Data-Driven Robotic Grasping in the Wild

Adithyavairavan Murali

CMU-RI-TR-20-49

September 21, 2020

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

Thesis Committee:
Professor Abhinav Gupta, *Carnegie Mellon University* (chair)
Professor Oliver Kroemer, *Carnegie Mellon University*
Professor David Held, *Carnegie Mellon University*
Professor Dieter Fox, *University of Washington*
Professor Sonia Chernova, *Georgia Institute of Technology*

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

Copyright © 2020 Adithyavairavan Murali. All rights reserved.

To my parents, Bharathy and Murali.

Abstract

Robotic grasping has seen tremendous advancements in recent years. Yet, the current paradigm of manipulation research is typically some form of table-top manipulation in constrained setups or in simulation. Building general purpose personal robots that can autonomously grasp unknown objects in unstructured environments like homes is an open problem. In this thesis, we explore important directions in scaling data-driven grasping to the diversity and constraints imposed by the real world.

We first discuss how we can go beyond picking individual objects in isolation to 6-DOF grasping in clutter. Most existing methods train policies on datasets collected in curated settings (in lab or simulation) and hence may not cope with the mismatch in data distribution when deployed in the wild. We build and open-source a low-cost mobile manipulator platform to parallelize data collection in challenging settings like homes and show that policies trained on this data generalize to novel objects in unseen homes. As a result, we also discuss ideas for scaling robot learning with several robots and transferring policies between different hardware. Yet, we hypothesize that visual perception alone is insufficient for robustness and present a self-supervised tactile-based re-grasping framework to close the loop on grasp execution. Lastly, we strive to go beyond robotic pick-and-place and generalize to diverse semantic manipulation tasks. We do so by scaling task-oriented grasping datasets with crowdsourcing and learning from semantic information like knowledge graphs.

Acknowledgments

I would first like to thank my advisor Abhinav Gupta for the incredible couple of years in grad school. He has provided invaluable advice, ideas, encouragement and set a high bar for research. I'm always inspired by this wisdom, passion for embodied AI and fierce commitment for making robots work in the real world. He gave me the space to explore high-risk-high-reward ideas at the crossroads of robotics, machine learning, knowledge systems, computer vision and robotic hardware. He keeps inspiring me with his creativity, commitment to research and the effort he puts into nurturing his students.

I would also like to thank my committee and several faculty members who helped me in my research journey. I would like to thank Dieter Fox for the opportunity to intern at NVIDIA and his encouragement to focus on robotic grasping in my PhD. His work on multi-modal sensing (RGB-D, tactile) and philosophy on using structured knowledge in learning systems (without learning from scratch) inspired the approaches I took in my papers. I had the great opportunity to work with Oliver Kroemer on several robotics classes at CMU. Working on the first iteration of *Learning for Manipulation* taught me so many applications of machine learning in robotics. I'm also incredibly proud that the locobots introduced in this thesis were used in teaching *Robot Autonomy*. Thanks to Oliver for pushing me to be a better communicator and educator. David Held has been constantly supportive of my research from the start, serving on all my committees and giving valuable feedback. Sonia Chernova encouraged me to think beyond functional problems in robotics, but also in terms of semantics and using prior knowledge. I've also had the opportunity to work with her lab, namely Weiyu Liu who really taught me everything I know about semantic knowledge in robotics. I also received valuable advice from several other faculty members at CMU who shaped my thoughts over the years: Chris Atkeson, Martial Hebert, Tom Mitchell, Kris Kitani, Deva Ramanan, Sidd Srinivasa, Drew Bagnell, Max Likhachev, Ralph Hollis and Katharina Muelling.

I would like to thank the incredible colleagues and collaborators in my lab who had a major impact on my graduate experience: Kenneth Marino, Tao Chen, Lerrel Pinto, Dhiraj Gandhi, Yin Li, Judy (Yufei) Ye, Victoria Dean, Helen Jiang, Sudeep Dasari, Shikhar Bahl, Gunnar Sigurdsson, Sam Powers, Gaurav Pathak, Wenxuan Zhou, Senthil Purushulwalkam, Pratyusha Sharma, Nadine Chang, Nilesh Kulkarni, Xinlei Chen, Ishan Misra, Xiaolong Wang, Jacob Walker and Abhinav Shrivastava. Thanks for all the discussions on AI, life and everything in between. Thanks for being patient with me and hope you forgive me for all the mess I made in lab!

I was also extremely lucky to have collaborated with industry partners. My internship at NVIDIA Seattle Robotics Lab was a terrific experience. I am thankful to Arsalan Mousavian and Clemens Eppner for being excellent mentors, role models and teaching me so much about sim2real transfer. My discussions with Ankur Handa, Chris Paxton, Tucker Hermans, Nathan Ratliff, Yu Xiang, Jonathan Tremblay, Stan Burchfield, Xinke Deng, Keunhong Park, Chris Xie, Muhammad Asif Rana were very valuable and shaped my perspectives on perception. Thanks to Saurabh Gupta, Kalyan Vasudev and Soumith Chintala at Facebook AI Research for their mentorship and help in launching the PyRobot project. I would also like to thank Shubam Tulsiani, Jessica Hodgins, Stuart Anderson and Mustafa Mukadam for their collaboration and mentorship.

While it be challenging to list everyone, I would also like to thank all my friends and colleagues from the CMU Smith Hall and Robotics community: Dinesh Reddy, Leonid Keselman, Raaj Yadav, Aayush Bansal, Jason Zhang, Pragna Mannam, Achal Dave, Peiyun Hu, Alex Spitzer, Rogerio Bonatti, Brian Okorn, Ben Newman, Adam Harley, Wen Sun, Thomas Weng, Allie Del Giorno, Xingyu Liu, Shivam Vats, Samuel Clarke, Arpit Agarwal, Suren Jayasuriya. Thanks to Achal, Senthil, Dinesh and Ankit for the amazing mid-afternoon Halo tournaments (and graciously losing to me on a number of occasions). Everything I know about robot hardware comes from Roberto Shu, Dhiraj Gandhi and Ankit Bhatia. I'd also like to thank Jacky Liang, Tim Lee, Anirudh Vemula for countless discussions and helping me whenever I was in a tight spot (sometimes quite literally). Thanks also to Judy, Brian and Tim for motivating me to keep a active lifestyle and pick up running, spinning, groupx, amongst others.

My interest in robotics was kindled during my undergraduate years at UC Berkeley. Big thanks to my undergraduate advisors Ken Goldberg and Pieter Abbeel for nurturing my interests and encouraging me to continue with graduate studies in robotics. I'd also like to thank the graduate students I worked with during my time at Berkeley and from whom I think I got my healthy dose of academic scepticism: Animesh Garg, Michael Laskey, Jeff Mahler, Sanjay Krishnan, Sidd Sen, Ben Kehoe, Florian Pokorny, Sachin Patil. While I did not have the foresight of working on grasping back in Berkeley, the theses of Jeff Mahler and Ben Kehoe heavily influenced my PhD research.

I owe a huge thanks to Suzanne Muth, Christine Downey, David Wettergreen and Alison Day for their support through key milestones in my PhD program.

I would also like to thank several researchers from the international robotics community who have mentored me and whose research motivated my own: Tapo Bhattacharjee, Sergey Levine, Ronald Fearing, David Hsu, Peter Allen, Samarth Brahmbhatt, Berk Calli, Wenzhen Yuan, Hyosang Lee, amongst several others.

Thanks also goes to my mentors in high school who pushed me to focus on science research: Sathyan Subbiah, Nikolai Yakovlev, Rebecca Carrier, Jason Tan, Koh Siak Peng, Law Hock Ling and Otto Fong.

Last but not least, no words can express my gratitude to my parents, family and friends from Singapore, India and Berkeley. They knew nothing about robotics but their love and support kept me going through the ups and downs.

Thanks everyone!

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction & Background

> "Supposedly you really can't tell, except by looking at the hands. They haven't perfected the hands yet."
> — Response from Peter when John is unable to tell a robot from a human in the Delos amusement park, Westworld (1973). In this futuristic sci-fi Western film, roboticists have managed to build humanoids that resemble humans, except for the grippers!

Humans can effortlessly grasp a wide variety of objects in diverse environments. On the other hand, robotic grasping has been extremely challenging in practice and is far from matching human dexterity. This puzzling gap is nicely captured by Moravec's paradox, which states that low-level sensorimotor skills demand enormous computational resources when compared to high-level reasoning [172]. In the case of AI research, computers have surpassed humans at games like Go [219] which demand complex decision making, but a robot will still struggle to manipulate a stone on the gameboard. A controversial comment at many robotics conferences (often as ice breaker) is that "grasping is solved" [1]. If this was indeed the case, why bother researching or even writing a thesis on the topic? In reality, grasping is still an active area of research with over four decades of prior work, and the number of papers increase every year [76]. Despite recent progress in the community, most research is still largely focused on constrained setups: picking individual objects on a table top setting. This is in stark contrast to the overall ambition of roboticists in building a general-purpose personal robot that can manipulate objects in unstructured environments. This robot has enormous promise with applications in logistics, hospitals, retail, warehouses and assistive care for the disabled and elderly. This thesis aims to scale data-driven grasping to the diversity and

[1] This was famously said by former DARPA program manager Gill Pratt at IROS 2012. He has since clarified that position and does not regret saying that. Source: IEEE Spectrum [98]

constraints imposed by real world environments.

1.1 Grasping Preliminaries

Grasping is an empirical and multidisciplinary problem encompassing hardware, software, theory and algorithms. Grasp planning can be described by three stages: 1) grasp synthesis, the process of generating grasp candidates from a large (infinite) search space, 2) grasp analysis which is ranking the candidates based on some metric and 3) grasp execution to apply the predicted grasp with a motion planner and controller on real robot hardware [32, 33, 180]. There are several factors affecting grasp success, including object geometry, material, contacts, surface friction, mass distribution, amongst others. Visual perception is crucial in acquiring observations for these factors, especially the object geometry which is needed for grasp synthesis and analysis. There are many ways of parameterizing grasps, but a popular formulation is the pose of the gripper. Typically, the gripper has its fingers open when positioned at a planned grasp pose, which will result in a successful grasp when they close. This grasp can be executed by commanding the robot to that specific pose with the help of motion planning and a controller. Barring a few papers [120, 142], these methods typically rely on good camera calibration and proprioceptive feedback to execute grasps. A parallel line of work investigates new gripper hardware [30], compliance and exploiting stiffness in the environment for grasping [76]. These are complementary themes to this thesis.

1.2 Classical Grasping

Summarizing classical grasp analysis in a few paragraphs would not do justice to its rich history. We refer readers to Bicchi and Kumar [32], Miller [165], Murray et al. [180] for a detailed treatment of the subject. Instead, we will describe the key insights and shortcomings of this school of thought. Classical approaches provide analytic solutions with guarantees, and they are formulated as a constrained optimization problem given accurate models of the object geometry and hand kinematics [180]. They are also based on practical assumptions such as simplified contact models, rigid body modeling and Coulomb friction.

The first key drawback of these classical approaches is their lack of robustness to positional errors during contact. In practice, robotic systems have several systematic and random errors, due to inaccurate kinematic, dynamics, sensor and calibration models. These errors are accentuated with wear and tear from contact interactions. This precludes any notion of "perfect execution" of grasps to the all but simulations or well-calibrated factory robots working in curated environments. While some works have modelled compliance [76] and caging [205] to reduce the reliance on accurate positioning, they were mainly shown to work in simulation and/or planar grasping. Several papers have demonstrated that classical approaches do not necessarily transfer from simulation to the real world due to these assumptions [26, 95, 171]. The Columbia grasp database [95] contains grasps for several thousand objects and multiple hands. The best grasps (ranked by analytic metrics) failed to transfer to known and similar (but not identical) objects [95]. This was due to execution error from camera calibration and overreliance on precise geometry, which can be different even for similar looking objects. Balasubramanian et al. [26] found that robust grasps resulting from human kinesthetic demonstrations could not be explained by analytic metrics. Heuristics like the alignment between the hand with the object's principal axis better captured these stable grasps. While object pose estimation has evolved with deep learning [66, 238], they have historically been prone to errors [200], may not generalize to novel objects, and can be slow to accurately register noisy point cloud data to known shape models during execution [94].

The second shortcoming with classical grasping are the assumptions on object information. To acquire object shape in the wild, we need to visually segment the object from the scene and integrate noisy observations from multiple views to get a complete 3D representation (without self-occlusions, etc.) [32, 165, 180]. This is very challenging in general, even in table top settings with known objects, and sometimes impossible in unstructured settings with clutter. We also may not know other physical parameters like friction coefficients, weight, center of mass and weight distribution. Acquiring such information from vision is an open problem.

1.3 Data-driven Grasping

Data-driven grasping has shown tremendous potential for generalization. Compared to analytic approaches, data-driven approaches focus on learning a feature representation of objects and grasps. They typically sample grasp candidates and rank them using a learnt evaluator [33, 153]. State-of-the-art algorithms have shown generalization to object instances [142, 153, 190], viewpoints[142], DOF constraints [144, 173, 179, 231], unknown environments [99] and even adversarial objects [242]. We refer to Bohg et al. [33] for a more detailed survey of the literature pre-2016.

There have been several macro trends that have contributed to the success of these techniques. First, deep learning [136, 139] and domain randomization [236] have allowed grasping models to learn a mapping from raw visual observations to grasp poses, and transfer to unknown objects in novel poses. Second, the complementary development of cloud computing [152], physical simulators (GraspIt! [165], FleX[164], Bullet [56]), visual renderers [36, 173] and photorealistic simulation [238] have allowed training on large scale datasets. Last, the price of robotic hardware is also steadily decreasing [91, 99], further reducing the entry barrier to widely deploying grasping systems. This includes low-cost collaborative manipulators and commodity 3D RGBD sensors like the Kinect and Realsense [123]. As a result of these trends, one can also just execute grasps on a low-cost robot to collect interactive data for self-supervised learning or for extensive testing.

Data-driven grasping has limitations that accompany any application of machine learning. Datasets suffer from bias [99] and overfitting to the specific hardware (e.g. gripper, cameras), environment they are trained on (labs, simulators) and may not transfer to real data. For more complex applications such as task-oriented grasping, there is a significant bottleneck in data, since we are reliant on semantic labelling by humans [178]. State-of-the-art algorithms have reported grasp performances of 96% on unknown objects [120]. While this is an impressive feat, we still need to optimize for robustness (on the scale of 99.99% accuracy) and long-tail data to deploy in the real world. With papers reporting a saturation of performance with more visual grasp data [142, 190], it is unclear whether more data is the solution.

1.4 Grasping Applications in Practice

Grasping has typically been used in pick-and-place systems in factories, where the robot is programmed to grasp a known object in constrained poses and place it elsewhere. Recently, empirical competitions like the Amazon Picking Challenge have exposed the significant gap between classical theory and grasping in practice. In fact, the winning teams for several years deployed a combination of vacuum suction grippers (abandoning fingers entirely), deep learning and point cloud heuristics (grasping along the surface normal) [50, 72]. They outperformed teams using the classical grasping pipeline of 3D pose estimation of known objects followed by executing pre-computed grasps [22, 203]. It is also worth noting that the contributions introduced in this thesis transcend the fingers vs. suction debate and can be applied for both systems.

At the time of writing this thesis, the world is dealing with the fallout from the COVID19 global pandemic. This has greatly accelerated the need for more robotic automation in warehouses and other settings and has created a new market for medical and delivery robots [97]. Despite the progress in robotic picking, robots are still effectively in a "cage": a stationary table-top environment with a well-calibrated hardware setup. It is unclear if existing industrial art can be confidently applied in unconstrained settings like homes on a personal robot.

1.5 Thesis Goal and Contributions

While factory and warehouse automation are the most practical applications of robotic grasping, the wider ambition of the robotics community is in building general-purpose robots that can work in unstructured settings like homes and hospitals. I argue that both prior waves of classical and data-driven grasping are insufficient in building this next generation of manipulators. In such spirit, the goal of this thesis can be summarized as follows: To scale data-driven grasping to the diversity and constraints imposed by real environments, in terms of robustness to grasp execution, clutter, robots and tasks.

We approach this problem from several important directions. First, we need to go beyond reasoning about individual objects to grasping objects in clutter, where occlusion and collision avoidance are important considerations. Second, we typically use expensive well-calibrated hardware to perform a handful of grasps on a single robot in the lab. As promised by science fiction, we will have several models of personal robots (with their own collision model, kinematics, dynamics and textures) in the future. We show how to transfer policies between different robots, use low-cost robots to scale up data-collection and finally demonstrate how to improve grasp sampling efficiency on high-dimensional hardware systems. We also present open source hardware and software frameworks to democratize grasping research. Third, we hypothesize that visual perception alone is insufficient for robust grasping and present a data-driven method of closing the loop on the grasp execution process with tactile sensing.

Despite the enormous progress and generalization in robotic grasping in recent years, there is still a large gap between robot and human dexterity. When humans grasp an object, we do so with a particular purpose or task in mind and grasping is just the first step to that end. We grasp objects for prototypical uses, such as grasping a cup by the handle for drinking, and creative applications like scooping with a bowl. Motivated by this and as the final lap of the thesis, we strive to go beyond robotic pick-and-place by generalizing grasping for tasks. We do so by scaling task-oriented grasping datasets with crowdsourcing and improvising supervision with semantic knowledge graphs.

The primary contributions of this thesis are detailed as follows:

- We present a learning-based approach for 6-DOF grasp synthesis of novel objects in structured clutter. We additionally propose a learnt collision checker conditioned on the gripper information and on the raw occluded point cloud of the scene. Trained only with synthetic data, our framework achieves a grasp accuracy of around 80 % on the real robot with several cluttered scenes of unknown objects.
- We assemble low-cost mobile manipulators for scaling interactive dataset collection in unstructured environments and for benchmarking progress in grasping
- We also present the first systematic effort in collecting a robotic grasping dataset inside homes using these low-cost robots. We demonstrate how data collected from these diverse home environments leads to superior performance and requires little-to-no domain adaptation when testing on novel homes with unseen objects
- A framework for policy learning from noisy data collected from multiple low-cost robots which improves overall grasping performance
- We propose the Hardware Conditioned Policies (HCP) algorithm for the more general problem

of transfer learning policies between robots of completely different kinematics and dynamics

- We present the PyRobot software framework for writing hardware agnostic code to improve the sharing of datasets and algorithms for robotic manipulation. It is designed to be beginner-friendly to democratize grasping (along with the low-cost robots)
- A curriculum learning approach to improving the grasp sampling complexity for high-dimensional hardware systems like multi-fingered grippers
- A framework for grasping unknown objects in novel poses from tactile sensing alone. The object is localized using particle filtering with contact sensing. The grasps are iteratively improved in a closed loop manner using a learnt re-grasping policy. We also demonstrate that tactile re-grasping can be used to substantially improve robustness when applied on top of vision-based grasping.
- We propose to scale supervision for task-oriented grasping with crowdsourcing, presenting the TaskGrasp dataset which expands upon prior datasets by an order of magnitude. This allows us to study generalization in terms of objects and tasks.
- We also demonstrate that semantic knowledge in the form of knowledge graphs and semantic embeddings can be used for generalizing task-oriented grasping.

1.6 Thesis Organization

This thesis is organized into four main parts.

Part I Generalization to Clutter: In Chapter 2, we present a 6-DOF grasp generation approach in structured clutter. Grasping in cluttered environments is challenging since it requires both reasoning about unseen object parts and potential collisions with the manipulator. Our method plans 6-DOF grasps for any desired object in a cluttered scene from partial point cloud observations. Though only trained on simulated data, our approach achieves a grasp success of 80.3%, outperforming baseline approaches by 17.6% in clearing multiple cluttered table scenes of unknown objects when tested on a real robotic platform. In corner cases when the target object is initially not reachable, we reason about moving blocking objects out of the way to finally grasp the target object.

Part II Generalization with Robots: It is extremely challenging to demonstrate grasping on real robots but most real datasets are overfit to the specific hardware and environments they are collected in. We use low-cost robots to scale up data-collection, show how to transfer policies between different robots and finally on how we can improve sampling efficiency on high-dimensional hardware systems.

- In Chapter 3, we investigate the problem of scaling self-supervised grasping approaches with curriculum learning on control space. To overcome the curse of dimensionality, we would need to either scale up data collection efforts or use a clever sampling strategy for training. We present a novel approach - Curriculum Accelerated Self-Supervised Learning (CASSL) - which orders the sampling of training data based on control dimensions: the learning and sampling are focused on few control parameters before others. The right curriculum for learning is suggested by variance-based global sensitivity analysis of the control space. Our experimental results indicate that CASSL provides significant improvement compared to baseline methods such as staged curriculum learning (8% increase) and end-to-end learning (14% improvement).

- In Chapter 4, we present the first systematic effort in collecting large scale robot data inside diverse environments like peoples homes. We first build a low cost mobile manipulator assembled for under 3K USD to parallelize data collection. Second, data collected using low-cost robots suffer from noisy labels due to imperfect execution and calibration errors. To handle this, we develop a framework which factors out the noise as a latent variable. The models trained with our home dataset showed a marked improvement of 43.7% over a baseline model trained with data collected in lab when tested in unseen homes. Our architecture which explicitly models the latent noise in the dataset also performed 10% better than one that did not factor out the noise.
- In Chapter 5, we introduce PyRobot and LoCoBot in our efforts to democratize grasping and encourage benchmarking. LoCoBot is the improved version of the low cost mobile manipulator used in Chapter 4. PyRobot is a light-weight, high-level interface on top of ROS that provides a consistent set of hardware independent mid-level APIs to control different robots. PyRobot abstracts away details about low-level controllers and inter-process communication, and allows non-robotics researchers (ML, CV researchers) to focus on building high-level AI applications. PyRobot aims to provide a research ecosystem with convenient access to robotics datasets, algorithmic implementations and models that can be used to quickly create a state-of-the-art baseline. We believe PyRobot, when paired up with low-cost robot platforms, will reduce the entry barrier into robotics, and democratize robotics.

Part III Generalization with Robustness: We hypothesize that visual perception alone cannot guarantee robust grasp executions. In Chapter 6, we present a self-supervised tactile-based approach for closing the loop for grasping. Specifically, we study the challenging problem of grasping novel objects without prior knowledge of their location or physical properties. Our key idea is to combine touch based object localization with tactile based re-grasping. Our re-grasping model learns to progressively improve grasps with tactile feedback based on the learned features. This network learns to estimate grasp stability and predict adjustment for the next grasp. Re-grasping is thus performed iteratively until our model identifies a stable grasp without slippage. Finally, we demonstrate extensive experimental results on grasping a large set of novel objects using tactile sensing alone. Most importantly, we demonstrate robustness by showing that re-grasping significantly boosts the overall performance by 10.6% when applied on top of a vision-based policy.

Part IV Generalization to Semantic Tasks: How do we go beyond pick-and-place applications and use grasping for completing more complex tasks? Despite the enormous progress in robotic grasping in recent years, existing methods have yet to generalize task-oriented grasping to the same extent. This is largely due to the scale of the datasets both in terms of the number of objects and tasks studied. In Chapter 7, we address these concerns with the TaskGrasp dataset which is more diverse both in terms of objects and tasks, and an order of magnitude larger than previous datasets. The dataset contains $250K$ task-oriented grasps for 56 tasks and 191 objects along with their RGB-D information. We take advantage of this new breadth and diversity in the data and present the GCNGrasp framework which uses the semantic knowledge of objects and tasks encoded in a knowledge graph to generalize to new object instances, classes and even new tasks. Our framework shows a significant improvement of around 12% on held-out settings compared to baseline methods which do not use semantics. We demonstrate that our dataset and model are applicable for the real world by executing task-oriented grasps on a real robot on unknown objects.

The specific publications for each chapter are listed below:

Chapter 2: 6-DOF Grasping for Target-driven Object Manipulation in Clutter, ICRA 2020 [179].

Chapter 3: CASSL: Curriculum Accelerated Self-Supervised Learning, ICRA 2018 [176].

Chapter 4: Robot Learning in Homes: Improving Generalization and Reducing Dataset Bias, NuerIPS 2018 [99]. Though not featured in this thesis, the following work also has several related contributions and ideas: Hardware Conditioned Policies for Multi-Robot Transfer Learning [45].

Chapter 5: PyRobot: An Open-source Robotics Framework for Research and Benchmarking, 2019 [177].

Chapter 6: Learning to Grasp Without Seeing, ISER 2018 [175].

Chapter 7: Same Object, Different Grasps: Data and Semantic Knowledge for Task-Oriented Grasping, Submitted. [178].

