach line $(\mathbf{p}, \mathbf{v})$ passes through $\mathbf{a}$ and is orthogonal to the base of $\mathcal{C}$. Then, we make the following assumptions to determine a final static contact configuration of $\mathcal{C}$ that forms a complete seal against $\mathcal{M}$ (see Fig. 7.1):

- **The perimeter springs** of $\mathcal{C}$ must not deviate from the original undeformed regular n-gon when projected onto a plane orthogonal to $\mathbf{v}$. This means that their locations can be computed by projecting them along $\mathbf{v}$ from their original locations onto the surface of $\mathcal{M}$.
- **The apex**, $\mathbf{a}$, of $\mathcal{C}$ must lie on the approach line $(\mathbf{p}, \mathbf{v})$ and, given the locations of $\mathcal{C}$'s base vertices, must also lie at a location that keeps the average distance along $\mathbf{v}$ between $\mathbf{a}$ and the perimeter vertices equal to h.

See Section 7.6 for computational details.

Given this configuration, a seal is feasible if:

- The cone faces of $\mathcal{C}$ do not collide with $\mathcal{M}$ during approach or in the contact configuration.
- The surface of $\mathcal{M}$ has no holes within the contact ring traced out by $\mathcal{C}$'s perimeter springs.
- The energy required in each spring to maintain the contact configuration of $\mathcal{C}$ is below a real-valued threshold E modeling the maximum deformation of the suction cup material against the object surface.

We threshold the energy in individual springs rather than the total energy for $\mathcal{C}$ because air gaps are usually caused by local geometry.

7.2.2 Wrench Space Analysis

To determine the degree to which the suction cup can resist external wrenches such as gravity, we analyze the set of wrenches that the suction cup can apply.

Wrench Resistance

The object wrench set for a grasp using a contact model with m basis wrenches is:

$$\Lambda = \{\mathbf{w} \in \mathbb{R}^6 \mid \mathbf{w} = G\alpha \text{ for some } \alpha \in \mathcal{F}\}$$

where $G \in \mathbb{R}^{6\times m}$ is a set of basis wrenches in the object coordinate frame, and $\mathcal{F} \subseteq \mathbb{R}^m$ is a set of constraints on contact wrench magnitudes [121].

Definition 6. *A grasp* $\mathbf{u}$ *achieves wrench resistance with respect to* $\mathbf{w}$ *if* $-\mathbf{w} \in \Lambda$ *[90, 121].*

We encode wrench resistance as a binary variable W such that $W = 0$ if $\mathbf{u}$ resists $\mathbf{w}$ and $W = 0$ otherwise.

Suction Contact Model

Many suction contact models acknowledge normal forces, vacuum forces, tangential friction, and torsional friction [4, 86, 112, 161] similar to a point contact with friction or soft finger model [121]. However, under this model, a single suction cup cannot resist torques about axes in the contact tangent plane, implying that any torque about such axes would cause the suction cup to drop an object (see Section 7.7 for a detailed proof). This defies our intuition since empirical evidence suggests that a single point of suction can robustly transport objects [40, 60].

We hypothesize that these torques are resisted through an asymmetric pressure distribution (force per unit length) on the ring of contact between the suction cup and object, which occurs due to passive elastic restoring forces in the material. Fig. 7.3 illustrates the suction ring contact model. The grasp map G is defined by the following basis wrenches:

1. **Actuated Normal Force (f_z):** The force that the suction cup material applies by pressing into the object along the contact z axis.

2. **Vacuum Force (V):** The magnitude of the constant force pulling the object into the suction cup coming from the air pressure differential.

3. **Frictional Force ($f_f = (f_x, f_y)$):** The force in the contact tangent plane due to the normal force between the suction cup and object, $f_N = f_z + V$.

4. **Torsional Friction (τ_z):** The torque resulting from frictional forces in the ring of contact.

5. **Elastic Restoring Torque ($\tau_e = (\tau_x, \tau_y)$):** The torque about axes in the contact tangent plane resulting from elastic restoring forces in the suction cup pushing on the object along the boundary of the contact ring.

The magnitudes of the contact wrenches are constrained due to (a) the friction limit surface [76], (b) limits on the elastic behavior of the suction cup material, and (c) limits on the vacuum force. In the suction ring contact model, $\mathcal{F}$ is approximated by a set of linear constraints for efficient computation of wrench resistance:

$$\begin{array}{rlll} \textbf{Friction:} & \sqrt{3}|f_x| \leqslant \gamma f_N & \sqrt{3}|f_y| \leqslant \gamma f_N & \sqrt{3}|\tau_z| \leqslant r\gamma f_N \\ \textbf{Material:} & \sqrt{2}|\tau_x| \leqslant \pi r \kappa & \sqrt{2}|\tau_y| \leqslant \pi r \kappa & \\ \textbf{Suction:} & f_z \geqslant -V & & \end{array}$$

Here γ is the friction coefficient, r is the radius of the contact ring, and κ is a material-dependent constant modeling the maximum stress for which the suction cup has linear-elastic behavior. These constraints define a subset of the friction limit ellipsoid and cone of admissible elastic torques under a linear pressure distribution about the ring of the cup. Furthermore, we can compute wrench resistance using quadratic programming due to the linearity of the constraints. See Section 7.7 detailed derivation and proof.

7.2.3 Robust Wrench Resistance

We evaluate the robustness of candidate suction grasps by evaluating seal formation and wrench resistance over distributions on object pose, grasp pose, and disturbing wrenches:

Definition 7. *The robust wrench resistance metric for* $\mathbf{u}$ *and* $\mathbf{x}$ *is* $\lambda(\mathbf{x}, \mathbf{u}) = \mathbb{P}(W \mid \mathbf{x}, \mathbf{u})$*, the probability of success under perturbations in object pose, gripper pose, friction, and disturbing wrenches.*

In practice, we evaluate robust wrench resistance by taking J samples, evaluating binary wrench resistance for each, and computing the sample mean: $\frac{1}{J}\sum_{j=1}^{J} W_j$.

7.3 Dex-Net 3.0 Dataset

To learn to predict grasp quality from noisy point clouds, we generate the Dex-Net 3.0 training dataset of point clouds, grasps, and grasp reward labels by sampling tuples $(R_i, \mathbf{u}_i, \mathbf{y}_i)$ from a dataset generation distribution $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$.

7.3.1 Dataset Generation Distribution

We model $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ as the product of the environment distribution $q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ described in Section 7.1 and an action candidate distribution $q(\mathbf{u} \mid \mathbf{x})$ modeling a uniform random distribution over contact points on the object surface. The details of each distribution are listed in Table 7.1. The parameters of the sampling distributions and compliant suction contact model $(n, r, h, E, V, \gamma, \kappa, \epsilon)$ (see Section 7.2) were set by maximizing average precision of the Q values using grid search for a set of grasps attempted on an ABB YuMi robot on a set of known 3D printed objects (see Section 7.5.1).

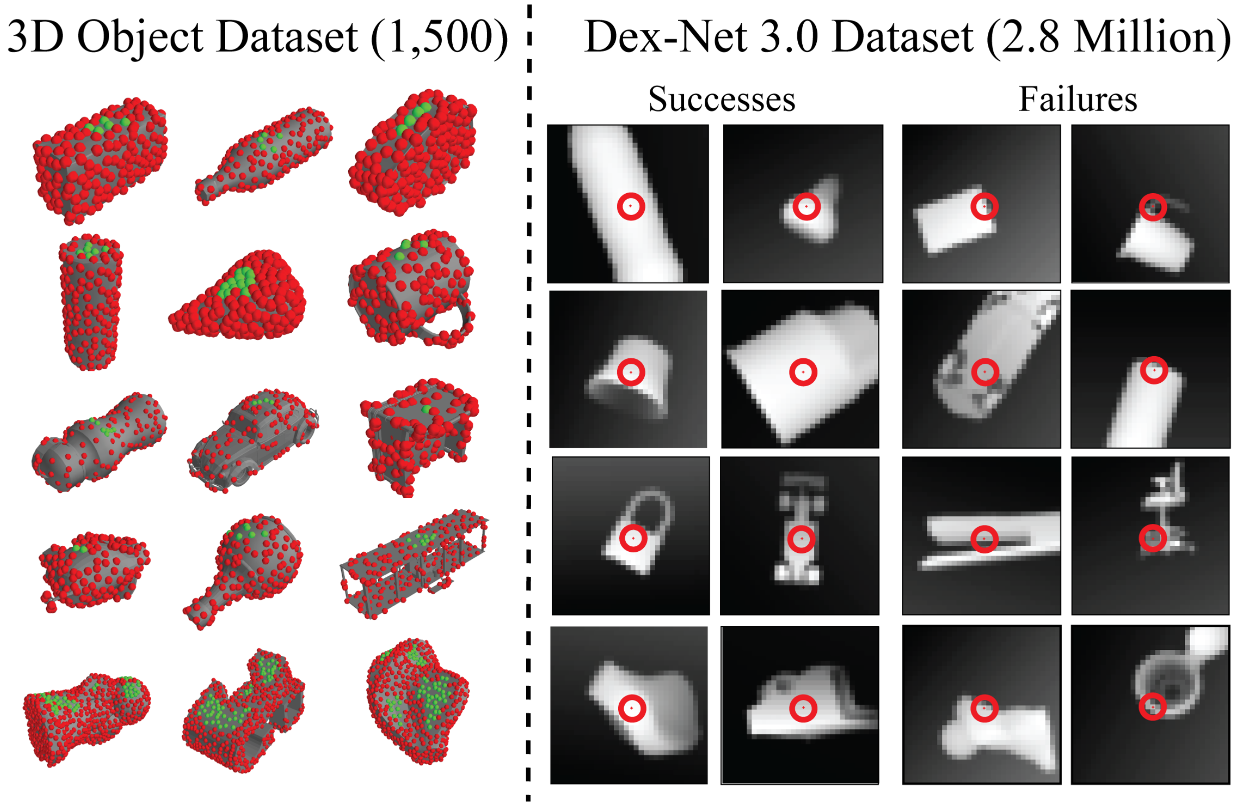


Figure 7.4: Dex-Net 3.0 dataset. (Left) The Dex-Net 3.0 object dataset contains approximately 350k unique suction target points across the surfaces of 1,500 3D models from the KIT object database [78] and 3DNet [180]. Each suction grasp is classified as robust (green) or non-robust (red). Robust grasps are often above the object center-of-mass on flat surfaces of the object. (Right) The Dex-Net 3.0 point cloud dataset contains 2.8 million tuples of point clouds and suction grasps with reward labels, with approximately 11.8% positive examples.

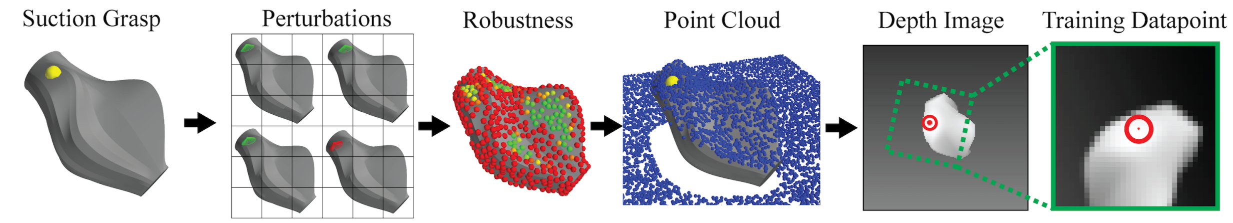


Figure 7.5: Computational pipeline for generating the Dex-Net 3.0 dataset (left to right). We first sample a candidate suction grasp from the object surface and evaluate the ability to form a seal and resist gravity over perturbations in object pose, gripper pose, and friction. The samples are used to estimate the probability of success, or robustness, for candidate grasps on the object surface. We render a point cloud for each object and associate the candidate grasp with a pixel and orientation in the depth image through perspective projection. Training datapoints are centered on the suction target pixel and rotated to align with the approach axis to encode the invariance of the robustness to image locations.

Distribution	Description
$q(\gamma)$	truncated Gaussian distribution over friction coefficients
$q(\mathcal{O})$	discrete uniform distribution over 3D object models
$q(\mathbf{T}_o \vert \mathcal{O})$	continuous uniform distribution over the discrete set of object stable poses and planar poses on the table surface
$p(\mathbf{T}_c)$	continuous uniform distribution over spherical coordinates for radial bounds $[r_\ell, r_u]$ and polar angle in $[0, \delta]$

Table 7.1: Details of the distributions used in the graphical model for generating the Dex-Net 3.0 training dataset.

7.3.2 Computational Pipeline

The pipeline for generating training tuples is illustrated in Fig. 7.5. We first sample state by selecting an object at random from a database of 3D CAD models and sampling a friction coefficient, planar object pose, and camera pose relative to the worksurface. We generate a set of grasp candidates for the object by sampling points and normals uniformly at random from the surface of the object mesh. We then set the binary reward label $R = 1$ if a seal is formed and robust wrench resistance (described in Section 7.2.3) is above a threshold value ϵ. Finally, we sample a point cloud of the scene using rendering and a model of image noise [104]. The grasp success labels are associated with pixel locations in images through perspective projection [57].

7.4 Learning a Deep Robust Grasping Policy

We use the Dex-Net 3.0 dataset to train a GQ-CNN that takes as input a single-view point cloud of an object resting on the table and a candidate suction grasp defined by a target 3D point and approach direction, and outputs the grasp quality, or estimated probability of success, for the grasp on the visible object.

7.4.1 Architecture

The GQ-CNN architecture is identical to Dex-Net 2.0 [105] except that we modify the pose input stream to include the angle between the approach direction and the table normal. The point cloud stream takes a depth image centered on the target point and rotated to align the middle column of pixels with the approach orientation similar to a spatial transforming layer [67]. The end-effector depth from the camera and orientation are input to a fully connected layer in a separate pose stream and concatenated with conv features in a fully connected layer. We train the GQ-CNN using stochastic gradient descent with momentum using an 80-20 training-to-validation image-wise split of the Dex-Net 3.0 dataset. Training

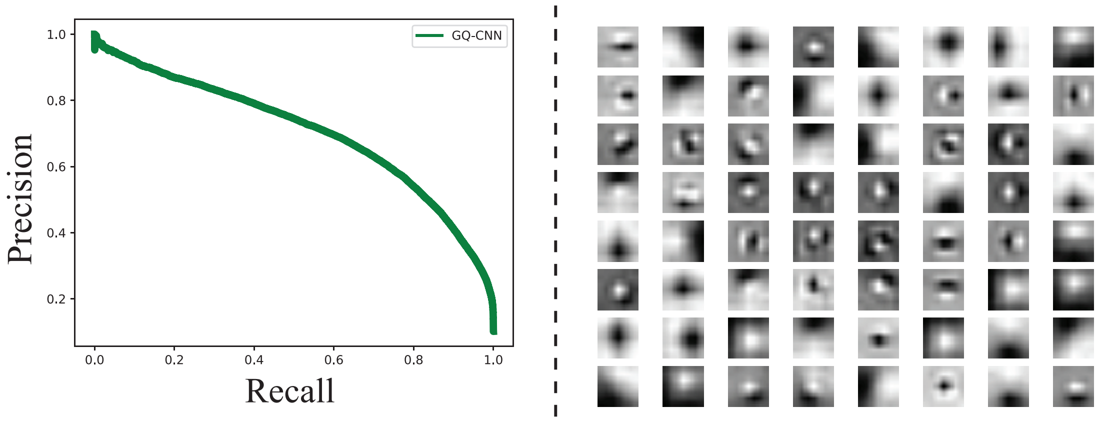


Figure 7.6: Performance of the Dex-Net 3.0 GQ-CNN. (Left) Precision-recall curve for the GQ-CNN trained on Dex-Net 3.0 on the validation set of 552,000 pairs of grasps and images. (Right) The 64 conv1_1 filters of the GQ-CNN. Each is 7×7. We see that the network learns circular filters which may be used to assess the surface curvature about the ring of contact between the suction cup and object.

took approximately 12 hours on three NVIDIA Titan X GPUs. The learned GQ-CNN achieves 93.5% classification accuracy on the held-out validation set.

7.4.2 Policy

We use the GQ-CNN in a deep robust grasping policy to plan suction target grasps from point clouds on a physical robot. The policy uses the Cross Entropy Method (CEM) [98, 105, 145]. CEM samples a set of initial candidate grasps uniformly at random from the set of surface points and inward-facing normals on a point cloud of the object, then iteratively re-samples grasps from a Gaussian Mixture Model fit to the grasps with the highest predicted probability of success.

7.4.3 Performance

The GQ-CNN trained on Dex-Net 3.0 had an accuracy of 93.5% on a held out validation set of approximately 552,000 datapoints. Fig. 7.6 shows the precision-recall curve for the GQ-CNN validation set and the optimized 64 conv1_1 filters, each of which is 7×7. Fig. 7.7 illustrates the probability of success predicted by the GQ-CNN on candidates grasps from several real point clouds.

7.5 Experiments

We ran experiments to characterize the precision of robust wrench resistance when object shape and pose are known and the precision of our deep robust grasping policy for planning grasps from point clouds for three object classes.

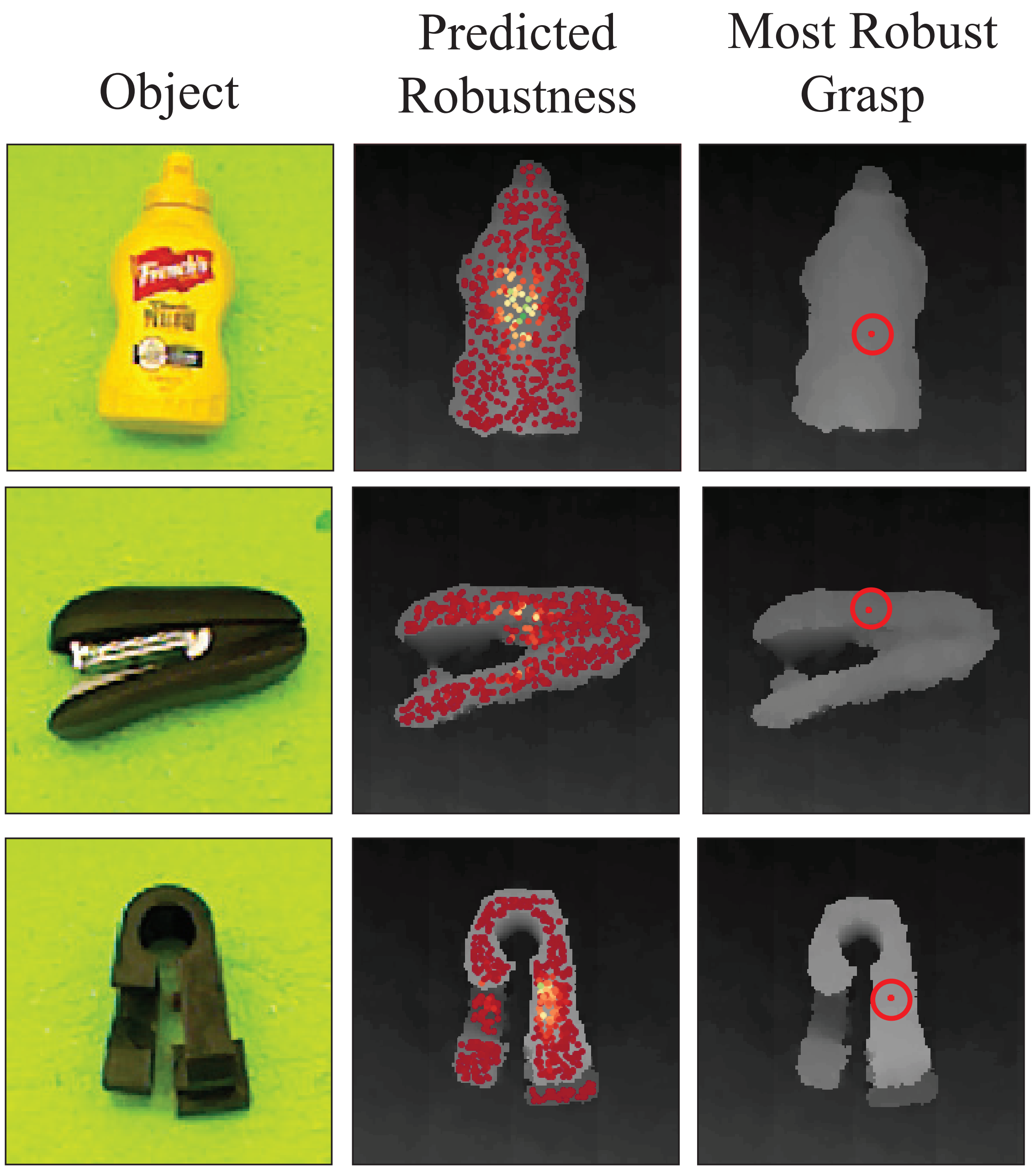


Figure 7.7: Example grasps planned with the Dex-Net 3.0 GQ-CNN-based policy on RGB-D point clouds. (Left) The robot is presented an object in isolation. (Middle) Initial candidate suction target points colored by the predicted probability of success from zero (red) to one (green). Robust grasps tend to concentrate around the object centroid. (Right) The policy optimizes for the grasp with the highest probability of success using the Cross Entropy Method.