Part I

Generalization to Clutter

Chapter 2

6-DOF Grasping for Object Manipulation in Clutter

2.1 Introduction

Grasping is a fundamental robotic task, but is challenging in practice due to imperfections in perception and control. Most commonly, grasp planning involves generating gripper pose configurations (3D position and orientation) that maximize a grasp quality metric on a target object in order to find a stable grasp. There are several factors that affect grasp stability, including object geometry, material, gripper contacts, surface friction, mass distribution, amongst several others [32, 196]. Most traditional approaches to grasping assume a separate perception system that can perfectly [196], or with some uncertainty [151], infer object information such as pose and shape. This is followed by physics-based grasp analysis [195, 196] or nearest-neighbour lookup on a database of pre-computed grasps [53]. These methods are slow [94], prone to perception error and do not generalize to novel objects.

Grasp synthesis is much harder in clutter, such as the example in Fig 6.1. The target object has to be grasped without any unwanted collisions with surrounding objects or the environment. In a real world application, a personal robot might be commanded to grasp a specific beverage from a narrow kitchen cabinet packed with other items. Grasps sampled agnostic of the clutter could end up in collision with the environment. Even if the gripper pre-shape is not in collision, it may be challenging to plan a collision-free and kinematically feasible path for the manipulator to achieve the gripper configuration. One would have to generate a diverse set of grasps since not all the grasps will be kinematically feasible to execute in the environment. Most model-based approaches in the grasping and task and motion planning literature assume perfect object knowledge or use

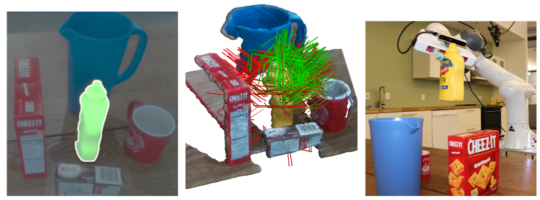

Figure 2.1: Given an unknown target object (*left*) our proposed method leads to robust grasping (*right*) despite challenging clutter and occlusions. This is enabled by explicitly reasoning about successful and colliding grasps (*center*).

an occupancy-grid representation for collision checking, which may not be reliable or practical in real-world settings [29, 70, 128, 196].

A large part of the difficulty lies in perception. In clutter, large and important parts of object geometry are occluded by other objects. Traditional shape matching techniques will find it extremely challenging to operate in such conditions, even when object geometry is known. In addition, getting quality 3D information is challenging and previous methods resort to using high quality depth sensors [153] or using multiple views [232], which would require observation-gathering exploratory movements impossible in confined spaces. This limits the deployment of such systems outside of controlled environments.

Recent works have explored data-driven methods for grasping unknown objects [33, 99, 120, 142, 153, 190, 232]. However, they mainly focus on the limited setting of planar-grasping and bin-picking. Some recent methods tackle the more difficult problem of generating grasps in $SE(3)$ from 2D (image) [176], 2.5D (depth, multi-view) [147, 232, 250] and 3D (point cloud) [48, 144, 173] data. These works primarily consider the problem from an object-centric perspective or in bin-picking settings. We consider the problem of 6-DOF grasp generation in structured clutter using a learning-based approach. Our method uses instance segmentation and point cloud observation from just a single view. We follow a cascaded approach to grasp generation in clutter, first reasoning about grasps at an object level and then checking the cluttered environment for collisions. We use a learned collision checker, which evaluates grasps for collisions from just raw partial point cloud observations and works under varying degrees of occlusion. Specifically, we present the following contributions:

- A learning-based approach for 6-DOF grasp synthesis for novel objects in structured clutter, which uses a learned collision checker conditioned on the gripper information and on the raw point cloud of the scene.
- Showing that our approach, trained only with synthetic data, achieves a grasp accuracy of 80.3% with 23 real-world test objects in clutter. It also outperforms a clutter-agnostic baseline

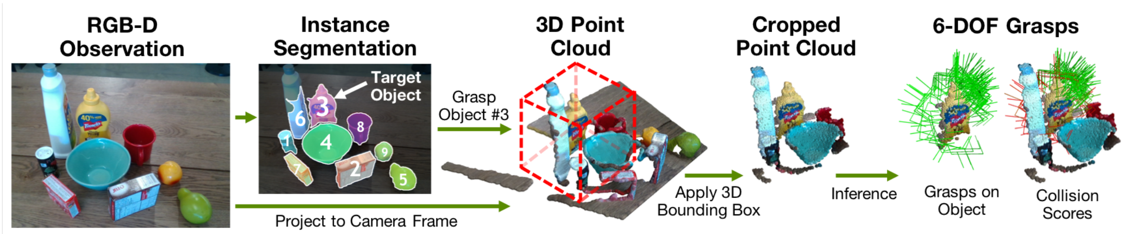

Figure 2.2: Overview of our cascaded grasping framework. A local point cloud centered on the target object is cropped from the scene point cloud using instance segmentation. 6-DOF grasps are then generated and ranked by collisions with the scene.

approach of 6-DOF GraspNet [173] with state-of-the-art instance segmentation [248] by 17.6%.

- Demonstrating an application of our approach in moving blocking objects away out of the way to grasp a target object that is initially occluded and impossible to grasp.

2.2 Related Work

Grasping is a widely studied field in robotics ([32, 33, 196]). In the following we will focus our comparison on existing approaches that are data-driven and the aspects in which they differ from the proposed method.

Grasping in clutter vs. isolated objects: Among learning-based methods for grasping a significant amount focuses on dealing with isolated objects on planar surfaces ([121, 139, 147, 173, 250]). Our approach specifically tackles the problem of grasping objects from a cluster of multiple objects. This problem is significantly harder since the accessible workspace around the target object is severely limited, occlusions are more likely to hamper perception and predicting outcomes might be more difficult due to contact interactions between objects. Although multiple learning-based approaches for dealing with grasping in clutter exist ([142, 152, 190, 232]) we will show in the following that they differ from our approach in multiple aspects.

Bin-picking vs. structured clutter: Most learning-based grasping approaches for clutter deal with rather small and light objects that are spread in a bin ([96, 142, 152, 190]). In contrast our approach focuses on *structured* clutter. We define structured clutter as packed configurations of mostly larger, heavier objects. Examples include kitchen cupboards or supermarket shelves. Compared to the bin-picking setup successful grasps are more sparsely distributed in structured clutter scenarios. Collisions and unintended contact is often more catastrophic since objects have fewer stable equilibria when they are not located on a pile. Since avoiding collision becomes more important, structured clutter is more prominent in evaluations of model-based task-and-motion-planning. Our approach explicitly predicts grasp configurations that are in collision and can do so despite occlusions.

Planar vs. spatial grasping in clutter: Many learning-based grasp algorithms for clutter are limited to planar grasps, representing them as oriented rectangles or pixels in the image ([142, 150, 190, 257]). As a result, grasps lack diversity and picking up an object might be impossible given additional task or arm constraints. This limitation is less problematic in bin-picking scenarios where objects are small and light. In structured clutter, spatial grasping is unavoidable, otherwise pre-grasp manipulations are needed [71]. Those learning-based approaches that plan full grasp poses are either based on hand-crafted features ([106, 129, 156]) or have non-learned components [232]. Our approach uses a learned grasp sampler that predicts the full 6D grasp pose and accounts for unseen parts due to occlusions.

Model-based vs. model-free: A lot of planning approaches exist that tackle scenarios of grasping in structured clutter ([19, 55, 71, 124, 128]). These approaches rely on full observability and prior object knowledge. In contrast, our method does not require any object models and poses; grasps are planned based on raw depth images. In that regard, it is similar to other data-driven methods for clutter ([142, 150, 190, 232, 257]) but differs from techniques that use hand-engineered features ([86, 106, 129, 156]).

Target-agnostic vs. target-driven: Few approaches focus on grasping specific objects in clutter ([62, 114]). Our method is target-driven as it uses instance segmentation [248] to match grasps with objects.

2.3 6-DOF Grasp Synthesis for Objects in Clutter

We consider the problem of generating 6-DOF grasps for unknown objects in clutter. The input to our approach is the depth image of the scene and a binary mask indicating the target object. In particular, we aim to estimate the posterior grasp distribution $P(G^*|X)$, where X is the point cloud observation and G^* is the space of successful grasps. We represent $g \in G^*$ as the grasp pose $(R, T) \in SE(3)$ of an opened parallel-yaw gripper that results in a robust grasp when closing its fingers.

The distribution of successful grasps is complex, multi-modal and discontinuous. The number of modes for a new object is not known a-priori and is determined by the geometry, size, and physics of the object. Additionally, small perturbations of a robust grasp could lead to failure due to collision or slippage from poor contact. Finally, cluttered scenes limit the robot workspace significantly. Although a part of an object might be visible it could be impossible to grasp if the gripper itself is a large object (such as the Franka Panda robot hand we use in our experiments) that leads to collisions with surrounding objects.

2.3.1 Overview of Approach

Our cascaded grasp synthesis approach factors the estimation of $P(G^*|X)$ by separately learning the grasp distribution for a single, isolated object $P(G^*|X_o)$ and a discriminative model $P(C|X, g)$ which we call *CollisionNet* that captures collisions C between gripper at pose g and clutter observed as X. X is the cropped point cloud of the scene and $X_o = \mathcal{M}_o(X)$ is the segmented object point cloud, where $\mathcal{M}_o$ is the instance mask of the target object.

The advantage of this factorization is twofold. First, it allows us to build upon prior work [173] which successfully infers 6-DOF grasp poses for single, unknown objects. Second, by explicitly disentangling the reasons for grasp success, i.e., the geometry of the target object and a collision-free/reachable gripper pose, we can reason beyond simple pick operations. As shown in a qualitative experiment in Sec. 2.4.3 we can use our approach to infer which object to remove from a scene to maximize grasp success of the target object.

Fig. 7.3 shows an overview of our approach. During inference, a target object can be selected based on a state-of-the-art segmentation algorithm [248]. Given this selection we infer possible successful grasps for the object ignoring clutter, and combine it with the collision results provided by CollisionNet.

In the following two sections, we will present both of these models. Note that our particular design decisions are based on comparisons with alternative formulations. In Sec. 2.4.1 we will show how our approach outperforms variants that do not distinguish between grasp failures due to collisions and target geometry, or use non-learned components.

2.3.2 6-DOF Grasp Synthesis for Isolated Objects

We first want to learn a generative model for the grasps given the point cloud observation of the cluttered scene. Though this generative model is learned from a reference set of positive grasps, it is not completely perfect due to several reasons. As a result, we follow the approach presented in [173] to have a second module to evaluate and further improve these generated grasps. Conditioned on the point cloud and grasp, the evaluator predicts a quality score for the grasp. This information could also be used to incrementally refine the grasp. We also explore the importance of object instance information in all stages of the 6-DOF grasping pipeline, from grasp generation to evaluation, in the ablation study.

Variational Grasp Sampling: The grasp sampler is a conditional Variational Autoencoder [126] and is a deterministic function that predicts the grasp g given a point cloud X_o and a latent variable z. $P(z) = \mathcal{N}(0, I)$ is a known probability density function of the latent space. The likelihood of the grasps can be written as such:

$$P(G|X_o) = \int P(G|X_o, z)P(z)dz \tag{2.1}$$

Optimizing Eqn 2.1 is intractable as we need to integrate over all the values of the latent space [126]. To make things tractable, an encoder $Q(z|X_o, g)$ is used to map each pair of point cloud X_o and grasp g to the latent space z while the decoder reconstructs the grasp given the sampled z. The encoder and decoder are jointly trained to minimize the reconstruction loss $\mathcal{L}(\hat{g}, g)$ between the ground truth grasps $g \in G^+$ and predicted grasps $\hat{g}$, with the KL-divergence penalty between the distribution Q and the normal distribution $\mathcal{N}(0, I)$:

$$\mathcal{L}_{VAE} = \sum_{z \sim Q, g \sim G^*} \mathcal{L}(\hat{g}, g) - \alpha \mathcal{D}_{KL}[Q(z|X_o, g), \mathcal{N}(0, I)] \tag{2.2}$$

Note that the input to the VAE is the point cloud of the target object segmented from the scene with instance mask.

To combine the orientation and translation loss, we define the reconstruction loss as $\mathcal{L}(\hat{g}, g) = \frac{1}{n} \sum ||\mathcal{T}(g; p) - \mathcal{T}(\hat{g}; p)||$ where $\mathcal{T}$ is the transformation of a set of predefined points p on the robot gripper. During inference, the encoder Q is discarded and latent values are sampled from $\mathcal{N}(0, I)$. Both the encoder and decoder are based on the PointNet++ architecture [197], where each point has a feature vector along with 3D coordinates. The features of each input point of the point cloud are concatenated to the grasp g and the latent variable z in the encoder and decoder respectively.

Though instance information can give a strong prior about the object, it is not perfect in practice. This is especially the case in cluttered scenarios where objects are occluded or very close to each other, resulting in noisy under and over-segmentation. When rendering the segmentation in simulation, we add random salt-and-pepper noise to the object boundaries and randomly merge partially occluded objects to neighboring ones in image space, to mimic the imperfections of instance segmentation methods on the real images.

Grasp Evaluation: Though the grasp sampler is trained with only positive grasps, it may still predict failed grasps which need to be identified and removed. We train an evaluator that predicts a grasp score $P(S|X_o, g)$, with the training data consisting of positive $G_S^+ = G^+$ and negative $G_S^- = G^-$ grasps. The evaluator's input is X_o, the point cloud of the object segmented from the full scene. Since the space of all possible 6-DOF grasp poses is large, it is not possible to sample all the negative grasps for training the grasp evaluator $P(S|X_o, g)$. Therefore, during training we sample from true negatives but also sample hard negative grasps by perturbing positive grasps with a small translation and orientation and choosing those that are in collision with the object or are too far from the object to grasp the object. At test time on the robot, the grasps are ranked by their evaluator scores and only those above a threshold are selected.

Grasp Refinement: A significant proportion of the grasps rejected by the evaluator are actually in close proximity to robust grasps. This insight could be exploited to perform a local search in the region of g to iteratively improve the evaluator score. We concretely want to sample Δg to increase the probability of success, i.e., $P(S|\Delta g + g, X_o) > P(S|g, X_o)$. The refinement was found using gradient descent in [173]. In practice, computing gradients is not fast. Instead, we use Metropolis-Hastings sampling where a random Δg is sampled and with probability of $\frac{P(S|g+\Delta g, X_o)}{P(S|g,x)}$ grasp $g + \Delta g$ is accepted. We observe that this sampling scheme yields similar performance to the gradient-based one while it is computationally twice as fast.

2.3.3 Collision Detection for Grasps in Clutter: CollisionNet

CollisionNet predicts a clutter-centric collision score $P(C|X, g)$ given the full scene information X. The training data for CollisionNet is $G_C^+ = \{g|g \in G_{ref}, \neg\Psi(g, \mathbf{x})\}$ and $G_C^- = \{g|g \in G_{ref}, \Psi(g, \mathbf{x})\}$. The ground truth labels are generated in simulation with a collision checker Ψ assuming full state information **x**. In each batch, we used balanced sampling of grasps within the subsets of the reference set G_{ref}, which consists of the positive and negative sets (G^+, G^-), hard-negatives generated by perturbing positive grasps (G_{hn}^-) and grasps in free space G_{free}. We observed that balanced sampling improved the stability of training and generalization at test time over uniform sampling from $G^+ \cup G^-$. Similar to the grasp evaluator, the scene/object point cloud X/X_o and gripper point cloud X_g are combined into a single point cloud by using an extra indicator feature that denotes whether a point belongs to the object or to the gripper. The PointNet++ [197] architecture then uses the relative information between gripper pose g and point clouds for classifying the grasps. CollisionNet is optimized using cross entropy loss.

2.3.4 Implementation Details

Training data is generated online by arranging multiple objects randomly at their stable poses. Objects are added to the scene with rejection sampling poses to ensure they are not colliding with existing clutter. In order to generate grasps for the scenes, we combine the positive and negative grasps of each object from [173] which includes a total of 126 objects from several categories (boxes, cylinders, bowls, bottles, etc.). From each scene we take multiple 3D crops centered on the object (with some noise) along with grasps that are inside the crop. The cropped point cloud of the 3D box is down-sampled to 4096 points. All the samplers and VAEs are based on PointNet++ [197] architecture and the dimension of latent space is set to 2. During inference, object instances are segmented with [248]. The VAE sampler generates the grasps given the point cloud of the target object by sampling 200 latent values. Grasps are further refined with 20 iterations of Metropolis-Hastings. The whole inference takes 2.5s on a desktop with NVIDIA Titan XP GPU.

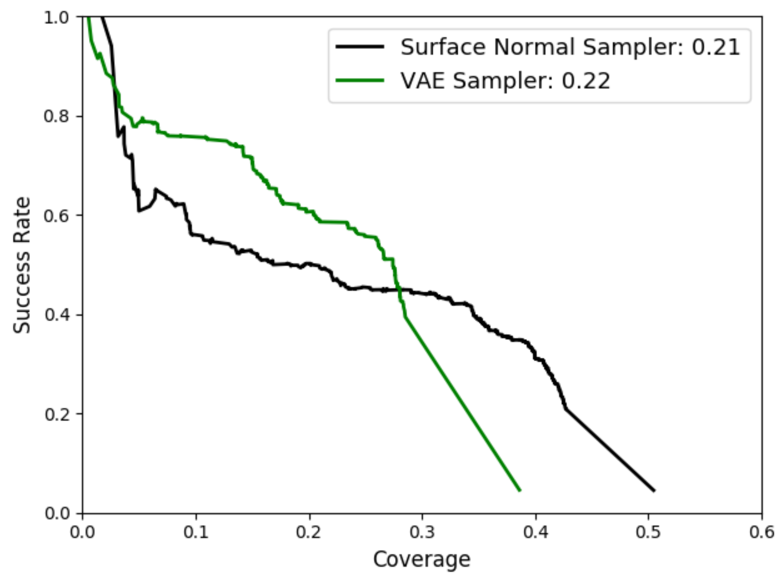

Figure 2.3: Comparing the VAE sampler and Surface Normal Sampler. The number next to the legend is the area under curve (AUC) and the VAE sampler has a higher AUC.

2.4 Experimental Evaluation

2.4.1 Ablation analysis and Discussion

Evaluation Metrics: Following [173], we used two metrics for evaluating the generated grasps: success rate and coverage. Success rate is the percentage of grasps that succeed grasping the object without colliding and coverage is the percentage of sampled ground truth grasps that are within $2cm$ of any of the generated grasps. The ablations were done in simulation with a held-out test set of 15 unseen objects of the same categories arranged at random stable poses in 100 different scenes. Physical interactions are simulated using FleX [164]. Area under curve (AUC) of the success-coverage plot is used to compare different variation of the methods in the ablations.

Learned vs. Surface Normal Based Grasp Sampler: The first ablation study we consider is the effect of using a learned VAE to sample grasps in comparison with a geometric baseline. This baseline generates grasps by sampling random points on the object along surface normals, with random standoff, and random orientation along the surface normal. Fig. 2.3 shows that our learned VAE sampler yields more grasp coverage. It is worth noting that the surface-normal based sampler performed well for simpler shapes like boxes but failed to generate grasps for more complex geometry with rim, handles, etc.

CollisionNet vs Voxel-Based Collision Checking: We compared the effectiveness of CollisionNet with a voxel-based heuristic commonly used (such as in MoveIt! [47]) for obstacle avoidance in

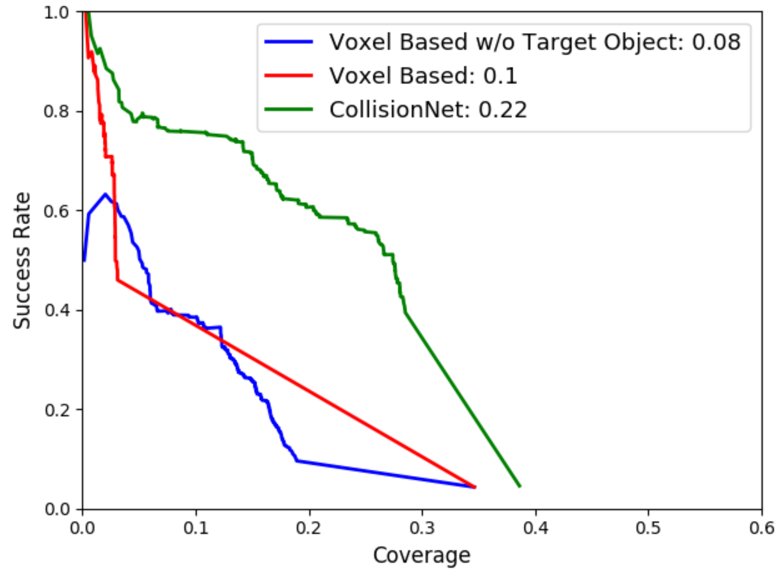

Figure 2.4: CollisionNet outperforms the Voxel-based approach in both success and coverage. The Voxel-based without Target Object ablation only considers collisions with the scene.

unknown 3D environments. In our case, from each object, 100 points are sampled using farthest point sampling. Each sampled point is represented by a voxel cube of size $2cm$. Collision checking is done by checking the intersection of the gripper mesh with any of the voxels. As shown in Fig. 2.4, CollisionNet outperforms the voxel-based heuristic in terms of precision and coverage. Qualitatively, we observed that the voxel-based representation fails to capture collision when the gripper mesh intersects with occluded parts of objects, or if there is missing depth information (see Fig. 2.5). In cases where the voxel-based collision checking fails, CollisionNet has 89.7% accuracy in classifying the collisions correctly.

The voxel-based approach also has several false negatives by rejecting good grasps that are slightly penetrating voxels corresponding to points on the target object, as the voxels expand the spatial region for collision checking. Without considering the voxels on the target object for collisions, the coverage decreases marginally (blue curve in Fig. 2.4). The grasp success also decreases as grasps that are actually colliding with the target object are not pruned out. CollisionNet does not suffer from such biases and can reason about relative spatial information in the partial point clouds.

Single-stage vs. Cascaded Evaluator: Instead of a cascaded grasp generation approach, one could also use a *single-stage* sampler and evaluator with object instance information. Once the grasps are sampled, there is only a single evaluator that directly estimates $P(S, \neg C|X, g)$. The positive training set is $G^+_{SC} = \{g|g \in G^+, \neg\Psi(g, \mathbf{x})\}$ while the negative set is $G^-_{SC} = \{g|g \in G^+, \Psi(g, \mathbf{x})\} \cup \{g|g \in G^-\}$. As a result, some positive grasps $g \in G^+$ will be in collision resulting

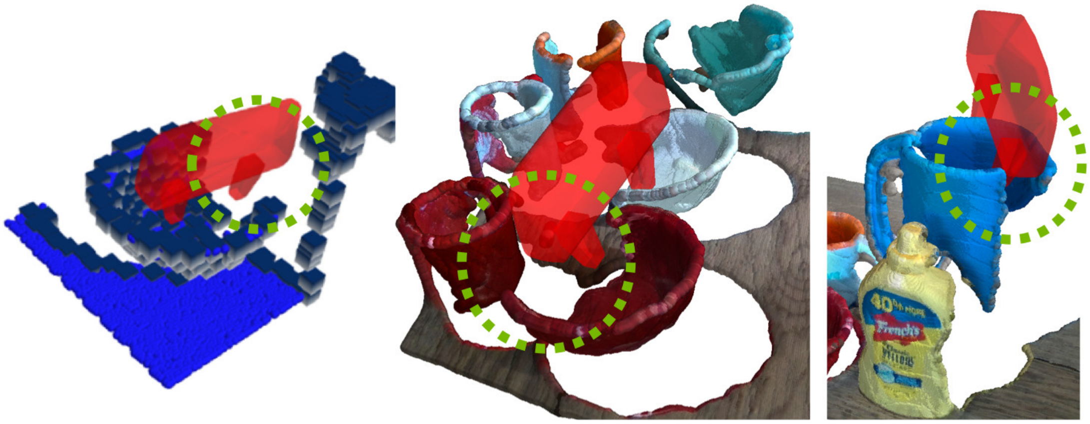

Figure 2.5: Examples where the voxel-based heuristic fails to predict collisions but CollisionNet succeeds. These false positives are due to missing points (region highlighted by dotted circle) from occlusion. These grasps will lead to critical collisions if executed.

in lower scores. An example of the input data to this baseline is shown in Fig. 2.7(b), where the indicator mask of the target object is passed as an additional feature to the PointNet++ architecture. We found that the cascaded model outperformed the single-stage model, as shown in Fig 2.6.

This improvement is due to two factors. First, the VAE is far more proficient in learning grasps from an object-centric observation than from scene-level information. Second, the cascaded architecture imposes an abstraction between having a grasp evaluator that is singly proficient in reasoning about grasp robustness and CollisionNet that is proficient in predicting collisions.