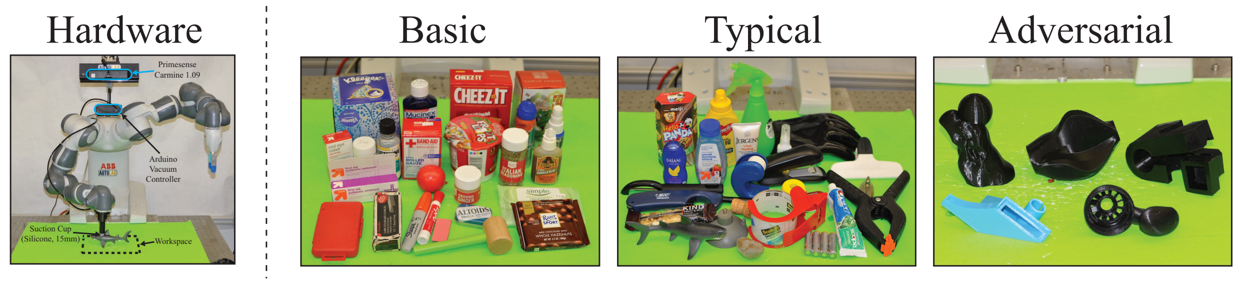


Figure 7.8: Physical benchmark used to evaluate the suction grasping policy. (Left) The experimental setup with an ABB YuMi equipped with a suction gripper. (Right) The 55 objects used to evaluate suction grasping performance. The objects are divided into three categories to characterize performance: Basic (e.g. prismatic objects), Typical, and Adversarial.

7.5.1 Object Classes

We created a dataset of 55 rigid and non-porous objects including tools, groceries, office supplies, toys, and 3D printed industrial parts. We separated objects into three categories, illustrated in Fig. 7.8:

1. *Basic:* Prismatic solids (e.g. rectangular prisms, cylinders). Includes 25 objects.
2. *Typical:* Common objects with varied geometry and many accessible, approximately planar surfaces. Includes 25 objects.
3. *Adversarial:* 3D-printed objects with complex geometry (e.g. curved or narrow surfaces) that are difficult to access. Includes 5 objects.

For object details, see http://bit.ly/2xMcx3x.

7.5.2 Experimental Protocol

We ran experiments with an ABB YuMi with a Primesense Carmine 1.09 and a suction system with a $15mm$ diameter silicone single-bellow suction cup and a VM5-NC VacMotion vacuum generator with a payload of approximately $0.9kg$. The experimental workspace is illustrated in the left panel of Fig. 7.8. In each experiment, the operator iteratively presented a target object to the robot and the robot planned and executed a suction grasp on the object. The operator labeled successes based on whether or not the robot was able to lift and transport the object to the side of the workspace. All experiments ran on a Desktop running Ubuntu 16.04 with a 3.4 GHz Intel Core i7-6700 Quad-Core CPU and an NVIDIA GeForce 980 GPU.

7.5.3 Performance Metrics

For each method, we measured:

1. **Average Precision (AP).** The area under the precision-recall curve, which measures precision over possible thresholds on the probability of success predicted by the policy. This is useful for industrial applications where a robot may take an alternative action (e.g. asking for help) if the planned grasp is predicted to fail.

2. **Success Rate.** The fraction of all grasps that were successful.

These metrics alone do not give a complete picture of how useful a suction grasp policy would work in practice. Average Precision (AP) penalizes a policy for having poor recall (a high rate of false negatives relative to true positives), and success rate penalizes a policy with a high number of failures. However, not all failures should be treated equally: some failures occur because a robust suction grasp cannot be found while others are the result of an overconfident prediction.

In practice, a suction grasp policy would be part of a larger system (e.g. a state machine) that could decide whether or not to execute a grasp based on the continuous probability of success output by the GQ-CNN. As long as the policy is not overconfident, such as system can detect failures before they occur and take an alternative action such as attempting to turn the object over, asking a human for help, or leaving the object in the bin for error handling. At the same time, if a policy is too conservative and never predicts successes, then the system will handle be able to handle very few test cases.

We illustrate this tradeoff by also plotting a Success-Attempt Rate curve which plots:

1. **Success Rate.** The fraction of fraction of grasps that are successful if the system only executes grasps have predicted probability of success greater than a confidence threshold τ.

2. **Attempt Rate.** The fraction of all test cases for which the system attempts a grasp, if the system only attempts grasps with predicted probability of success greater than a confidence threshold τ.

over all possible values of the confidence threshold τ. There is typically an inverse relationship between the two metrics: a higher confidence threshold will reduce false positives but will also reduce the frequency of grasp attempts, increasing runtime and decreasing the diversity of cases that the robot is able to successfully handle.

7.5.4 Performance on Known Objects

To assess performance of our robustness metric independent of the perception system, we evaluated whether or not the metric was predictive of suction grasp success when object shape and pose were known using the 3D printed Adversarial objects. The robot was presented one of the five Adversarial objects in a known stable pose, selected from the top three most probable stable poses. We hand-aligned the object to a template image generated by rendering the object in a known pose on the table. Then, we indexed a database of grasps

Metric	AP (%)	Success Rate (%)
PC3D	88	80
SS	89	84
WR	93	80
RWR	**100**	**92**

Table 7.2: Performance of robust grasping policies with known state (3D object shape and pose) across 75 physical trials per policy on the Adversarial object dataset. The policies differ by the metric used to rank grasps, and each metric is computed using the known 3D object geometry. The robust wrench resistance metric, which considers the ability of a suction cup to form a seal and resist gravity under perturbations, has very high precision. In comparison, the Planarity-Centroid heuristic achieves only 88% precision and 80% success.

precomputed on 3D models of the objects and executed the grasp with the highest metric value for five trials. In total, there were 75 trials per experiment.

We compared the following metrics:

1. **Planarity-Centroid (PC3D).** The inverse distance to the object centroid for sufficiently planar patches on the 3D object surface.
2. **Spring Stretch (SS).** The maximum stretch among virtual springs in the suction contact model.
3. **Wrench Resistance (WR).**
4. **Robust Wrench Resistance (RWR).**

The results are detailed in Table 7.2 and the Success vs Attempt Rate curve is plotted in Fig. 7.9. A policy based on the robust wrench resistance metric with our compliant suction contact model had a 100% success rate for a large percentage of possible test cases, whereas a heuristic based on planarity and the distance to the object center of mass had success rates as low as 67%, indicating that the real-valued distance to the center of mass is not well correlated with grasp success.

7.5.5 Performance on Novel Objects

We also evaluated the performance of GQ-CNNs trained on Dex-Net 3.0 for planning suction target points from a single-view point cloud. In each experiment, the robot was presented one object from either the Basic, Typical, or Adversarial classes in a pose randomized by shaking the object in a box and placing it on the table. The object was imaged with a depth sensor and segmented using 3D bounds on the workspace. Then, the grasping policy executed the most robust grasp according to a success metric. In this experiment the human operators were blinded from the method they were evaluating to remove bias in human labels.

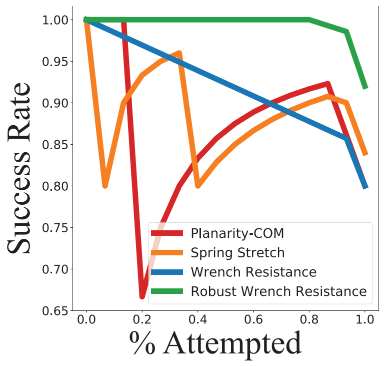


Figure 7.9: Success rate vs Attempt Rate for grasp quality metrics on known 3D objects in known poses.

We compared policies that used the following metrics:

1. **Planarity.** The inverse sum of squared errors from an approach plane for points within a disc with radius equal to that of the suction cup.

2. **Centroid.** The inverse distance to the object centroid.

3. **Planarity-Centroid.** The inverse distance to the centroid for sufficiently planar patches on the 3D object surface.

4. **GQ-CNN (ADV).** A GQ-CNN trained on synthetic data from only the Adversarial objects (to assess the ability of the model to fit complex objects).

5. **GQ-CNN (DN3).** A GQ-CNN trained on synthetic data from the 3DNet [180], KIT [78], and Adversarial object datasets.

Table 7.3 details performance on the Basic, Typical, and Adversarial objects, and Fig. 7.10 illustrates the Success-Attempt Rate tradeoff. We see that the Dex-Net 3.0 policy has the highest AP across the Basic and Typical classes. Also, the GQ-CNN trained on the Adversarial objects significantly outperforms all methods on the Adversarial dataset, suggesting that our model is able to exploit knowledge of complex 3D geometry to plan robust grasps. Furthermore, the Success-Attempt Rate curve suggests that the continuous probability of success output by the Dex-Net 3.0 policy is highly correlated with the true success label and can be used to detect failures before they occur on the Basic and Typical object classes. The Dex-Net 3.0 policy took an average of approximately 3 seconds to plan each grasp.

Interestingly, in all experiments the Dex-Net 3.0 GQ-CNN policy performs similarly to the suction heuristic based on planarity and proximity to the object centroid. This suggests that there may be a relationship between analytic grasp reward functions defined in 3D space

	Basic		Typical		Adversarial	
	AP (%)	**Success Rate** (%)	**AP** (%)	**Success Rate** (%)	**AP** (%)	**Success Rate** (%)
Planarity	81	74	69	67	48	47
Centroid	89	92	80	78	47	38
Planarity-Centroid	98	94	94	**86**	64	62
GQ-CNN (ADV)	83	77	75	67	**86**	**81**
GQ-CNN (DN3)	**99**	**98**	**97**	82	61	58

Table 7.3: Performance of image-based grasping policies for 125 trials each on the Basic and Typical datasets and 100 trials each on the Adversarial datasets. We see that the GQ-CNN trained on Dex-Net 3.0 has the highest average precision on the Basic and Typical objects but has lower precision on the adversarial objects, which are very different than common objects in the training dataset. A GQ-CNN trained on the Adversarial dataset significantly outperforms all methods on these objects, suggesting that our model is able to capture complex geometries when the training dataset contains a large proportion of such objects.

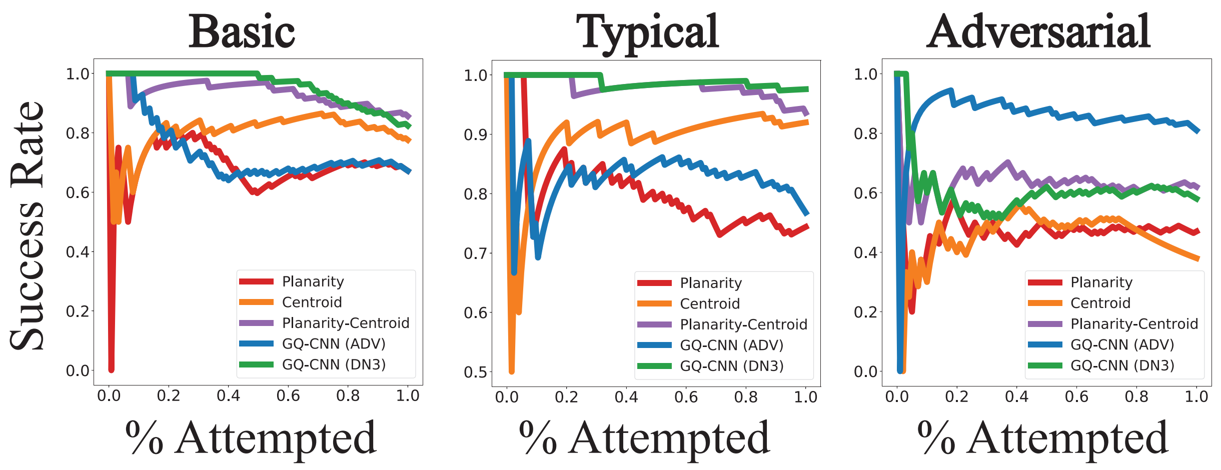


Figure 7.10: Success vs Attempt Rate for 125 trials on each of the Basic and Typical object datasets and 100 trials each on the Adversarial object dataset. The GQ-CNN trained on Dex-Net 3.0 has near 100% precision on the Basic and Typical classes for a significant portion of attempts, suggesting that the GQ-CNN is able to predict when it is likely to fail on novel objects. The GQ-CNN trained on the Adversarial objects has a significantly higher precision on the Adversarial class but does not perform as well on the other objects.

and 2D image-based heuristics. Knowledge of the principles behind 3D prasp analysis could perhaps inform the design of future, more robust heuristics.

7.5.6 Classification Performance on Known Objects

To assess performance of our robustness metric on classifying grasps as successful or unsuccessful, we evaluated whether or not the metric was able to classify a set of grasps sampled randomly from the 3D object surface using the known 3D object geometry and pose of the Adversarial objects. First, we sampled a set of 1000 grasps uniformly at random from the surface of the 3D object meshes. Then robot was presented one of the five Adversarial ob-

Metric	AP (%)	Accuracy (%)	Rank Correlation
PC3D	71	68	0.36
SS	75	74	0.49
WR	78	**77**	0.52
RWR	**80**	75	**0.62**

Table 7.4: Performance of classification and correlation with successful object lifts and transports for various metrics of grasp quality based on 3D object meshes. The metrics SS, WR, and RWR all use our compliant suction contact model, and RWR uses our entire proposed method: checking seal formation, analyzing wrench resistance using the suction ring model, and computing robustness with Monte-Carlo sampling.

jects in a known stable pose, selected from the top three most probable stable poses. We hand-aligned the object to a template image generated by rendering the object in a known pose on the table. Then, the robot executed a grasp uniformly at random from the set of reachable grasps for the given stable pose. In total, there were 600 trials, 125 per object.

We compared the predictions made for those grasps by the following metrics:

1. **Planarity-Centroid (PC3D).** The inverse distance to the object centroid for sufficiently planar patches on the 3D object surface.

2. **Spring Stretch (SS).** The maximum stretch among virtual springs in the suction contact model.

3. **Wrench Resistance (WR).**

4. **Robust Wrench Resistance (RWR).**

We measured the Average Precision (AP), classification accuracy, and Spearman's rank correlation coefficient (which measures the correlation between the ranking of the metric value and successes on the physical robot). Table 7.4 details the performance of each metric. We see that the robust wrench resistance metric with our compliant spring contact model has the highest average precision and correlation with successes on the physical robot.

7.5.7 Failure Modes

The system was not able to handle many objects due to material properties. We broke up the failure objects into two categories:

1. **Imperceptible Objects:** Objects with (a) surface variations less than the spatial resolution of our Primesense Carmine 1.09 depth camera or (b) specularities or transparencies that prevent the depth camera from sensing the object geometry. Thus the point-cloud-based grasping policies were not able to distinguish successes from failures.

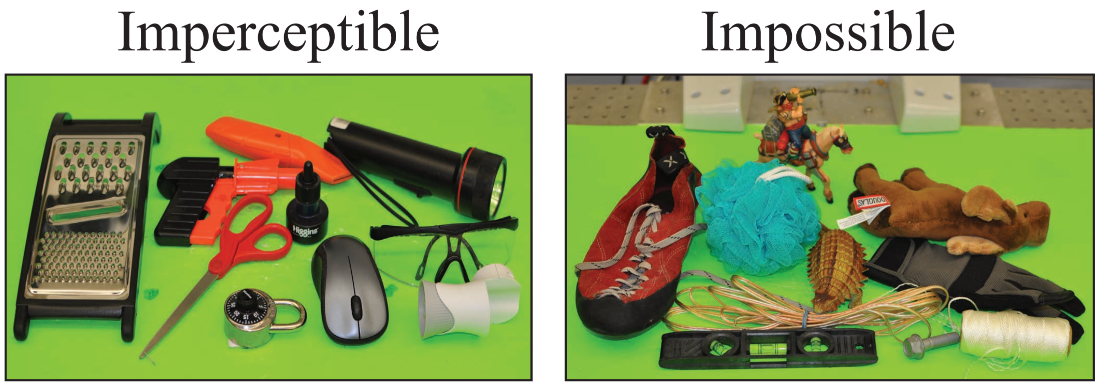


Figure 7.11: Two categories of objects that cannot be handled by any of the point-cloud based suction grasping policies. (Left) Imperceptible objects, which cannot be handled by the system due to small surface variations that cannot be detected by the low-resolution depth sensor but do prevent seal formation. (Right) Impossible objects, which cannot be handled by the system due to non-porosity or lack of an available surface to form a seal.

2. **Impossible Objects:** Objects for which a seal cannot be formed either because objects are (a) non-porous or (b) lack a surface patch for which the suction cup can achieve a seal due to size or texture.

A set of example objects within each of the failure modes are illustrated in Fig. 7.11.

7.6 Details of Quasi-Static Spring Seal Formation Model

In this section, we derive a detailed process for statically determining a final configuration of C that achieves a complete seal against mesh $\mathcal{M}$. We assume that we are given a line of approach ℓ parameterized by $\mathbf{p}$, a target point on the surface of $\mathcal{M}$, and $\mathbf{v}$, a vector pointing towards P along the line of approach.

First, we choose an initial, undeformed configuration of C. In this undeformed configuration of C, all of the springs of C are in their resting positions, which means that the structural springs of C form a right pyramid with a regular n-gon as its base. This perfectly constrains the relative positions of the vertices of C, so all that remains is specifying the position and orientation of C relative to the world frame.

We further constrain the position and orientation of C such that ℓ passes through $\mathbf{a}$ and is orthogonal to the plane containing the base of C. This leaves only the position of $\mathbf{a}$ and a rotation about ℓ as degrees of freedom. For our purposes, the position of $\mathbf{a}$ along ℓ does not matter so long as C is not in collision with $\mathcal{M}$ and the base of C is closer to $\mathcal{M}$ than the apex is. In general, we choose $\mathbf{a}$ such that $\|\mathbf{p} - \mathbf{a}\| > x + h$, where x is the largest extent of the object's vertices. For the rotation about ℓ, we simply select a random initial angle.

Theoretically, the rotation could affect the outcome of our metric, but as long as n is chosen to be sufficiently large, the result is not sensitive to the chosen rotation angle.

Next, given the initial configuration of C, we compute the final locations of the perimeter springs on the surface of M under two main constraints:

- The perimeter springs of C must not deviate from their initial locations when projected back onto the plane containing the base of C's initial right pyramid.
- The perimeter springs of C must lie flush against the mesh M.

Essentially, this means that the perimeter springs will lie on the intersection of $\mathcal{M}$ with a right prism K whose base is the base of the initial configuration's right pyramid and whose height is sufficient such that K passes all the way through $\mathcal{M}$. The base vertices of C will lie at the intersection of $\mathcal{M}$ and K's side edges, and the perimeter springs of C will lie along the intersection of $\mathcal{M}$ and K's side faces.

Finally, given a complete configuration of the perimeter vertices of C as well as the paths of the perimeter springs along the surface of $\mathcal{M}$, we compute the final location of the cup apex $\mathbf{a}$. We work with three main constraints:

- $\mathbf{a}$ must lie on ℓ.
- $\mathbf{a}$ must not be below the surface of $\mathcal{M}$ (i.e. $\mathbf{v}^T(\mathbf{a}-\mathbf{p}) \leqslant 0$).
- $\mathbf{a}$ should be chosen such that the average displacement between $\mathbf{a}$ and the perimeter vertices along $\mathbf{v}$ remains equal to h.

Let $\mathbf{a}^* = \mathbf{p} - t^*\mathbf{v}$. Then, the solution distance t^* is given by

$$t^* = \min\left(\left[\frac{1}{n}\sum_{i=1}^{n}(\mathbf{c}_i - \mathbf{p})^T\mathbf{v}\right] - h, 0\right).$$

When thresholding the energy in each spring, we use a per-spring threshold of a 10% change in length, which was used as the spring stretch limit in [137].

7.7 Derivation of Compliant Suction Contact Model

The basis of contact wrenches for the suction ring model is illustrated in Fig. 7.3. The contact wrenches are not independent due to the coupling of normal force and friction, and they may be bounded due to material properties. In this section we prove that wrench resistance can be computed with quadratic programming, we derive constraints between the contact wrenches in the suction ring model, and we explain the limits of the soft finger suction contact models for a single suction contact.

7.7.1 Computing Wrench Resistance with Quadratic Programming

The object wrench set for a grasp using a contact model with m basis wrenches is $\Lambda = \{\mathbf{w} \in \mathbb{R}^6 \mid \mathbf{w} = G\alpha \text{ for some } \alpha \in \mathcal{F}\}$, where $G \in \mathbb{R}^{6\times m}$ is a set of m basis wrenches in the object coordinate frame, and $\mathcal{F} \subseteq \mathbb{R}^m$ is a set of constraints on contact wrench magnitudes [121]. The grasp map G can be decomposed as $G = AW$ where $A \in \mathbb{R}^{6\times 6}$ is the *adjoint transformation* mapping wrenches from the contact to the object coordinate frame and $W \in 6 \times m$ is the *contact wrench basis*, a set of m orthonormal basis wrenches in the contact coordinate frame [121].

Definition 8. *A grasp* $\mathbf{u}$ *achieves wrench resistance with respect to* $\mathbf{w}$ *if* $-\mathbf{w} \in \Lambda$.

Proposition 1. *Let G be the grasp map for a grasp $\mathbf{u}$. Furthermore, let $\epsilon^* = argmin_{\alpha\in\mathcal{F}}\|G\alpha + \mathbf{w}\|_2^2$. Then $\mathbf{u}$ can resist $\mathbf{w}$ iff $\epsilon^* = 0$.*

Proof. ($\Rightarrow$). Assume $\mathbf{u}$ can resist $\mathbf{w}$. Then $-\mathbf{w} \in \Lambda$ and therefore $\exists \alpha \in \mathcal{F}$ such that $G\alpha = -\mathbf{w} \Rightarrow G\alpha + \mathbf{w} = \mathbf{0}$. ($\Leftarrow$). Assume $\epsilon^* = 0$. Then $\exists \alpha \in \mathcal{F}$ such that $G\alpha + \mathbf{w} = \mathbf{0} \Rightarrow G\alpha = -\mathbf{w} \Rightarrow -\mathbf{w} \in \Lambda$. □

When the set of admissible contact wrench magnitudes $\mathcal{F}$ is defined by linear equality and inequality constraints, the $\min_{\alpha\in\mathcal{F}}\|G\alpha + \mathbf{w}\|_2^2$, is a Quadratic Program which can be solved exactly by modern solvers.

Our suction contact model assumes the following:

1. Quasi-static physics (e.g. inertial terms are negligible).
2. The suction cup contacts the object along a circle of radius r (or "ring") in the $xy-$plane of the contact coordinate frame.
3. The suction cup material behaves as a ring of infinitesimal springs per unit length. Specifically, we assume that the pressure along the $z-$axis in contact coordinate frame satisfies $p(\theta) = k\delta_z(\theta)$ where δ_z is displacement along the z-axis and $k \in \mathbb{R}$ is a spring constant (per unit length). The cup does not permit deformations along the x or y axes.
4. The suction cup material is well approximated by a spring-mass system. Furthermore, points on the contact ring are in static equilibrium with a linear displacement along the $z-$axis from the equilibrium position: $\delta_z(\theta) = \delta_0 + ar\cos(\theta) + br\sin(\theta)$. Together with Assumption 3, this implies that:

$$p(\theta) = p_0 + p_x\cos(\theta) + p_y\sin(\theta)$$

   for real numbers p_0, p_x, and p_y.

5. The force on the object due to the vacuum is a constant $-V$ along the $z-$ axis of the contact coordinate frame.

6. The object exerts a normal force on the object $f_N = f_z + V$ where f_z is the force due to actuation. This assumption holds when analyzing the ability to resist disturbing wrenches because the material can apply passive forces but may not hold when considering target wrenches that can be actuated.

The magnitudes of the contact wrenches are constrained due to (a) the friction limit surface [76], (b) limits on the elastic behavior of the suction cup material, and (c) limits on the vacuum force.

Friction Limit Surface

The values of the tangential and torsional friction are coupled through the planar external wrench and thus are jointly constrained. This constraint is known as the *friction limit surface* [76]. We can approximate the friction limit surface by computing the maximum friction force and torsional moment under a pure translation and a pure rotation about the contact origin.

The tangential forces have maximum magnitude under a purely translational disturbing wrench with unit vector $\hat{\mathbf{v}}$ in the direction of the velocity:

$$\begin{aligned}
f_x &\leqslant \int_0^{2\pi} \mu \hat{\mathbf{v}}_x p(\theta) d\theta \\
&\leqslant \int_0^{2\pi} \mu \hat{\mathbf{v}}_x \left(p_0 + p_x \cos(\theta) + p_y \sin(\theta)\right) d\theta \\
&\leqslant 2\pi\mu\hat{\mathbf{v}}_x p_0 \\
f_y &\leqslant 2\pi\mu\hat{\mathbf{v}}_y p_0 \\
\|f_f\|_2^2 &\leqslant (2\pi\mu\hat{\mathbf{v}}_x p_0)^2 + (2\pi\mu\hat{\mathbf{v}}_y p_0)^2 \\
&= (2\pi\mu p_0)^2 \left(\hat{\mathbf{v}}_x^2 + \hat{\mathbf{v}}_y^2\right) \\
&= (2\pi\mu p_0)^2 \\
&= \mu f_N
\end{aligned}$$

The torsional moment has a maximum moment under a purely rotational disturbing wrench about the contact $z-$ axis. This disturbing wrench can be described with a unit vector $\hat{\mathbf{v}}(\theta) = (\sin(\theta), -\cos(\theta), 0)$. Thus the torsional moment is bounded by:

$$\begin{aligned}|\tau_z| &\leqslant \int_0^{2\pi} \mu r p(\theta) d\theta = \int_0^{2\pi} \mu r \left(p_0 + p_x \cos(\theta) + p_y \sin(\theta)\right) d\theta \\ &\leqslant 2\pi\mu r p_0 \\ &\leqslant r\mu f_N\end{aligned}$$

We can approximate the friction limit surface by the ellipsoid [75, 76]:

$$\frac{\|f_t\|_2^2}{(\mu f_N)^2} + \frac{|\tau_z|^2}{(r\mu f_N)^2} \leqslant 1$$

While this constraint is convex, in practice many solvers for Quadratically Constrained Quadratic Programs (QCQPs) assume nonconvexity. We can turn this into a linear constraint by bounding tangential forces and torsional moments in a rectangular prism inscribed within the ellipsoid:

$$\begin{aligned}|f_x| &\leqslant \frac{\sqrt{3}}{3}\mu f_N \\ |f_y| &\leqslant \frac{\sqrt{3}}{3}\mu f_N \\ |\tau_z| &\leqslant \frac{\sqrt{3}}{3}r\mu f_N\end{aligned}$$

Elastic Restoring Torques

The torques about the x and y axes are also bounded. Let $\mathbf{w}(\theta) = (r\cos(\theta), r\sin(\theta), 0)$. Then:

$$\begin{aligned}
\tau_t &= \int_0^{2\pi} (\mathbf{w}(\theta) \times \mathbf{e}_z) p(\theta) d\theta \\
\tau_x &= \int_0^{2\pi} r\sin(\theta) p(\theta) d\theta \\
&= \int_0^{2\pi} r\sin(\theta) \left(p_0 + p_x\cos(\theta) + p_y\sin(\theta)\right) d\theta \\
&= \int_0^{2\pi} r p_y \sin^2(\theta) d\theta \\
&= \pi r p_y \\
\tau_y &= \pi r p_x \\
\|\tau_e\|_2^2 &= \pi^2 r^2 (p_x^2 + p_y^2) \\
&\leqslant \pi^2 r^2 \kappa^2
\end{aligned}$$

where κ is the *elastic limit* or *yield strength* of the suction cup material, defined as the stress at which the material begins to deform plastically instead of linearly.

Vacuum Limits

The ring contact can exert forces f_z on the object along the z axis through motor torques that transmit forces to the object through the ring of the suction cup. Under these assumptions, the normal force exerted on the object by the suction cup material is:

$$\begin{aligned}
f_N &= \int_0^{2\pi} p(\theta) d\theta = \int_0^{2\pi} \left(p_0 + p_x\cos(\theta) + p_y\sin(\theta)\right) d\theta \\
&= 2\pi p_0
\end{aligned}$$

Note also that $f_N = f_z + V$, where f_z is the z component of force on the object, since the normal force must offset the force due to vacuum V even when no force is being applied on the object.

Constraint Set

Taking all constraints into account, we can describe $\mathcal{F}$ with a set of linear constraints:

$$\begin{array}{rlll} \textbf{Friction:} & \sqrt{3}|f_x| \leqslant \mu f_N & \sqrt{3}|f_y| \leqslant \mu f_N & \sqrt{3}|\tau_z| \leqslant r\mu f_N \\ \textbf{Material:} & \sqrt{2}|\tau_x| \leqslant \pi r \kappa & \sqrt{2}|\tau_y| \leqslant \pi r \kappa & \\ \textbf{Suction:} & f_z \geqslant -V & & \end{array}$$

Since these constraints are linear, we can solve for wrench resistance in the our contact model using Quadratic Programming. In this paper we set $V = 250N$ and $\kappa = 0.005$.

7.7.2 Limits of the Soft Finger Suction Contact Model

The most common suction contact model in the literature [86, 112, 161, 172, 182] considers normal forces from motor torques, suction forces from the pressure differential between inside the cup and the air outside the object, and both tangential and torsional friction resulting from the contact area between the cup and the object. Let $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$ be unit basis vectors along the x, y, and z axes. The contact model is specified by:

$$W = \begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{e}_z \end{bmatrix}$$

$$\alpha = (f_x, f_y, f_z, \tau_z) \in \mathcal{F} \text{ if and only if:}$$

$$\sqrt{f_x^2 + f_y^2} \leqslant \mu f_z$$

$$|\tau_z| \leqslant \gamma |f_z|$$

The first constraint enforces Coulomb friction with coefficient μ. The second constraint ensures that the net torsion is bounded by the normal force, since torsion results from the net frictional moment from a contact area. Unlike contact models for rigid multifinger grasping, f_z can be positive or negative due to the pulling force of suction.

Proposition 2. *Under the soft suction contact model, a grasp with a single contact point cannot resist torques about axes in the contact tangent plane.*

Proof. The wrench $\mathbf{w} = (\mathbf{0}, \tau_e)$ is not in the range of W because it is orthogonal to every basis wrench (column of W). □

The null space of W is spanned by the wrenches $\mathbf{w}_1 = (\mathbf{0}, \mathbf{e}_x)$ and $\mathbf{w}_2 = (\mathbf{0}, \mathbf{e}_y)$, suggesting that a single suction contact cannot resist torques in the tangent plane at the contact. This contradicts empirical evidence suggesting that a single point of suction can reliably hold and transport objects to a receptacle in applications such as the Amazon Picking Challenge [40, 60].

7.7.3 Details of Environment Model

We follow the state model of [105], which we repeat here for convenience. The parameters of the sampling distributions were set by maximizing average precision of the Q values using grid search for a set of grasps attempted on an ABB YuMi robot on a set of known 3D printed objects (see Section 7.5.4).

We model the state distribution as $q(\mathbf{x}) = q(\mu)q(\mathcal{O})q(\mathbf{T}_o|\mathcal{O})q(\mathbf{T}_c)$. We model $q(\mu)$ as a Gaussian distribution $\mathcal{N}(0.5, 0.1)$ truncated to $[0, 1]$. We model $q(\mathcal{O})$ as a discrete uniform distribution over 3D objects in a given dataset. We model $q(\mathbf{T}_o|\mathcal{O}) = q(\mathbf{T}_o|\mathbf{T}_s)p(\mathbf{T}_s|\mathcal{O})$, where is $q(\mathbf{T}_s|\mathcal{O})$ is a discrete uniform distribution over object stable poses and $q(\mathbf{T}_o|\mathbf{T}_s)$ is uniform distribution over 2D poses: $\mathcal{U}([-0.1, 0.1] \times [-0.1, 0.1] \times [0, 2\pi))$. We compute stable poses using the quasi-static algorithm given by Goldberg et al. [49]. We model $q(\mathbf{T}_c)$ as a uniform distribution on spherical coordinates $r, \theta, \varphi \sim \mathcal{U}([0.5, 0.7] \times [0, 2\pi) \times [0.01\pi, 0.1\pi])$, where the camera optical axis always intersects the center of the table. The parameters of the sampling distributions were set by maximizing average precision of the Q values using grid search for a set of grasps attempted on an ABB YuMi robot on a set of known 3D printed objects (see Section 7.5.4).

The grasp candidate model $q(\mathbf{u} \mid \mathbf{x})$ is a uniform distribution over points samples on the object surface, with the approach direction defined by the inward-facing surface normal at each point.

We follow the observation model of [105], which we repeat here for convenience. Our observation model $q(\mathbf{y} \mid \mathbf{x})$ model images as $\mathbf{y} = \alpha * \hat{\mathbf{y}} + \epsilon$ where $\hat{\mathbf{y}}$ is a rendered depth image created using OSMesa offscreen rendering. We model α as a Gamma random variable with shape= 1000.0 and scale=0.001. We model ϵ as Gaussian Process noise drawn with measurement noise $\sigma = 0.005$ and kernel bandwidth $\ell = \sqrt{2}px$.

Our grasp reward model $q(R \mid \mathbf{u}, \mathbf{x})$ specifies a distribution over wrench resistance due to perturbations in object pose, gripper pose, friction coefficient, and the disturbing wrench to resist. Specifically, we model $q(R \mid \mathbf{u}, \mathbf{x}) = q(R \mid \hat{\mathbf{x}}, \hat{\mathbf{u}}, \mathbf{w})q(\hat{\mathbf{x}} \mid \mathbf{x})q(\hat{\mathbf{u}} \mid \mathbf{u})p(\mathbf{w})$. We model $q(\mathbf{w})$ as the wrench exerted by gravity on the object center-of-mass with zero-mean Gaussian noise $\mathcal{N}(\mathbf{0}_3, 0.01\mathbf{I}_3)$ assuming as mass of 1.0kg. We model $q(\hat{\mathbf{u}} \mid \mathbf{u})$ as a grasp perturbation distribution where the suction target point is perturbed by zero-mean Gaussian noise $\mathcal{N}(\mathbf{0}_3, 0.001\mathbf{I}_3)$ and the approach direction is perturbed by zero-mean Gaussian noise in the rotational component of Lie algebra coordinates $\mathcal{N}(\mathbf{0}_3, 0.1\mathbf{I}_3)$. We model $q(\hat{\mathbf{x}} \mid \mathbf{x})$ as a state perturbation distribution where the pose $\mathbf{T}_o$ is perturbed by zero-mean Gaussian noise in Lie algebra coordinates with translational component $\mathcal{N}(\mathbf{0}_3, 0.001\mathbf{I}_3)$ and rotational component $\mathcal{N}(\mathbf{0}_3, 0.1\mathbf{I}_3)$ and the object center of mass is perturbed by zero-mean Gaussian noise $\mathcal{N}(\mathbf{0}_3, 0.0025\mathbf{I}_3)$. We model $q(R \mid \hat{\mathbf{x}}, \hat{\mathbf{u}}, \mathbf{w})$ a Bernoulli with parameter 1 if $\hat{\mathbf{u}}$ resists $\mathbf{w}$ given the state $\hat{\mathbf{x}}$ and parameter 0 if not.

7.7.4 Implementation Details

To efficiently implement sampling, we make several optimizations. First, we precompute the set of grasps for every 3D object model in the database and take a fixed number of samples of grasp success from $q(R \mid \mathbf{x}, \mathbf{u})$ using quadratic programming for wrench resistance evaluation. We convert the samples to binary success labels by thresholding the sample mean by $\tau = 0.5$. We also render a fixed number of depth images for each stable pose independently of grasp success evaluation. Finally, we sample a set of candidate grasps from the object in each depth image and transform the image to generate a suction grasp thumbnail centered on the target point and oriented to align the approach axis with the middle column of pixels for GQ-CNN training.

7.8 Discussion

In this chapter we developed a dataset generation model for training policies that plan grasps for a vacuum suction cup gripper from point clouds. Experiments suggest that this method generalizes well to novel objects on a physical robot, similar to previous results with a parallel-jaw gripper. This suggests that dataset generation methods can be generally extended to other grippers by developing a gripper-specific reward distribution based on the physics of the gripper. Furthermore, while this chapter considers only grasping a single object from a tabletop, the method can be extended to sequential grasping in clutter using the methods of Chapter 5.

The development of deep robust grasping policies for multiple grippers suggests a natural next question: can policies for the individual grippers be combined into a single composite policy that intelligently decides between the grippers based on observations of objects in the environment?

Chapter 8

Learning Deep Composite Policies with Analytic Supervision

Universal picking, the ability of robots to rapidly and reliably grasp a wide variety of objects, can benefit applications where the set of objects is constantly growing and changing such as e-commerce order fulfillment, flexible manufacturing, and home decluttering. This is challenging due to sensor noise and partial observability which make it highly difficult to infer the identity and pose of objects in the environment. Further complicating the problem are limitations of individual robot grippers due to the size, shape, and actuation mechanism. In practice, a single gripper may only be able to robustly grasp a subset of objects. For example, vacuum-based suction cup grippers can easily grasp objects with large and flat surfaces like boxes, but they cannot grasp objects smaller than the cup diameter (e.g. pencils), objects with high surface variation (e.g. mesh pencil cup), or porous objects (e.g. cloth).

A common approach to handling a wide variety of objects is to equip robots with a set of grippers and to use a high-level composite policy to select which gripper to use based on the available objects [60, 120, 184]. However, this raises a new difficulty: how can a robot reliably decide between grippers? Directly ranking grasps using heuristics or gripper-specific quality metrics for each gripper may be difficult because the metrics measure fundamentally different quantities. On the other hand, learning a composite policy from grasps rewards using human labels [187] or reinforcement learning [185] requires expensive data collection and may be prone to corrupt training examples.

In this chapter, we propose a hybrid method for learning composite policies by combining the idea of gripper-agnostic reward functions from reinforcement learning with analytic grasp quality metrics that can be scaled to rapidly generate millions of training examples. In particular, we unify the gripper-specific dataset generation distributions of Chapter 5 and Chapter 7 into a single model that uses a gripper-agnostic quality metric based on the grasp wrench space [121], the set of forces and torques that the gripper can apply through contact with an object. The metric can be evaluated for each gripper by computing the wrench set using gripper-specific geometry and contact models. By differentiating between grippers at the level of contact models, the grasp quality metrics are naturally calibrated to one another and can be directly compared to inform a decision between grippers. We use this model to generate the Dex-Net 4.0 training dataset of approximately 5 million parallel-jaw and suction cup grasps associated with synthetic point clouds and reward metrics computed over simulated heaps containing over 1,600 unique object CAD models.

The contributions of this chapter are:

1. A sequential grasping environment for Universal Picking with a suction cup and parallel-jaw gripper based on a gripper-agnostic reward metric.
2. Dex-Net 4.0, a dataset of approximately 5 million synthetic point clouds, grasps, and reward labels collected from the Universal Picking environment. The dataset also contains a new set of watertight and manifold 3D mesh models from Thingiverse augmented with synthetic cardboard backings, to simulate common products.
3. A composite grasping policy trained on Dex-Net 4.0 that maximizes the predicted grasp quality using a separate Grasp Quality CNN (GQ-CNN) for each gripper.
4. Experiments with bin picking on an ABB YuMi evaluating performance of the Dex-Net 4.0 composite policy on a physical robot, including comparisons with state-of-the-art heuristics, naive ranking of grasps with the GQ-CNNs of Chapter 5 and Chapter 7, and a policy fine-tuned on approximately 13k datapoints collected from physical grasp attempts.

The experiments suggest that the Dex-Net 4.0 has the highest success rate on a physical robot, with over 95% success and 310 mean picks per hour on heaps of 25 novel objects.

8.1 Problem Statement

We consider the problem of planning a sequence of grasps (Section 2.1.3) to iteratively move a set of objects from a cluttered bin to a receptacle using a set of available robot grippers such as a vacuum-based suction cup or parallel-jaws. A key subproblem is deciding which gripper to use to execute each grasp.

8.1.1 Objective

The objective is to find a policy π to maximize the success rate for iteratively lifting and transporting a single object from a cluttered bin up to a maximum of T grasp attempts, as introduced in Section 2.1.3:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}}\ \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} R(\mathbf{x}_t, \pi(\mathbf{y}_t), \mathbf{x}_{t+1})\right] \tag{8.1.1}$$

where $\mathbf{x}_t$ denotes that state at time t, $\mathbf{y}_t$ denotes the point cloud observation at time t, and the expectation is taken with respect to the joint distribution:

$$p(R_0, \mathbf{x}_0, \mathbf{y}_0, ..., \mathbf{x}_T, \mathbf{y}_T \mid \pi) = p(\mathbf{x}_0) \prod_{t=0}^{T-1} p(\mathbf{y}_t \mid \mathbf{x}_t) p(R_t \mid \mathbf{x}_t, \pi(\mathbf{y}_t), \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \pi(\mathbf{y}_t))$$

In particular, in this chapter we consider learning a policy that generalizes over a diverse distribution of possible initial object states $p(\mathbf{x}_0)$ by selecting from a set of grippers with different capabilities.

8.2 Dataset Generation

To learn a composite policy to plan grasps from point clouds, we automatically synthesize a massive training dataset of point clouds, grasps, and reward labels. Rather than sampling long time-sequences of grasps, we focus on increasing the diversity of the training dataset by maximizing the computational efficiency of generating a single training datapoint. The training dataset is generated by sampling tuples $(R_i, \mathbf{u}_i, \mathbf{y}_i)$ from a dataset generation distribution $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ using a one-timestep Monte-Carlo reward evaluation for a large set of actions over a diverse set of object states in cluttered heaps. We model $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ as the product of and environment distribution $q(R, \mathbf{x}, \mathbf{y} \mid \mathbf{u})$ and an action candidate distribution $q(\mathbf{u} \mid \mathbf{x}, \mathbf{y})$ based on an algorithmic supervisor to guide sampling toward actions that lead to high reward. We generate millions of samples from $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ to produce Dex-Net 4.0, a large-scale supervised training dataset for learning composite robot policies for bin picking.