Role of Object Instance Segmentation: We compared our cascaded grasp sampling approach to an instance-agnostic baseline. Without instance information, the baseline is a single-stage grasp planner that uses the point cloud of the scene, since we cannot get a object-centric input. An example of the input data to this baseline is shown in Fig. 2.7(a). From the ablation shown in Fig. 2.6, we found that our cascaded grasp sampler (using instance information and CollisionNet) had a AUC of 0.22 and outperformed the object instance-agnostic baseline in terms of both success and coverage, which had a AUC of 0.02. A common failure mode of the instance-agnostic model is that the variational sampler gets confused as to which object to grasp in the scene, with the latent space being potentially mapped to grasps for multiple objects and degrading the overall grasp quality for all the objects.

2.4.2 Real Robot Experiments

In our robot experiments, we wanted to show that our cascaded grasp synthesis approach (1) transfers to the real world despite being trained only in simulation; (2) has competitive performance

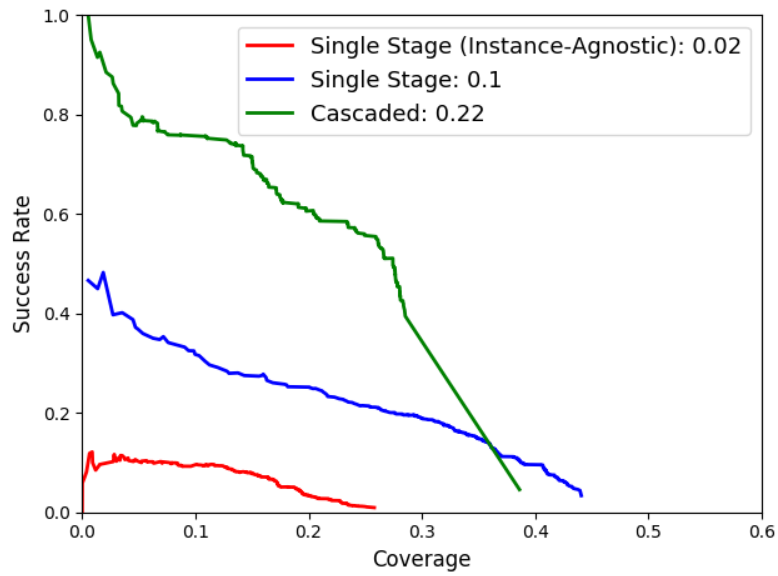

Figure 2.6: Our cascaded approach demonstrates much higher success and coverage compared to a single-stage and instance-agnostic model.

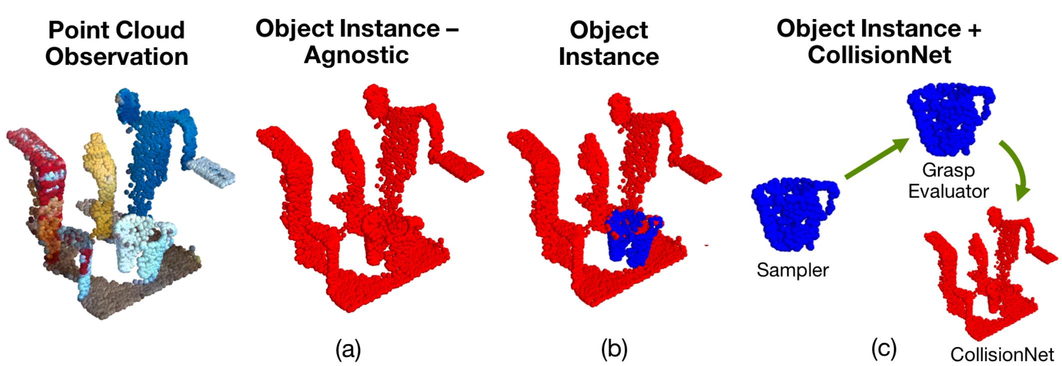

Figure 2.7: Representative example of the data provided to the different grasping architectures a) single-stage model without object instance information b) single-stage model with object instance mask used as a feature vector along with the point cloud c) our cascaded model to sample with object-centric point cloud and evaluate for collisions with clutter-centric data. The target object is colored in blue.

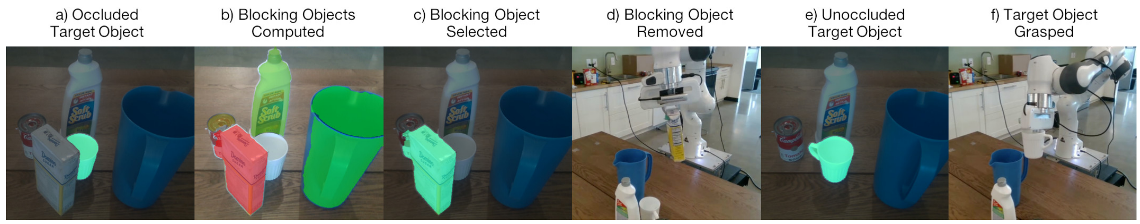

Figure 2.8: Application of our approach in retrieving a partly occluded mug (highlighted in (a)). The blocking objects are ranked (colored in (b), red being most inhibiting) and removed from the scene. The target object is finally grasped in (f).

Figure 2.9: Scenes used for testing. See accompanying video for grasp performance.

for target-driven grasping in real clutter scenes and (3) outperforms baseline methods using the clutter-agnostic 6-DOF GraspNet implementation [173] with instance segmentation and voxel-based collision checking. Our physical setup consists of a 7-DOF Franka Panda robot with a parallel-jaw gripper. We used a Intel Realsense D415 RGB-D camera mounted on the gripper wrist for perception. We execute the grasps in a open-loop fashion where the robot observes the scene once, generates the grasps and then executes solely based on the accurate kinematics of the robot. We found open-loop execution to work reasonably well in our setting. CollisionNet only considers collisions between the gripper and the clutter. We also use occupancy voxel-grid collision checking on top of CollisionNet to make sure that the rest of the manipulator arm does not collide with the clutter and table during motion planning. We compared the performance of the method on 9 different scenes (see Fig 2.9) with the fixed order (pre-computed randomly) of objects to be grasped. A grasp was considered a success if the robot could grasp the object within two consecutive attempts on the same scene. One could choose the order in which all the target objects are completely visible. To make the problem more challenging, half of the chosen target objects were occluded. To generate grasps, a batch size of 200 latents were sampled and the grasps that have scores lower than a threshold for each of the evaluator is filtered out. From the remainder of grasps, the one with maximum score is chosen to be executed.

Table 2.1: Real Robot experiments

Approach	Trials	Performance (%)
6-DOF GraspNet [173] + Ins. Segmentation [248]	32/51	62.7
Object Instance	31/51	60.7
Object Instance + CollisionNet (Ours)	41/51	80.3

The results are summarized in Table 2.1. Our framework achieves a success rate of 80.3% and outperforms the baseline 6 DOF-GraspNet approach by 17.6%. Furthermore, without CollisionNet, our model performance degrades substantially. The two failure cases are the grasps that are colliding with the object but object centric evaluator predicts high score for them. These grasps are filtered out by CollisionNet. The second failure mode pertains to the fact that the voxel-based representation cannot capture all collisions.

2.4.3 Application: Removing Blocking Objects

Consider scenarios such as that shown in Fig. 2.8, where the target object is being blocked by other objects and none of the sampled grasps are kinematically feasible. To accomplish this task, the model needs to generate potential grasps for the target object even though the target object is not physically reachable (detected by low scores from CollisionNet). Given the potential grasps, we can identify which objects are interfering with the generated grasps for the target object. The blocking object j is chosen to be the one with the largest increase in collision scores when removing the corresponding object points from the scene point cloud i.e. $\alpha_j = P(C|\hat{X}_j, g) - P(C|X, g)$. The objects are colored by this ranking metric α_j in Fig. 2.8(b), with the red object being the most blocking. The modified point cloud $\hat{X}_j$, which hallucinates the scene without object j, is implemented by merging the object's instance mask with that of the table and projecting corresponding points to the table plane. Grasps are then generated for the blocking object and removed from the scene. Collision-free grasps can now be generated for the unoccluded target object for the robot to recover it. Target objects can be specified by any down-stream task but in this use case, it is specified by choosing the segmentation corresponding to the target object.

2.5 Conclusion

We present a learning-based framework for 6-DOF grasp synthesis for novel objects in structured clutter settings. This problem is especially challenging due to occlusions and collision avoidance which is critical. Our approach achieves a grasp accuracy of 80.3% in grasping novel objects in clutter on a real robotic platform despite being only trained in simulation. A key failure mode of our approach is that it only considers gripper pre-shape collisions by design and hence motion planning could still fail on generated grasps. In future work, we hope to consider trajectory generation in grasp planning and explore the use of our approach in task planning applications. We also aim to apply this framework in grasping objects from challenging environments like kitchen cabinets and handle the case of retrieving stacked objects in structured clutter.

Part II

Generalization with Robots

Chapter 3

Curriculum Learning for High Dimensional Grasping

3.1 Introduction

With the advent of big data in robotics [20, 140, 142, 190], there has been an increasing interest in self-supervised learning for planning and control. The core idea behind these approaches is to collect large-scale datasets where each data-point has the current state (e.g. image of the environment), action/motor-command applied, and the outcome (success/failure/reward) of the action. This large-scale dataset is then used to learn policies, typically parameterized by high-capacity functions such as Convolutional Neural Networks (CNNs) that predict the actions of the agent from input images/observations. But what is the right way to collect this dataset for self-supervised learning?

Most self-supervised learning approaches use random exploration. That is, first a set of random objects is placed on the table-top followed by a random selection of actions. However, is random sampling the right manner for training a self-supervised system? Random exploration with few thousand data points will only work when the output action space is low-dimensional. In fact, the recent successes in self-supervised learning which shown experiments on real robots (not just simulation) use a search space of only 3-6 dimensions [1] for output action space. Random exploration is also sub-optimal since it leads to a very sparse sampling of the action-space.

We focus on the problem of sampling and self-supervised learning for high-level, high-dimensional control. One possible approach is to collect and sample training data using staged-training [190] or on-policy search [228]. In both these approaches, random sampling is first used to train an initial policy. This policy is then used to sample the next set of training points for learning. However, such

[1] [20, 142, 190] use 3,4,5-dim search space respectively

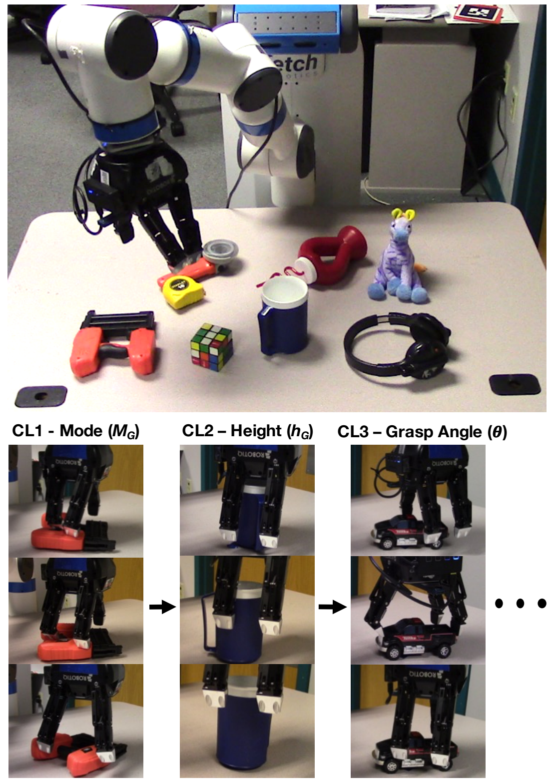

Figure 3.1: Given a table-top scene, our robot learns to grasp objects by Curriculum Accelerated Self-Supervised Learning (CASSL). Given the various control dimensions, such as mode, height, grasp angle, etc., our robot focuses on learning to predict the easier dimensions earlier. We used a Fetch-robot with an adaptive 3-fingered gripper from Robotiq.

approaches are hugely biased due to initial learning from random samples and often sample points from a small search space. Therefore, recent papers have investigated other exploration strategies, such as curiosity-driven exploration [110]. However, data sparsity in high-dimensional action space still remains a concern.

Let's take a step back and think how do humans deal with high-dimensional control. We note that the action space of human control grows continually with experience: the search does not start in high-dimensions but rather in a small slice of the high-dimensional space. For example, in the early stages of human development, when hand-eye coordination is learned, a single mode of grasping (palmar-grasp) is used, and we gradually acquire more complex, multi-fingered grasping modalities [90]. Inspired by this observation, we propose a similar strategy: order the learning in control parameter space by fixing few dimensions of control parameters and sampling in the remaining dimensions. We call this strategy curriculum learning in control space, where the curriculum decides which control dimensions to learn first [2]. We use a sensitivity analysis based approach to define the curriculum over control dimensions. We note that our framework is designed to infer high-level control commands and use planners/low-level controllers to achieve desired commands. In future work, the curriculum learning of low-level control primitives, such as actuator torques, could be explored.

We demonstrate the effectiveness of our approach for the task of adaptive multi-fingered grasping (See Fig 3.1). Our search space is 8-dimensional and we sample the training points for learning control in 6-dimensions (x, y is done via region-proposal sampling, as explained later). We show how a robust model for grasping can be learnt with very few examples. Specifically, we illustrate that defining a curriculum over the control space improves overall grasping rate compared to that of random sampling and staged-training strategy by a significant margin. To the best of our knowledge, this is the first work applications of curriculum learning on a physical robotic task.

3.2 Related Work

Curriculum Learning: For biological agents, concepts are easier to learn when provided in a structured manner instead of an arbitrary order [220]. This idea has been formalized for machine learning algorithms by Elman et al. [75] and Bengio et al. [28]. Under the name of Curriculum Learning (CL) [28], the core idea is to learn easier aspects of the problem earlier while gradually increasing the difficulty. Most curriculum learning frameworks focus on the ordering of training data: first train the model on easy examples and then on more complex data points. Curriculum over data has been shown to improve generalization and speed up convergence [46, 143]. In our work, we propose curriculum learning over the control space for robotic tasks. The key idea in our method is that in higher dimensional control spaces, some modalities are easier to learn and are uncorrelated with other modalities. Our variance-based sensitivity analysis exposes these easy to learn modalities which are learnt earlier while focusing on harder modalities later.

Intrinsic Motivation: Given the challenges for reinforcement learning in tasks with sparse extrinsic reward, there have been several works that have utilized intrinsic motivation for exploration and learning. Recently, Pathak et al. [186] learned a policy for a challenging visual-navigation task by optimizing with intrinsic rewards extracted from self-supervized future image/state prediction error. Sukhbaatar et al. [225] proposed a asymmetric self-play scheme between two agents to improve data efficiency and incremental exploration of the environment. In our work, the curriculum is defined over the control space to incrementally explore parts of the high-dimensional action space.

[2]Note our curriculum is defined in control space as opposed to standard usage where easy examples are used first followed by hard examples for training. In our case, the objects being explored, though diverse and numerous, remain fixed.

Ranking Functions: An essential challenge in CL is to construct a ranking function, which assigns the priority for each training datapoint. In situations with human experts, a stationary ranking function can be hand defined. In Bengio et al. [28], the ranking function is specified by the variability in object shape. Some other methods like Self-Paced Learning [137] and Self-Paced Curriculum Learning [117] dynamically update the curriculum based on how well the agent is performing. In our method, we use a stationary ranking that is learned from performing sensitivity analysis [211] on some data collected by sampling the control values from a quasi-random sequence. This stationary ranking gives priority ordering on control parameters. Most formulations of curriculum training use a linear curriculum ordering. A recent work by Svetlik et al. generated a directed acyclic graph of curriculum ordering and showed improved data efficiency for training an agent to play Atari games with reinforcement learning [229].

Grasping: We demonstrate data-efficiency of CASSL on the grasping problem. Refer to [32, 33] for surveys of prior work. Classical foundational approaches focus on physics-based analysis of stability [184]. However, these methods usually require explicit 3D models of the objects and do not generalize well to unseen objects. To perform grasping in complex unstructured environments, several data-driven methods have been proposed [142, 152, 190]. For large-scale data collection both simulation [152] and real-world robots [142, 190] have been used. However, these large scale methods operate on lower dimensional control spaces (planar grasps are often 3 dimensional in output space) since high-dimensional grasping requires significantly more amount of data. In our work, we hypothesize and show that CASSL requires lesser data and can also learn on higher dimensional grasping configurations.

Robot Learning: The proposed method of Curriculum Accelerated Self-Supervised Learning (CASSL) is not specific to the task of grasping and can be applied to a wide variety of robot learning, manipulation and self-supervised learning tasks. The ideas of self-supervised learning have been used to push and poke objects [20, 191]. Nevertheless, a common criticism of self-supervised approaches is their dependency on large scale data. While reducing the amount of data for training is an active area for research [193], CASSL may help in reducing this data dependency. Deep reinforcement learning [74, 169, 217] methods have empirically shown the ability of neural networks to learn complex policies and general agents. Unfortunately, these model-free methods often need data in the order of millions to learn their perception-based control policies.

3.3 Curriculum Accelerated Self-Supervised Learning (CASSL)

We now describe our curriculum learning approach for high-level control. First, we discuss how to obtain priority ordering of control parameters followed by how to use the curriculum for learning.

Figure 3.2: A small subset of the data processed by the model during training can be seen here. Note that during training, we use a wide variety of objects with different sizes, shapes and rigidity.

3.3.1 CASSL Framework

Our goal is to learn a policy $v = \pi(I)$ and scoring function $y = \mathcal{F}(I, v)$, which given the current state represented by image I and action v predicts the likelihood of success y for the task. Note that in the case of high-dimensional control $v = v_1, v_2....v_K$ where K is the dimensionality of the action space. For the task of grasping an object, y can be the grasp success probability given the image of object (I) and control parameters of the grasp configuration (v). The high-level control dimensions for grasping are the grasping configuration, gripper pose, force, grasping mode, etc. as explained later.

The core idea is that instead of randomly sampling the training points in the original K-dim space and learning a policy, we want to focus learning on specific dimensions first. So, we will sample more uniformly (high exploration) in the dimensions we are trying to learn; and for the other dimensions we use the current model predictions (low exploration). Consequently, the problem is reduced to the challenge of finding the right ordering of the control dimensions. One way of determining this ranking is with expert human labeling. However, for the tasks we care about, the output function $\mathcal{F}(I, v)$ is often too complex for a human to infer rankings due to the complex space of grasping solutions. Instead, we use global sensitivity analysis on a dataset of physical robotic grasping interactions to determine this ranking. The key intuition is to sequentially select the dimension that is the most independent and interacts the least with all other dimensions, hence is easier to learn.

3.3.2 Sensitivity Analysis

For defining a curriculum over control dimensions, we use variance-based global sensitivity analysis. Mathematically, for a model of the form $y = \mathcal{F}(I, v = \{v_1, v_2, \cdots, v_K\})$, global sensitivity analysis aims to numerically quantify how uncertainty in the scalar output (e.g. grasp success probability in our context) can be expressed in terms of uncertainty in the input variables (i.e. the control dimensions) [212]. The first order index, denoted by $S_j^{(1)}$, is the most preliminary metric of sensitivity and represents the uncertainty in y that comes from v_j alone. Another metric of interest is the total sensitivity index $S_j^{(T)}$, which is the sum of all sensitivity indices (first and higher order terms) involving the variable v_j. As a result, it captures the interactions (pairwise, tertiary, etc.) of v_j with other variables. Detailed description on monte carlo estimators for the indices and proofs can be found in [212]. Obtaining the sensitivity metrics requires the model $\mathcal{F}$ or an approximate version of it. Instead, we use Sobol sensitivity analysis [105] implementation in `SAlib` and propose a data-driven method for estimating the sensitivity metrics. In Sobol sensitivity analysis, the control input is sampled from a quasi-random sequence, as it provides a better coverage/exploration of the control space compared to a uniform random distribution.

3.3.3 Determining the Curriculum Ranking

Given a large control space, an intuitive curriculum would be to learn control dimensions in the *descending* order of their sensitivity. However, when designing a curriculum, we also care about the interactions between a control dimension and others. Hence, we need to optimize on getting dimensions that have high sensitivity and low correlation with other dimensions. One way to do this is to minimize higher order (>1) terms (i.e. $S_i^{(T)} - S_i^{(1)}$) and the pairwise interactions between variables $S_i^{(2)}$. Given sensitivity values for each control dimension, we choose the subset of dimensions Ψ which minimize the heuristic Eqn 3.1 below:

$$\min_{\Psi} E(\Psi) = \sum_{i \in \Psi} (S_i^{(T)} - S_i^{(1)}) + \sum_{i \in \Psi} \sum_{j \in (\Omega - \Psi)} |S_{ij}^{(2)}| \tag{3.1}$$

Here Ω is the set of all control dimensions (i.e. $\Omega = \{v_1, v_2, \cdots, v_K\}$), and Ψ is a subset of dimensions. We evaluate all possible $2^K - 1$ subsets and choose the subset with the minimum value as the first set of control dimensions in the curriculum. We then recompute the term for subsets of remaining control dimensions and iteratively choose the next subset (as seen in Algorithm 1). The intuition behind Eqn 3.1 is that we want to choose the subset of control dimensions on which the output y depends the most and is least correlated with the remaining dimensions.

3.3.4 Modeling the Policy

The policy function $v = \pi(I)$ takes the image as input I and outputs the desired action v. Inspired by the approach in [190], we use a CNN to model the policy function. However, since CNNs have been shown to work better on classification than regression, we employ classification instead of regressing control outputs. To this end, each control space is discretized into n_i bins as given in Table 3.1.

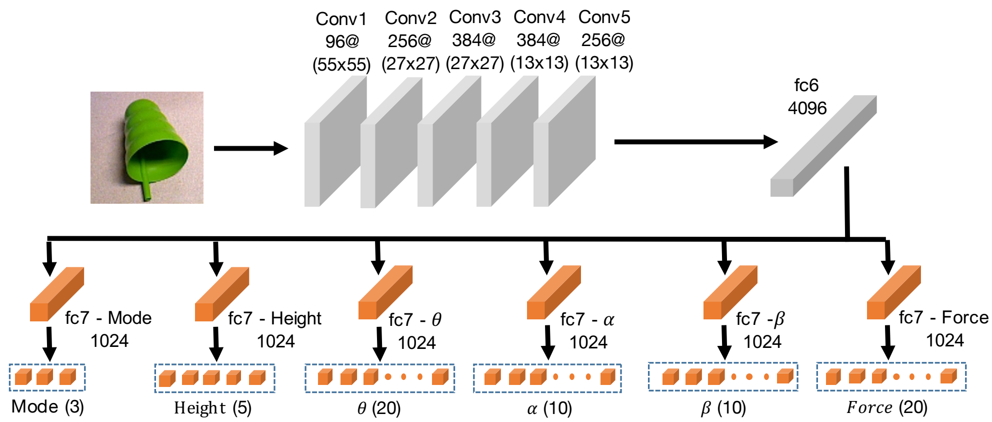

Figure 3.3: We employ a deep neural network to learn the action policy. The convolutional layers and the first fully connected layer (fc6) are shared (in grey). The fc7 and output control layers are trained (in orange) to learn control-specific weights.

Our network design is based on AlexNet [136], where the convolutional layers are initialized with ImageNet [209] pre-trained weights as done before in [190, 192]. We used ImageNet pre-trained features as they been proven to be effective for transfer learning in a number of visual recognition tasks [93, 201]. The network architecture is shown in Fig 3.3. The fully-connected layer's weights are initialized from a normal distribution. While we could have had separate networks for each control parameter, this would enormously increase the size of our model and make the prediction completely independent. Instead, we employ a shared architecture commonly used in multi-task learning [191, 192], such that the non-linear relationship between the different parameters could be learned. Each parameter has a separate fc7 layer and this ensures that the network learns a shared representation of the task until the fc6 layer. The fc8 ouputs are finally sent through and normalized by a sigmoidal function. Predicting the correct discretized value for each control parameter is formulated as a multi-way classification problem. More specifically, $p_{ij} = \pi(I, u_{ij})$ is akin to a Q value function that returns the probability of success when the action corresponding to the j^{th} discrete bin for control dimension i is taken.

3.3.5 Curriculum Training

Algorithm 1 describes the complete training structure of our method. First, initial data is collected to perform sensitivity analysis and given this priority ordering, we begin the training procedure for our policy models. Apart from diversity in the objects seen, we still need to enforce exploration in the action space through all stages of the curriculum training.