8.2.1 Assumptions

The stochastic model is based on the following assumptions:

1. Quasi-static physics (e.g. inertial terms are negligible) with Coulomb friction.
2. Objects are rigid and made of non-porous material.
3. The robot has a single overhead depth sensor with known intrinsics, position, and orientation
4. The robot has two end-effectors with known geometry: a vacuum-based grippers with a single disc-shaped linear elastic suction cup, and a parallel-jaw gripper with a soft-finger contact model.

8.2.2 Definitions

States. Let $\mathbf{x} = (\mathcal{O}_1, ...\mathcal{O}_m, \mathcal{C}, \mathbf{w}_1, ...\mathbf{w}_m)$ denote the state of the environment at time t consisting of a single overhead depth camera, a set of objects, and a perturbation wrench on each object (e.g. gravity, disturbances). Each object state $\mathcal{O}_i$ specifies the geometry $\mathcal{M}_i$, pose $\mathbf{T}_{o,i}$, friction coefficient γ_i, and center of mass $\mathbf{z}_i$. The camera state $\mathcal{C}$ specifies the intrinsic parameters $\mathcal{I}$ and pose $\mathbf{T}_c$. Each wrench $\mathbf{w}$ is specified as a vector $\mathbf{w} \in \mathbb{R}^6$.

Grasp Actions. Let $\mathbf{u}_s \in \mathcal{U}_s$ denote a suction grasp in 3D space defined by a suction gripper $\mathcal{G}_s$ and a rigid pose of the gripper $\mathbf{T}_s = (\mathbf{R}_s, \mathbf{t}_s) \in SE(3)$, where the rotation $\mathbf{R}_s \in SO(3)$ defines the orientation of the suction tip and the translation $\mathbf{t}_s \in \mathbb{R}^3$ specifies the target location for the center of the suction disc Let $\mathbf{u}_p \in \mathcal{U}_p$ denote a parallel-jaw grasp in 3D space defined by a suction gripper $\mathcal{G}_s$ and a rigid pose of the gripper $\mathbf{T}_p = (\mathbf{R}_p, \mathbf{t}_p) \in SE(3)$, where the rotation $\mathbf{R}_p \in SO(3)$ defines the grasp axis and approach direction, and the translation $\mathbf{t}_p \in \mathbb{R}^3$ specifies the target center point of the jaws. The set of all possible grasps is $\mathcal{U} = \mathcal{U}_s \bigcup \mathcal{U}_p$.

Point Clouds. Let $\mathbf{y} = \mathbb{R}_+^{H \times W}$ be a 2.5D point cloud represented as a depth image with height H and width W taken by a camera with known intrinsics [57].

State Distribution. The initial state distribution $q(\mathbf{x})$ is equivalent to the model used in Chapter 5, which we repeat here for completeness. We model $q(\mathbf{x})$ as the product of distributions on:

1. *Object Count (m):* Poisson distribution with mean λ.

2. *Object Heap ($\mathcal{O}$):* Uniform distribution over a discrete set of m 3D triangular meshes $\{\mathcal{M}_0, ...\mathcal{M}_{m-1}\}$ and the pose from which each mesh is dropped into the heap.

3. *Depth Camera ($\mathcal{C}$):* Uniform distribution over the camera pose and intrinsic parameters.

4. *Coulomb Friction (α):* Truncated Gaussian constrained to $[0, 1]$.

The initial state is sampled by (1) sampling an object count m and a set of m 3D CAD models, (2) sampling a planar pose for the heap center and planar pose offsets from the pile center for each of the objects, and (3) dropping the objects one by one from a fixed height h_0 above the table and running dynamic simulation until all objects come to rest (all velocities are zero).

Reward Distribution. Binary rewards occur for grasps for which a quasi-static equilibrium is feasible when the object is perturbed by an external wrench (e.g. due to gravity or inertia). Let $\mathcal{O}_i \in \mathbf{x}_t$ be an object contacted by the gripper when executing action $\mathbf{u}_t$. Then we measure grasp success with a binary-valued metric $R(\mathbf{x}_t, \mathbf{u}_t) \in \{0, 1\}$ that measures whether or not:

- The gripper geometry in the pose specified by $\mathbf{u}_t$ is collision-free.

- The gripper contacts exactly one object $\mathcal{O}_i$ when executing the grasp parameterized by $\mathbf{u}_t$.

- The grasp can resist a random disturbing force and torque (wrench) $\mathbf{w}_t = \mathbf{w}_g + \epsilon_w$ on the grasped object with over 50% probability, where $\mathbf{w}_g$ is the fixed wrench due to gravity and ϵ_g is a random wrench sampled from a zero-mean Gaussian $\mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I})$.

8.2.3 Environment Distribution

A key advantage of hybrid grasp planning methods is the ability to rapidly generate training datasets. However, as found in Chapter 5, forward simulating the state of the heap for every grasp requires almost an order of magnitude more computation time per datapoint than the direct indexing method of Chapter 4. In Chapter 5, we attempted to address this by initializing the GQ-CNN with a set of pre-trained weights from training on millions of datapoints in Dex-Net 2.0 and fine-tuning on a smaller set of datapoints collected in heaps. However, fine-tuning is prone to effects such as catastrophic forgetting which can lead to unpredictable failures of the grasping policy [84].

In this chapter, we explore *action enumeration*, a computationally efficient approximation that performs a one-timestep Monte-Carlo evaluation of grasp quality for a large set of actions on each state of the heap. The tradeoff is that the set of states used in training will not reflect the effects of the policy's actions, but the training dataset can be orders of magnitude larger for the same computational budget. In theory this could lead to sequential learning phenomena such as the covariate shift, but we do not observe such effects as detailed in Section 8.4.

Therefore, the environment distribution used to generate the Dex-Net 4.0 dataset only evaluates states, observations, and rewards from the initial state distribution:

$$q(R, \mathbf{x}, \mathbf{y} \mid \pi) = q(\mathbf{x}) q(\mathbf{y} \mid \mathbf{x}) q(R \mid \mathbf{x}, \pi).$$

Unified Reward Distribution

In previous chapters, the analytic grasp quality metric was fundamentally different for the suction cup and parallel-jaw grippers. Suction cup grasps were measured by wrench resistance, the ability to resist a specific random disturbance, while parallel-jaw grasps were measured by epsilon quality, which analyzes the worst case ratio of contact forces applied through the grasp to the magnitude of the wrench that the contact forces exert. This is problematic because the metrics are not calibrated to one another. For example, the epsilon quality analyzes worst-case behavior and therefore may be more pessimistic than wrench resistance, which only considers a particular target wrench. Training a composite policy in such an environment may lead to overuse of the gripper which is measured with the more forgiving metric, and while this problem could be addressed with calibration, this could require hand-tuning and domain expertise.

We address this with a reward distribution based on gripper-agnostic grasp quality metrics that use gripper-specific contact models to determine the set of wrenches that can be applied to the object. Fig. 8.1 illustrates the reward distribution. Given an object consisting

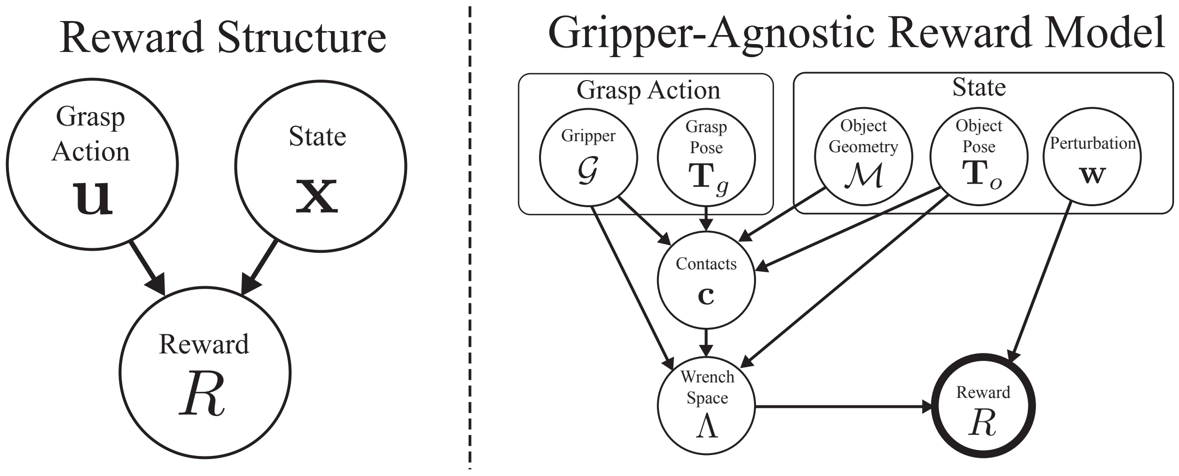

Reward Structure
Grasp Action $\mathbf{u}$
State $\mathbf{x}$
Reward R
Gripper-Agnostic Reward Model
Grasp Action
Gripper $\mathcal{G}$
Grasp Pose $\mathbf{T}_g$
State
Object Geometry $\mathcal{M}$
Object Pose $\mathbf{T}_o$
Perturbation $\mathbf{w}$
Contacts $\mathbf{c}$
Wrench Space Λ
Reward R

Figure 8.1: Gripper-Independent reward distribution for generating the Dex-Net 4.0 training dataset.

of a geometry $\mathcal{M}$ in pose $\mathbf{T}_o$, the gripper $\mathcal{G}$ (geometry and physical parameters such as friction) and grasp pose $\mathbf{T}_g$ are used to determine the contacts $\mathbf{c}$, or set of points and normals between the fingers and object. This is used to compute the set of wrenches Λ that the grasp can apply to the object under quasi-static physics and a point contact model. Specifically, the wrench set for grasp $\mathbf{u}$ using a contact model with m basis wrenches is:

$$\Lambda(\mathbf{u}) = \{\mathbf{w} \in \mathbb{R}^6 \mid \mathbf{w} = G(\mathbf{u})\alpha \text{ for some } \alpha \in \mathcal{F}(\mathbf{u})\}$$

as defined in Chapter 7. To recap, the grasp matrix $G(\mathbf{u}) \in \mathbb{R}^{6\times m}$ is a set of basis wrenches in the object coordinate frame specifying the set of wrenches that the grasp can apply through contact via active (e.g. joint torques) and passive (e.g. inertia) means. The wrench constraint set $\mathcal{F}(\mathbf{u}) \subseteq \mathbb{R}^m$ limits contact wrench magnitudes based on the capabilities of the gripper [121]. Finally, the grasp wrench set is used to measure grasp reward based on wrench resistance, or the ability of the grasp to resist a perturbation wrench $\mathbf{w}$ (e.g. due to gravity) as defined in Section 7.2.2 [90, 121]. The grasp reward $R = 1$ if the robust wrench resistance (Section 7.2.3) is greater than 50% over M samples from the graphical model.

8.2.4 Action Candidate Distribution

Rather than sampling only pre-computed grasps as in Chapters 4, 5, 7, we sample either pre-computed actions from the supervisor or actions generated from the point cloud observation to better reflect the distribution of grasps that the policy will have to evaluate on the physical robot. Formally, the supervisor-guided action candidate distribution is:

$$q(\mathbf{u} \mid \mathbf{x}, \mathbf{y}) = \begin{cases} \Omega(\mathbf{x}) & \text{with prob. } \epsilon \\ Unif(\mathcal{C}(\mathbf{y})) & \text{otherwise} \end{cases}$$

where the $\mathcal{C}(\mathbf{y})$ is the set of all candidate actions sampled from the point cloud with equal numbers of suction and parallel-jaw grasps. Note that we sample actions from the point cloud rather than the 3D object mesh surfaces as in past chapters. We found that this can improve the robustness of the resulting policy because the set of actions in the training dataset better reflects the set of actions that the policy evaluates when planning grasps based on point clouds. We use $\epsilon = 0.1$ to favor sampling grasps directly from the point clouds, which we found to work well empirically in Chapter 7.

Composite Supervisor

To guide grasp evaluation toward more promising candidates, we pre-compute a stochastic robust grasping supervisor $\Omega(\mathbf{x})$ that plans grasps using full knowledge of the state. An optimal supervisor maximizes the long-term reward of taking a sequence of actions from the current state. However, computing this policy is expensive due to long-term dependencies. We compute a greedy composite supervisor that can be sampled efficiently online by pre-computing a set of grasps for a set of known 3D objects in a database (such as Dex-Net [104]) that are robust to different possible orientations of each object. Since the state of each object $\mathbf{x}$ in the heap is not known ahead of time, the supervisor computes quality by taking the expectation with respect a surrogate reward distribution $\hat{q}(R \mid \mathbf{x}, \mathbf{u}) = \hat{q}(R \mid \tilde{\mathbf{x}}, \tilde{\mathbf{u}})\hat{q}(\tilde{\mathbf{x}} \mid \mathbf{x})\hat{q}(\tilde{\mathbf{u}} \mid \mathbf{u})$, where:

- The reward distribution $\hat{q}(R \mid \tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ deterministically computes the reward using analytic wrench resistance under gravity.
- The state distribution $\hat{q}(\tilde{\mathbf{x}} \mid \mathbf{x})$ samples an orientation of the object uniformly at random and a disturbing wrench $\mathbf{w}_t = \mathbf{w}_g + \epsilon_w$, where $\mathbf{w}_g$ is the fixed wrench due to gravity and ϵ_g is a random wrench sampled from a zero-mean Gaussian $\mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I})$.
- The action distribution $\hat{q}(\tilde{\mathbf{u}} \mid \mathbf{u})$ perturbs the grasp pose by a zero mean perturbation according to the Lie Algebra method used in Chapter 3.

The supervisor is implemented by first sampling a set of grasps for each gripper on each object $\mathcal{O}_i$ in the database. Suction grasps are sampled uniformly from the object mesh surface as in Section 7.3 and parallel-jaw grasps are sampled from the set of antipodal pairs on the object mesh as in Section 4.1.1. We denote by $\mathcal{U}_s(\mathcal{O}_i)$ and $\mathcal{U}_p(\mathcal{O}_i)$ the set of precomputed grasps for the suction and parallel-jaw gripper, respectively. The sizes of $\mathcal{U}_s(\mathcal{O}_i)$ and $\mathcal{U}_p(\mathcal{O}_i)$ are constrained to be equal to avoid biasing the training dataset toward on gripper over another. Given this set, the supervisor computes the quality of each grasp in $\mathcal{U}(\mathcal{O}_i) = \mathcal{U}_s(\mathcal{O}_i) \bigcup \mathcal{U}_p(\mathcal{O}_i)$ by setting the state to $\mathbf{x} = (\mathcal{O}_i, \mathbf{w})$ and evaluating:

$$\hat{Q}(\mathcal{O}_i, \mathbf{u}) = \mathbb{1}\left(\mathbb{E}_{\hat{q}}\left[R(\mathbf{x}, \mathbf{u})\right] > \tau\right)$$

using a sample approximation. We use a threshold $\tau = 0.75$. A higher threshold increases the probability that a supervisor grasp will be successful given a particular state of objects in a heap, but reduces the number of supervisor grasps for each state, making positive examples more rare in samples from q.

Given a full state of the heap $\mathbf{x}$, the supervisor plans grasps by first computing the set of available robust grasps for each object in world reference frame:

$$\mathcal{U}_i^*(\mathbf{x}) = \left\{ \mathbf{T}_{o_i}\mathbf{u} \mid \mathbf{u} \in \mathcal{U}(\mathcal{O}_i), \hat{Q}(\mathcal{O}_i, \mathbf{u}) = 1, collfree(\mathbf{x}, \mathbf{u}) \right\}$$

and then sampling a grasp from the set of all robust candidates uniformly at random:

$$\Omega(\mathbf{x}) \sim Unif\left(\bigcup_{i=1}^{m} \mathcal{U}_i^*(\mathbf{x}) \right)$$

8.3 Learning a Composite Policy

We learn a composite policy $\pi_\theta(\mathbf{y}_t) = \mathbf{u}_t$ that is composed of separate quality functions for each gripper Q_{θ_s} and Q_{θ_p} parameterized by Grasp Quality Convolutional Neural Networks (GQ-CNNs), where the policy parameters are defined by $\theta = (\theta_s, \theta_p)$ [185, 187]. The policy selects actions by maximizing quality over all available grippers:

$$\pi_\theta(\mathbf{y}_t) = \text{argmax}\ \left(\max_{\mathbf{u}_s \in \mathcal{U}_s} Q_{\theta_s}(\mathbf{y}_t, \mathbf{u}_s), \max_{\mathbf{u}_p \in \mathcal{U}_p} Q_{\theta_p}(\mathbf{y}_t, \mathbf{u}_p) \right)$$

8.3.1 Learning Objective

We train the composite policy by optimizing the weights of Q_{θ_s} and Q_{θ_p} on a large supervised training datset sampled from the composite grasp dataset generation distribution described in Section 8.2. Specifically, we sample a training dataset $\mathcal{D} = \{(R_i, \mathbf{y}_i, \mathbf{u}_i)\}_{i=1}^N$ from $q(R, \mathbf{x}, \mathbf{y}, \mathbf{u})$ and minimize the cross-entropy loss $\mathcal{L}$ over $\mathcal{D}$ separately for each gripper:

$$\theta_s^* = \underset{\theta_s \in \Theta_s}{\text{argmin}} \sum_{\mathbf{u}_{s,i} \in \mathcal{D}_s} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_{s,i})) \tag{8.3.1}$$

$$\theta_p^* = \underset{\theta_p \in \Theta_p}{\text{argmin}} \sum_{\mathbf{u}_{p,i} \in \mathcal{D}_p} \mathcal{L}(R_i, Q_\theta(\mathbf{y}_i, \mathbf{u}_{p,i})) \tag{8.3.2}$$

where $\mathcal{D}_s$ and $\mathcal{D}_p$ are the subsets of the training dataset $\mathcal{D}$ containing only the suction or parallel-jaw grasps, respectively.

8.3.2 Dex-Net 4.0 Training Dataset

The Dex-Net 4.0 training dataset is generated by sampling from q (defined in Section 8.2 for a large and diverse set of 3D object models.

The 3D object models, illustrated in Fig. 8.2, reflect a broad range of products that could be encountered in applications such as warehousing, manufacturing, or home decluttering. The dataset is augmented with synthetic “skinpack” meshes to reflect cardboard-backed products encountered in retail applications. Augmentation was performed by placing each

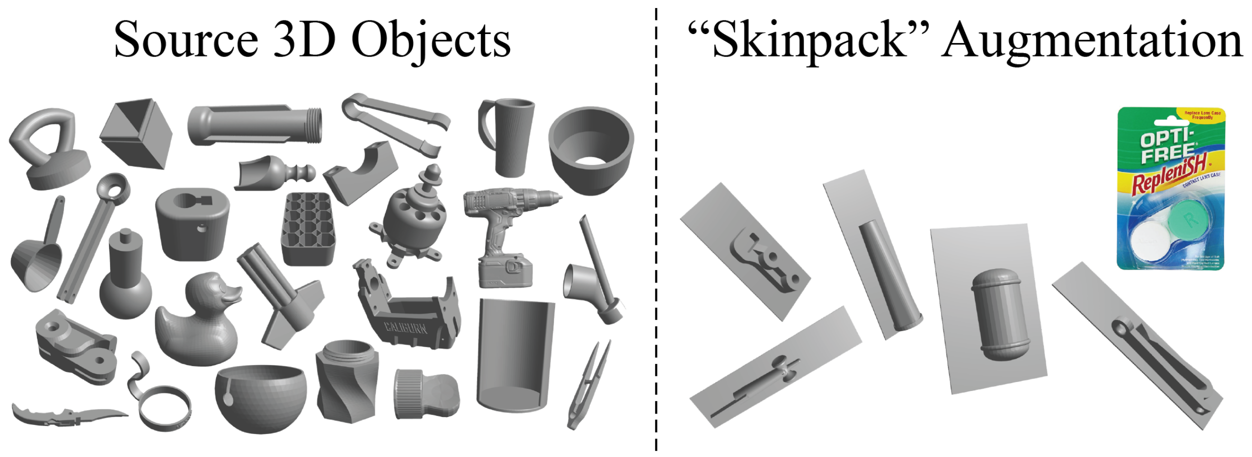


Figure 8.2: Subset of the 1,600 3D object models used to generate the Dex-Net 4.0 training dataset. (Left) The source meshes consist of approximately 800 3D triangular meshes selected from the set of freely available watertight and manifold meshes available on Thingiverse. (Right) The dataset is augmented with synthetic “skinpack” meshes to reflect cardboard-backed products encountered in retail applications.

source mesh in a quasi-static stable resting pose [49]) on an infinite planar worksurface and attaching a rectangular planar segment to the mesh at the triangle touching the worksurface.

To increase the computational efficiency of generating a single datapoint, the dataset contains a large set of labeled actions for each point cloud. Every state sampled from $q(\mathbf{x})$ has 5 associated depth images in Dex-Net 4.0 representing 3D point clouds captured from randomized camera poses and intrinsic optical parameters. Each image sampled from $q(\mathbf{y} \mid \mathbf{x})$ has a set of labeled actions for each gripper with associated quality metrics. Fig. 8.3 illustrates a training depth image from Dex-Net 4.0 with the set of labeled grasps. The intrinsic parameters for the virtual cameras are sampled around the nominal values for a Photoneo PhoXi S industrial depth sensor. Images are converted to 96×96 training thumbnails that are translated to move the grasp center to the thumbnail center pixel and rotated to align the grasp approach direction or axis with the middle row of pixels for the suction and parallel-jaw grippers, respectively.

In total, the Dex-Net 4.0 dataset contains approximately 5 million synthetic point clouds, grasps, and reward labels generated from 2,500 unique object heaps, with approximately 2.5 million datapoints for each gripper.

8.3.3 Optimization

The GQ-CNN architectures are similar to those used in Dex-Net 2.0 [105] and Dex-Net 3.0 [106] with two primary changes. First, we remove local response normalization as experiments suggest that it does not affect training performance. Second, we modify the sizes and pooling of the following layers : conv1_1 (16 9×9 filters, 1× pooling), conv1_2 (16 5×5 filters, 2× pooling), conv2_1 (16 5×5 filters, 1× pooling), conv2_2 (16 5×5 filters, 2× pooling), fc3 (128 output neurons), pc1 (16 output neurons), and fc4 (128 output neurons).

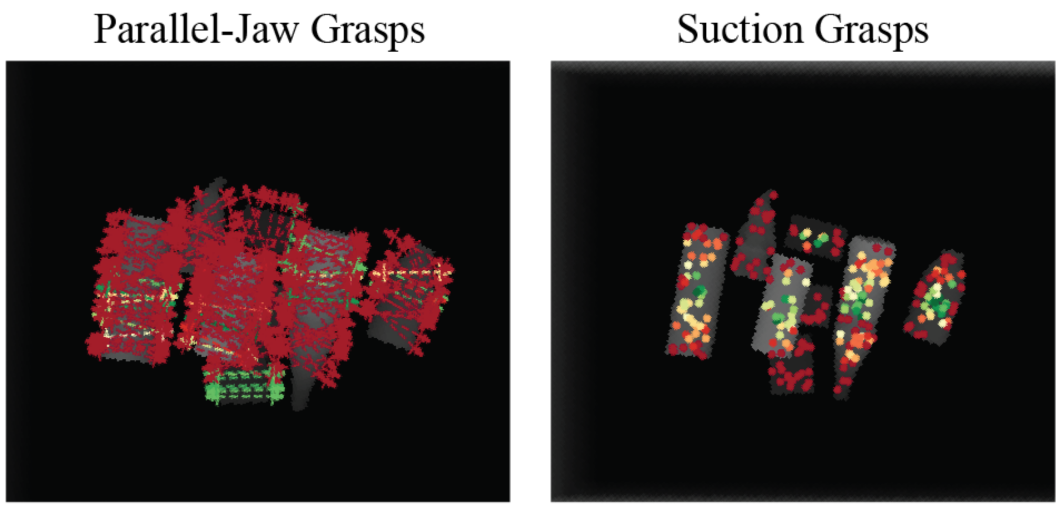


Figure 8.3: Example training image from Dex-Net 4.0 with associated labeled grasps for each gripper. Each grasp is colored by the robust wrench resistance metric reflecting the probability of successfully lifting and transporting a single object under disturbances due to gravity and random perturbations. Green indicates high probability while red indicates low probability. The binary grasp rewards are computed by thresholding the metric by 50%.

We train each GQ-CNN using stochastic gradient descent with momentum for 50 epochs using an 80-20 training-to-validation image-wise split of the Dex-Net 4.0 dataset. We use a learning rate of 0.01 with an exponential decay of 0.95 every 0.5 epochs, a momentum term of 0.9, and an $\ell 2$ weight regularization of 0.0005. We initialize the weights of the model by sampling from a zero mean Gaussian with variance $\frac{2}{n_i}$, where n_i is the number of inputs to the i-th network layer [59]. To augment the dataset during training, we reflect the image about its vertical and horizontal axes and rotate each image by 180° since these lead to equivalent grasps. We do not sample point cloud noise during training as in [104]. Training took approximately 24 hours on a single NVIDIA Titan Xp GPU. The learned GQ-CNNs achieves 96.2% and 97.5% classification accuracy for the suction and parallel-jaw grippers, respectively, on the held-out validation set.

8.3.4 Policy Deployment

We use the trained GQ-CNNs to plan grasps from point clouds on a physical robot by using derivative-free optimization to search for the highest quality grasp across both grippers. In particular, the policy uses the Cross Entropy Method (CEM) [98, 105, 106, 145]. CEM samples a set of initial candidate grasps uniformly at random from the set of surface points and inward-facing normals on a point cloud of the object, then iteratively re-samples grasps from a Gaussian Mixture Model fit to the grasps with the highest predicted probability of success. We optimize for the highest quality grasp from each gripper separately using the parameters of [106] for the suction gripper and the parameters of [105] for the parallel-jaw gripper, and then execute the grasp with the highest quality from the two planned grasps.

8.4 Experiments

We ran over 5,000 trials on a physical robot system to characterize the success rate of the Dex-Net 4.0 policy for a bin picking task across a dataset of 75 novel test objects. Our experiments aim to evaluate (1) the reliability of the Dex-Net 4.0 policy compared to previous iterations of Dex-Net and advanced heuristic grasp planners used in picking challenges and (2) the sensitivity of the Dex-Net 4.0 policy to the number of objects in the bin, the diversity of the training dataset, the GQ-CNN architecture, the availability of a training dataset of over 10k images collected empirically, and a memory system to avoid repeated failures.

8.4.1 Hardware Setup

The experimental setup is illustrated in Fig. 8.4. The benchmark hardware system consisted of an ABB YuMi bimanual industrial collaborative robot with an overhead Photoneo PhoXi S industrial 3D scanner, a custom suction gripper, and custom 3D printed parallel-jaw fingers with silicone fingertips [53]. The suction gripper consisted of a $20mm$ diameter silicone single-bellow suction cup seated in a 3D printed housing mounted to the end of the right arm. The vacuum was created by supplying compressed air from a Jun Air 18-40 quiet air compressor to a VacMotion MSV-35 vacuum generator. The payload of the suction system was approximately $0.9kg$ and a vacuum flow of approximately 8 standard cubic feet per minute. Objects were grasped from a picking bin mounted atop of a set of LoadStar load cells that tracked the weight of the bin with a resolution of approximately $5g$. Each gripper had a separate receptacle in which to drop the objects, one on each side of the bin.

8.4.2 Test Object Datasets

We created a dataset of 75 objects including common retail products, groceries, tools, office supplies, toys, and 3D printed industrial parts. We separated objects into three categories of 25 objects each, illustrated in the top panel of Fig. 8.5:

1. *Basic:* Lightweight prismatic solids (e.g. rectangular prisms, cylinders).
2. *Typical:* Common objects with clear plastic covers (“skinpack”), varied geometry, masses up to $500g$, and accessible, approximately planar surfaces.
3. *Adversarial:* Objects with complex geometry (e.g. curved or narrow surfaces) that are difficult to access or that do not satisfy the precise assumptions of the dataset generation distribution. For example, some objects have moving parts (e.g. can opener), porous surfaces (e.g. plush duck toy), deformable materials (e.g. fake grapes) or transparent surfaces (e.g. sink brush).

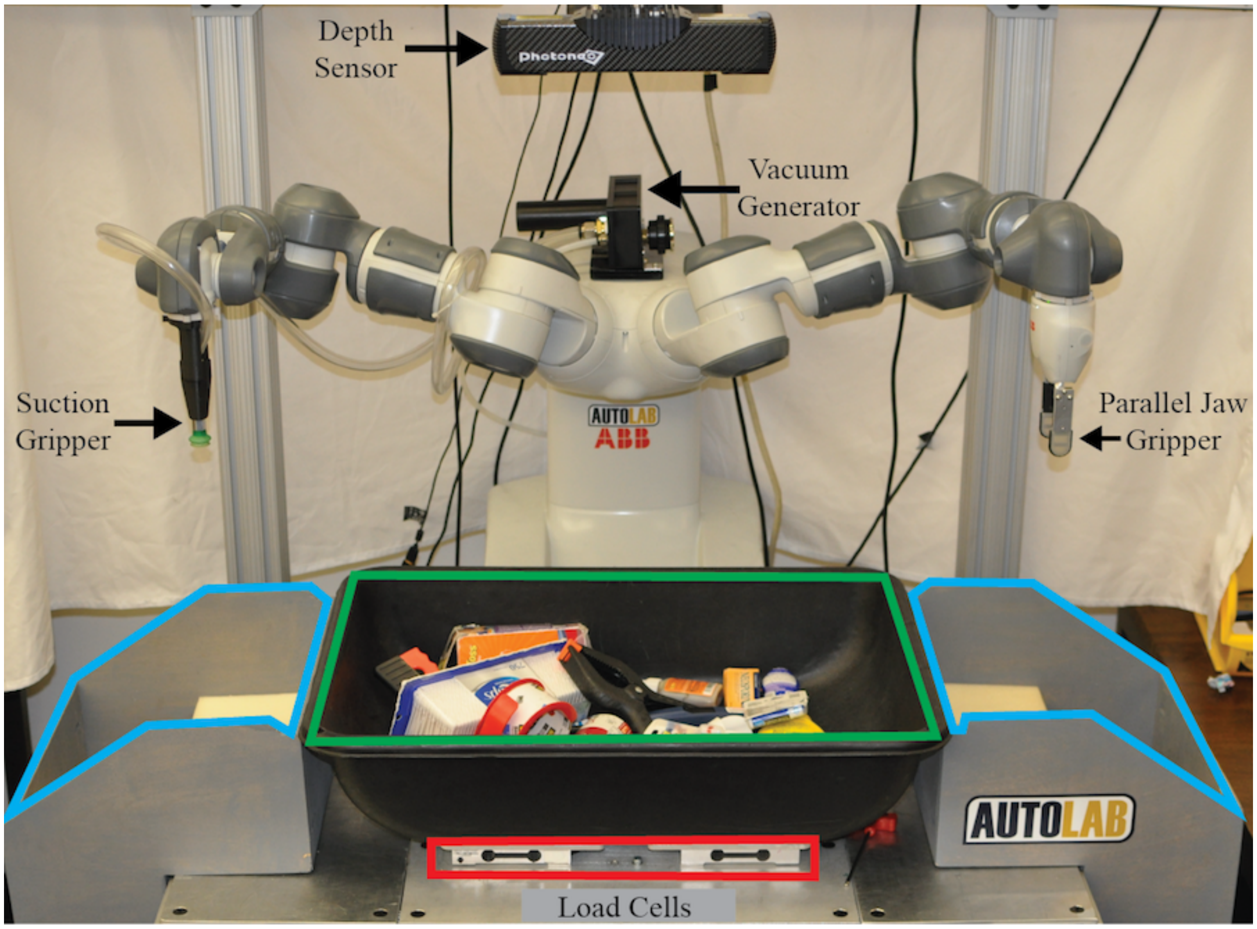


Figure 8.4: Physical benchmark used for evaluating composite bin picking policies. The goal was for the robot to iteratively transport each object from the picking bin (green highlight) to a receptacle (blue highlight) by planning a grasping pose for either the suction cup or parallel-jaw gripper based on a point cloud from an overhead Photoneo PhoXi S industrial depth sensor.

8.4.3 Experiment Protocol

Each experiment consisted of 5 independent trials in which the policy under evaluation attempted to pick each object from a test object dataset from the source bin and transport it to a receptacle. Before each experiment, the camera pose was registered to the robot using a chessboard. In each trial, the operator set the full dataset of objects in the bin by shaking the objects in a box, placing the box upside down in the bin, and mixing the bin by hand, ensuring that the objects rested below the rim of the bin. Then the robot iteratively attempted to pick objects from the bin. On each attempt, the grasping policy received as input a point cloud of the objects in the bin and returned a grasp action for exactly one of the grippers consisting of a pose for the gripper relative to the base of the robot. Then a common motion planning method was used to approach the target grasping pose along a linear end-effector trajectory and the robot moved to the pose, established contact with the object, and planned a motion to the receptacle. The operator labeled successes based on whether or not the robot was able to lift and transport the object to the receptacle on side of the workspace and also labeled the identity of each grasped object. If the object crossed over the boundary of the bin and bounced out of the receptacle, the grasp was still considered successful. A trial was considered complete after either all objects were removed, 75 total attempts, or 10 consecutive failures. All experiments ran on a Desktop running Ubuntu 16.04 with a 3.4 GHz Intel Core i7-6700 Quad-Core CPU and an NVIDIA Titan Xp GPU.

8.4.4 Description of Baselines

We compared performance with three baselines:

1. **Heuristic (Suction).** Ranks planar grasps based on the inverse distance to the centroid of an object [120], where the object centroid was estimated as the mean pixel of an object instance segmask from a tuned Euclidean Clustering segmentation algorithm from the Point Cloud Library. Candidate grasps planarity was determined by evaluating the mean squared error (MSE) of all 3D points within a sphere of radius $10mm$ (based on the suction cup size) to the best fit plane for the points. Then grasps were considered planar if either (a) the MSE was less than an absolute threshold or (b) the MSE was within the top 5% of all candidate grasps. The hyperparameters were hand-tuned to optimize performance for bin picking.

2. **Heuristic (Composite).** Ranks grasps planned with both the suction heuristic above and a parallel-jaw heuristic based on antipodality. The parallel-jaw heuristic was similar to the suction heuristic, ranking antipodal grasps based on the inverse distance to the estimated centroid of an object and determining antipodality based on estimated point cloud surface normals. The height of the gripper above the bin was set using a hand-coded formula to determine the offset from the point cloud withing the region of the grasp. The decision policy for the grippers was based on which grasp was closer to the estimated object centroid.

3. **Dex-Net 2&3 Composite.** Ranks grasps based on the estimated quality from separate GQ-CNNs trained to estimate the quality of parallel-jaw and suction cup grasps in clutter by fine-tuning the Dex-Net 2.0 and 3.0 base networks using the method of [103].

8.4.5 Bin Picking on Novel Objects

We evaluated the performance of the Dex-Net 4.0 policy versus the three baseline methods across the three test object datasets. Fig. 8.5 shows the results. Dex-Net 4.0 achieved the highest success rates across all object datasets with a success rate of 97%, 95%, and 63% on the Basic, Typical, and, Adversarial object datasets, respectively, compared to a success rate of 93%, 80%, and 50% for the next best baseline method. This corresponds to over 310 mean picks per hour (Section 2.1.3) for the Basic and Typical object datasets. The composite policy based on Dex-Net 2.0 and 3.0 does not perform as well. This may be attributed to the fact that the base features were trained using simulated noise for a low resolution Primesense Carmine sensor and evaluated on a high-resolution sensor.

To provide insight into the performance differences, we plot the mean cumulative reward, or mean number of objects picked versus the number of total pick attempts. We see that the baselines take longer to clear the last few objects from the bin, sometimes failing to clear several of the objects. Many of the failures of the Dex-Net 4.0 policy on the adversarial objects were due to consecutive failures. For example, it would repeatedly suction the plush duck toy without reacting to the failure. This can be attributed to the fact that the policy is greedy, failing to consider both the history of grasp attempts and the expected long-term reward.

8.4.6 Sensitivity to Amount of Clutter

The high success rate of the Dex-Net 4.0 policy on the Basic and Typical objects raised the question: is this performance maintained with larger amounts of clutter? To investigate this question, we evaluated performance on a dataset of 50 test objects combining all objects from both the Basic and Typical datasets. We evaluated the Dex-Net 4.0 policy on 5 independent trials in which each policy was allowed up to 100 total attempts and compared with the heuristics for a reference. Fig. 8.6 displays the results. We see that Dex-Net 4.0 has the highest success rate at 90%. In comparison, the performance of the heuristics is relatively unchanged with success rates near 80%. The cumulative reward plot suggests that the Dex-Net 4.0 policy makes most mistakes when the bin is nearly full. This is due to failed attempts to lift objects from underneath others. In comparison, the heuristics also fail when the bin is ful, but also have a significant reduction in success rate as the bin becomes close to empty.

8.4.7 Sensitivity to Training Dataset Diversity

We also investigated the necessity of the full 1600 3D object models and 2500 unique object heaps per gripper. We trained variants of the Dex-Net 4.0 policy on three variants of training datasets:

1. **Fewer Unique Objects.** 5 million training datapoints generated from 100 unique 3D objects in 2500 unique heaps.
2. **Very Few Unique Heaps.** 5 million training datapoints generated from 1600 unique 3D objects in 100 unique heaps.
3. **Fewer Unique Heaps.** 5 million training datapoints generated from 1600 unique 3D objects in 500 unique heaps.

We evaluated performance on the Basic and Typical test object datasets using the same experimental procedure as Section 8.4.5.

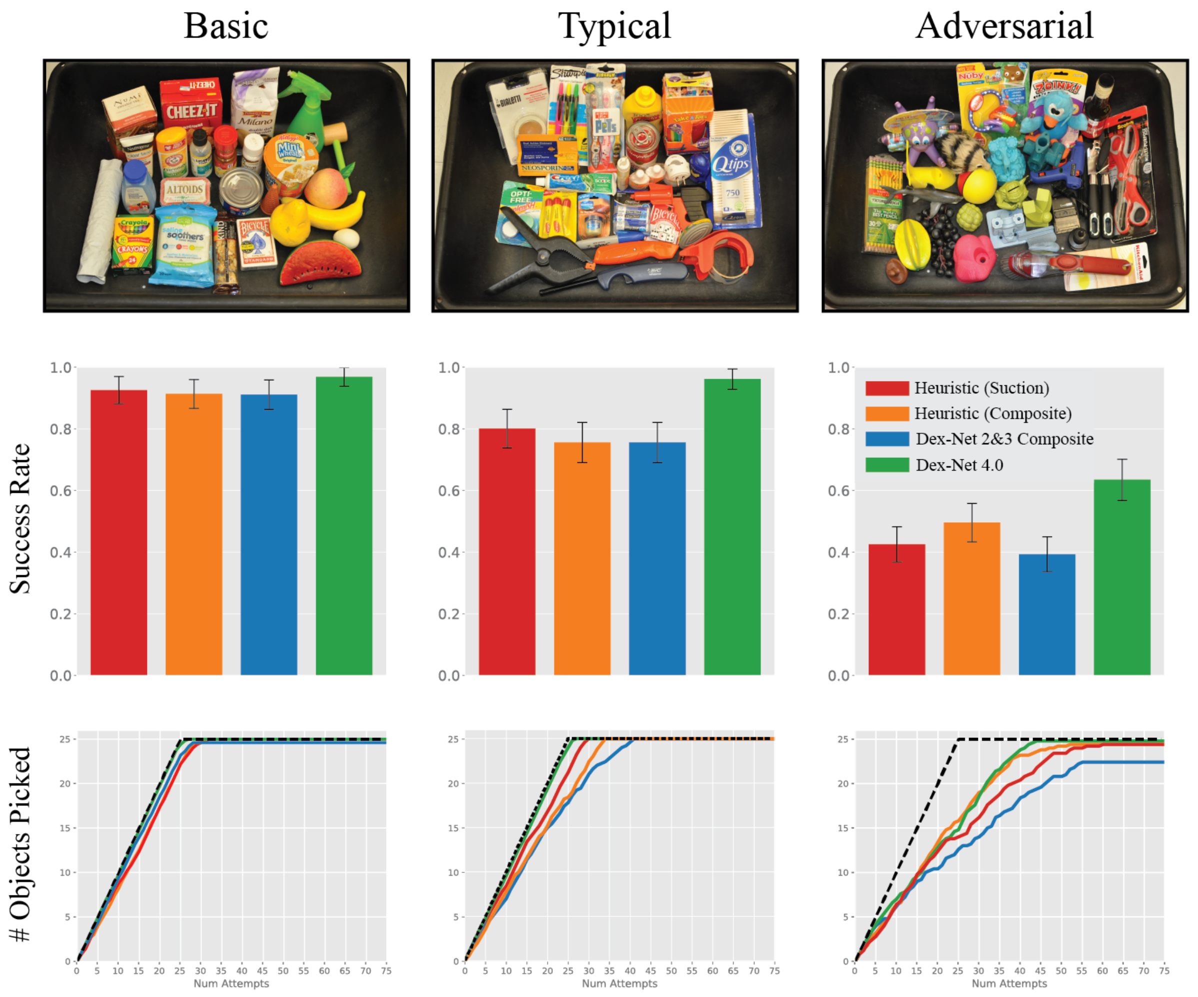


Figure 8.5: Results on bin picking benchmark across three object datasets for the Dex-Net 4.0 policy compared with the performance of the three baseline methods: (1) a heuristic for only the suction cup based on planarity and the distance to an estimated object centroid, (2) a heuristic for selecting between the suction heuristic and a parallel-jaw heuristic based on antipodality, and (3) a composite policy based on ranking grasps from GQ-CNNs fine-tuned from the Dex-Net 2.0 and 3.0 base networks using the method of [103]. (Middle Row) The overall success rates suggest that the Dex-Net 4.0 policy significantly outperforms the baselines on the Typical and Adversarial object datasets. (Bottom Row) The average number of objects picked versus the number of attempts suggest that the Dex-Net 4.0 policy makes fewer mistakes on the last few objects in the bin. For reference, the best possible performance (succeeding on every pick) is illustrated with a black-dotted line.