As described in Algorithm 1, the greedy action corresponds to executing whatever control values the network predicts. The hyper-parameters, ϵ_{post} = 0.15 and ϵ_{pre} = 0.7, determine the

Algorithm 1 Curriculum Accelerated Self-Supervised Learning (CASSL)

Given: ξ, ϵ_{pre}, ϵ_{post}, $D = \{\}$
Collect: dataset d_0 with quasi-random control samples
Initialize: aggregated dataset $D \leftarrow D \cup \{d_0\}$
$[S^{(1)}, S^{(2)}, S^{(T)}] \leftarrow$ SensitivityAnalysis(D)
Find curriculum $\mathcal{C}$ using $[S^{(1)}, S^{(2)}, S^{(T)}]$
Train: Models M_i^0 with D $\forall i$
for *control (indexed by k) in* $\mathcal{C}$ **do**
 Collect new dataset d_k with the policy below:

$$\pi_{CASSL} = \begin{cases} \epsilon_{post}\text{-Greedy with } M_i^{k-1} & i < k \\ \text{Importance sampling of fc\_8} & i = k \\ \epsilon_{pre}\text{-Greedy with } M_i^{k-1} & i > k \end{cases}$$

 Aggregate new dataset $D = \{D, d_k\}$
 Update Model M_k with D
end

probability of choosing a random action vis-à-vis the greedy one given by the policy. Therefore, for the control dimensions already learned, we are more likely to select the policy via the network. In our framework, for parameters that have already been learned in the curriculum (i.e $i < k$), they will have little exploration. In contrast, for control parameters with $i > k$, they have a great deal of exploration so that the data collected captures the higher order effects between control parameters. When $i = k$, the control is chosen with importance sampling explained as follows. The grasping policy is parameterized as a multi-class classifier on a discretized action space. As a result, the output value p_{ij} from the final sigmoid layer for the j^{th} discrete bin for control i can be treated as a bernoulli random variable with probability p_{ij}. Here, the control value u_i that is selected is the one which the model is most uncertain about and hence has the highest variance i.e u_i = argmax_j p_{ij}(1-p_{ij})). Taking the analytic derivative, the uncertainity is maximized when p_{ij}=0.5. This approach is similar to previous works such as [**?**], where actions were taken based on what the agent is most "curious"/uncertain about and the curiosity reward is defined as the prediction error of the next state given the current state and action. Similarly, in [110], the actions that maximize information gain about the agent's belief of the environment dynamics were taken.

At each stage of the curriculum learning, we also aggregated the training dataset similar to DAgger [208] and prior work [190]. On stage *k* of the curriculum, the network was fine-tuned on D_k =$\{D_{k-1}$, $d_k\}$, where d_k is the dataset collected in the current stage of the curriculum. We sample d_k 2.5 times more than D_{k-1} to give more importance to new datapoints.

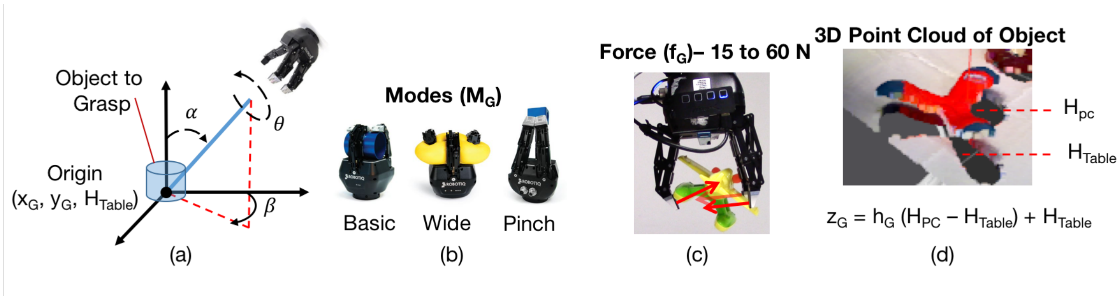

Figure 3.4: Our grasping problem formulation involves the high dimensional control of the adaptive gripper. (a) describes the translational and rotational control dimensions (x_G and y_G are however subsumed in input samples). (b) describes the various modes of grasping, i.e. basic, wide and pinch modes. (c) illustrates the force the gripper is allowed to apply on the objects. (d) describes the gripper's commanded height with respect to the table and the object.

3.4 CASSL for Grasping

We now describe the implementation of CASSL for the task of grasping objects. The grasping experiments and data are collected on a Fetch mobile manipulator [245]. Visual data is collected using a PrimeSense Carmine 1.09 short-range RGBD sensor and we use a 3-finger adaptive gripper from Robotiq. The Expanding Space Tree (ESTk) planner from MoveIt is used to generate collision-free trajectories and state estimation is hand-designed similar to prior work [190] - using background subtraction to detect newly placed objects on the table. We further use depth images to obtain an approximate value for the height of objects.

3.4.1 Adaptive Grasping

The robotiq gripper has three fingers that can be independently controlled and has two primary grasp modalities - encompassing and finger-tip grips. As shown in Fig 3.4, there are three operational modes for the gripper - Pinch, Normal and Wide. Pinch mode is meant for precision grasping of small objects and is limited to finger-tip grasps. Normal grasping mode is the most versatile and can grasp a wide range of objects with encompassing and finger-tip grasps. Similarly, Wide mode is adept at grasping circular or large objects. While the fingers can be individually controlled, we only command the entire gripper to open/close, and the proprietary planner handles the lower-level control for the fingers. The fingers are operated at a speed of 110mm/sec.

The adaptive mechanisms of the gripper also allow it to better handle the uncertainty in the object's geometry and pose. As a result of the adaptive closing mechanism, some of the grasps end up being similar to push-grasps [70]. The gripper fingers sweep the region containing the object, such that the object ends up being pushed inside the fingers regardless of its starting position. Sometimes, such grasps may not have force closure and the object could slip out of the gripper.

Table 3.1: Control parameters, range and discretization

Parameter	Min	Max	# of Discrete Bins
θ	$-180°$	$180°$	20
α	$-10°$	$10°$	10
β	$-30°$	$30°$	10
h_G (Height)	0	1	5
M_G (Mode)	0	2	3
f_G (Force)	15N	60N	20

3.4.2 Grasping Problem Definition

We formulate our problem in the context of table-top grasping, where we infer high-level grasp control parameters based on the image of the object. There are three parameters that determine the location of the grasp (x_G, y_G and h_G), three parameters that determine the approach direction and orientation of the gripper (α, β and θ) and two others that involve the configuration (Mode M_G and Force f_G). The geometric description of the three angles with respect to the object pose is shown in Fig 3.4 and details of each parameter are provided in Table 3.1. θ is very sensitive to asymmetrical, elongated objects while α - the angle from the vertical axis - allows the gripper to tilt its approach direction to grasp the objects from the side. The camera's point cloud data gives a noisy estimate of the object height, denoted by H_{pc}. Let H_{Table} be the height of the table with respect to the robot base. Then, h_G is a scaling parameter (between 0 and 1) that interpolates between these two values, where the final height of the object is $z_G = h_G \cdot (H_{pc} - H_{Table}) + H_{Table}$. The height of a grasp is crucial in ensuring that the gripper moves low enough to make contact with the object in the first place. However, note that the error in the height depends on both h_G and the noisy depth measurement from the camera. As shown in Fig 3.4, there were only three discrete modes for the gripper provided by the manufacturer.

Although the total space of grasp control is 8 dimensional, two of the translational controls (x_G and y_G) are subsumed in the sampling. Given an input image of the entire scene I_S, 150 patches I_P are sampled which correspond to the different values of x_G and y_G. Though this increases the inference time (since we have to input multiple samples), it also massively decreases the search space as a lot of the scene ($\{x_G, y_G\}$ corresponding to the background) is empty. Hence, only 6 dimensions of control $\{h_G, \alpha, \beta, \theta, M_G, f_G\}$ are learned for our task of grasping.

3.4.3 Sensitivity Analysis on Adaptive Grasping

As described in Section 3.3.2, we collect a dataset of 1960 grasp interactions using the sobol quasi-random sampling scheme with an accuracy of 21% during data collection. The results for the $S_i^{(1)}, S_i^{(T)}$ and $S_{ij}^{(2)}$ indices for all control parameters are shown in Table 3.2. While the sensitivity

Table 3.2: Sensitivity Analysis results

	f_G	M_G	α	β	θ	h_G
$S^{(1)}$	0.014	0.109	0.040	0.087	0.164	0.124
$S^{(T)}$	0.799	0.985	0.892	1.130	0.850	0.788
$S^{(2)}$						
f_G	-	0.0125	-0.195	-0.216	-0.153	0.0956
M_G	-	-	-0.0859	0.163	-0.190	0.0385
α	-	-	-	-0.0904	-0.194	-0.236
β	-	-	-	-	-0.280	-0.0519
θ	-	-	-	-	-	-0.260
h_G	-	-	-	-	-	-

analysis was limited to 10 objects, they were diverse in their properties - shape, deformable vs. rigid, large vs. small. Given sensitivity indices for each control parameter, the objective function in Eqn 3.1 is optimized to determine the optimal ordering of the control parameters to learn. The ordering that minimizes Eqn 3.1 is: $[h_G, \theta, f_G, M_G, \alpha, \beta]$ in decreasing order of priority.

3.4.4 Training and Model Inference

Eqn 6.2 is the joint loss function that is optimized. $\hat{y}$ corresponds to the success/failure label, $D(i)$ gives the number of discretized bins for control parameter i (see Table 3.1), K (=6) is the number of control parameters, B is the batch size and σ is the sigmoid activation. $\delta(k, u_{i,j})$ is an indicator function and is equal to 1 when the control parameter i corresponding to bin j is applied. $y_{i,j}^{fc7}$ is the corresponding feature vector that is passed into the final sigmoid activation.

$$L = \sum_{i=1}^{K} \sum_{j=1}^{D(i)} \sum_{k=1}^{B} \delta(k, u_{i,j}) \cdot \text{Cross-Entropy}(\sigma(y_{i,j}^{fc7}), \hat{y}) \tag{3.2}$$

Note that for each image datapoint, the gradients for all six control parameters are back-propagated throughout training. For each stage of the curriculum, the network is trained for 15-20 epochs with a learning rate of 0.0001 using the ADAM optimizer [125]. For inference, once we have the bounding box of the object of interest, 150 image patches are sampled randomly within this window and are re-sized to 224 $\times$ 224 dimensions for the forward pass through the CNN. For each control parameter, the discrete bin with the highest activation is selected and interpolated to obtain the actual continuous value. The networks and optimization are implemented in TensorFlow [17]. As a good practice when training deep models, we used dropout(0.5) to reduce model over-fitting.

Figure 3.5: Set A contains 10 objects seen in training. Set B and C contain 10 and 20 novel objects respectively not used in training

3.5 Experimental Evaluation

Experimental Settings: To quantitatively evaluate the performance of our framework, we physically tested the learned models on a set of diverse objects and measured their grasp accuracy averaged over a large number of trials. We have three test sets (shown in Fig 3.5): 1) Set A containing 10 objects seen by the robot during training 2) Set B containing 10 novel objects and 3) Set C with 20 novel objects. For Sets A and B, 5 grasps were attempted for each object placed in various random initial configurations and the results are detailed in Table 3.3. CL0 in Table 3.3 refers to the model that was trained on the 1960 grasps collected for sensitivity analysis. Fig 3.6 shows some of the successful grasps executed with the robot using the final model trained with CASSL (i.e. CL6). Given the long physical testing time on the largest test set C, we took the best performing model and baselines on test sets A and B and tested them on Set C. As summarized in Table 3.3, the values reported for each model were averaged for a total of 160 physical grasping trials (8 per object). When testing, the object was placed in 8 canonical orientations (NSWE,NE,SE,SW and NW) with respect to the same reference orientation.

Curriculum Progress: The grasp accuracy increases with each stage of curriculum learning on Set A and B, as shown in Fig 3.7. Starting with CL0 at 41.67%, the accuracy topped 70.0% on Set A (Seen objects) and 62% on Set B (Novel objects) at the end of the curriculum for the CL6 model. Note that at each stage of the curriculum, the model trained on the previous stage was used to collect around 460-480 grasps as explained in Algorithm 1. As expected, the performance of the models on Set A with seen objects was better than that of the novel objects in Set B. Yet, the strong grasping performance on unseen objects suggests that the CNN was able to learn a generalized visual representation to scale its inference to novel objects. There was a dip in accuracy for CL2, possibility owing to over-fitting on one of the control dimensions, but the performance recovered in subsequent stages since the models are trained with all the aggregated data.

Table 3.3: Results on test set with seen and novel objects

	Training	Testing		
		Set A	Set B	Set C
CL0	20.9	42.0	42.0	-
CASSL(Ours) - CL6(β)	51.1	**70.0**	**62.0**	66.9
CASSL(Random1)	42.7	56.0	54.0	55.6
CASSL(Random2)	37.1	54.0	50.0	-
Staged Learning [142, 190]	26.85	66.0	54.0	56.9
Random Exploration	25.8	48.0	48.0	-

Figure 3.6: Some successful grasps achieved by model trained with CASSL.

Baseline Comparison: We evaluated against four baselines, all of which are provided equal or more data than that given to CASSL. 1) *Random Exploration* - Training the network from scratch with 4756 random grasps. 2) *Staged Learning* [142, 190] - We first trained the network with data from sensitivity analysis (i.e. CL0) and used this learned policy to sample the next 2796 grasp data points, as done in prior work. The policy was then fine-tuned with the aggregated data (4756 examples). In the third and final stage, 350 new grasp data points were sampled. This staged baseline was the training methodology used in prior work [142, 190]. 3) *CASSL (Random 1 & 2)* - Instead of using sensitivity analysis to define the curriculum, two sets of randomly ranked control parameters were trained with CASSL and the performance of the final trained models is reported in Table 3.3. The ordering for Random 1 and 2 is $[M_G, \alpha, \theta, \beta, f_G, h_G]$ (in decreasing order of priority) and $[\beta, f_G, \alpha, h_G, M_G, \theta]$ respectively. In addition to the baselines above, the CL0 model achieves a grasping rate of around 20.86% and this could be roughly considered as the performance of random grasping trained with 1960 datapoints.

All the curriculum models (except CL0, CL2) outperformed the random exploration baseline's

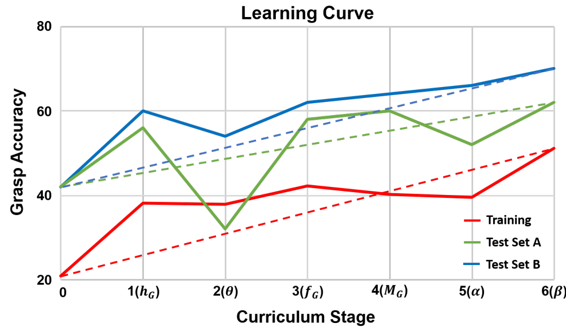

Figure 3.7: Variation in grasp accuracy with respect to stages in learning

accuracy of 48%. On the Set B (novel objects), CL6 showed a marked increase of 14%, 8% and 12% vis-à-vis the random exploration, staged learning and CASSL (Random 2) baselines respectively. For the results on the larger Set C, CL6 still outperformed staged learning by about 10% and CASSL (Random 1) by 11.3%. The curriculum optimized with sensitivity analysis outperformed the random curriculum, illustrating the importance of choosing the right curriculum ranking, the lack of which can hamper learning performance.

3.6 Conclusion

We introduce Curriculum Accelerated Self-Supervised Learning (CASSL) for high-level, high-dimensional control in this work. In general, using random sampling or staged learning is not optimal. Instead, we utilize sensitivity analysis to compute the curriculum ranking in a data-driven fashion and assign the priority for learning each control parameter. We demonstrate effectiveness of CASSL on adaptive, 3-fingered grasping. On novel test objects, CASSL outperformed baseline random sampling by 14%, on-policy sampling by 8% and a random curriculum baseline by 12%. In future work, we hope to explore the following: 1) Modify the existing framework to include dynamically changing curriculum instead of a pre-computed stationary ordering 2) Investigate applications in hierarchical reinforcement learning, where high-level policy trained with CASSL is used alongside a low-level controller 3) Scale CASSL for learning in high dimensional manipulation tasks such as in-hand manipulation.

Chapter 4

Robot Learning in Homes

4.1 Introduction

Powered by the availability of cheaper robots, robust simulators and greater processing speeds, the last decade has witnessed the rise of data-driven approaches in robotics. Instead of using hand-designed models, these approaches focus on the collection of large-scale datasets to learn policies that map from high-dimensional observations to actions. Current data-driven approaches mostly focus on using simulators since it is considerably less expensive to collect simulated data than on an actual robot in real-time. The hope is that these approaches will either be robust enough to domain shifts or that the models can be adapted using a small amount of real world data via transfer learning. However, beyond simple robotic picking tasks [187, 194, 236], there exist little support to this level of optimism. One major reason for this is the wide "reality gap" between simulators and the real world.

Therefore, there has concurrently been a push in the robotics community to collect real-world physical interaction data [85, 140, 142, 176, 181, 190**?**] in multiple robotics labs. A major driving force behind this effort is the declining costs of hardware which allows scaling up data collection efforts for a variety of robotic tasks. This approach has indeed been quite successful at tasks such as grasping, pushing, poking and imitation learning. However, these learned models have often been shown to overfit (even after increasing the number of datapoints) and the performance of these robot learning methods tends to plateau fast. This leads us to an important question: why does robotic action data not lead to similar gains as we see in other prominent areas such as computer vision [226] and natural language processing [57]?

The key to answering this question lies in the word: "real". Many approaches claim that the data collected in the lab is real-world data. But is this really true? How often do we see white table-clothes or green backgrounds in real-world scenarios? We argue that current robotic datasets lack

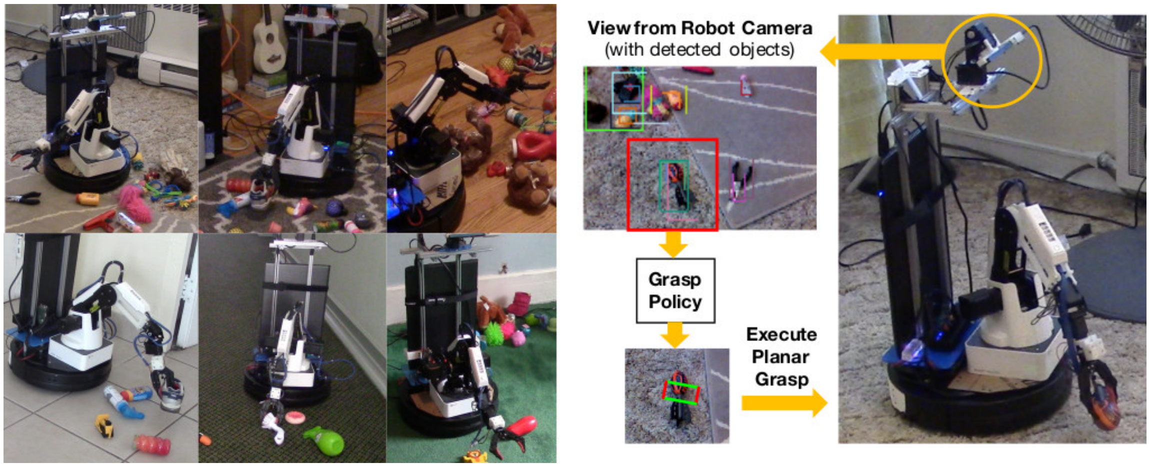

Figure 4.1: We built multiple low-cost robots and collected a large grasp dataset in several homes.

the diversity of environments required for data-driven approaches to learn invariances. Therefore, the key lies in moving data collection efforts from a lab setting to real-world homes of people. In this chapter, we argue that learning based approaches in robotics need to move out of simulators and labs and enter the homes of people where the "real" data lives.

There are however several challenges in moving the data collection efforts inside the home. First, even the cheapest industrial robots like the Sawyer or the Baxter are too expensive (¿20K USD). In order to collect data in homes, we need a cheap and compact robot. But the challenge with low-cost robots is that the lack of accurate control makes the data unreliable. Furthermore, data collection in homes cannot receive 24/7 supervision by humans, which coupled with external factors will lead to more noise in the data collection. Finally, there is a chicken-egg problem for home-robotics: current robots are not good enough to collect data in homes; but to improve robots we need data in homes.

In this work, We propose to break this chicken-egg problem and present the first systematic effort in collecting a dataset inside the homes. Towards this goal: (a) we assemble a robot which costs less than $3K$ USD; (b) we use this robot to collect data inside 6 different homes for training and 3 homes for testing; (c) we present an approach that models and factors the noise in labeled data; (d) we demonstrate how data collected from these diverse home environment leads to superior performance and requires little-to-no domain adaptation. We hope this effort drives the robotics community to move out of the lab and use learning based approaches to handle inaccurate cheap robots.

4.2 Related Work

Large scale robot learning: Over the last few year there has been a growing interest in scaling up robot learning with large scale robot datasets. The *Cornell Grasp Dataset* [139] was among the first works that released a hand annotated grasping dataset. Following this, Pinto and Gupta [190] created a self-supervized grasping dataset in which a Baxter robot collected and self-annotated the data. Levine et al. [142] took the next step in robotic data collection by employing an *Arm-Farm* of several industrial manipulators to learn grasping using reinforcement learning. All of these works, use data in a restrictive lab environment using high-cost data labelling mechanisms. In our work, we show how low-cost data in a variety of homes can be used to train grasping models. Apart from grasping, there has also been a significant effort is collecting data for other robotic tasks. **?**], Finn et al. [85], and Pinto and Gupta [191] collected data of a manipulator pushing objects on a table. Similarly, Nair et al. [181] collects data for manipulating a rope on a table while Yahya et al. [249] used several robots in parallel to train a policy to open a door. Erickson et al. [77], Murali et al. [175], and Calandra et al. [40] collected a dataset of robotic tactile interactions for material recognition and grasp stability estimation. Again, all of this data is collected in a lab environment. We also note several pioneering work in lifelong robotics like Hawes et al. [102], Veloso et al. [240]. In contrast to our work, they focus on navigation and long-term autonomy. **Grasping:** Grasping is one of the fundamental problems in robotic manipulation and we refer readers to recent surveys Bicchi and Kumar [32], Bohg et al. [33] for a comprehensive review. Classical approaches focused on physics-based analysis of stability [184] and usually require explicit 3D models of the objects. Recent papers have focused on data-driven approaches that directly learn a mapping from visual observations to grasp control [139, 142, 190]. For large-scale data collection both simulation [36, 80, 153**?**] and real-world robots [142, 190] have been used. Mahler et al. [153] propose a versatile grasping model, that achieves 90% grasping performance in the lab for the bin-picking task. However since this method uses depth as input, we demonstrate that it is challenging to use it for home robots which may not have accurate depth sensing in these environments.

Learning with low cost robots: Given that most labs run experiments with standard collaborative or industrial robots, there is very limited research on learning on low cost robots and manipulators. Deisenroth et al. [64] used model-based RL to teach a cheap inaccurate 6 DOF robot to stack multiple blocks. Though mobile robots like iRobot's Roomba have been in the home consumer electronics market for a decade, it is not clear whether they use learning approaches alongside mapping and planning.

Modelling noise in data: Learning from noisy inputs is a challenging problem that has received significant attention in computer vision. Nettleton et al. [183] show that training models from noisy data detrimentally impacts performance. However, as the work in Frénay and Verleysen [88] points out, the noise can be either independent of the environment or statistically dependent on the environment. This means that creating models that can account for and correct noise [168, 247] are valuable. Inspired from Misra et al. [168], we present a model that disentangles the noise in the training grasping data to learn a better grasping model.

4.3 Overview

The goal of this work is to highlight the importance of diversifying the data and environments for robot learning. We want to show that data collected from homes will be less biased and in turn allow for greater generalization. For the purposes of this work, we focus on the task of grasping. Even for simple manipulation primitive tasks like grasping, current datasets suffer from strong biases such as simple backgrounds and the same environment dynamics (friction of tabletop etc.). We argue that current learning approaches exploit these biases and are not able to learn truly generalizable models.

Of-course one important question is what kind of hardware should we use for collecting the large-scale data inside the homes. We envision that since we would need to collect data from hundreds and thousands of homes; one of the prime-requirement for scaling is significantly reducing the cost of the robot. Towards this goal, we assembled a customized mobile manipulator as described below.

Hardware Setup: Our robot consists of a Dobot Magician robotic arm [1] mounted on a Kobuki mobile base [4]. The robotic arm came with four degrees of freedom (DOF) and we customized the last link with a two axis wrist. We also modified the original pneumatic gripper with a two-fingered electric gripper [2]. The resulting robotic arm has five DOFs - x, y, z, roll & pitch - with a payload capacity of 0.3kg. The arm is rigidly attached on top of the moving base. The Kobuki base is about 0.2m high with 4.5kg of payload capacity. An Intel R200 RGBD [123] camera was also mounted with a pan-tilt attachment at a height of 1m above the ground. All the processing for the robot is performed an on-board laptop [5] attached on the back. The laptop has intel core i5-8250U processor with 8GB of RAM and runs for around three hours on a single charge. The battery in the base is used to power both the base and the arm. With a single charge, the system can run for 1.5 hours.