Fig. 8.7 displays the results. All variants lead to reduced performance, with fewer unique heaps leading to the greatest reduction in performance. This suggests that further increasing the diversity of the training dataset could lead to improved performance.

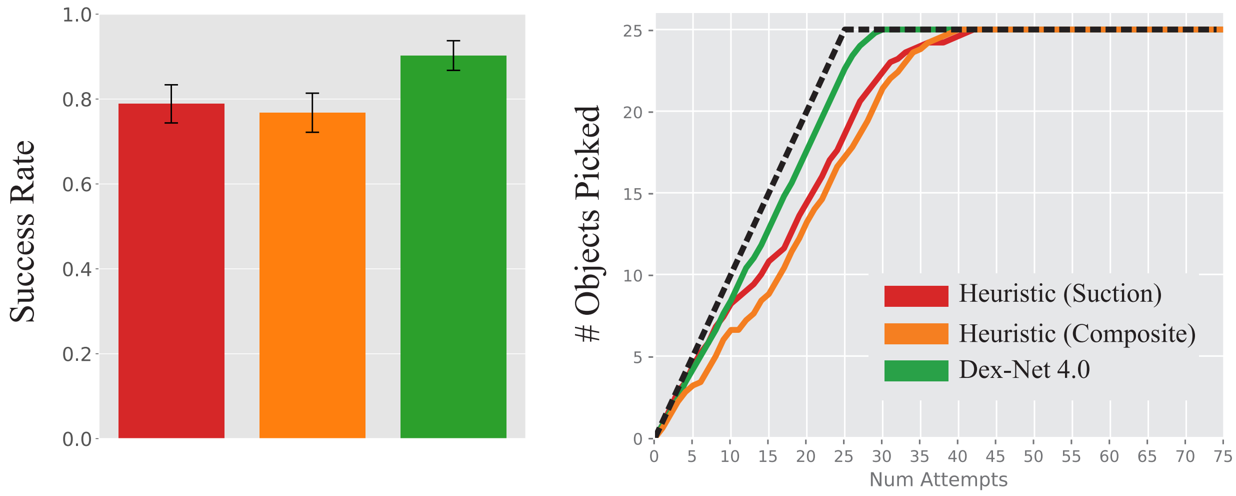


Figure 8.6: Bin picking benchmark results on a large dataset of 50 novel test objects combining the Basic and Typical datasets for the Dex-Net 4.0 policy and two heursistic baselines. The Dex-Net 4.0 policy has reduced performance than with 25 objects, but still significantly outperforms the baselines with over 290 mean picks per hour.

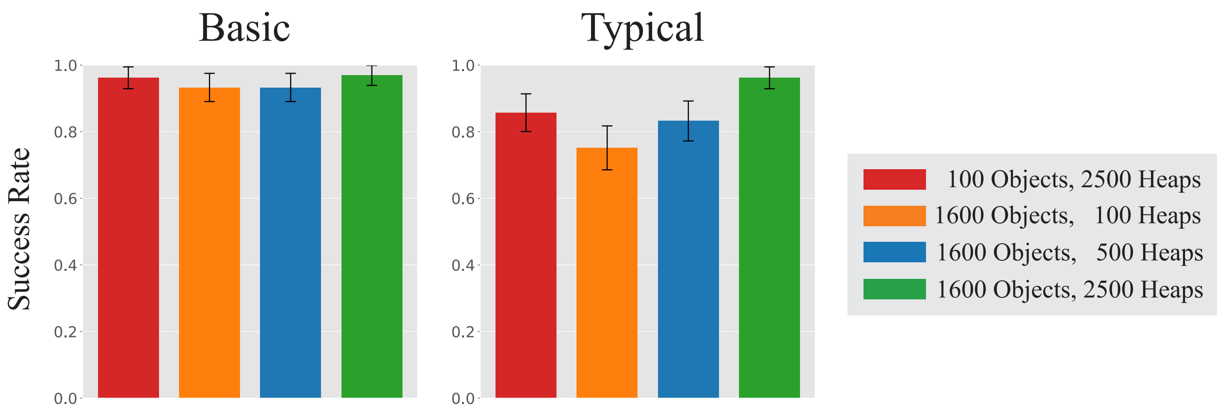


Figure 8.7: Bin picking benchmark results for variants of the Dex-Net 4.0 policy trained with less diverse training datasets on the Basic and Typical datasets. Lack of training dataset diversity appears to lead to reduced performance on more complex objects, and reducing the number of unique heaps appears to affect performance more than the number of unique objects.

8.4.8 Sensitivity to Neural Network Architecture

We also studied whether or not changes to the neural network architecture affected the performance of the resulting policy by training the Improved GQ-CNN model [68] on the Dex-Net 4.0 dataset and evaluating performance on all three 25 object test datasets. Fig. 8.8 displays the results. We see that the architecture maintains high performance, with no statistically significant difference between the Improved GQ-CNN architecture and standard GQ-CNN architecture across all datasets. This suggests that the high success rate may be more related to the Dex-Net 4.0 dataset itself than the network architecture.

8.4.9 Effects of Training on Physical Grasp Outcomes

A longstanding question in robot grasping is: to what extent can performance be improved from training on labeled grasp attempts on a physical system? We studied this question using the Dex-Net 4.0 Empirical Dataset, a dataset of over 13k labeled grasp attempts collected over the course of the Dex-Net 4.0 experiments and demonstrations of the Dex-Net 4.0 system in 2018 over hundreds of unique objects, including all objects used in the experiments. Approximately 5k datapoints were labeled by human operators in experiments and the remaining datapoints were labeled automatically by thresholding weight differences measured with high-precision load cells mounted underneath the bin.

The dataset collection policies were purely exploitative – every datapoint was the result of the best possible grasp according to the policy running on the robot. The reasons for this were twofold. First, in real industrial applications, exploration may lead to costly performance reductions such as lower mean picks per hour. Second, as collecting data on a physical system is expensive, we elected to use all data available from the hardware platform which was collected during exploitative performance evaluations rather than collecting a much smaller subset of data.

We trained ten variants of the Dex-Net 4.0 policy on varying amounts of empirical data: training from scratch on empirical data and fine-tuning with a reinitialized layers after the conv2_2 layer and after the fc4 layer on different ratios of real to synthetic data: 1-to-0, 1-to-1, 1-to-10, and 1-to-100. We tuned learning parameters to optimize performance on a fixed held-out validation set of datapoints from the empirical dataset. We then evaluated each policy on the Adversarial object dataset, since that was where the Dex-Net 4.0 network performed most poorly. The best performing policy, referred to as "Dex-Net 4.0 (FT-Empirical)", was the fine-tuned network with layers after fc4 reinitialized and a 1-to-10 ratio of real to synthetic data.

Fig. 8.8 shows the results. Surprisingly, the empirical network only achieved 65% success on the Adversarial objects. This is not a statistically significant gain in performance over the standard Dex-Net 4.0 policy. Furthermore, performance actually decreased on the Basic and Typical objects to 93% and 89%, respectively.

We believe that there are several possible explanations. The first is that the relatively small size of the dataset and exploitative dataset collection policy has caused the policy to fail to generalize to the large number of candidate grasps it considers during evaluation. Since the GQ-CNNs estimate the probability of success for 1,000 candidate grasps on every policy evaluation, there are likely to be grasp candidates that are dissimilar to other empirical datapoints. If the GQ-CNN is overconfident on any of these predictions as a result of overfitting, then those grasps will be selected for execution. The second hypothesis is that the GQ-CNN filters have learned a representation specific to synthetic data that cannot be

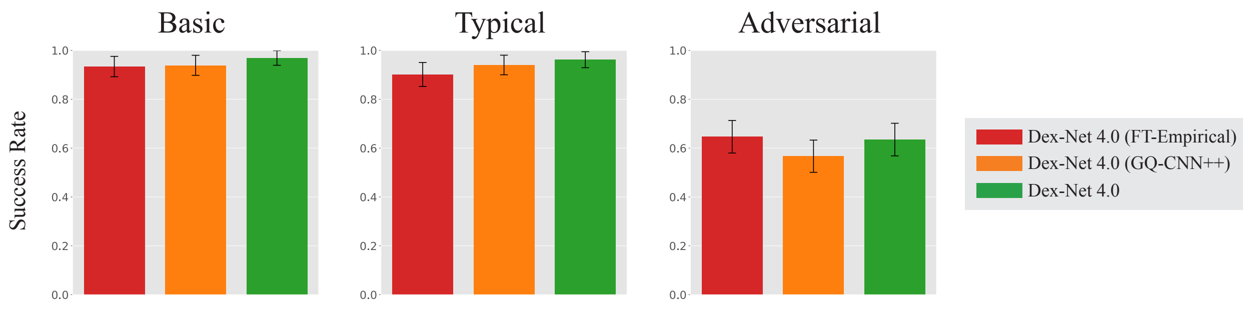


Figure 8.8: Bin picking benchmark results for two alternative Dex-Net 4.0 policies: the highest performing network fine-tuned on a dataset of 13k labeled grasp attempts on a physical robot and a Dex-Net 4.0 policy trained with the Improved GQ-CNN architecture [68]. The architectural variant maintains similar performance. Surprisingly, the empirically trained network does not lead to significant performance gains. This may be due to the relatively small size of the dataset or the skewed distribution of positive and negative examples between the synthetic Dex-Net 4.0 dataset and grasp attempts collected from experiments, which have many more successes.

generalized to real datapoints with simple fine-tuning. This could in theory be remedied by training on only empirical datapoints. However, our previous analyses in Chapter 4 suggest that this is not the case.

We wish to note emphasize that these experiments should not discourage the use of empirical data to improve performance in general. There are many other more sophisticated transfer learning methods that could help such as domain adaptation [170] or Generative Adversarial Networks [12]. Furthermore, collecting larger numbers of training examples or using reinforcement learning to guide the dataset collection distribution could increase performance. Further experiments will be necessary to understand this phenomenon. Nonetheless, it is clear that learning from real data is nontrivial, and it simply may not be feasible when data is collected from purely exploitative policies. A similar phenomenon was recently published for the task of re-orienting a cube with a multifingered hand [127] in which training on data from the physical Shadow Hand did not lead to increased performance, suggesting that training on only synthetic data may have better reliability on a physical robot.

8.4.10 Effects of Memory

Since we observed that the majority of failures of the Dex-Net 4.0 policy on the Adversarial objects were due to repeated failures, we studied the effect of using a simple memory-based policy to avoid attempting near-identical grasps on the same object several times in a row. We used an instance recognition system based on segmentation and matching objects in feature space [185]. The memory system tracked failures by monitoring the weight of objects in the bin using the load cells. A failure was marked when the change in weight was less than $5g$. When a failure occurred, the point cloud was segmented using Euclidean Clustering from the PCL library and the segmentation was used to mask out the region of a grayscale

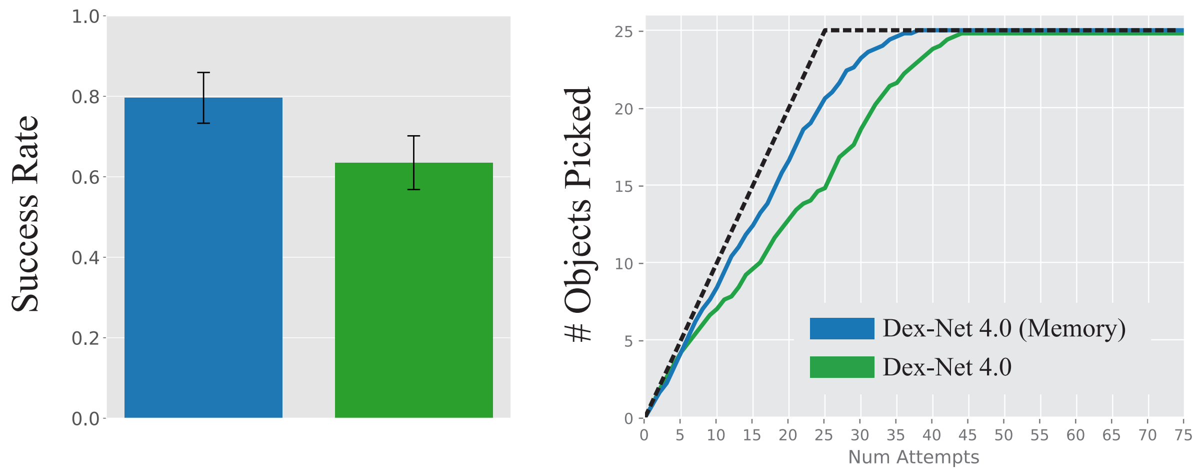


Figure 8.9: Results on the bin picking benchmark using a variant of the Dex-Net 4.0 policy augmented with memory and the ability to push objects to adapt to failures. The memory-based policy achieves 80% success on the Adversarial objects at over 250 mean picks per hour and does not leave any objects behind in the bin.

image of the objects in the bin corresponding the to the segmask containing the grasp center pixel. The segmented image patch was featurized using the VGG-16 network and stored in a failure database corresponding to the gripper.. On the next grasp attempt, the point cloud was pre-segmented and each segment was featurized using VGG-16 and matched to the failure database. If the distance in feature space was less than a threshold and the segmentation mask was within X pixels of the object when the failure occurred, then the matching region in the current image was converted to a constraint on the grasp sampler for the Dex-Net 4.0 policy – no candidate grasps were allowed to be sampled from the failure region. Furthermore, if greater than three consecutive failures occurred then the memory system rejected the planned grasp and used the pushing policy of [29] to perturb the objects in the bin. A detailed description and experimental characterization of this memory system is left for future work.

Fig. 8.9 shows the results. The memory system increased the success rate on the Adversarial to 80%, significantly higher than any other method. This suggests that a straightforward memory-based policy can lead to performance gains on difficult objects that cause repeated failures.

8.4.11 Pathological Objects

Over the course of experiments we noted that several classes of objects could not be reliably handled with Dex-Net 4.0 but are possible to be handled with the combination of a suction and parallel-jaw gripper. Fig. 8.10 displays a selected subset. One class of objects has transparent surfaces that trigger the parallel-jaw policy to attempt a grasp that appears to

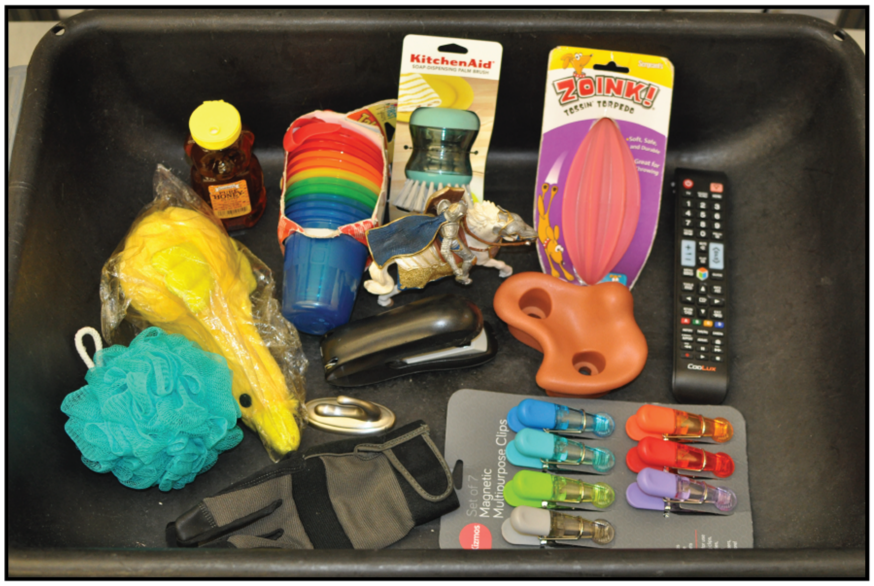


Figure 8.10: Several objects that cause failures across composite policies that plan grasps for suction and parallel-jaw grippers based on point cloud geometry alone.

have clearance when imaged with a depth sensor, but leads to a collision with the transparent surface. Another class has very structured small variations in the surface, such as parallel lines of stacked cups, buttons on a remote, or detailed embellishments of action figures, which the suction network tends to ignore and attempt grasps on. This is likely due to the lack of such objects in the training dataset of 3D object models. Other classes include (1) objects with loose transparent packaging that cannot be sensed with structured light or suctioned with the existing suction hardware, (2) porous objects that appear to have flat surfaces, and (3) reflective objects.

8.5 Discussion

In this chapter, we presented a method for learning composite grasping policies that can decide between a set of alternative grippers by training on synthetic datasets generated with analytic supervision and a reward model that measures grasp success in a gripper-independent manner. We implemented the method and used it to generate Dex-Net 4.0, a dataset of approximate 5 million training datapoints for a suction cup and parallel-jaw grippers. Thousands of experiments on a physical ABB YuMi suggest that policies trained on Dex-Net 4.0 have very high success rates on novel objects on a physical robot, with a success rate of over 95% on heaps of 25 novel Basic and Typical objects at over 310 mean picks per hour. Furthermore, a memory system to avoid repeated failures can achieve 80% on adversarial objects. Surprisingly, training on only synthetic data outperforms a many variants of policies fine-tuned on a dataset of 13k real grasp attempts.

Results suggest several areas for future work. Extending the model to evaluate grasp reward based on all contacts between objects could reduce failures due to grasping objects that are blocked by other objects in the heap. Extending the model to new grippers such as suction cup arrays, underactuated two finger grippers, or soft hands could increase the range of objects that hybrid composite policies can handle. Extending the model to synthetic color images could enable the system to sense transparencies, specularities, or highly textured surfaces. Extending the model to deformable and porous objects could help avoid overconfident grasps with a suction cup gripper. Nonetheless, significant performance gains on the task of bin picking with novel objects may require larger changes to the method: taking into account the history and future expected rewards that we know are required to achieve optimality in partially observed problems, training on empirically collected data to reduce the simulation-to-reality gap, and using feedback policies that can react on the fly using force and tactile sensing instead of executing grasps in open loop.

Part IV

Conclusion

I didn't mean to take up all your sweet time,
I'll give it right back one of these days.
If I don't meet you no more in this world,
I'll meet you in the next one, and don't be late.

JIMI HENDRIX - VOODOO CHILD (SLIGHT RETURN)

Chapter 9

Discussion

9.1 Overview

This thesis studied a hybrid grasp planning method: to use physics-based analytic models of robot grasping to supervise robust robot grasping policies that generalize to novel objects. We showed that robust analytic models can be used to efficiently sample and evaluate millions of grasps across a diverse set of 3D object geometries using distributed computation in the Cloud. Using these models, we constructed an end-to-end dataset generation distribution to randomize training examples of images, grasps, and analytic rewards over sensor parameter, physics parameters, and thousands of unique objects. To evaluate performance we implemented this technique in the Dexterity Network (Dex-Net) project, which includes code and datasets for hybrid grasp planning. We sampled massive training datasets from Dex-Net and learned the weights of a deep neural network to predict grasp quality directly from images.

On physical robot benchmarks for single object grasping and bin picking, we studied performance across parallel-jaw and vacuum suction cup grippers when planning grasps for a varying number of objects from depth images taken with an overhead RGB-D sensor. Results demonstrate that robot policies that plan a grasp to maximize a quality function learned from Dex-Net can transfer to a physical robot with no additional training, and the policies achieve state-of-the-art success rates on the custom grasping benchmarks. This suggests that hybrid grasp planning based on analytic supervision can be used to train fast and reliable robot grasping policies that generalize across robots, gripper types, and picking tasks.

9.2 Takeaways

While this thesis presented thousands of empirical results suggesting that the hybrid grasp planning method works well, an important question has not yet been addressed: why does the proposed method work? To confront this question, it is important to re-visit some of the unique assumptions and properties of the method for the specific grasping tasks considered in this thesis.

9.2.1 Relevance of Analytic Grasp Quality Metrics

First, the results of this thesis suggest that analytic grasp quality metrics are relevant to predicting grasp success on a physical robot Nonetheless, quasi-static wrench space metrics have been criticized for making too strong of assumptions for considering only a necessary, not sufficient, condition for dynamic grasp stability [11].

This thesis suggests that the assumptions of grasp quality metrics should not be conflated with the assumptions of the grasp planner. Using full knowledge of the geometric, material, and mass properties of objects can be useful when used as a means to generate data for discriminative learning rather than requiring a state estimation system to estimate these properties at runtime. Furthermore, it is known that grasps with a feasible quasi-static equilibrium can be made stable by constructing a feedback controller composed of virtual springs at the contact points [123]. The results of this thesis suggest that on a physical robot with low acceleration, dynamic stability is often achieved. Where does the stability come from? Perhaps this is solved in hardware, where compliance in the fingertips acts naturally as a passive stability controller under some pre-loading of the fingers. Perhaps this is also facilitated by area contacts which provide additional stability.