One unavoidable consequence of significant cost reduction is the inaccurate control due to cheap motors. Unlike expensive setups such as Sawyer or Baxter, our setup has higher calibration errors and lower accuracy due to in-accuracte kinematics and hardware execution errors. Therefore, unlike existing self-supervised datasets; our dataset is diverse and huge but the labels are noisy. For example, the robot might be trying to grasp at location x, y but to due to noise the execution is at $(x + \delta_x, y + \delta_y)$. Therefore, the success/failure label corresponds to a different location. In order to tackle this challenge, we present an approach to learn from noisy data. Specifically, we model noise as a latent variable and use two networks: one which predicts the likely noise and other that predicts the action to execute.

4.4 Learning on Low Cost Robot Data

We now present our method for learning a robotic grasping model given low-cost data. We first introduce the patch grasping framework presented in Pinto and Gupta [190]. Unlike the data collected in industrial/collaborative robots like the Sawyer and Baxter, there is a higher tendency for noisy labels in the datasets collected with cheap robots. This error in position control can be attributed to a myraid of factors: hardware execution error, inaccurate kinematics, camera calibration, proprioception, wear and tear, etc. We present an architecture to disentangle the noise of the low-cost robot's actual and commanded executions.

4.4.1 Grasping Formulation

Similar to [190], we are interested in the problem of planar grasping. This means that every object in the dataset is grasped at the same height (fixed cartesian z) and perpendicular to the ground (fixed end-effector pitch). The goal is find a grasp configuration (x, y, θ) given an observation I of the object. Here x and y are the translational degrees of freedom, while θ represents the rotational degrees of freedom (roll of the end-effector). Since our main baseline comparison is with the lab data collected in Pinto and Gupta [190], we follow a model architecture similar to theirs. Instead of directly predicting (x, y, θ) on the entire image I, several smaller patches I_P centered at different locations (x, y) are sampled and the angle of grasp θ is predicted from this patch. The angle is discretized as θ_D into N bins to allow for multimodal predictions.

For training, each datapoint consists of an image I, the executed grasp (x, y, θ) and the grasp success label g. This is converted to the image patch I_P and the discrete angle θ_D. A binary cross entropy loss is then used to minimize the classification error between the predicted and ground truth label g. We use a Imagenet pre-trained convolutional neural network as initialization.

4.4.2 Modeling Noise as Latent Variable

Unlike [190] where a relatively accurate industrial arm is used along with well calibrated cameras, our low-cost setup suffered from inaccurate position control and calibration. Though the executions are noisy, there is some structure in the noise which is dependent on both the design and individual robots. This means that the structure of noise can be modelled as a latent variable and decoupled during training [168]. Our approach is summarized in Fig 4.2.

The conventional approach [190] models the grasp success probability for image patch I_P at angle θ_D as $P(g|I_P, \theta_D; \mathcal{R})$. Here $\mathcal{R}$ represents variables of the environment which can introduce noise in the system. In the case of standard commercial robots with high accuracy, $\mathcal{R}$ does not play a significant role. However, in the low cost setting with multiple robots collecting data in parallel,

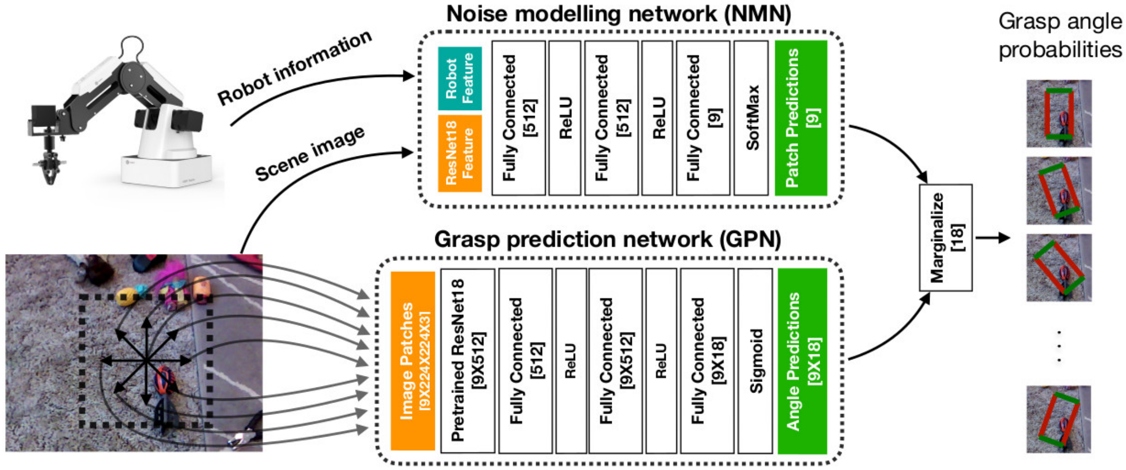

Figure 4.2: Our architecture consists of three components - a) the Grasp Prediction Network (GPN) which infers grasp angles based on the image patch of the object b) the Noise Modelling Network (NMN) which estimates the latent noise given the image of the scene and robot information and the c) marginalization layer computing the final grasp angles.

it becomes an important consideration for learning. For instance, given an observed execution of patch I_P, the actual execution could have been at a neighbouring patch. Here, z models the latent variable of the actual patch executed, and $\widehat{I_P}$ belongs to a set of possible hypothesis neighbouring patches $\mathcal{P}$. We considered a total of nine patches centered around I_P, as explained in Fig 4.2.

The conditional probability of grasping at a noisy image patch I_P can hence be computed by marginalizing over z:

$$P(g|I_P, \theta_D, \mathcal{R}) = \sum_{\widehat{I_P} \in \mathcal{P}} P(g|z = \widehat{I_P}, \theta_D, \mathcal{R}) \cdot P(z = \widehat{I_P}|\theta_D, I_P, \mathcal{R}) \tag{4.1}$$

Here $P(z = \widehat{I_P}|\theta_D, I_P, \mathcal{R})$ represents the noise which is dependent on the environment variables $\mathcal{R}$, while $P(g|z = \widehat{I_P}, \theta_D, \mathcal{R})$ represents the grasp prediction probability given the true patch.

The first part of the equation is implemented as a standard grasp network, which we refer to as the Grasp Prediction Network (GPN). Specifically, we feed in nine possible patches and obtain their respective success probability distribution. The second probability distribution over noise is modeled via a separate network, which we call Noise Modelling Network (NMN). The overall grasp model *Robust-Grasp* is defined by GPN $\otimes$NMN, where $\otimes$ is the marginalization operator.

4.4.3 Learning the latent noise model

Thus far, we have presented our *Robust-Grasp* architecture which models the true grasping distribution and latent noise. What should be the inputs to the NMN network and how should it be trained?

We assume that z is conditionally independent of the local patch-specific variables (θ_D, I_P) given the global information $\mathcal{R}$, i.e $P(z = \widehat{I_P}|\theta_D, I_P, \mathcal{R}) \equiv P(z = \widehat{I_P}|\mathcal{R})$. Apart from the patch I_P and grasp information (x, y, θ), other auxiliary information such as the image of the entire scene, *ID* of the specific robot that collected a datapoint and the raw pixels location of the grasp are stored. The image of the whole scene might contain essential cues about the system, such as the relative location of camera to the ground which may change over the lifetime of the robot. The identification number of the robot might give cues about errors specific to a particular hardware. Finally, the raw pixels of execution contain calibration specific information, since calibration error is coupled with pixel location, since we do least squares fit to compute calibration parameters.

It is important to emphasize that we do not have explicit labels to train NMN. Since we have to estimate the latent variable z, one could use Expectation Maximization (EM) [65]. But inspired from Misra et al. [168], we use direct optimization to jointly learn both NMN and GPN with the noisy labels from our dataset. The entire image of the scene along with the environment information is passed into NMN. This outputs a probability distribution over the patches where the grasps might have been executed. Finally, we apply the binary cross entropy loss on the overall marginalized output GPN$\otimes$NMN and the true grasp label g.

4.4.4 Training details

We used PyTorch [185] to implement our models. Instead of learning the visual representations from scratch, we finetune on a pretrained ResNet-18 [103] model. For the noise modelling network (NMN), we concatenate the 512 dimensional ResNet feature with a one-hot vector of the robot's ID and the raw pixel location of the grasp. This passes through a series of three fully connected layers and a SoftMax layer to convert the correct patch predictions to a probability distribution. For the grasp prediction network (GPN), we extract nine candidate correct patches to input. One of these inputs is the original noisy patch, while the others are equidistant from the original patch. The angle predictions for all the patches are passed through a sigmoid activation at the end to obtain grasp success probability for a specific patch at a specific angle.

We train our network in two stages. First, we only train GPN using the noisy patch which allows it to learn a good initialization for grasp prediction and in turn provide better gradients to NMN. This training is done over five epochs of the data. In the second stage, we add the NMN and marginalization operator to simultaneously train NMN and GPN in an end-to-end fashion. This is done over 25 epochs of the data. We note that this two-stage approach is crucial for effective training of our networks, without which NMN trivially selects the same patch irrespective of the input. The optimizer used for training is Adam [125].

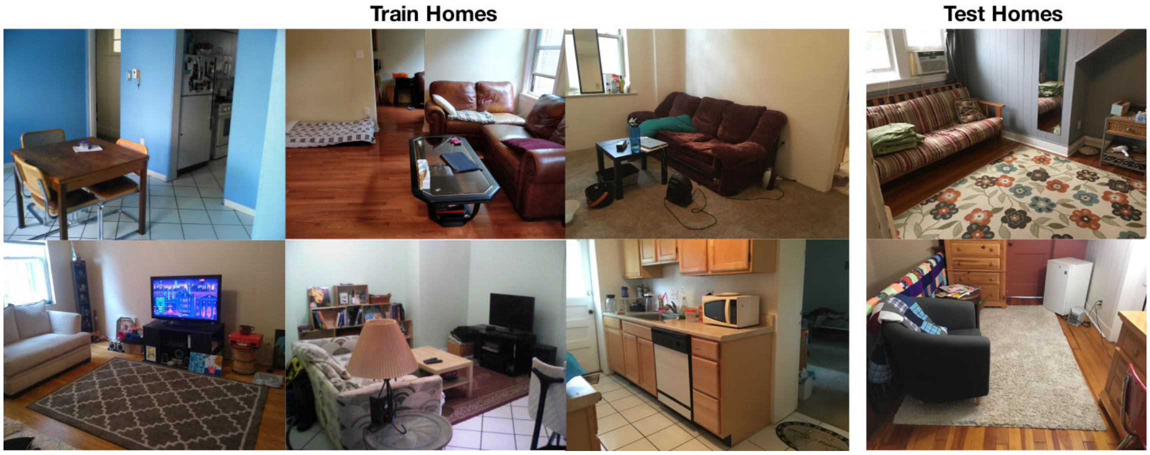

Figure 4.3: Homes used for collecting training data and environments where models were tested

4.5 Experimental Evaluation

In our experimental evaluation, we demonstrate that collecting data in diverse households is crucial for our learned models to generalize to unseen home environments. Furthermore, we also show that modelling the error of low cost robots in our *Robust-Grasp* architecture significantly improves grasping performance. We here onwards refer to our robot as the Low Cost Arm (LCA).

Data Collection: First, we describe our methodology for collecting grasp data. We collected a diverse set (see Fig 4.3) of planar grasping in six homes. Each home has several environments and the data was collected in parallel using multiple robots. Since we are collecting data in homes which have very unstructured visual input, we used an object detector (specifically tiny-YOLO, due to compute and memory constraints on LCA) [202]. This results in bounding box predictions for the objects amidst clutter and diverse backgrounds, of which we only use the 2D location and discard the object class information. Once we have the location of the object in image space, we first sample a grasp and then compute the 3D grasp location from the noisy PointCloud. The motion planning pipeline is carefully designed since our under-constrained robot only has 5 DOFs. When collecting training data, we scattered a diverse set of objects and let the mobile base randomly move and grasp objects. The base was constrained to a 2m wide area to prevent the robot from colliding with obstacles beyond its zone of operation. We collected a dataset of about 28K grasps.

Quantitative Evaluation: For quantitative evaluation, we use three different test settings:

- Binary Classification (*Held-out Data*): For our first test, we collect a held-out test set by performing random grasps on objects. We measure the performance of binary classification where given a location and grasp angle; the model has to predict whether the grasp would be successful or not. This methodology allows us evaluate a large number models without needing to run them on a real robot. For our experiments, we use three different environments/set-ups for held-out data. We collected two held-out datasets using LCA in lab and LCA in home environments. Our third dataset is publicly available Baxter robot data [190].

- Real Low Cost Arm (*Real-LCA*): We evaluated the physical grasping performance of our learned models on the low cost arm in this setting. For testing, we used 20 novel objects in four canonical orientations in three homes not seen in training. Since both the homes and the objects are not seen in training, this metric tests the generalization of our learned model.
- Real Sawyer (*Real-Sawyer*): In the third metric, we measure the physical grasping performance of our learned models on an industrial robotic arm (Sawyer). Similar to the *Real-LCA* metric, we grasp 20 novel objects in four canonical orientations in our lab environment. The goal of this experiment is to show that training models with data collected in homes also improves task performance in curated environments like the lab. Since the Sawyer is a more accurate and better calibrated, we evaluate our *Robust-Grasp* model against the model which does not disentangle the noise in the data.

Baselines: Next we describe the baselines used in our experiments. Since we want to evaluate the performance of both the home robot dataset (*Home-LCA*) and the *Robust-Grasp* architecture, we used baselines for both the data and model. We used two datasets for the baseline: grasp data collected by [190] (*Lab-Baxter*) as well as data collected with our low cost arms in a single environment (*Lab-LCA*). To benchmark our *Robust-Grasp* model, we compared to the noise independent patch grasping model [190], which we call *Patch-Grasp*. We also compared our data and model with DexNet-3.0 from Mahler et al. [153] (*DexNet*) for a strong real-world grasping baseline.

4.5.1 Experiment 1: Performance on held-out data

To demonstrate the importance of learning from home data, we train a *Robust-Grasp* model on both the *Lab-Baxter* and *Lab-LCA* dataset and compare it to the model trained with the *Home-LCA* dataset. As shown in Table 4.1, models trained on only lab data overfit to their respective environments and do not generalize to the more challenging *Home-LCA* environment, corresponding to a lower binary classification accuracy score. On the other hand, the model trained on *Home-LCA* perform well on both home and curated lab environments.

Table 4.1: Results of binary classification on different test sets

Model	**Train Dataset**	**Test Accuracy (%)**		
		Lab-Baxter	Lab-LCA	Home-LCA
Patch-Grasp [190]	Lab-Baxter [190]	76.9	55.1	54.3
Patch-Grasp	Lab-LCA	58.0	69.1	56.5
Patch-Grasp	Home-LCA	71.5	71.3	69.9
Robust-Grasp	Lab-LCA	55.0	71.2	56.1
Fine-tuned	Lab-LCA, Home-LCA	74.6	52.1	59.7
Robot-ID Conditioned	Home-LCA	73.5	71.1	70.6
Robust-Grasp (**Ours**)	Home-LCA (**Ours**)	75.2	71.1	73.0

To illustrate the importance of collecting a large *Home-LCA* dataset, we compare to a common domain adaptation baseline: fine-tuning the model learned on *Lab-LCA* with 5K home grasps ('Fine-tuned' in Table 4.1). We notice that this is significantly worse than the model trained with just home data from scratch. Our hypothesis is that the feature representation learned from Lab data is insufficient to capture the richer variety present in Home Data.

Further, to demonstrate the importance of the NMN for noise modelling, we compare to a baseline model without NMN and feed the robot_id to the grasp prediction network directly ('Robot-ID Conditioned' in Table 4.1), similar to *Hardware Conditioned Policies* [45]. This baseline gives competitive results while testing on *Lab-LCA* and *Lab-Baxter* datasets, however it did not fare as well as *Robust-Grasp*. This demonstrates the importance of NMN and sharing data across different LCAs.

4.5.2 Experiment 2: Performance on Real LCA Robot

In *Real-LCA*, our most challenging evaluation, we compare our model against a pre-trained *DexNet* baseline model and the model trained on the *Lab-Baxter* dataset. The models were benchmarked based on the physical grasping performance on novel objects in unseen environments. We observe a significant improvement of 43.7% (see Table 4.2) when training on the *Home-LCA* dataset over the *Lab-Baxter* dataset. Moreover, our model is also 33% better than *DexNet*, though the latter has achieved state-of-the-art results in the bin-picking task [153]. The relatively low performance of *DexNet* in these environments can be attributed to the high quality depth sensing it requires. Since our robots are tested in homes which typically have a lot of natural light, the depth images are quite noisy. This effect is further coupled with the cheap commodity RGBD cameras that we use on our robot. We used the *Robust-Grasp* model to train on the *Home-LCA* dataset.

Table 4.2: Results of grasp performance in novel homes (*Real-LCA*)

Environment	Model		
	Home-LCA (Ours)	Lab-Baxter [190]	DexNet [153]
1	58.75	31.25	38.75
2	57.5	11.25	26.25
3	70.0	12.50	21.25
Overall	**62.08**	18.33	28.75

4.5.3 Does factoring out the noise in data improve performance?

To evaluate the performance of our *Robust-Grasp* model vis-à-vis the *Patch-Grasp* model, we would ideally need a noise-free dataset for fair comparisons. Since it is difficult to collect noise-free data on our home robots, we use *Lab-Baxter* for benchmarking. The Baxter robot is more accurate and better calibrated than the LCA robot and thus has less noisy labels. Testing is done on the Sawyer robot to ensure the testing robot is different from both training robots.

Results for the *Real-Sawyer* are reported in Table 4.3. On this metric, our *Robust-Grasp* model trained on *Home-LCA* achieves 77.5% grasping accuracy. This is a significant improvement over the 56.25% grasping accuracy of the *Patch-Grasp* baseline trained on the same dataset. We also note that our grasp accuracy is similar to the performance reported (around 80%) in several recent learning to grasp papers [142]. However unlike these methods, we train in a completely different environment (homes) and test in the lab. The improvements of the *Robust-Grasp* model is also demonstrated with the binary classification metric in Table 4.1, where it outperforms the *Patch-Grasp* by about 4% on the *Lab-Baxter* and *Home-LCA* datasets. Moreover, our visualizations of predicted noise corrections in Fig 4.4, show that the corrections depend on both the pixel locations of the noisy grasp and the specific robot.

Table 4.3: Results of grasp performance in lab on the Sawyer robot (*Real-Sawyer*)

Robust-Grasp (Home-LCA)	Patch-Grasp (Home-LCA)	Patch-Grasp (Lab-Baxter)
77.50 (Ours)	56.25	1.25

4.6 Conclusion

In summary, we present the first effort in collecting large scale robot data inside diverse environments like people's homes. We first assemble a mobile manipulator which costs under $3K$ USD and collect a dataset of about $28K$ grasps in six homes under varying environmental conditions. Collecting data with cheap inaccurate robots introduces the challenge of noisy labels and we present an architectural framework which factors out the noise in the data. We demonstrate that it is crucial to train models

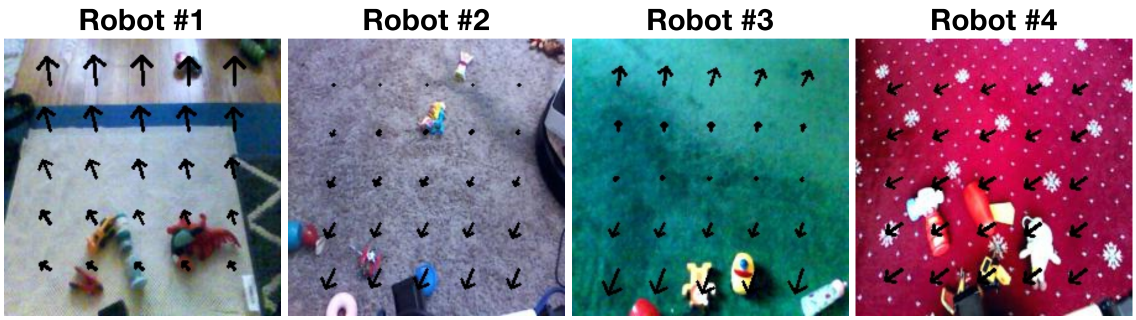

Figure 4.4: We visualize the predicted corrections made by the Noise Modelling Network (NMN). The arrows indicate the NMN learned direction of correction for noisy patches uniformly sampled in the image for multiple robots. This demonstrates that the NMN outputs are both, dependent on the raw pixel location of the noisy grasp and, dependent on the robot ID.

with data collected in households if the goal is to eventually test them in homes. To evaluate our models, we physically tested them by grasping a set of 20 novel objects in lab and in three unseen home environments from *Airbnb*. The model trained with our home dataset showed a 43.7% improvement over a model trained with data collected in the lab. Furthermore, our framework performed 33% better than a baseline *DexNet* model, which struggled with the typically poor depth sensing in common household environments with a lot of natural light. We also demonstrate that our model improves grasp performance in curated environments like the lab. Our model was also able to successfully disentangle the structured noise in the data and improved performance by about 10%.

Chapter 5

Democraticizing Robotics with PyRobot

5.1 Introduction

Over the last few years there have been significant advances in AI, specifically in the fields of machine learning, computer vision, natural language processing and speech. Most of these advancements have been fueled by high-capacity neural networks and the availability of large-scale datasets. However, an often overlooked reason for this fast-paced progress has been the development of a conducive research ecosystem. Platforms such as Caffe [116], PyTorch [185], TensorFlow [16] have reduced the entry barrier, which has democratized and accelerated research in these fields. For example, a new researcher in computer vision can get started with training state-of-the-art detectors using PyTorch and MSCOCO [145] in less than a day. Common platforms and datasets have also led to standardized evaluations and benchmarks which also helps quantify progress in these areas.

The field of data-driven robotics has also seen tremendous excitement and energy in the past several years [20, 21, 84, 99, 113, 140, 142, 153, 190, 192, 193, 258]. However, compared to other areas in AI, it has been relatively hard for a new researcher to get started and contribute to the progress in robotics. Why is that the case? One obvious reason is that researchers have to set up significant hardware infrastructure. This creates a high entry-barrier for researchers both in terms of financial cost and development time. Fortunately, there has been substantial progress on this front with the development of low-cost robots such as Blue [91], LoCoBot [99] and others [12, 251]. In fact, the cost of a robot is now comparable to that of the cost of a GPU! However even with these low-cost robots, getting started in robotics is still hard due to the lack of research platforms and a self-sustaining ecosystem.

Frameworks such as ROS [198] have made setting up robots substantially easier by providing a common mid-level communication layer and tools that are agnostic to low-level hardware and program context. However, there are two issues with such open-source frameworks:

ROS requires expertise: Dominant robotic software packages like ROS and MoveIt! are complex and require a substantial breadth of knowledge to understand the full stack of planners, kinematics libraries and low-level controllers. On the other hand, most new users do not have the necessary expertise or time to acquire a thorough understanding of the software stack. A light weight, high-level interface would ease the learning curve for AI practitioners, students and hobbyists interested in getting started in robotics.

Lack of hardware-independent APIs: Writing hardware-independant software is extremely challenging. In the ROS ecosystem, this was partly handled by encapsulating hardware-specific details in the Universal Robot Description Format (URDF) which other downstream services could read from. Yet, from the perspective of high-level AI applications, most robotics code is still hardware dependent. As a community, we lack a research platform and a common API that we can use to share code, datasets and models.

In this work, we attempt to tackle these challenges via an open-source research platform – **PyRobot**. PyRobot is a light weight, high-level interface on top of ROS that provides hardware independent mid-level APIs and high-level examples for manipulation and navigation. PyRobot also provides libraries for hand-eye calibration, tele-operation, trajectory tracking, and SLAM-based navigation. We believe PyRobot combined with the recently released LoCoBot robot will reduce both the financial cost and development time – leading to democratization of data-driven robotics. The hardware-independent API will lead to development of code and datasets that can be shared across the community. While the current PyRobot release interfaces with LoCoBot and Sawyer, we plan to release integration with several new robots like the UR5 [7] and Franka [11], and simulator platforms like MuJoCo [237] and Habitat [158].

5.2 Related Work

Robotics Software Design. The robotics community has embraced a layered hierarchical software design from the early days [38] and re-usability has been a core design principle [155]. We refer readers to Tsardoulias and Mitkas [239] for a comprehensive review. There have been several motion planning libraries such as OpenRave [68], MoveIt! [47], OMPL [224] which provide hardware-agnostic core functionalities that can be compiled for each specific robot. In the likes of ROS, there have also been robotics ecosystems, such as OROCOS [39] and the Microsoft Robotics Studio that support kinematic libraries, distributed processes, state machines for the real time control of robots.