Regardless of the exact reason this thesis substantiates the claim that analytic metrics based on unrealistic assumptions can still be useful when used in the right way. Under certain parameter settings, this thesis found empirically that grasp quality metrics can act as a useful proxy for sufficient conditions for grasp stability, rarely predicting false positives. Despite the theoretical shortcomings, these simple and computationally efficient models from physics and geometry can give us a principled way to design useful grasp reward functions. By understanding the assumptions of grasping in new applications, it may be possible to develop new computationally efficient task-specific reward functions.

9.2.2 Bias-Variance Tradeoff in Dataset Collection

Second, as described in Chapter 2 the hybrid grasp planning method is based on a key hypothesis: that collecting a massive amount of consistent training data from a model can lead to better generalization than collecting large dataset of unbiased sensor, robot, object, and environment-specific data from a physical system or human labelers. This result may be unsettling even after so many pages of experiments because it is obvious that the model used to generate the data is not precisely modeling the behavior of the real world system. Clearly there are limits, so it may be tempting to believe that training on only real data eventually will eventually outperform method and that long-term research should focus efforts on efficient methods for unbiased dataset collection.

While reducing dataset collection time is clearly valuable in the near term, the results of this thesis suggest that there may be a deeper reason for the reliability of the hybrid method.

In particular, the findings bear similarity to a classic result in machine learning: the bias-variance tradeoff. Although the tradeoff is typically analyzed in terms of the function class, this thesis suggests that a similar result may hold with respect to the dataset used to collect training data: using a biased model for automating data collection may improve the scale and consistency of training datasets and lead to better performance on a physical system in comparison to methods based on smaller, noisy training datasets. At least for the tasks considered in this thesis, increasing dataset bias seems to lead to much higher reliability for specific sets of objects, grippers, and cameras that are well-approximated by analytic models. Perhaps decreasing dataset bias is not the only factor worth considering in robot learning for manipulation. There is reason to believe that increasing bias by generating massive synthetic training datasets with computationally efficient models from physics can actually result increased reliability of learned grasping policies on a physical system. A theoretical argument is left for future work.

9.2.3 Visual Grasp Affordances Derived from Physics

Third, the results of this thesis suggest that there is a deep connection between the idea of grasp affordances and analytic grasp planning. A large body of research has considered how to define affordances such as handles, boxes, and bars that facilitate grasping novel objects, but much of the work has been based on heuristics or empirical learning from human labels. This had been thought to be distinct from analytic grasp planners that compute the wrench space based on exact object shape. By turning analytic metrics into a reward signal for discriminative learning, this thesis suggests a re-interpretation of affordances: geometric structures that are indicative of the ability to apply forces and torques to a given object. These affordances are a natural outcome of generating a dataset of images, grasps, and analytic reward labels from a collection of common 3D object geometries, and deep neural networks are particularly good at learning to recognize them. In other words, the results of this thesis suggest that certain geometric features of objects that facilitate grasping have a very distinct appearances when considering the subset of shapes that are likely to be found in common industrial and home environments.

9.2.4 Abstractions for Reliable Robot Learning

Fourth, efficient learning of robust robot grasping policies was only possible in this thesis by carefully considering the design of a framework to use machine learning where it could provide the most benefit. While one could approach the grasp planning problem by learning an entire torque-controller to move an arm and gripper through contact with objects based on vision, significant data could be needed to simply learn motions in free space that can be executed reliably using established techniques from motion planning and control. In contrast, the hybrid approach considered herein only planned a minimal parameterization of a grasp as a gripper pose, and this parameterization could be used to define and entire motion plan and controller to grasp and transport objects. By focusing learning on only generalizing planning of the final gripper pose to novel objects, every datapoint contributed toward the aspect of grasping that could benefit from learning rather than re-learning known capabilities. Furthermore, rather than using complex architectures or training methods to regress directly to a grasp pose, we considered framing the problem as simple binary classification which is well-understood in theory to the machine learning community and well-understood in practice to the image classification community in computer vision. These abstractions also reduced the complexity of sequential learning by reducing the frequency and difficulty of learned decisions which are known to be important factors for reducing phenomena such as the covariate shift. Overall, carefully designed abstractions may have been a major contributor to the reliability and generalization ability of policies learned in this thesis.

9.3 Opportunities for Future Research

This thesis revealed several limitations of the hybrid method and opportunities for future work.

9.3.1 Extensions to the Grasping Environment

The grasping environments for synthetic dataset generation could be extended in a variety of ways to model a wider range of possible grasping problems:

- **Color Image Sensors.** This thesis only considered depth images and segmentation masks, but recent research on domain randomization [169] suggests that a very similar stochastic synthetic dataset generation method could be used to generate training datasets with color images. This can be applied to 3D models by randomizing the texture of the object. To generalize to the complex texture on real objects (e.g. logos, text), one possibility is to sample textures for the 3D CAD models from color images of real world objects.

- **Tactile Sensors.** Recent developments in tactile sensing such as GelSight [71] give higher resolution tactile data than was previously possible. Recent research suggests that regrasping policies can be learned from this data to adapt to possible failures on the fly [65]. Future research could study whether or not this tactile data could be simulated to learn grasp quality functions for a regrasping policy.

- **New Grippers.** Recently, soft and underactuated robot grippers have become increasingly popular due to their robustness to object shape and positioning error. Building an analytic data generation distribution for such grippers may challenging due to the lack of analytic models for the gripper. Computationally efficient models for simulating these grippers, such as raytracing for fast evaluation of contact areas, could be useful for implementing a hybrid approach. On the other hand, multifingered grippers are challenging due to the number of degrees of freedom. Recent research suggests that the hybrid method could be extended to multifinger grippers by framing the problem as probabilistic inference in a deep neural network [101]. Another possibility would be learn a grasp quality function that evaluates grasps defined by a pre-grasping configuration of a multifingered hand and a fixed policy for closing the fingers (sometimes referred to as a "grasp synergy").

- **Complex Actions.** In this thesis, actions were parameterized by a gripper pose and grasps were executed by controlling the robot to reach the desired pose, close the gripper, and lift the object. To increase success rate, it may be fruitful to explore continuous servoing methods. This has been studied by [98, 74, 176], but additional research on 3D servoing with eye-in-hand cameras could be interesting to consider. Another set of actions that may be interesting to consider are "multi-hand" grasps, in which a robot uses more than one gripper to lift a large object.
- **Inter-Object Contacts.** This thesis only considered contacts between a single gripper and object, but the high-level hybrid method generalizes to the case where reward functions assess chains of contact between objects in clutter. A reward function that considers the feasibility of quasi-static equilibria with the resultant wrench from all objects resting on one another based on a static kinematic contact chain could be a starting point. Modeling such phenomena could be useful to increase grasp success rates in dense clutter or for extensions of the method to tasks that involve careful placement amidst contact such as stacking.
- **Kinematics.** The environments presented in this thesis are agnostic to the robot kinematics, assuming that a gripper pose can always be reached by controlling the joints of the arm. In practice, this may not be true. Learning to represent the set of kinematically feasible grasps may challenging. For example, the grasp pose is no longer rotationally or translationally invariant. A starting point could be to equip the grasping environments with kinematic models, but a complete solution may need to consider new network architectures that leverage the structure of the kinematic chain for efficient learning.
- **Reactive Policies.** The results of this thesis make heavy use of simple action parameterizations that specify an open loop grasping policy. For more complex manipulation it may be important to use a policy that reacts to sensor inputs while executing a grasp. Hybrid training of such fine-grained feedback policies raises several interesting challenges such as making use of additional sensors that can, for example, measure slip or shear, and optimizing for long-term reward.

9.3.2 Extensions to New Tasks

It may be fruitful to consider implementing the hybrid method for the following novel grasp planning problems:

- **Mobile Picking.** Applications of mobile robots in medicine and in the home will require robots to lift and transport novel objects from human environments In theory, these environments be modeled using techniques developed in this thesis. The camera pose could be randomized about where the mobile robot camera is expected to be mounted. The background objects could be randomized by placing objects in synthetic scenes such as those considered in [191]. The object poses could be randomized by selecting stables poses on flat surfaces in the static environment. Potential challenges include: (1) handling sensor noise due to natural light and clutter and (2) being robust to large uncertainties in the target gripper pose due to imprecision in the robot base.

- **Regrasping.** A fundamental problem in robot manipulation is regrasping [18], or reconfiguring an object relative to a gripper. These problems could be approached by framing the problem as planning a sequence of grasps, object placements, and pushes to reach a goal pose of the object relative to the hand. For example, Dex-Net 2.0 could be extended to rotate an object between stable poses by pick-and-place regrasping on a tabletop.

- **Mechanical Search.** Another important problem in manipulation is searching for a particular object among clutter and moving it to a target position. This may require the definition of additional grasp primitives, such as rummaging, and additional perception capabilities, such as object instance recognition. Also, as this task is sequential, it may require reinforcement rather than supervised learning.

- **Dynamic Tasks.** This thesis only considered quasi-static tasks in which inertial effects are negligible. An extension to dynamic grasping tasks such as throwing could be an interesting direction for future work.

9.3.3 Extensions to the Learning Method

Purely supervised learning may not work for all tasks such as sequential problems for which the choice of actions at the current timestep have a significant affect on the the reward received later, or problems for which the action space is large and must be explored. The following extensions may be fruitful:

- **Reinforcement Learning.** Any method from the diverse array of reinforcement learning techniques such as policy gradients [151], deep Q learning [118], or meta-learning [42] could be used to improve performance by learning directly in a sequential grasping environment.

- **Transfer Learning.** This thesis considered fine-tuning but a number of transfer learning techniques such as domain adaptation [170] or Generative Adversarial Networks (GANs) [12] could be considered. One possibility is to learn a generative model of sensor noise to augment synthetic data.

- **Optimizing the Generative Model.** The dataset generation distributions considered in this thesis all depend on a set of parameters. These parameters were "trained" to maximize the likelihood of images, grasps, and rewards collected from a physical robot under the synthetic model. A principled optimization approach to learning these parameters could lead to better performance. Furthermore, active data collection to jointly optimize the generative model and performance of the learned policy could reduce the number of examples necessary to learn a robust policy.
- **Continuous Learning.** A final possibility is to continuously improve the performance of the policy based on the results of grasp attempts on a physical system. A number of questions arise. Should data be collected on policy, which maximizes task performance but discourages exploration, or off policy, which will execute potentially failing and dangerous grasps on occasion? How can we aggregate data from multiple robots with different sensors, calibrations, objects, or even tasks? While it may be possible to train one generic model, performance on each individual robot could be maximized by tailoring training for that particular robot. Furthermore, how can we ensure data privacy and integrity as massive amounts of data are collected from real robots deployed in industrial applications?

9.4 The Broader Picture

Given the limited availability of hardware and lack of theory proving the correctness of grasp planning algorithms, this thesis does not claim to validate the hybrid method beyond the robots, grippers, objects, and grasping tasks considered in experiments. In fact, the only way to evaluate this method on a new grasp planning problem is to build a dataset generation distribution for that problem, which requires significant time, domain expertise, and attention to detail. Implementing a similar method for a new manipulation task could require years of research, and this may seem unnecessary in the midst of the deep learning wave of artificial intelligence, where significant amounts of research are predicated on the belief that robots can learn arbitrary new tasks by random exploration inspired by toddlers.

After the thousands and thousands of experiments in this thesis on the seemingly innocuous task of lifting novel objects, such a purely empirical approach does not seem reasonable for reliable operation in real-world tasks. Data collection on physical robots is difficult – so difficult, in fact, that it takes a tremendous amount of engineering effort to even benchmark grasp planning methods across a few hundred objects in academic research labs today. Often grasping research degenerates to evaluation on a small dataset of hand-picked examples. If simply evaluating a single policy on a physical robot is so difficult, how can we expect to train policies on physical data to generalize across robots, grippers, sensors, and millions of unique objects?

A general approach based on statistical machine learning can only take one approach: to build a scalable system for collecting, maintaining, and analyzing data collected from a huge number of robots deployed in real applications. If no such system exists, then it will not be possible to adequately benchmark methods to know whether or not they work in the first place. Why not study models for benchmarking robot learning, which can be used to simulate massive numbers of trials, to benchmark the performance of various learning methods, and to understand the limitations of methods?

This thesis tells two lessons that may be more generally applicable to robot learning. First, assumptions informed by real world applications can be used to design synthetic environments based on stochastics, geometry, and physics for learning policies that transfer to a physical robot without any real data. Second, diverse and clean training datasets are not a given in robotics, unlike the fields of computer vision, speech, and reinforcement learning. In robotics, we must establish data collection protocols for a given robot, sensor, and task before we can even begin to think about using machine learning. The problem formulation considered in this thesis suggests that protocols for data collection on a physical robot are indistinguishable from models for dataset generation from the perspective of a robot grasping policy. This suggests that by designing a benchmark we are, in some sense, designing a model for a task. After all, most machine learning results in robotics to date involve careful choices of tasks and reward signals that improve learning efficiency and reliability, such as monochrome objects, planar manipulation without occlusions, and goal poses derived from kinematics.

Learning algorithms are not the only important factor for training robot policies. The results of this thesis suggest that the distribution of data used to train a policy is also a fundamental component of robot learning systems. If nothing else, we hope that this inspires future research on the design and optimization of dataset collection distributions for robot learning.

Appendix A

Gaussian Process Implicit Surfaces

In this appendix we review Gaussian process implicit surfaces (GPISs). A signed distance function (SDF) [122] describes the shape of an object by storing the signed distance from every point in space to the nearest point on the surface. SDFs are defined as a real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(\mathbf{x}) > 0$ outside the object, $f(\mathbf{x}) = 0$ on the object surface, and $f(\mathbf{x}) < 0$ inside the object. A GPIS is a Gaussian distribution over SDFs formed by Gaussian process regression (GPR) on noisy observations of an SDF [179]. In this work we will use $d = 2$ and restrict evaluations of the SDF f to an $M \times M$ 2-dimensional grid with square cells [9, 35]. In practice M might be set based on the resolution of the sensor used to acquire measurements [122].

A.0.1 Gaussian Process Regression (GPR)

Gaussian process regression (GPR) is used in machine learning as a nonparametric regression method for estimating continuous functions from sparse and noisy data [138]. For a GPIS, a training set consists of a set of input spatial locations $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$, and signed distance observations $\mathbf{y} = \{y_1, \ldots, y_n\}, y_i \in \mathbb{R}$. In practice, $\mathbf{y}$ can be acquired using KinectFusion, which uses ray tracing to compute an SDF from RGBD point clouds [122], or by segmenting an object from the environment and performing a Euclidean distance transformation [45, 177].

A GPIS is specified by a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$, also referred to as a *kernel*, which measures the similarity in signed distance between spatial locations. Given a set of training data $\mathcal{D} = \{\mathcal{X}, \mathbf{y}\}$, mean $m(\cdot)$, kernel $k(\cdot, \cdot)$, and measurement noise σ_m, the posterior distribution on SDF f_* at a test location $\mathbf{x}_*$ is [138]:

$$\begin{aligned} ap(f_* \mid \mathbf{x}_*, \mathcal{D}) &\sim \mathcal{N}\Big(\mu(\mathbf{x}_*), \sigma^2(\mathbf{x}_*)\Big) \\ \mu(\mathbf{x}_*) &= m(\mathbf{x}_*)+ \\ &\quad k(\mathcal{X}, \mathbf{x}_*)^\intercal (K + \sigma_m^2 I)^{-1}(\mathbf{y} - m(\mathcal{X})) \qquad \text{(A.0.1)} \\ \sigma^2(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_*)- \\ &\quad k(\mathcal{X}, \mathbf{x}_*)^\intercal (K + \sigma_m^2 I)^{-1} k(\mathcal{X}, \mathbf{x}_*) \qquad \text{(A.0.2)} \end{aligned}$$

where $K \in \mathbb{R}^{n \times n}$ is a matrix with entries $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. $k(\mathcal{X}, \mathbf{x}_*) = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)]^\intercal$, and $m(\mathcal{X}) = [m(\mathbf{x}_1), ..., m(\mathbf{x}_n)]^\intercal$. This derivation can also be used to predict the mean and variance of the SDF gradient by differentiating the kernel function, which can be used to obtain GPIS surface normals [35, 138, 158].

Following Dragiev et al. [35], we use the squared exponential kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = C \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\ell^2}\right)$$

which specifies the correlation of the signed distance between two spatial points. This kernel depends on a scale $C \in \mathbb{R}$ and bandwidth $\ell \in \mathbb{R}$, which we set using maximum-likelihood estimation [179]. Other common kernels relevant to GPIS are the thin-plate splines kernel [179] and the Matern kernel [9].

A.0.2 GPIS Construction from Point Clouds

We review the construction of 2D GPIS models from RGBD point clouds from a single viewpoint of a Primesense Carmine sensor [61]. The scenario we consider consists of objects lying flat on a table imaged from above. Our method assumes that the object can be identified in the point clouds and segmented from the background. Furthermore, we assume that missing measurements in the point cloud are caused by surface properties of the object instead of the environment, because in our scenario the table is measured accurately by the Primesense. Thus, we use a constant negative mean function for the GPIS to bias areas of missing measurements to be part of the object.

To construct a GPIS from point clouds, we first combine several point clouds by averaging to remove the effects of small zero-mean noise in the depth values, similar to KinectFusion [122]. Then we create a segmentation mask for the object in both the RGB and depth point clouds. For the objects in Fig. **??** this segmentation is performed by hand, and in our physical experiments we use RGB and depth thresholding. We also create a measurement noise map, which specifies the variance of 0-mean measurement noise, based on a noise model of the Primesense [61]. We then combine the two segmentation masks in an image with each pixel weighted by its inverse variance to form an occupancy grid, and compute an SDF using a Euclidean distance transformation of the occupancy map [61, 177]. Finally, we run GPR on the SDF values and measurement noise map to construct a GPIS. Points with missing measurements (e.g. NaN values in the depth map) are considered part of the object in the occupancy map but are not used for the GPIS construction.

This construction procedure results in high uncertainty in (a) areas where the RGB and depth segmentations disagree (e.g. Object A) and (b) areas with missing measurements (e.g. Object B). GPIS could also be constructed directly from the SDF and confidence weights used in KinectFusion [122].

Bibliography

[1] Martín Abadi et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Brenna D Argall et al. “A survey of robot learning from demonstration”. In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[3] Jean-Yves Audibert and Sébastien Bubeck. “Best arm identification in multi-armed bandits”. In: *COLT-23th Conference on Learning Theory-2010*. 2010, 13–p.

[4] B_ Bahr, Y Li, and M Najafi. “Design and suction cup analysis of a wall climbing robot”. In: *Computers & electrical engineering* 22.3 (1996), pp. 193–209.

[5] Ravi Balasubramanian et al. “Physical human interactive guidance: Identifying grasping principles from human-planned grasps”. In: *Robotics, IEEE Transactions on* 28.4 (2012), pp. 899–910.

[6] Timothy D Barfoot and Paul T Furgale. “Associating uncertainty with three-dimensional poses for use in estimation problems”. In: *IEEE Trans. Robotics* 30.3 (2014), pp. 679–693.

[7] Christopher Batty. *SDFGen*. `https://github.com/christopherbatty/SDFGen`.

[8] Ulrich Bauer, Michael Kerber, and Jan Reininghaus. *PHAT - Persistent Homology Algorithm Toolbox*. 2013. URL: `https://code.google.com/p/phat/`.

[9] Marten Bjorkman et al. “Enhancing visual perception of shape through tactile glances”. In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 3180–3186.

[10] Jeannette Bohg and Danica Kragic. “Learning grasping points with shape context”. In: *Robotics and Autonomous Systems* 58.4 (2010), pp. 362–377.

[11] Jeannette Bohg et al. “Data-driven grasp synthesis: A survey”. In: *IEEE Trans. Robotics* 30.2 (2014), pp. 289–309.

[12] Konstantinos Bousmalis et al. “Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping”. In: *arXiv preprint arXiv:1709.07857* (2017).

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. “Collaborative grasp planning with multiple object representations”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2011, pp. 2851–2858.

[15] Eric Brown et al. “Universal robotic gripper based on the jamming of granular material”. In: *Proceedings of the National Academy of Sciences* 107.44 (2010), pp. 18809–18814.

[16] Berk Calli et al. “Benchmarking in Manipulation Research: The YCB Object and Model Set and Benchmarking Protocols”. In: *arXiv preprint arXiv:1502.03143* (2015).

[17] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. “Acting optimally in partially observable stochastic domains”. In: *AAAI*. Vol. 94. 1994, pp. 1023–1028.

[18] Nikhil Chavan-Dafle and Alberto Rodriguez. “Sampling-based planning of in-hand manipulation with external pushes”. In: *arXiv preprint arXiv:1707.00318* (2017).

[19] Chao Chen and Michael Kerber. “Persistent homology computation with a twist”. In: *Proceedings 27th European Workshop on Computational Geometry*. Vol. 11. 2011.