Low-cost Mobile Manipulators. There has been very limited research on learning on low-cost robots, given that most researchers use standard industrial or collaborative robots. Deisenroth *et al.* [64] used model-based RL to teach a cheap inaccurate 6 DOF robot to stack multiple blocks and a previous iteration of LoCoBot was used in Gupta *et al.* [99] to learn visual grasping policies with real data collected in people's homes. Recently, Gealy *et al.* [91] proposed a compliant low-cost arm using quasi-direct drive actuation.

Open Source Manipulators. There has been very limited work in open sourced manipulators. Raven is a open architecture surgical research robot [207]. Recently, the Open Manipulator project from Robotis allows one to build their own low cost robot with custom kinematics and design [12].

Research Ecosystems in AI Fields. Research in a number of AI fields has benefited from there being common tasks (such as object detection in computer vision or parsing in NLP), common datasets (such as BSDS [162], ImageNet [210], PASCAL VOC [79] and MSCOCO [145] in computer vision, or Penn Tree Bank [159], GLUE [241], SentEval [52] and WMT in NLP, *etc.*), and common code bases to experiment with (DPMs [92], Caffe [116], Stanford CoreNLP [157], spaCy [109], *etc.*). While some people argue that such use of common tasks and datasets can prevent creative progress, at the same time, it has lead to rapid progress in these fields, as researchers can quickly replicate results and build upon each other work.

Benchmarking in Robotics. Benchmarking in robotics is extremely challenging given the vast scope of applications and diversity of physical test conditions (hardware, objects, environment, etc.). It is a well acknowledged concern within the robotics community that we are yet to develop reliable benchmarking metrics that can be widely adopted to quantify research progress. Several workshops have tried to stimulate discourse towards this end [14, 15] and different task specific metrics have been proposed for grasping [154], gripper design [135], SLAM [15], etc. Research has also benefited from creating object datasets with shape and grasp information, such as the Columbia Grasp Database [95], DexNet [152] and KIT Object Models [122], which could be used for perception and motion planning. The YCB dataset went a step further by distributing a physical dataset of household and kitchen objects with corresponding meta data (shape, RGBD scans, etc) [41]. While there is no consensus yet on benchmarking in robotics, we hope that the combination of PyRobot and LoCoBot will facilitate further discussion.

5.3 PyRobot Framework

PyRobot is a python-based robotics framework that isolates the ROS system [198] from the user-end and supports the same API across different robots (see Figure 5.1 for an overview). Essentially, it provides a python wrapper around the mid-level features provided by ROS and the low-level C++/C controllers and driver backends. PyRobot has common utility functions for all robots, such as joint position control, joint velocity control, joint torque control, cartesian path planning, forward kinematics and inverse kinematics (based on the robot URDF file), path planning, visual SLAM, among other features. Though it abstracts away the complexity of the underlying software

Listing 5.1 PyRobot example for position control on LoCoBot and Sawyer.

```
# LoCoBot - Arm
from pyrobot import Robot
bot = Robot('locobot')
target_joints = [0, 0, 0, 0, 0]
bot.arm.set_joint_positions(target_joints)

# LoCoBot - Base
target_position = [1, 1, 1]
bot.base.go_to_absolute(target_position)

# Sawyer
from pyrobot import Robot
bot = Robot('sawyer',
            use_arm=True,
            use_base=False,
            use_camera=False,
            use_gripper=True)
target_joints = [0, 0, 0, 0, 0, 0, 0]
bot.arm.set_joint_positions(target_joints)
```

stack, users still have the flexibility to use components at varying levels of the hierarchy, such as commanding low-level velocities and torques by-passing a planner. We summarize the design philosophy behind PyRobot below.

Beginner-friendly. Ideally, new users should be able to start commanding a robot in just a few lines of code, as shown in the Listing 5.1, without learning ROS or the underlying software and firmware stack.

Hardware-agnostic design. PyRobot is designed to easily accommodate common robotic manipulators and mobile bases. Currently, it supports LoCoBot, a low-cost mobile robot with a 5-DOF manipulator and a Sawyer robot. Each robot has a YACS [13] configuration file that specifies the necessary robot-specific parameters: joint names, ROS topics to get state and set commands, base frame, end-effector frame, planner configuration, inverse kinematics solution tolerance, whether it has an arm or base or camera, *etc.*. A PyRobot object requires the config file for initialization. As shown in Listing 5.1, the Sawyer robot can be commanded in a manner identical to that of LoCoBot.

Open Source. Robotics systems development has typically been constrained to robotics experts in academia and industry with access to expensive and niche robotics systems. However, the extensive scope of artificial intelligence requires strong collaboration between researchers to build and maintain these large systems and one can contribute to all layers of the stack with open sourcing. Apart from the open software, LoCoBot works as an affordable open hardware that can be easily assembled for use with PyRobot. While simulation is useful for software testing and running experiments, writing software that works on the real robot is the eventual goal of the field and has

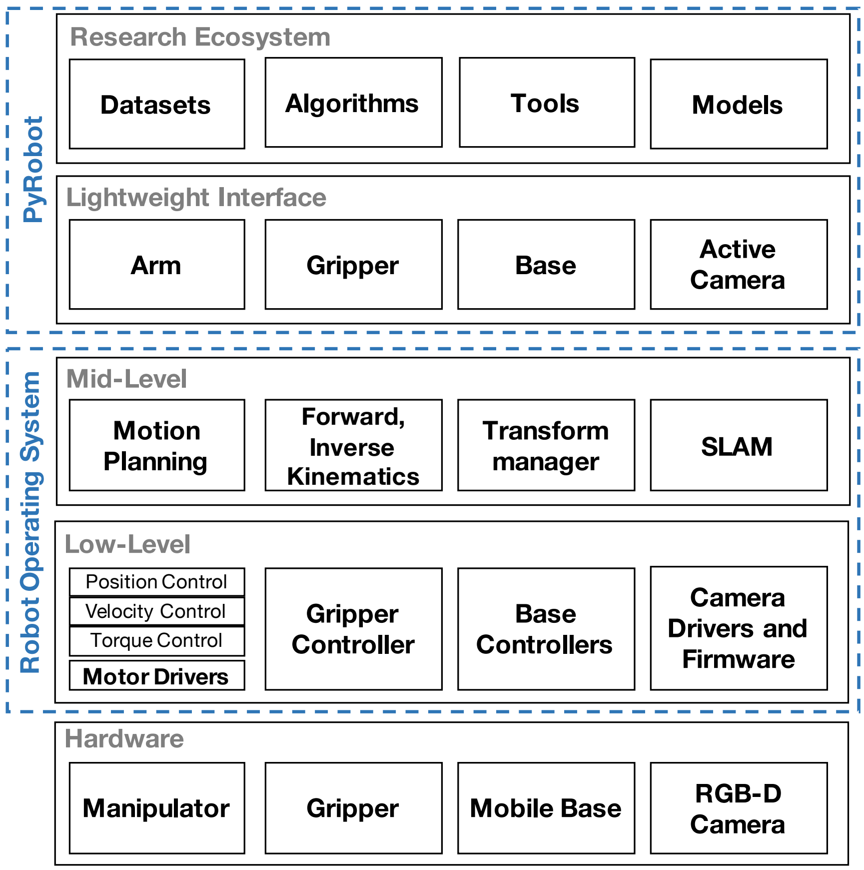

Figure 5.1: Overview of PyRobot system architecture.

severe challenges. As more developers have access to both open hardware and software, high quality applications tested on real robots can be publicly shared.

5.4 Supported Hardware and Simulators

PyRobot is currently integrated with the following robots. In addition to real robots, PyRobot can also be used to control robots in simulators like Gazebo.

LoCoBot: LoCoBot, shown in Figure 5.2 (left), is a low-cost mobile manipulator platform built for easy setup and benchmarking robot learning research. It consists of a Trossen Widow X robotic arm [8] assembled with Dynamixel XM-430 and XL-430s servo motors. The arm has five degrees of freedom (DOFs) - with a working payload of $0.2\,\mathrm{kg}$ and a maximum reach of $0.55\,\mathrm{m}$. The robot comes in two versions, with the arm rigidly mounted on a Kobuki mobile base [4]. The Kobuki base is about $0.12\,\mathrm{m}$ high with payload capacity of around $4.5\,\mathrm{kg}$. For visual perception, an Intel Realsense D435 RGBD camera [123] is mounted with a pan-tilt attachment at a height of about $0.6\,\mathrm{m}$ above the ground. An automatic camera calibration routine is implemented in the software suite. LoCoBot also comes with a Intel NUC (i5, 8GB RAM) machine rigidly attached on the base, which could be used for on-board compute. Kobuki base is powered through its own battery that can run base for about 2 hours. We use a 185 Wh battery pack [9] to power the arm, pan-tilt mount, and the on-board computer. On a full charge, the complete system is able to run for 50-60 minutes. LoCoBot-Lite, shown in Figure 5.2 (right), is a cheaper version of LoCoBot that uses the Create2 base [10] instead of the Kobuki base.

Sawyer: The Sawyer is a 7-DOF collaborative robot arm from Rethink Robotics [6]. PyRobot interfaces with the Intera SDK provided with the Sawyer.

Simulators: PyRobot currently supports Gazebo simulator [130], a 3D rigid body simulator popular in the robotics community. For LoCoBot and LoCoBot-Lite, PyRobot supports tight integration with Gazebo *i.e.*, the same code can be run on both Gazebo and the real robot.

5.5 PyRobot Controllers

While a number of robots come with their own implementations for low-level control, PyRobot implements basic controllers for differential drive bases. It also interfaces with planners such as MoveIt! [47] and Movebase [160]. We measure the performance of these controllers and planners implemented in PyRobot for the LoCoBot base and arm.

Table 5.1: Base position control performance for LoCoBot and LoCoBot-Lite. We report translation and rotation error for different motion types for the different controllers for base position control implemented in PyRobot. Lower errors are better.

	Error with respect to motion capture			Error with respect to odometry		
Controllers	ILQR	Proportional	Movebase	ILQR	Proportional	Movebase
LoCoBot						
Linear motion						
Translation (mm)	17 ± 5	46 ± 23	89 ± 16	3 ± 1	41 ± 32	102 ± 2
Rotation (deg)	0.43 ± 0.25	1.77 ± 1.46	10.81 ± 2.19	0.12 ± 0.10	1.65 ± 1.37	10.63 ± 2.19
Rotation motion						
Translation (mm)	6 ± 0	6 ± 4	4 ± 2	0 ± 0	5 ± 1	2 ± 1
Rotation (deg)	1.32 ± 0.68	2.48 ± 0.98	12.53 ± 1.09	1.45 ± 0.24	2.54 ± 1.02	13.08 ± 1.18
Combined motion						
Translation (mm)	16 ± 2	65 ± 52	78 ± 2	6 ± 1	55 ± 50	87 ± 15
Rotation (deg)	0.29 ± 0.19	3.2 ± 2.69	11.59 ± 1.3	0.84 ± 0.20	2.35 ± 2.94	11.65 ± 1.63
LoCoBot-Lite						
Linear motion						
Translation (mm)	144 ± 8	142 ± 7	260 ± 81	9 ± 5	34 ± 5	99 ± 31
Rotation (deg)	1.79 ± 1.59	2.82 ± 0.52	7.34 ± 8.19	1.6 ± 1.5	1.61 ± 0.34	5.21 ± 3.13
Rotation motion						
Translation (mm)	3 ± 2	3 ± 2	3 ± 1	2 ± 2	3 ± 3	3 ± 1
Rotation (deg)	6.97 ± 1.71	3.07 ± 3.47	9.94 ± 1.46	1.44 ± 1.12	4.59 ± 2.78	3.42 ± 1.66
Combined motion						
Translation (mm)	123 ± 7	99 ± 4	230 ± 57	5 ± 6	93 ± 19	93 ± 21
Rotation (deg)	2.8 ± 1.68	1.19 ± 0.95	5.87 ± 8.22	2.57 ± 1.31	1.57 ± 1.15	4.18 ± 3.45

5.5.1 Accuracy of Base Control

PyRobot implements position controllers to command the robot base to a desired target position (parameterized as a 3-DOF pose, (x, y) location of the base and its heading θ: $[x, y, \theta]$). We implement the following three controllers:

DWA Controller from Movebase: We implemented Dynamic Window Approach Controller (DWA) [87] for our robot through Movebase [160] navigation engine. In this approach, we repeatedly sample a discrete sequence in the robot's control space with the highest score and execute the sequence until the target is reached.

Proportional Controller: We decompose the motion into an on-spot rotation, linear motion and a final on-spot rotation at the target location. Each segment of this motion is executed using a proportional controller that applies velocities proportional to the tracking error. For smooth motion, we bound the velocities and the change in velocities.

Linear Quadratic Regulator: We analytically compute a trajectory (a sharp one that breaks the motion into on-spot rotation, straight motion and a final on-spot rotation; or a smooth one by fitting a bézier curve between the stating state and the ending state). We sample this trajectory to obtain a state trajectory using constraints on maximum linear and angular velocities. We linearize the dynamics of the robot (assumed to be a bicycle model [25]) around this state trajectory, and construct a LQR feedback controller [25] to track this state trajectory.

We conducted trials on the robot to quantify the accuracy of each of these different position controllers on both LoCoBot and LoCoBot-Lite. We measured accuracy using the difference in commanded state *vs.* the achieved state as measured using a Vicon motion capture system. The error was factored into translation (difference in (x, y) location), and rotation (difference in the heading θ). We report these errors in Table 5.1. We group trials into the following three categories: *a) Linear motion*: 5 trials each with targets $2\,\mathrm{m}$ in front ($[2, 0, 0]$), or $2\,\mathrm{m}$ behind ($[-2, 0, 0]$); *b) On-spot rotation*: 5 trials each with target being left rotation by $\pi/2$ ($[0, 0, \pi/2]$), right rotation by $\pi/2$ ($[0, 0, -\pi/2]$); *c) Combined linear and rotation motion*: 5 trials each with targets $[1, 1, 0]$ and $[-1, -1, 0]$.

Table 5.1 reports translation and rotation errors for the different controllers for the two robots for these different cases. We generally note that errors are lower for LoCoBot *vs.* LoCoBot-Lite. Additionally, LQR and proportional controller generally perform better than the DWA controller from Movebase. As all these controllers close the loop on the base odometry, we additionally include errors with respect to base odometry in right part of the table. We observe that the LQR controller is more effective at closing the loop.

PyRobot also implements trajectory tracking (using feedback controllers as described above). We show qualitative comparisons between different controllers in Figure 5.4.

5.5.2 Repeatability Tests for Manipulator

Compared to expensive industrial and collaborative robots, low-cost manipulators like LoCoBot suffer from control errors that can be attributed to a range of factors: manufacturing and assembling error, gear backlash, hardware execution error, kinematics inaccuracy, hand-eye calibration error, motor wear and tear, etc. The position-control repeatability was analyzed by commanding the arm to 4 different 3D poses (and the home pose) in a 2D grid at a fixed height without carrying a payload for a total of 10 repetitions per pose. The ground truth positions were measured using a Vicon motion capture system at $120\,\mathrm{Hz}$. The arm always started at the home pose (when the joint angles are all 0) before moving to the commanded end pose. The results are summarized in Table 5.2. Overall, the arm had a repeatability error of $0.33\,\mathrm{mm}$ to $0.58\,\mathrm{mm}$, computed based on ISO9283 standard. Poses 1 and 3 were closer to the robot torso and had lower error compared to Pose 2 and 4 where the arms were extended at the extremities of the workspace. The standard deviation along the z axis was also higher across all poses due to gravity. For comparison, the Sawyer and UR5 robots are reported to have a repeatability of $0.1\,\mathrm{mm}$ [6, 7]. The position control in the initial release only relies on proprioceptive feedback, and using feedforward model-based control in future release could reduce the error further. The PID gain settings are exposed to the user for more specialized robot or task-specific tuning.

Table 5.2: Locobot Arm Pose Repeatability

Std Dev.(mm)	Poses				
	1	2	3	4	Home
x	0.12	0.13	0.07	0.11	0.15
y	0.13	0.07	0.10	0.14	0.27
z	0.21	0.33	0.22	0.31	0.24
Repeatability (mm)	0.41	0.58	0.33	0.50	0.52

5.6 High-Level AI Applications

We discuss implementation of a few example high-level AI applications through the PyRobot API.

5.6.1 Visual SLAM

Visual SLAM algorithms provide more accurate odometry as compared to odometry that is derived purely from inertial sensors on the base. We deployed ORB-SLAM2 [174], a leading visual SLAM systems in the PyRobot library. ORB-SLAM2 is a feature-based indirect visual SLAM system that uses ORB features to perform tracking, mapping, and loop closing. We adapt the open-source ORB-SLAM2 code into a ROS package. This package saves RGB and depth images of the keyframes and continuously publishes camera trajectory and camera pose. PyRobot uses this published pose information to return the robot base state and trajectory. This state derived from visual SLAM can be used in downstream controllers or algorithms for more accurate behavior. PyRobot also supports dense map reconstruction, by integrating depth image observations using the ORB-SLAM2 estimated camera pose. This can be used for motion planning for navigation tasks.

5.6.2 Navigation via SLAM and Path Planning

We deployed Movebase [160] ROS package on LoCoBot and LoCoBot-Lite for safe navigation in environments with obstacles. We use the occupancy map as obtained from visual SLAM, to compute a 2D cost-map that denotes regions of the environment where the robot is safe to move. Movebase uses this cost-map to generate collision free trajectories to goals specified in the environment. These trajectories can be executed using any of the controllers implemented in PyRobot. These steps are run continuously, and the plan is updated if it becomes infeasible as the robot perceives previously unseen parts of the environment.

5.6.3 Learned Visual Navigation

We deploy learned policies for visual navigation on LoCoBot using PyRobot API. We work with the cognitive mapping and planning policy (CMP) from Gupta *et al.* [101]. Given an input goal location, CMP policy takes in the current image from the on-board camera to output one of four macro-actions (stop, turn left, turn right or go straight). We use the base position control interface in PyRobot API to execute these actions. Listing 5.2 shows simplified code, and Figure 5.6 shows frames from a sample execution.

5.6.4 Grasping

We deploy a learned-based grasping algorithm to grasp objects placed on the ground from RGB images using the PyRobot API. The model is trained on data from people's homes [99] and is robust to a wide variety of objects and backgrounds. This model outputs a grasp in the image space. This grasp is parameterized by 2D location in the image and the gripper orientation. We convert this 2D location and orientation into the *grasp position* (3D location and orientation) using known camera parameters, and the depth image. We command the robot to the *pre-grasp location*, that is a few centimeter above the grasp position, lower the arm to reach the object, and close the gripper to grasp the object. Listing 5.3 shows simplified code, and Figure 5.7 shows sample grasps using the LoCoBot.

5.6.5 Pushing

We deploy a heuristic-based pushing algorithm using PyRobot. It relies on the depth sensor, and thus the quality of the pushing depends on how well the stereo-based depth sensor behaves in different background. To achieve the best performance, it is best to place the robot on a floor with non-uniform texture.

The algorithm can be summarized with the following steps: (1) Move the arm out of the camera's field of view. (2) Filter the point cloud seen by the RGBD camera, specifically removing points too far away and those that correspond to the floor by coordinate thresholding. (3) Project the remaining point cloud onto the xy-plane and use DBSCAN [78] algorithm to automatically cluster the projected points. (4) Randomly select one cluster and choose a random push-start point on the enclosing bounding box of the cluster. (5) Move the gripper to the push-start point and move the gripper horizontally towards the center of the cluster. Listing 5.4 shows simplified code.

5.7 Conclusion

In this chapter, we describe the PyRobot framework, which provides a high-level hardware independent API to control different robots. We believe PyRobot when combined with low-cost robots such as LoCoBot, will reduce the barrier to entry into robotics. In the immediate future, we will continue to grow the functionality in PyRobot such as by interfacing with simulators (like AI Habitat [158], Gibson [246] and MuJoCo [237]), improving controllers such as be implementing gravity compensation for LoCoBot. But more broadly, we believe PyRobot will lead to the development of a research and teaching ecosystem.

PyRobot for robotics instruction. Having a beginner-friendly and open architecture is great for robotics education, as affordable robotic setups with LoCoBot and PyRobot could easily be assembled and scaled for hands-on instruction. 10 LoCoBots were used in the Spring 2019 offering of 16-662: Robot Autonomy (by Professor Oliver Kroemer) in the Robotics Institute at CMU, to support homework assignments and projects. We believe many more such courses will follow.

PyRobot as a research ecosystem. Compared to other fields, benchmarking in robotics is challenging due to several reasons. PyRobot's unified API and LoCoBot's standard hardware, will allow researchers to share their high level algorithmic implementations, models and datasets collected on a real robot. This will allow researchers to collaborate and iterate faster on robotics applications. We will continue to expand the set of pre-trained models. Hopefully, other researchers will find the PyRobot framework useful and contribute their models for others to use as well.

5.8 Code Listings

Listing 5.2 Visual navigation example using PyRobot API.

```
from pyrobot import Robot

# Construct Robot.
bot = Robot('locobot')

# Construct policy.
policy = CMP()

# Relative position for each action.
dv = 0.4          # Forward step size
dw = np.pi/2.     # Rotation step size
action_position = [[0., 0., 0.0],
                   [0., 0., -dw],
                   [0., 0., +dw],
                   [dv, 0., 0.0]]

# Set goal for policy.
policy.set_new_goal(goal)
while action != 0:
    # Get image.
    rgb = bot.camera.get_rgb()

    # Compute action.
    action = policy.compute_action(rgb)

    # Execute action.
    position = action_position[action]
    bot.base.go_to_relative(position)
```

Figure 5.2: LoCoBot (left) and LoCoBot-Lite (right). Both robots have a 5 DOF arm mounted on top of a mobile base (Kobuki or Create2). Robots are equipped with a RGB-D camera mounted on a pan-tilt stand. Robots come with a battery pack and an on-board computer.

Figure 5.3: LoCoBot is low-cost and hence scalable.

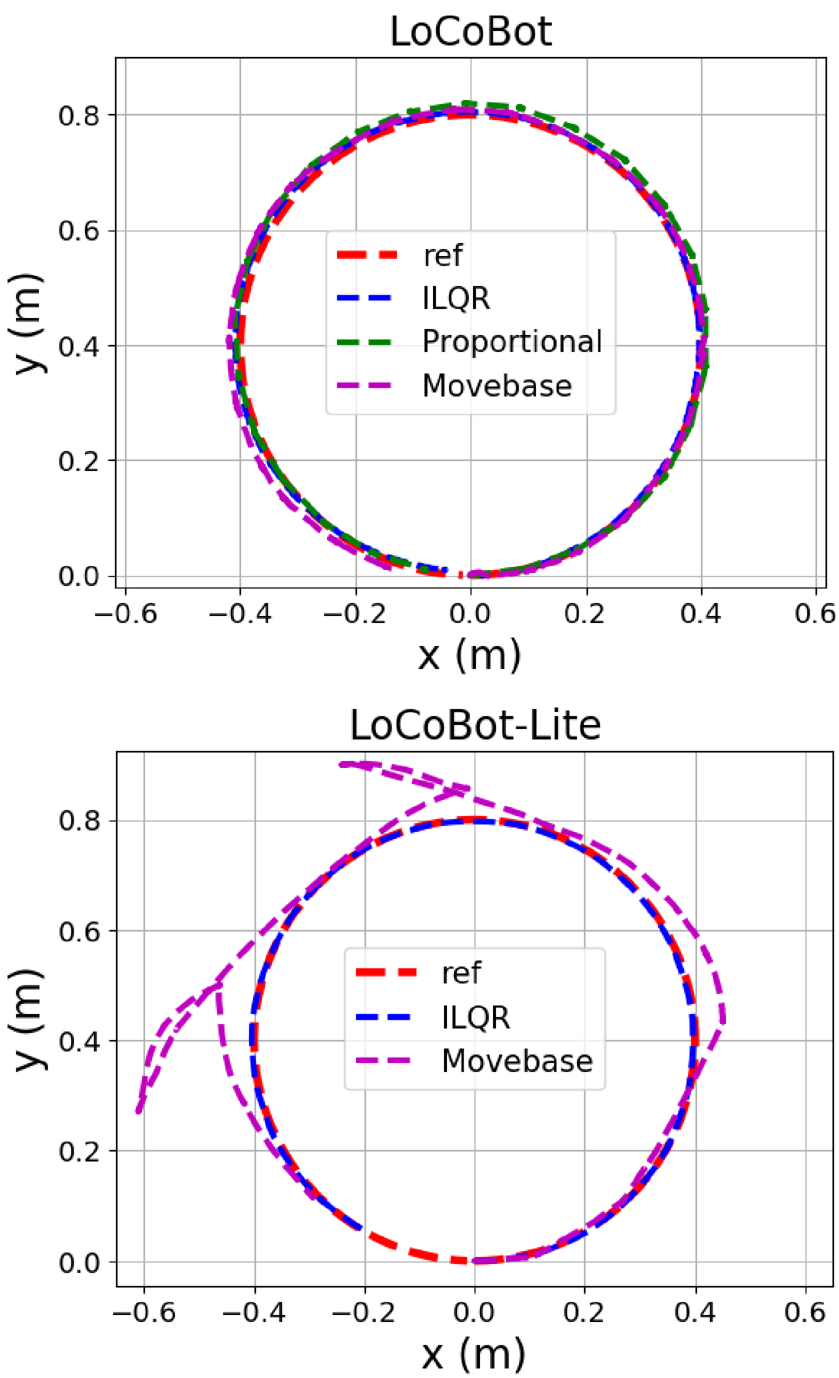

Figure 5.4: Qualitative comparisons for trajectory tacking for LoCoBot and LoCoBot-Lite. Reference trajectory (a circle of radius $0.4\,\mathrm{m}$) is shown in red.