[20] I-Ming Chen and Joel W Burdick. “Finding antipodal point grasps on irregularly shaped objects”. In: *IEEE Trans. Robotics and Automation* 9.4 (1993), pp. 507–512.

[21] Sanjiban Choudhury et al. “Adaptive Information Gathering via Imitation Learning”. In: *Proc. Robotics: Science and Systems (RSS)*. 2017.

[22] Brian Chu et al. “Best practices for fine-tuning visual classifiers to new domains”. In: *Computer Vision–ECCV 2016 Workshops*. Springer. 2016, pp. 435–442.

[23] Matei Ciocarlie et al. “The Velo gripper: A versatile single-actuator design for enveloping, parallel and fingertip grasps”. In: *Int. Journal of Robotics Research (IJRR)* 33.5 (2014), pp. 753–767.

[24] Matei Ciocarlie et al. “Towards reliable grasping and manipulation in household environments”. In: *Experimental Robotics*. Springer. 2014, pp. 241–252.

[25] Nikolaus Correll et al. “Analysis and observations from the first amazon picking challenge”. In: (2016).

[26] Erwin Coumans et al. “Bullet physics library”. In: *Open source: bulletphysics. org* 15.49 (2013), p. 5.

[27] Mark R Cutkosky and Paul K Wright. “Friction, stability and the design of robotic fingers”. In: *Int. Journal of Robotics Research (IJRR)* 5.4 (1986), pp. 20–37.

[28] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE. 2005, pp. 886–893.

[29] 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.

[30] Raphael Deimel and Oliver Brock. "A novel type of compliant and underactuated robotic hand for dexterous grasping". In: *Int. Journal of Robotics Research (IJRR)* 35.1-3 (2016), pp. 161–185.

[31] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 248–255.

[32] Renaud Detry et al. "Learning a dictionary of prototypical grasp-predicting parts from grasping experience". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 601–608.

[33] Mehmet Dogar and Siddhartha Srinivasa. "A framework for push-grasping in clutter". In: *Robotics: Science and systems VII* 1 (2011).

[34] 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)*. IEEE. 2014, pp. 1997–2004.

[35] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. "Gaussian process implicit surfaces for shape estimation and grasping". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2011, pp. 2845–2850.

[36] H. Edelsbrunner and J. Harer. "Persistent homology-a survey". In: *Contemporary mathematics* 453 (2008), pp. 257–282.

[37] Herbert Edelsbrunner. *Weighted alpha shapes*. University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

[38] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[39] Herbert Edelsbrunner and Dmitriy Morozov. *Persistent homology: theory and practice*. Tech. rep. Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2012.

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

[41] C. Ferrari and J. Canny. "Planning optimal grasps". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1992, pp. 2290–2295.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *arXiv preprint arXiv:1703.03400* (2017).

[43] Daniel Fiser. *libccd - Collision Detection Between Convex Shapes*. `http://libccd.danfis.cz/`.

[44] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[45] Yasutaka Furukawa and Jean Ponce. "Carved visual hulls for image-based modeling". In: *Europan Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 564–577.

[46] Douglas C Giancoli. *Physics: principles with applications*. Pearson Education, 2005.

[47] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[48] Robby Goetschalckx, Pascal Poupart, and Jesse Hoey. "Continuous correlated beta processes". In: *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*. Vol. 22. 1. Citeseer. 2011, p. 1269.

[49] Ken Goldberg et al. "Part pose statistics: Estimators and experiments". In: *IEEE Trans. Robotics and Automation* 15.5 (1999), pp. 849–857.

[50] Corey Goldfeder and Peter K Allen. "Data-driven grasping". In: *Autonomous Robots* 31.1 (2011), pp. 1–20.

[51] Corey Goldfeder et al. "The Columbia grasp database". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 1710–1716.

[52] Haiyun Guo et al. "Multi-view 3d object retrieval with deep embedding network". In: *IEEE Transactions on Image Processing* 25.12 (2016), pp. 5526–5537.

[53] Menglong Guo et al. "Design of parallel-jaw gripper tip surfaces for robust grasping". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2831–2838.

[54] Saurabh Gupta et al. "Aligning 3D models to RGB-D images of cluttered scenes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4731–4740.

[55] Martin Hägele et al. "Industrial robotics". In: *Springer handbook of robotics*. Springer, 2016, pp. 1385–1422.

[56] Awni Hannun et al. "DeepSpeech: Scaling up end-to-end speech recognition". In: *arXiv preprint arXiv:1412.5567* (2014).

[57] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[58] Elliot W Hawkes, Hao Jiang, and Mark R Cutkosky. "Three-dimensional dynamic surface grasping with dry adhesion". In: *Int. Journal of Robotics Research (IJRR)* 35.8 (2016), pp. 943–958.

[59] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[60] Carlos Hernandez et al. "Team Delft's Robot Winner of the Amazon Picking Challenge 2016". In: *arXiv preprint arXiv:1610.05514* (2016).

[61] C Herrera, Juho Kannala, Janne Heikkilä, et al. "Joint depth and color camera calibration with distortion correction". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 34.10 (2012), pp. 2058–2064.

[62] Alexander Herzog et al. "Learning of grasp selection based on shape-templates". In: *Autonomous Robots* 36.1-2 (2014), pp. 51–65.

[63] Stefan Hinterstoisser et al. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. IEEE. 2011, pp. 858–865.

[64] Matthew W Hoffman, Bobak Shahriari, and Nando de Freitas. "Exploiting correlation and budget constraints in Bayesian multi-armed bandit optimization". In: *arXiv preprint arXiv:1303.6746* (2013).

[65] Francois R Hogan et al. "Tactile Regrasp: Grasp Adjustments via Simulated Tactile Transformations". In: *arXiv preprint arXiv:1803.01940* (2018).

[66] Katsushi Ikeuchi et al. *Picking up an Object from a Pile of Objects.* Tech. rep. DTIC Document, 1983.

[67] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Proc. Advances in Neural Information Processing Systems.* 2015, pp. 2017–2025.

[68] Maciej Jaśkowski et al. "Improved GQ-CNN: Deep Learning Model for Planning Robust Grasps". In: *arXiv preprint arXiv:1802.05992* (2018).

[69] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014).

[70] 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).* IEEE. 2016, pp. 4461–4468.

[71] Micah K Johnson et al. "Microgeometry capture using an elastomeric sensor". In: *ACM Transactions on Graphics (TOG).* Vol. 30. 4. ACM. 2011, p. 46.

[72] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. "Planning and acting in partially observable stochastic domains". In: *Artificial intelligence* 101.1-2 (1998), pp. 99–134.

[73] Sham Machandranath Kakade et al. "On the sample complexity of reinforcement learning". PhD thesis. University of London London, England, 2003.

[74] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *arXiv preprint arXiv:1806.10293* (2018).

[75] 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.

[76] Imin Kao, Kevin Lynch, and Joel W Burdick. "Contact modeling and manipulation". In: *Springer Handbook of Robotics*. Springer, 2008, pp. 647–669.

[77] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. "Leveraging big data for grasp planning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2015.

[78] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. "The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics". In: *The International Journal of Robotics Research* 31.8 (2012), pp. 927–934.

[79] 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)*. IEEE. 2012, pp. 576–583.

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

[81] Ben Kehoe et al. "Cloud-based robot grasping with the google object recognition engine". In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE. 2013, pp. 4263–4270.

[82] Junggon Kim et al. "Physically based grasp quality evaluation under pose uncertainty". In: *Robotics, IEEE Transactions on* 29.6 (2013), pp. 1424–1439.

[83] Junggon Kim et al. "Physically based grasp quality evaluation under pose uncertainty". In: *IEEE Trans. Robotics* 29.6 (2013), pp. 1424–1439.

[84] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* (2017), p. 201611835.

[85] Edouard Klein et al. "Inverse reinforcement learning through structured classification". In: *Proc. Advances in Neural Information Processing Systems*. 2012, pp. 1007–1015.

[86] Ramesh Kolluru, Kimon P Valavanis, and Timothy M Hebert. "Modeling, analysis, and performance evaluation of a robotic gripper system for limp material handling". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.3 (1998), pp. 480–486.

[87] Sanjay Krishnan et al. "ActiveClean: interactive data cleaning for statistical modeling". In: *Proceedings of the VLDB Endowment* 9.12 (2016), pp. 948–959.

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[89] OB Kroemer et al. "Combining active learning and reactive control for robot grasping". In: *Robotics and Autonomous Systems* 58.9 (2010), pp. 1105–1116.

[90] 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)*. IEEE. 2017, pp. 1595–1600.

[91] Heinrich Kruger et al. "Partial closure grasps: Metrics and computation". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2011, pp. 5024–5030.

[92] 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)*. IEEE. 2017.

[93] Michael Laskey et al. "DART: Noise Injection for Robust Imitation Learning". In: *Conference on Robot Learning*. 2017.

[94] Michael Laskey et al. "Multi-Armed Bandit Models for 2D Grasp Planning with Uncertainty." In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. IEEE. 2015.

[95] 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)*. IEEE. 2016.

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

[97] Beatriz León et al. "Opengrasp: a toolkit for robot grasping simulation". In: *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 109–120.

[98] 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.

[99] Bo Li et al. "A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries". In: *Computer Vision and Image Understanding* 131 (2015), pp. 1–27.

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

[101] Qingkai Lu et al. "Planning Multi-Fingered Grasps as Probabilistic Inference in a Learned Deep Network". In: *arXiv preprint arXiv:1804.03289* (2018).

[102] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

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

[104] 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)*. IEEE. 2016, pp. 1957–1964.

[105] 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.

[106] 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.

[107] Jeffrey Mahler et al. “Energy-Bounded Caging: Formal Definition and 2D Lower Bound Algorithm Based on Weighted Alpha Shapes”. In: *IEEE Robotics & Automation Letters*. IEEE. 2016.

[108] Jeffrey Mahler et al. “Gp-gpis-opt: Grasp planning under shape uncertainty using gaussian process implicit surfaces and sequential convex programming”. In: (2015).

[109] Jeffrey Mahler et al. “Privacy-preserving Grasp Planning in the Cloud”. In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. IEEE. 2016, pp. 468–475.

[110] Jeffrey Mahler et al. “Synthesis of energy-bounded planar caging grasps using persistent homology”. In: (2018).

[111] Tanwi Mallick, Partha Pratim Das, and Arun Kumar Majumdar. “Characterizations of noise in Kinect depth images: A review”. In: *IEEE Sensors Journal* 14.6 (2014), pp. 1731–1740.

[112] Giacomo Mantriota. “Theoretical model of the grasp with vacuum gripper”. In: *Mechanism and machine theory* 42.1 (2007), pp. 2–17.

[113] Matthew T Mason. “Mechanics and planning of manipulator pushing operations”. In: *The International Journal of Robotics Research* 5.3 (1986), pp. 53–71.

[114] Matthew T. Mason. *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001. ISBN: 0-262-13396-2.

[115] Zoe McCarthy, Timothy Bretl, and Seth Hutchinson. “Proving path non-existence using sampling and alpha shapes”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2563–2569.

[116] Andrew T Miller and Peter K Allen. “Graspit! a versatile simulator for robotic grasping”. In: *Robotics & Automation Magazine, IEEE* 11.4 (2004), pp. 110–122.

[117] John W Miller, Rod Goodman, and Padhraic Smyth. “On loss functions which minimize to conditional expected values and posterior probabilities”. In: *IEEE Transactions on Information Theory* 39.4 (1993), pp. 1404–1408.

[118] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), p. 529.

[119] Luis Montesano and Manuel Lopes. “Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions”. In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 452–462.

[120] D Morrison et al. “Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge”. In: *arXiv preprint arXiv:1709.06283* (2017).

[121] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation.* CRC press, 1994.

[122] Richard A Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *IEEE Int. Symposium on Mixed and augmented reality (ISMAR).* IEEE. 2011, pp. 127–136.

[123] Van-Duc Nguyen. “Constructing stable grasps”. In: *Int. Journal of Robotics Research (IJRR)* 8.1 (1989), pp. 26–37.

[124] John Oberlin and Stefanie Tellex. “Autonomously Acquiring Instance-Based Object Models from Experience”. In: *Int. S. Robotics Research (ISRR).* 2015.

[125] John Oberlin et al. “Acquiring Object Experiences at Scale”. In: ().

[126] Lael U Odhner et al. “A compliant, underactuated hand for robust manipulation”. In: *Int. Journal of Robotics Research (IJRR)* 33.5 (2014), pp. 736–752.

[127] OpenAI. “Learning Dextrous In-Hand Manipulation”. In: *arXiv preprint arXiv:1808.00177* (2018).

[128] Christos H Papadimitriou and John N Tsitsiklis. “The complexity of Markov decision processes”. In: *Mathematics of operations research* 12.3 (1987), pp. 441–450.

[129] 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.

[130] Michael A Peshkin and Arthur C Sanderson. “The motion of a pushed, sliding workpiece”. In: *IEEE Journal on Robotics and Automation* 4.6 (1988), pp. 569–598.

[131] 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.

[132] Robert Platt Jr et al. “Belief space planning assuming maximum likelihood observations”. In: (2010).

[133] Florian T Pokorny, Kaiyu Hang, and Danica Kragic. “Grasp Moduli Spaces”. In: *Robotics: Science and Systems.* 2013.

[134] Florian T Pokorny and Danica Kragic. “Classical grasp quality evaluation: New algorithms and theory”. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on.* IEEE. 2013, pp. 3493–3500.

[135] Florian T. Pokorny and Danica Kragic. “Classical Grasp Quality Evaluation: New Theory and Algorithms”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[136] Domenico Prattichizzo and Jeffrey C Trinkle. “Grasping”. In: *Springer handbook of robotics*. Springer, 2008, pp. 671–700.

[137] Xavier Provot et al. “Deformation constraints in a mass-spring model to describe rigid cloth behaviour”. In: *Graphics interface*. Canadian Information Processing Society. 1995, pp. 147–147.

[138] Carl Edward Rasmussen. “Gaussian processes for machine learning”. In: (2006).

[139] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. “Imitation learning for locomotion and manipulation”. In: *Int’l Conf. on Humanoid Robots*. IEEE. 2007, pp. 392–397.

[140] Joseph Redmon and Anelia Angelova. “Real-time grasp detection using convolutional neural networks”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1316–1322.

[141] Colin Rennie et al. “A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place”. In: *IEEE Robotics & Automation Letters* 1.2 (2016), pp. 1179–1185.

[142] Elon Rimon and Andrew Blake. “Caging 2D bodies by 1-parameter two-fingered gripping systems”. In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. Vol. 2. IEEE. 1996, pp. 1458–1464.

[143] Elon Rimon and Joel W. Burdick. “Mobility of bodies in contact. I. A 2nd-order mobility index for multiple-finger grasps”. In: vol. 14. 5. 1998, pp. 696–708.

[144] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. “From caging to grasping”. In: *Int. Journal of Robotics Research (IJRR)* (2012), p. 0278364912442972.

[145] Reuven Y Rubinstein, Ad Ridder, and Radislav Vaisman. *Fast sequential Monte Carlo methods for counting and optimization*. John Wiley & Sons, 2013.

[146] Daniela Rus and Michael T Tolley. “Design, fabrication and control of soft robots”. In: *Nature* 521.7553 (2015), p. 467.

[147] Fereshteh Sadeghi and Sergey Levine. “CAD2RL: Real single-image flight without a single real image”. In: *Proc. Robotics: Science and Systems (RSS)*. 2017.

[148] Marcos Salganicoff, Lyle H Ungar, and Ruzena Bajcsy. “Active learning for vision-based robot grasping”. In: *Machine Learning* 23.2-3 (1996), pp. 251–278.

[149] Samuele Salti, Federico Tombari, and Luigi Di Stefano. “SHOT: Unique signatures of histograms for surface and texture description”. In: *Computer Vision and Image Understanding* 125 (2014), pp. 251–264.

[150] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. “Robotic grasping of novel objects using vision”. In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173.

[151] John Schulman et al. “Trust region policy optimization”. In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[152] Daniel Seita et al. “Large-Scale Supervised Learning of the Grasp Robustness of Surface Patch Pairs”. In: *Proc. IEEE Int. Conf. on Simulation, Modeling, and Programming of Autonomous Robots (SIMPAR)*. IEEE. 2016.

[153] Guy Shani, Joelle Pineau, and Robert Kaplow. “A survey of point-based POMDP solvers”. In: *Autonomous Agents and Multi-Agent Systems* (2013), pp. 1–51.

[154] Karun B Shimoga. “Robot grasp synthesis algorithms: A survey”. In: *The International Journal of Robotics Research* 15.3 (1996), pp. 230–266.

[155] Ashutosh Singh et al. “Bigbird: A large-scale 3d database of object instances”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2014.

[156] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. “Learning without state-estimation in partially observable Markovian decision processes”. In: *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 284–292.

[157] Gordon Smith et al. “Computing parallel-jaw grips”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1999.

[158] Ercan Solak et al. “Derivative observations in Gaussian process models of dynamic systems”. In: (2003).

[159] Adam J Spiers et al. “Single-grasp object classification and feature extraction with simple robot hands and tactile sensors”. In: *IEEE Transactions on Haptics* 9.2 (2016), pp. 207–220.

[160] Niranjan Srinivas et al. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *Proc. International Conference on Machine Learning (ICML)*. 2010.

[161] 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)*. IEEE. 2015, pp. 2279–2284.

[162] Hang Su et al. “Multi-view Convolutional Neural Networks for 3D Shape Recognition”. In: *arXiv preprint arXiv:1505.00880* (2015).

[163] Jianhua Su et al. “Vision-based caging grasps of polyhedron-like workpieces with a binary industrial gripper”. In: *IEEE Transactions on Automation Science and Engineering* 12.3 (2015), pp. 1033–1046.

[164] Raúl Suárez, Jordi Cornella, and Máximo Roa Garzón. *Grasp quality measures*. Institut d’Organització i Control de Sistemes Industrials Barcelona, Spain, 2006.

[165] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[166] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[167] The CGAL Project. *CGAL User and Reference Manual*. 4.6.1. CGAL Editorial Board, 2015.

[168] Sebastian Thrun. "Monte carlo pomdps". In: *Proc. Advances in Neural Information Processing Systems*. 2000, pp. 1064–1070.

[169] 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)*. IEEE. 2017, pp. 23–30.

[170] Eric Tzeng et al. "Adversarial discriminative domain adaptation". In: *Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2. 2017, p. 4.

[171] Mostafa Vahedi and A Frank van der Stappen. "Caging polygons with two and three fingers". In: *Int. Journal of Robotics Research (IJRR)* 27.11-12 (2008), pp. 1308–1324.

[172] Angel J Valencia et al. "A 3D vision based approach for optimal grasp of vacuum grippers". In: *Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2017 IEEE International Workshop of*. IEEE. 2017, pp. 1–6.

[173] Gino Van Den Bergen. "Proximity queries and penetration depth computation on 3d game objects". In: *Game developers conference*. Vol. 170. 2001.

[174] Laurens Van Der Maaten. "Accelerating t-SNE using tree-based algorithms." In: *Journal of machine learning research* 15.1 (2014), pp. 3221–3245.

[175] Jacob Varley et al. "Generating multi-fingered robotic grasps via deep learning". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 4415–4420.

[176] Ulrich Viereck et al. "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2017.

[177] Jun Wang and Ying Tan. "Efficient Euclidean distance transform algorithm of binary images in arbitrary dimensions". In: *Pattern Recognition* 46.1 (2013), pp. 230–242.

[178] Jonathan Weisz and Peter K Allen. "Pose error robust grasping from contact wrench space metrics". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 557–562.

[179] Oliver Williams and Andrew Fitzgibbon. "Gaussian process implicit surfaces". In: *Gaussian Proc. in Practice* (2007).

[180] Walter Wohlkinger et al. “3dnet: Large-scale object class recognition from cad models”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012.

[181] Zhirong Wu et al. “3D ShapeNets: A Deep Representation for Volumetric Shape Modeling”. In: *CVPR*. Vol. 1. 2. 2015, p. 3.

[182] Yu Yoshida and Shugen Ma. “Design of a wall-climbing robot with passive suction cups”. In: *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE. 2010, pp. 1513–1518.

[183] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[184] 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).

[185] 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.

[186] 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)*. IEEE. 2017, pp. 1386–1383.

[187] Andy Zeng et al. “Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching”. In: (2018).

[188] Liangjun Zhang, Young J Kim, and Dinesh Manocha. “Efficient cell labelling and path non-existence computation using C-obstacle query”. In: *The International Journal of Robotics Research* 27.11-12 (2008), pp. 1246–1257.

[189] Liangjun Zhang et al. “Generalized penetration depth computation”. In: *Computer-Aided Design* 39.8 (2007), pp. 625–638.

[190] Yu Zheng and Wen-Han Qian. “Coping with the grasping uncertainties in force-closure analysis”. In: *Int. Journal of Robotics Research (IJRR)* 24.4 (2005), pp. 311–327.

[191] Yuke Zhu et al. “Target-driven visual navigation in indoor scenes using deep reinforcement learning”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2017, pp. 3357–3364.