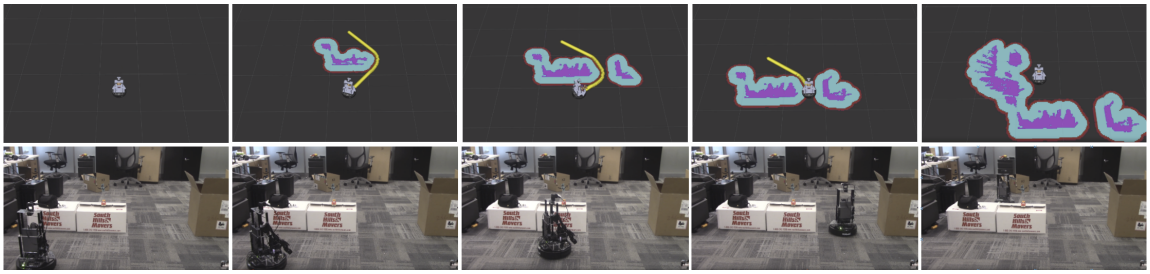

Figure 5.5: An example of Navigation via SLAM and Path Planning. First row corresponds to the 2-D map constructed using the on-board SLAM and the second row corresponds to the actual motion of the robot.

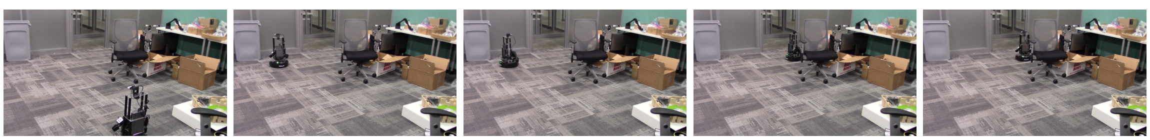

Figure 5.6: Snapshots from a run of visual navigation policy (CMP [101]) deployed on LoCoBot. See project website for videos.

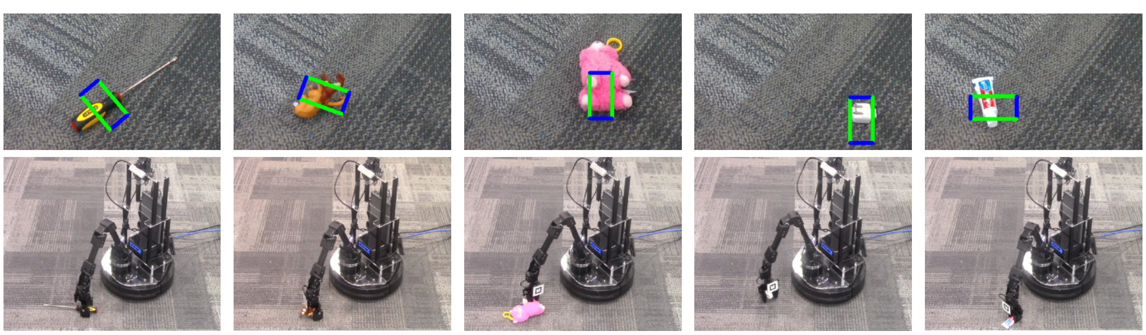

Figure 5.7: Grasps selected by the grasp model and execution by the robot.

Listing 5.3 Grasping example using PyRobot API.

```
from pyrobot import Robot

# Construct Robot.
bot = Robot('locobot')

# Set pregrasp and grasp height.
pregrasp_height = 0.2
grasp_height = 0.13

# Construct grasp model.
model = GraspModel()

# Move arm and camera to reset position.
reset_pos = [-1.5, 0.5, 0.3, -0.7, 0.]
bot.arm.set_joint_positions(reset_pos)
bot.camera.set_pan_tilt(0.0, 0.8)

# Get image.
rgb = bot.camera.get_rgb()

# Compute action.
grasp_img = model.compute_grasp(rgb)

# Convert grasp from Image space to
# robot workspace.
grasp_pose = cvt_space(grasp_img)

# Execute grasp.
# 1. Go to pre-grasp pose
pregrasp_position = [grasp_pose[0],
                     grasp_pose[1],
                     pregrasp_height]
grasp_angle = grasp_pose[2]
bot.arm.set_ee_pose_pitch_roll(
    position=pregrasp_position,
    pitch=np.pi / 2,
    roll=grasp_angle,
    plan=False,
    numerical=False)

# 2. Go to grasp pose.
grasp_position = [grasp_pose[0],
                  grasp_pose[1],
                  grasp_height]
bot.arm.set_ee_pose_pitch_roll(
    position=grasp_position,
    pitch=np.pi / 2,
    roll=grasp_angle,
    plan=False,
    numerical=False)

# 3. Grasp the object
bot.gripper.close()
```

Listing 5.4 Object pushing example using PyRobot API.

```
from pyrobot import Robot

# Construct Robot.
bot = Robot('locobot')

# Setup gripper, camera, arm.
bot.gripper.close()
bot.camera.set_pan_tilt(0, 0.7, wait=True)

# Move hand out of camera view.
ov_pos = [1.96, 0.52, -0.51, 1.67, 0.01]
bot.arm.set_joint_positions(ov_pos, plan=False)

# Get the point cloud(in base frame).
pts, colors = bot.camera.get_current_pcd(
                  in_cam=False)

# Compute push location, direction.
pre_push_pt, push_pt, obj_center = \
    get_push_direction(pts, colors)

# Move the gripper to pre-pushing pose
bot.arm.set_ee_pose_pitch_roll(
    position=pre_push_pt,
    pitch=np.pi / 2,
    roll=0,
    plan=False,
    numerical=False)

# Move the gripper vertically down.
down_disp = push_pt - pre_push_pt
bot.arm.move_ee_xyz(down_disp,
                    plan=False,
                    numerical=False)

# Move the gripper horizontally
# to push the object.
hor_disp = 2 * (obj_center - push_pt)
bot.arm.move_ee_xyz(hor_disp,
                    plan=False,
                    numerical=False)
```

Part III

Generalization with Robustness

Chapter 6

Tactile Re-grasping

6.1 Introduction

Consider the task of grasping a slippery glass bottle. We use vision to determine the object's location and its properties such as shape. Based on these estimates, we can even plan how to approach and make contact with the bottle. However, not until we get tactile feedback by touching, can we adjust our hands for a reliable grasp. In many cases, the hand completely occludes the object after contact, severely diminishing the use of hand-eye coordination; yet in all these cases we humans are invariably successful in grasping the objects. In fact, we are even capable of grasping objects solely based on touching. A good example is when we probe around on a nightstand for our phone. Haptics and the sense of touch plays a vital role in grasping. Yet, most of our currently existing grasping algorithms primarily builds on visual sensing (RGB-Depth or laser scanners). In fact, in the recent Amazon Picking Challenge, only one of 26 teams used a tactile sensor [54]. Can a robot learn to grasp solely based on touching and without even using vision? More importantly, can the robot incorporate both visual inputs and tactile feedback for robust grasping?

Sensory inputs affect the success of a grasp in all stages: *localization* of the object, *planning*[1] of the grasp control parameters (gripper pose, approach direction, etc.) and the *execution* of the grasp on the robot. Vision-based methods, such as object detection, segmentation and point cloud registration, are widely used for localization. Without using visual sensing, tactile exploration has demonstrated promising results on locating objects and estimating their 6 DOF poses [115, 119, 133, 188, 189, 213]. However, haptics has rarely been considered in the context of grasping beyond simple, individual objects. Recently, there has also been tremendous progress in data-driven grasp planning methods, namely in learning grasp policies from RGB-D images [142, 153, 190, 244]. But most of these approaches ignore haptic feedback during execution. In fact, tactile sensing has been

[1] Grasp planning refers to both analytic and data-driven techniques.

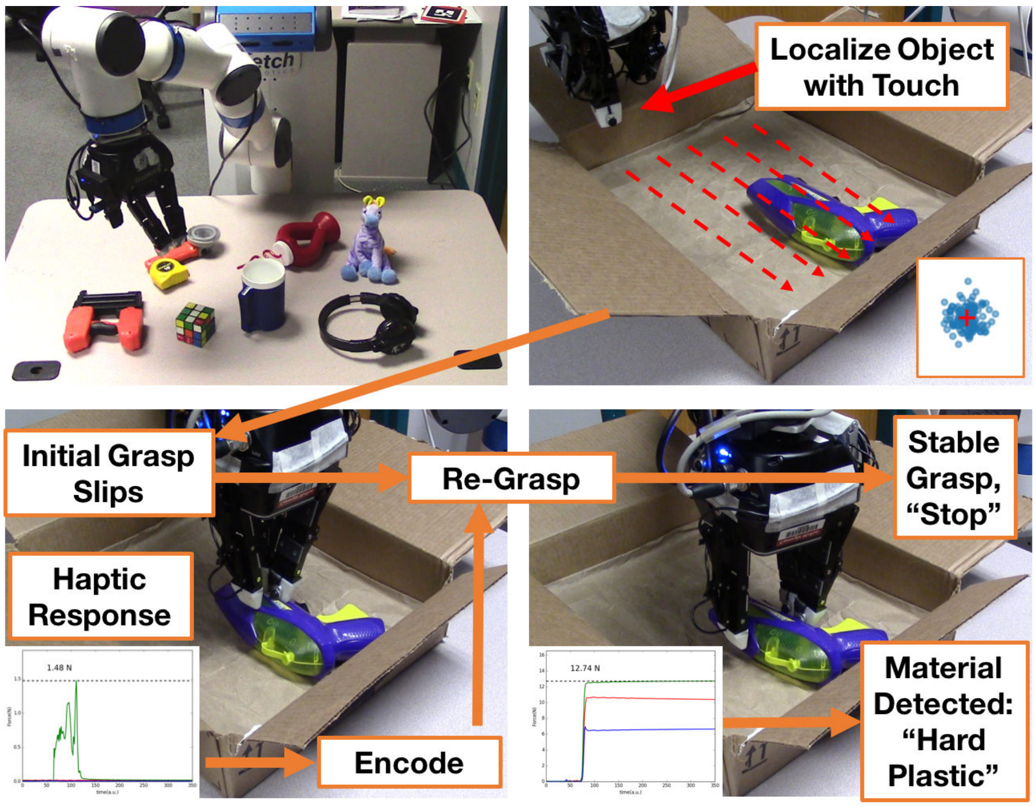

Figure 6.1: Our Fetch robot learns to localize and grasp a novel object of unknown shape from just tactile sensing. Our method estimates the target's location by touch-probing the workspace (top right), and establish an initial grasp (bottom left). We then learn to extract features from haptic feedback, and predict how to adjust the grasp (bottom right). This re-grasping process is repeated until our method identifies a stable grasp.

previously used for grasp execution, for instance in assessing grasp stability [27, 40, 60], and thus enabling the hand to adjust its posture and position online [59, 61, 111, 206]. Nonetheless, these methods assume either the initial grasp or the object information is inferred with vision, with few exceptions [83]. Felip et al. [83] presented a full system for tactile grasping using hand-crafted rules. In such light, no general learning framework exists for a complete grasp (localization, planning and execution) using solely touch sensors.

In this chapter, we present the first general framework for learning to grasp with only tactile sensing and without prior object knowledge. Our goal is to scale to a diverse set of unknown objects. To this end, we focus on 2D planar grasps of a single object. To start with, we design a *localization module* to obtain an approximate location of the object. Intuitively, we control the robot to sequentially "touch-scan" the grasp plane until hitting the object and we use a particle filter to aggregate the measurements and track the target location.

With all the uncertainty of object location, tactile sensing and kinematics, how can the robot reliably grasp the object? Our core idea is to treat grasping as a multi-step process with error recovery. Specifically, we propose a *re-grasping module* that refines the initial grasp with multiple re-trials. To extract rich meaningful features for the re-grasping task, we use a recurrent auto-encoder to learn an unsupervised representation from all the unlabelled data. These features are then fed to another neural network that simultaneously estimates grasp stability, and predicts the adjustment for the next grasp. Our framework will iterate on the grasps until our network estimates a high chance of success or the number of trials reaches a predefined limit.

Our high-capacity deep network requires a large-scale tactile dataset for training, which is missing in the community. We have thus created a new dataset of grasping with both tactile and visual sensing. Specifically, we record images, haptic measurements as the robot gripper encloses its fingers on an object, high-level re-grasp actions sent to the motion planner and labels of whether an object has been successfully grasped. Our publicly available dataset includes 7.8K interactions with 52 unique objects with material labels. We hope that it will serve as a major resource for future research on visio-haptic manipulation.

Our method is trained using our dataset, and tested on 20 unseen objects. We systematically vary components of our framework and benchmark the performance. First, we show that our unsupervised representation learning produces rich tactile features for a variety of passive (material recognition) and active (re-grasping) tasks. Next, we show that haptic based re-grasping improves a baseline policy, with the ground truth object location provided by vision-based localization. Finally, with touch based localization, our full method achieves a grasping accuracy of 40.0% using tactile sensing alone. We believe this is one of the first results of grasping a large set of unknown objects without seeing. Furthermore, we explore combining haptic and visual sensing for robust grasping. Our results indicate that our multi-step re-grasping with tactile feedback 1) improves the robustness of grasp execution and 2) offers an easy plug-in for existing grasp planning methods.

6.2 Related Work

Grasping is one of the fundamental problems in robotic manipulation and we refer readers to recent surveys [32, 33, 230].

Vision Based Grasping. Visual perception has been the primary modality for sensing, grasp planning and execution. Several work on model-based grasping make use of visual information like point clouds/images to estimate physical properties of objects (e.g., shape [166] or pose [51]), and finally to generate control commands for grasping. Sensing detailed physical properties from visual inputs can be exceedingly challenging, and might not be necessary for finding desired controls. Therefore, recent papers have focused on learning-based approaches [214, 244]. These methods directly map input visual data to the control signals for open-loop grasping. Recently, a lot of progress has been made in this direction by using deep models [140, 142, 153, 190]. However, using visual inputs alone leads to errors such as slippage due to low-friction or wrong grasp location due to self-occlusion.

Tactile Exploration. In contrast, humans make great use of tactile signals for grasping and can even grasp unknown objects without using visual sensing [118]. Therefore, recent work in robotics has also explored the use of haptics for sensing an object's shape, pose, location or attributes [49, 223, 256]. For example, if the location of an object is known, the shape can be estimated by actively touching its parts [163, 254]. Similarly, given the 3D models of objects, several recent work seek to infer the 6DOF pose of the objects with a series of information-gathering actions [115, 188, 213]. However, these results have neither been considered for the task of grasping nor can generalize to unknown objects. The most relevant work are from [189] and [119]. Pezzementi et al. [189] built occupancy grid mapping using tactile sensing of unknown 2D objects. Kaboli et al. [119] proposed a pre-touch strategy to localize novel objects in a 3D workspace. These work are similar to our touch localization step, yet they failed to complete the full pipeline of touch-based grasping.

Re-grasping with Tactile sensing. Haptic feedback is widely used for closed-loop control when executing a grasp, also known as re-grasping. Early work [82] focused on analytical solutions for 2D planar grasp given ideal tactile sensing of a known object shape. For real world tactile data, hand crafted rules can be highly effective if object shape is known [111]. Several recent works addressed the task of re-grasping or assessing grasp stability without prior object knowledge [27, 40, 44, 59, 73, 206]. However, they all rely on a good initial grasp given by another sensor modality. The most relevant work are from [61, 134] and [83]. Based on tactile feedback, Dang et al. [61] learned to predict grasp stability [60], which is further used to guide grasping. Their method can

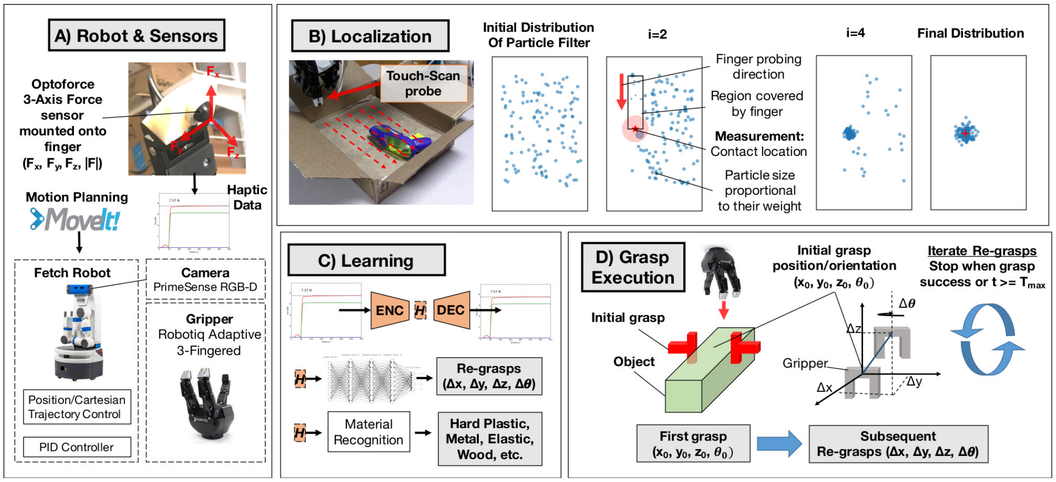

Figure 6.2: Overview of our system and approach. (a) Our robot and sensors: We equip a fetch robot with a Robotiq gripper and additional sensor packages. Our sensors include force sensor on the fingers of the gripper and RGB-D cameras on the head of the robot; (b) Our touch based object localization: We touch-probe a 2D grasp plane of the workspace, and use particle filtering to aggregate evidences of the object's location. An initial grasp is established given an estimate of the object's 2D location. (c) Our unsupervised learning scheme for haptic features: We learn to represent haptic data during grasping using an conditional auto-encoder. The learned features are fed into our re-grasping model to correct the initial grasp. (d) Our re-grasping model: Based on haptic features from current grasp, we estimate grasp stability and predict how to adjust the grasp. A new grasp is generated by applying the adjustment to the current grasp. This process repeats until our method predicts a stable grasp.

generalize to unknown objects but requires accurate object locations. Moreover, their approach only used simulated data with hand-designed features. Koval et al. [134] utilized haptic feedback to learn both pre and post contact push-grasping policies. Their method accounts for inaccurate sensing of object location and pose, yet is limited to objects with known shapes. Conversely, our method learns tactile based re-grasping policies with neither prior knowledge of the object (shape/physics) nor necessarily a good initial grasp. In addition, our approach makes use of large-scale real-world visual and haptic data to learn how to grasp. Moreover, Felip et al. [83] presented a full tactile grasping pipeline (exploration and re-grasping) with a wrist force-torque sensor, fingertip tactile sensors and a fully actuated multi-fingered gripper. They used a set of hand-crafted rules/features and demonstrated success on a small set of novel objects. Conversely, our tactile perception modules are learned from data and only uses the fingertip tactile sensors. We show that our learned model can be applied to successfully grasp a larger set of novel objects, including deformable and elastic ones.

Grasping Datasets. Alongside algorithmic developments, large-scale datasets have fueled the success of learning to grasp [140, 190]. However, when it comes to haptic datasets, there have been only few attempts such as [43, 77]. These datasets either focus on passive tasks e.g., material recognition [77], or are limited to grasping a small set of 2-3 objects with a small number of trials [42, 43]. As part of our effort, we created the first large-scale grasping dataset with both tactile and visual sensing to facilitate future research of visio-haptic grasping. As a result, our work is also deeply intertwined with the unsupervised learning of tactile feature representations. Previous work has primarily used hand-crafted features for haptic data [111]. Schneider et al. [216] constructed haptic features using bag-of-words. Madry et al. [149] explored unsupervised learning of haptic features using sparse coding. The learned representation has been shown effective for re-grasping [44], though it is intended for a specific class of sensors providing a matrix/image of tactile responses. We propose a novel method for learning haptic features using a deep recurrent network similar to [227].

6.3 Dataset

In this section, we present the effort on creating our visio-haptic dataset for grasping. Large-scale haptic dataset for grasping is important for learning high capacity deep models. Unfortunately, this kind of dataset is missing in the community. We seek to bridge this gap by collecting a new grasping dataset that includes both visual and haptic sensor data. Specifically, our dataset consists of 7800 grasp interactions with 52 different objects. Each grasp interaction lasts for 3.5-4 seconds and is recorded with:

- **RGB Frames:** We capture images of four specific events of a grasping: for the initial scene, before, during and after grasp execution. These images have a resolution of 1280x960.
- **Haptic Measurements:** Tactile signals are measured by force sensors mounted on each of the three fingers of the gripper. The sensor measures the magnitude (F) and the direction of forces (F_x, F_y, F_z) at 100Hz.
- **Grasping Actions and Labels:** We record the pose of all 2D planar grasps, including the initial grasp $(x_0, y_0, z_0, \theta_0)$ and subsequent re-grasps $(x_t, y_t, z_t, \theta_t)$. We also record whether the re-grasp succeeded.
- **Material Labels of Objects:** We label material categories (7) for each object, including metal, hard plastic, elastic plastic, stuffed fabric, wood, glass and ceramic.

Data Collection. To collect this dataset, we sample and execute a large set of grasps. The robot will lift up objects and automatically detect successful grasps. A major issue with this data collection process is how we can get more successful grasps. It is easy to collect failure cases by applying random grasps but it is difficult to collect successful grasps, which are critical for learning. To address this issue, we used an existing vision based grasping policy to sample an initial grasp from a pre-learned visual grasping policy [176]. We collect two sets of data and combine them to form our final dataset. The first set includes all 52 objects with 50-55 initial grasps. Each initial grasp is followed by a single random re-grasp. The grasps in this set have a higher rate of success. On the other hand, our second set contains a subset of 7 objects covering different types of materials. For each object in this set, we sample 80-100 initial grasps, and allow 2-3 random re-grasps, resulting in a higher failure rate.

Dataset Statistics. Overall, our dataset includes more than 30K RGB frames and over 2.8 million of tactile samples from 7800 grasp interactions of 52 objects. We provide grasping actions and labels for each interaction, as well as material labels for each object. To the best of our knowledge, this is by far the largest dataset for vision-haptic grasping. Our dataset is publicly available at: cs.cmu.edu/GraspingWithoutSeeing.

6.4 Overview

We present an overview of our framework in Fig 6.2. Our goal is to reliably grasp a target object using just fingertip tactile sensors and without knowing the location, pose or shape of the object. Similar to previous works, our framework has two main stages: grasp planning and grasp execution. For planning, we make use of particle filtering to localize an object based on a sequence of touch-probing. For grasp execution, we learn to iteratively adjust the grasp based on haptic feedback, using a deep neural network. Unlike other work in robot learning [140] which learn torque control, we infer position control commands and use a motion planner to reach that configuration. We also explore the benefit of applying our re-grasping model on top of a vision based grasping policy. Our methods for planning and execution are detailed in Section 6.5 and 6.6, respectively.

Platform. We implement our method on a real world robotic platform—a research edition of Fetch mobile manipulator [245], equipped with a 3-Finger adaptive gripper (Robotiq). We use ROS [198] and position control with the Expanding Space Tree (ESTk) motion planner from MoveIt to generate collision-free trajectories for the robot. For haptic sensing, we mount a 3-Axis Optoforce sensor onto each of the three Robotiq fingers. We made sure this mounting is rigid by using customized 3D-printed fixtures (see left panel of Fig 6.2). For vision, we use a PrimeSense Carmine 1.09 short-range RGB-D camera mounted on the robot's head. Note that visual data is not used in our method, except when we explore combining RGB frames from PrimeSense with haptic sensing for grasping.

6.5 Initial Grasp from Touching

We present our method for grasp planning. Traditionally, the goal of planning is to generate a good initial grasp of a target object. This usually requires the robot to sense the physical properties of the object, such as shape or pose. This is especially challenging with tactile sensing alone. Nevertheless, our key observations are that 1) we can infer a rough location of the object by probing the grasp plane and hitting the target multiple times; 2) even a poor initial grasp is often sufficient for successful grasping, if we allow the robot to correct the grasp a few times using haptic feedback. Thus, we propose a simple method for grasping. We first localize the object by touching, and then generate a random initial grasp. We will show that this method can be highly effective when combined with our learning based re-grasping policy.

6.5.1 Particle Filter for Touch Localization

The core of our grasp planning is a simple touch-based localization method using contact sensing. We consider the task of grasping a single target object within a known workspace–in our setting a constrained packaging box in which the object could be in any pose. In this case, we control the robot to line-scan a fixed 2D plane of the workspace using one of its fingers, which functions as a touch probe. The probe moves in a cartesian path until it detects a contact (defined by a threshold on the magnitude of force). Our method makes multiple contacts and uses particle filtering to infer the object's location $x \in R^2$ on the 2D plane.

The choice of a particle filter is tailored for our problem, as our contact measurement is highly non-linear and lacks analytic derivatives. Particle filters are a non-parametric formulation of the recursive Bayes filter:

$$bel(x_t) = \eta p(z_t|x_t, u_t) \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{6.1}$$

The belief $bel(x_t)$ is approximated using a finite set of particles $X_t = \{x_t^{[i]}\}_{i=1}^n \sim bel(x_t)$. x_t above denotes the target location at time t, u_t is the line scan action and z_t the contact sensing measurement. The touch-localization framework is summarized in Algorithm 2 and the detailed mechanisms of the particle filter could be found in [234]. At the end of touch-scanning, the centroid of the resampled $X_{N_{SCANS}}$ particles is returned as the final estimate of the target object's location.

Algorithm 2 Touch localization using Contact Sensing

$X_0 \leftarrow$ Uniform random samples
for t = $1{:}N_{scans}$ **do**
  $X_t \leftarrow \emptyset$
  Run linear scan u_t to get observation z_t
  for i = $1{:}N_{particles}$ *in* X_{t-1} **do**
    Sample from motion model $x_t^{[i]} \sim p(x_t|x_{t-1}^{[i]})$
    Update measurement $w_t^{[i]} \leftarrow p(z_t|x_t^{[i]}, u_t)$
    $X_t \leftarrow \{x_t^{[i]}\} \cup X_t$
  end
  $X_t \leftarrow$ Resample(X_t, w_t)
end
return: mean($X_t = \{x_t^{[i]}\}$) $\rightarrow$ object location

We present details of our measurement and motion models.

- **Motion model**: Touching the object might change its location. This displacement is usually small, yet is determined by how the robot moves (u_t), and the physical properties of the object and its environment. We simplify the motion model by assuming a Gaussian distribution independent of u_t: $p(x_t|x_{t-1}, u_t) = \mathcal{N}(x_{t-1}, \sigma^2 I)$, where σ is a small noise.
- **Measurement model**: Our measurement model tracks physical occupancy of probed locations. Any location on the 2D plane can be either free space (no contact) or occupied by the object (contact). We either increase (occupied) or decrease (free space) the weights of particles that lies within the vicinity (a sphere of radius 2.5cm for our experiments) of the location. An example is shown in Fig 6.2, where particles in swept area of the probe are down-weighted and particles near the contact point (red circle) are up-weighted.

Once we estimate the target location, our next step is to generate a grasp. Without prior object information, we select a grasp by randomly sampling from the rest of the parameter space. Executing such a grasp is highly likely to fail, as this sample can be far away from feasible grasps. Somewhat surprisingly, we will show that this random policy can produce a successful grasp, if we allow the robot to re-grasp a few times and adjust its controls each time based on tactile feedback.

6.6 Grasp Execution via Re-grasping

Given a noisy object location and a randomly selected grasp, how can the robot reliably grasp the object? To address this question, let us first look at what is measured by haptic sensors during grasping. Fig 6.3 shows haptic responses during the task of grasping. It is evident that these signals encode important information about the object in contact. For example, the magnitude of force

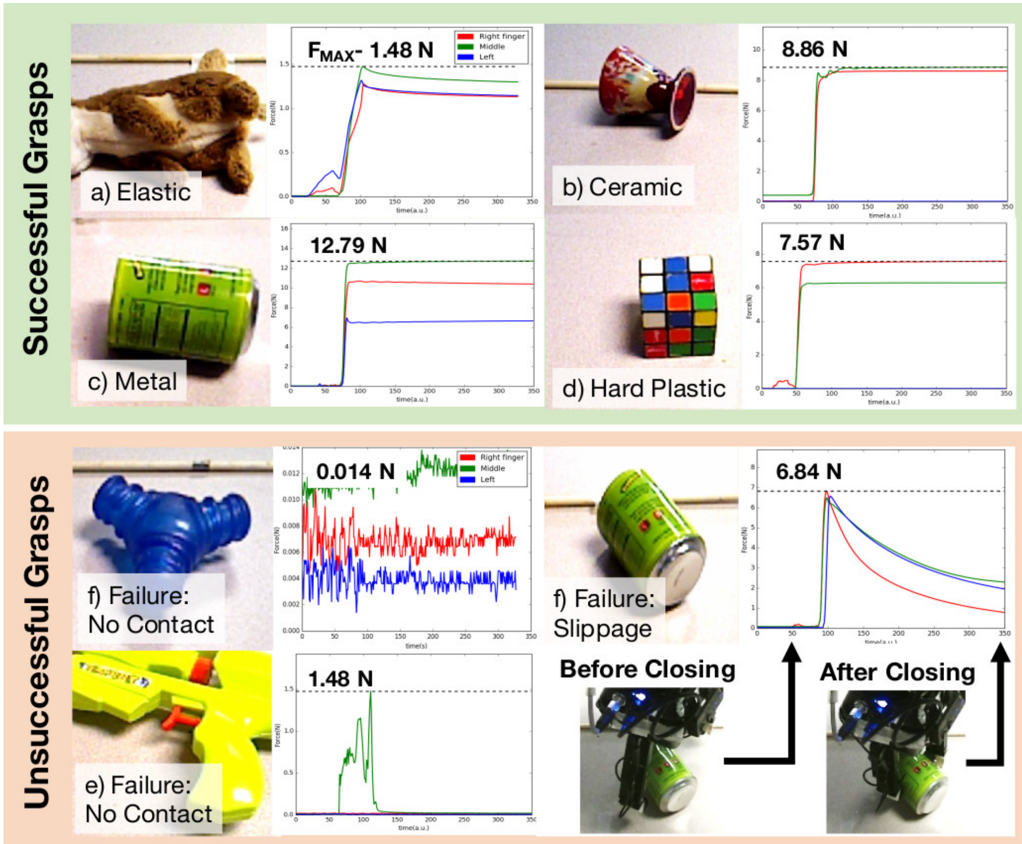

Figure 6.3: Tactile response from both successful and failed grasps. These grasps are from objects with varying shape/material/compliance properties. We plot the time series of force magnitude from our sensors on three fingers (red: right, green: middle, blue: left). The maximum force during grasping is also displayed. We record signals before and after the gripper closes (shown in bottom). These signals contain important information about the object (e.g., material, shape) and the grasping (e.g., grasp stability). And we explore using them to estimate how to correct a previous grasp.

implies the material of the object. And the temporal force variation across three fingers indicates the shape. These signals also capture critical aspects about the grasping. For example, we can predict the stability of the grasping by tracking the temporal structure of signals before and after contact. Therefore, we hypothesize that these tactile signals can be used to correct the initial grasp.

We will demonstrate that this is indeed possible if we consider grasping as a multi-stage process, and allow the robot to re-grasp a few times. Each new grasp is generated by adjusting a previous one using haptic feedback. Re-grasping thus helps to reduce the uncertainty of sensing. To this end, we propose a learning based approach for tactile based re-grasping. Our method learns representations from haptic data, estimate the grasp stability and predict the adjustment for next grasp, all using deep models. We now present our methods on haptic feature learning and tactile based re-grasping.

6.6.1 Learning Haptic Features

The next question is how do we learn a generalized representation of haptic data? Should we use hand-designed features or some task-specific representation? Raw tactile signals are in the form of a time series, with a low dimensional vector at each time step. Since they do not encode much global information compared to modalities like vision, it is challenging to consider haptic data without the context of the robot control applied. Therefore, what we need to learn is a conditional representation and to this end, we trained a conditional auto-encoder model over the haptic signals, shown in Fig 6.4. Both encoder and decoder in our model have a recurrent architecture (LSTMs [108]). Our encoder M_{ENC} takes a sequence of haptic data and control signals as inputs, and encodes them into a low dimensional latent space H. Our decoder M_{DEC} reconstructs the input haptic data from the latent space H.

By conditioning the reconstruction on control actions, the network must learn to embody the temporal structure of haptic data within the motion of the robot during grasping. This will allow us to re-use H to present haptic and control signals for re-grasping. Note that the learning is unsupervised in nature and does not require manual labeling.

More specifically, our haptic signals, denoted by $O = \{O_t\}$, include a 12D vector for each time step from all three fingers. Our control signals include the configuration of the gripper: $f = \{f_t\}$ and $m = \{m_t\}$. m_t is the mode of the adaptive gripper. m describes the angle between the fingers, and has categorical values of “pinch”, “normal” and “wide angle”. The under-actuated gripper fingers have three links each but only one DOF as f_t. f_t is valid when the gripper has been fully enclosed on the object. If no object was enclosed (grasp failure), f_t will take the maximum possible value. We use $L2$ loss and stochastic gradient descent for training. For feature extraction, we discard the decoder M_{DEC} and only use the encoder M_{ENC} to extract the hidden state H from a fixed size time window (3 seconds).

6.6.2 Learning to Re-grasp

We consider a multi-stage grasping problem, where each grasp is conditioned on the previous one. Formally, given a current grasp g, we measure the haptic data O and grasp configuration parameters (m, f) and encode them into $H = M_{ENC}(O, m, f)$. H is the hidden state that captures the haptic responses of the current grasp. Next, we learn the corrective action $\Delta g = \pi_{re-grasp}(H)$ that leads to better grasp stability and the architecture is shown in Fig 6.4. At the same time, we learn a score function $p = M_{stability}(H)$ to predict the grasp stability, which determines the empirical probability of grasp success. The score function $M_{stability}(H)$ is a simple feedforward networks with 5 fully connected layers of size $(512, 512, 256, 128, 64)$ and a final sigmoid function to estimate the probability. When testing, we iteratively apply the predicted Δg to current grasp g. We execute

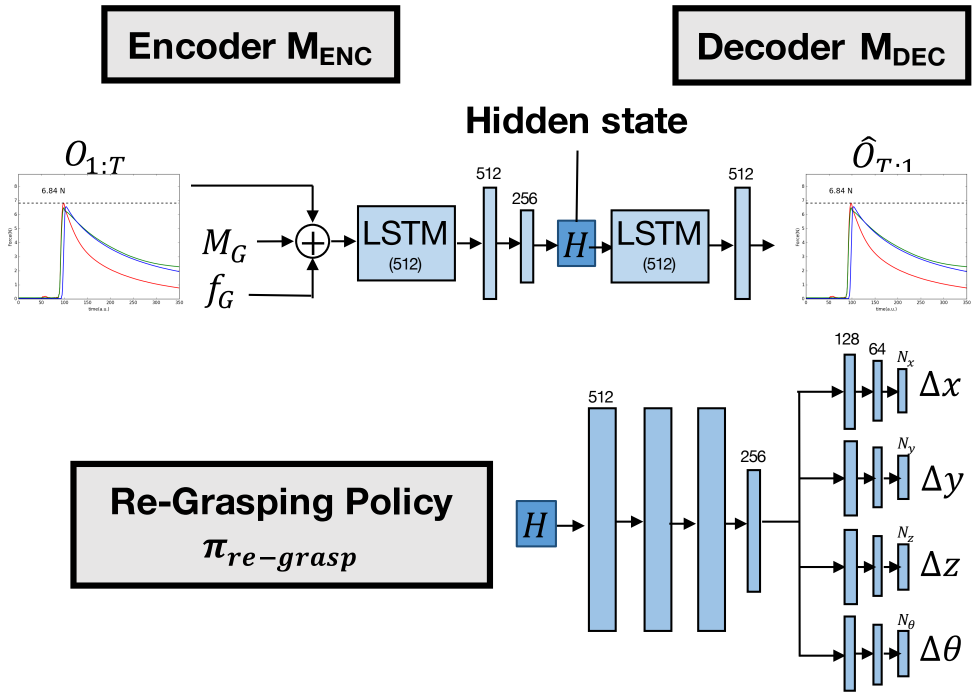

Figure 6.4: Network architectures for learning haptic features (top) and re-grasping policy (bottom). Our conditional auto-encoder M_{ENC}–M_{DEC} learns to reconstruct haptic data using both haptic signals and applied gripper control. We treat the learned latent space H as features for learning re-grasping policy $\pi_{re-grasp}$. Our re-grasping policy maps the hidden representation H to the adjustments of planar grasping parameters (Δx, Δy, Δz, $\Delta\theta$) (4D). These high level parameters are then executed using the motion planner to generate a new grasp.

$g + \Delta g$ until $M_{stability}$ predicts a high rate of success. Algorithm 3 summarizes our method.

Algorithm 3 Grasping Without Seeing

Localize object with vision/touch
Sample g_1 from π_{vision}/π_{random}
Execute g_1 on robot
Collect first haptic measurement O_1
for *i = 2:T_{max}* **do**
  Encode $H_{i-1} \leftarrow M_{ENC}(O_{i-1})$
  Compute p_{i-1} = $M_{stability}(H_{i-1})$
  if $p_{i-1} > p_{threshold}$ **then**
    break
  else
    Compute g_i = $\pi_{regrasps}(H_{i-1})$
    Execute g_i on robot
    Collect haptic measurement O_i
  end
end

Our output action Δg is parameterized by the change of the gripper's position (-0.025m $\leq$ (Δx, Δy, Δz) $\leq$ 0.025m) and orientation ($-\pi/4 \leq \Delta\theta \leq \pi/4$). Δg is thus a 4D vector. Given that the haptic measurement is only relevant in the local neighborhood of the current grasp, we constrain the range of these parameters to small adjustments tailored to our setting. During data collection, continuous values of the re-grasp (Δx, Δy, Δz, $\Delta\theta$) are sampled randomly. However, for the deep network we use a dicretized output space. Specifically, we discretize each control dimension into 5 bins. Thus, the learning of the policy function $\pi_{re-grasp}(H)$ is similar to multi-way binary classification.

$$L = \sum_{i=1}^{K} \sum_{k=1}^{B} \sum_{j=1}^{D(i)} \delta(k, u_{i,j}) \cdot \text{Cross-Entropy}(\sigma(y_{ij}^{final}), \hat{y}). \tag{6.2}$$

Eq 6.2 shows our loss for learning our policy function. $\hat{y}$ corresponds to the success/failure label while y_{ij}^{final} is the final dense layer before the sigmoid. $D(i) = 5$ gives the number of discretized bins for control parameter i, K (=4) is the number of control parameters, B is the batch size and σ is the sigmoid activation. $\delta(k, u_{i,j})$ is an indicator function and is equal to 1 when the control parameter i $u_{i,j}$ corresponding to bin j is applied. The learning rates for $\pi_{re-grasp}$, M_{ENC}/M_{DEC}, $M_{stability}$ are 5e-7, 1e-5 and 5e-5 respectively. All models are trained with ADAM optimizer [125] for around 20 epochs. The networks and optimization are implemented in TensorFlow [17] and Keras. Similarly, $M_{stability}$ is learned using a cross-entropy loss.

6.6.3 Improving Vision-Based Grasping with Re-grasping

Finally, for our experiments we also explore incorporating the haptic re-grasping module with vision based grasping. In practice, any vision-based policies could be used [142, 153, 190]. We adapt a variant of [176] (hereafter denoted as π_{vision}). π_{vision} is used to generate an initial grasp, followed by our re-grasping model. We also use this policy to collect our dataset. We sample control parameters from π_{vision} that are more likely to produce a success grasp to increase the number of successful grasps in our dataset.

Specifically, five control parameters are inferred from the object's image I_{obj} : x_{pixel}, y_{pixel}, θ, M_G, $h_G \sim \pi_{vision}(I_{obj})$. x_{pixel} and y_{pixel} are the 2-D grasp locations in image plane (converted to 3-D coordinates x_G and y_G with a calibrated depth camera). θ is the angle of the gripper about the vertical axis in a planar grasp (similar to [190]). M_G is the configuration of the gripper, which is also used for our learning of haptic features. And h_G is estimated height of the object from depth sensing. For both testing and data collection, we sampled $N_{patches} = 40$ parameters from π_{vision} and chose the command u_i for each control dimension i by $u_i = \text{argmax}_j \, \pi_{vision}(I_{obj}, u_{ij})$.

6.7 Experimental Evaluation

We now present our experimental results. Our experiments are divided into two parts. First, we evaluate the learned haptic features for two key tactile perception tasks of material recognition and grasp stability estimation. We compare against state-of-the-art haptic feature extraction methods, and benchmark the choice of classifiers. Second, we test our tactile based grasping framework. We report results for our re-grasping module, tactile-only grasping, and visio-haptic grasping.

Test Set for Grasping. To evaluate our grasping framework, we physically test grasping methods on a set of novel objects. We measure the grasp accuracy averaged over multiple trials per object as our evaluation criteria. This test setting is very challenging: testing objects are not presented in the training set and thus have not been seen by neither our $\pi_{re-grasp}$ model nor π_{vision}. Our testing set is divided into two parts, as shown in Fig 6.5. Each set consists of 10 different objects. Set A is more difficult than Set B, as it contains objects with more complex geometry, heterogeneous material distribution (e.g., plastic toy guns and stapler) and articulations. This test set is also used for grasp stability estimation.

6.7.1 Learning Haptic Features

Our first experiment tests our haptic feature learning scheme. Our decoder achieves a reconstruction error (L2 norm) of 0.81 and 1.1 on the training set and our held-out testing set (10% of the recorded data), respectively. This error (around 1 Newton of force) is reasonable when compared to ~ 0.2 Newton sensing noise from our force sensor. To further evaluate the learned haptic features, we consider two key tasks in tactile perceptions: (1) material recognition; and (2) grasp stability estimation. And we consider different combinations of haptic features and classifiers for both tasks.

Figure 6.5: Our test set of objects. These objects were not in the training data. We divide our test set into two parts. Set A contains slightly harder objects to grasp (such as the red and orange toy guns) compared to Set B.

Tactile Features. We compare our learned haptic features with two other baselines representation learning methods.

- **Auto-encoder.** This is our haptic features learned using a unsupervised recurrent auto-encoder. Once learned, only the encoder is used to extract features.
- **Sparse Coding.** This is a variant [204] of ST-HMP features [149]. These features are learned using dictionary learning and sparse coding on the spectrogram of 1D time series of tactile signals. Note that directly using ST-HMP is not feasible for us, as it requires 2D tactile images.
- **Hand Crafted.** This is from [223], where raw signals from three specific events (before contact, when the finger closing movement is stalled due to object-finger contact, after the fingers are in equilibrium) are extracted.

Choice of Classifiers. We further vary the classifiers used for both material recognition and grasp stability estimation.

- **Deep Network.** We train a five-layer neural network with cross entropy loss for classification.
- **SVM.** This is a linear classifier trained with hinge loss.

Material Recognition: The task is to classify 7 different materials in our dataset using tactile signals during grasping. All features are learned from the full training set, as no supervision is required. Our classifiers are trained on a subset of the training set (80%) and tested on the held out testing set (the remaining 20%). We report average class accuracy. The results are presented in Table 6.1.

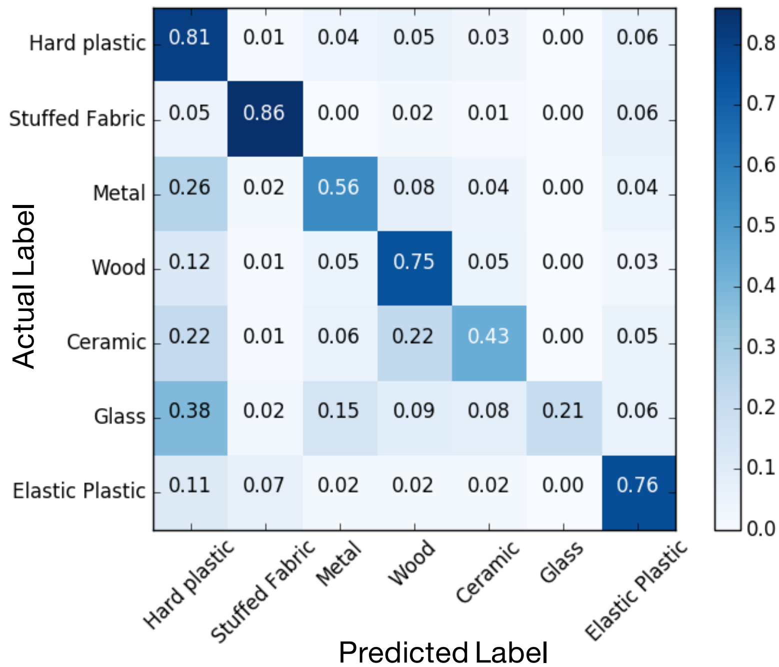

Figure 6.6: Confusion matrix for material recognition on a held out test set. Using our learned haptic features, we achieve an accuracy of 42.86%.

Table 6.1: Results of Material Recognition

Feature Type	Accuracy (%)	
	Deep Network	SVM
Auto-encoder (Ours)	**42.86**	40.68
Sparse Coding [149, 204]	36.35	35.93
Hand Crafted [223]	33.50	33.66

The features learned from our auto-encoder outperforms sparse coding and hand crafted features for both the deep network and SVM by a significant margin (at least 4.7%). The feed-forward network also performs at least comparably or slightly better than the SVM for all features. In particular, our haptic features with deep networks improves the traditional method of sparse coding with SVM by 5.8%. Furthermore, we show the confusion matrix for material recognition in Fig 6.6. The majority of the error comes from hard objects that are composed of wood/metal/glass being mis-classified as hard plastic. This result demonstrates that our haptic features encode physical properties of the object.

Grasp Stability Estimation: The task is to estimate whether the grasp will be successful given tactile signals during grasping. Again, all features are learned from the training set. We train the classifiers on the training set and apply them on our full test set (580 trials on 20 unseen objects). We report the accuracy for binary classification. The results are summarized in Table 6.2.

Table 6.2: Results of Grasp Stability Estimation

Feature Type	Accuracy (%)	
	Deep Network	SVM
Auto-encoder (Ours)	**85.92**	84.50
Sparse Coding [149, 204]	81.37	80.12
Hand Crafted [223]	82.54	82.66

The results of grasp stability follow the same trend of material recognition. Our haptic features significantly outperform other features. And the combination of our learned haptic features with deep network achieves the best accuracy. This result suggest that the learned haptic features contains important information for grasping. To better understand our tactile features for grasping, we visualize the t-SNE embedding of the learned features and plot example results of our grasp stability estimation in Fig 6.7. We observe that the main failure modes are from that (1) the part of the finger containing the haptic sensor may not come into contact with the object; and (2) the object may slip in the gripper.

Remarks: We demonstrate that our learned features are highly effective for two key tactile perception tasks. When compared to other haptic features, our feature learning can substantially improve the performance. We also show that deep networks are on average better than classical linear SVM with all haptic features. These results provide a strong support to our design of the re-grasping model, i.e. the combination of our learned haptic features and deep networks.

6.7.2 Tactile Based Grasping

Our second experiment focuses on the tactile based grasping framework. We first evaluate our core touch based re-grasping model. We then benchmark the full pipeline, and explore incorporating vision based grasping with our re-grasping model.

Re-grasping Model: We evaluate our core re-grasping model using the full test set (20 objects). Note in this case, we assume an oracle object location is given: we place each object in eight canonical orientations (N,S,W,E,NE,SE,SW and NW). Moreover, the initial grasp is randomly selected given the object location. We then compare three different settings: a single random re-grasp, multiple random re-grasps, and our re-grasping model. For fair comparison, we set the number of trials for random re-grasps equal to the maximum number of trials of our model. Both random re-grasp and our model are based on our grasp stability estimation.

Table 6.3: Re-grasping results with oracle object locations

Object Location	Initial Grasp	Re-grasp	Grasp Accuracy (%)		
			Set A	Set B	A+B
Vision	Random	-	16.3	32.5	24.4
Vision	Random	Random ($\leq$4 trials)	17.5	28.8	23.2
Vision	Random	Ours ($\leq$4 trials)	**33.8**	**41.3**	**37.5**

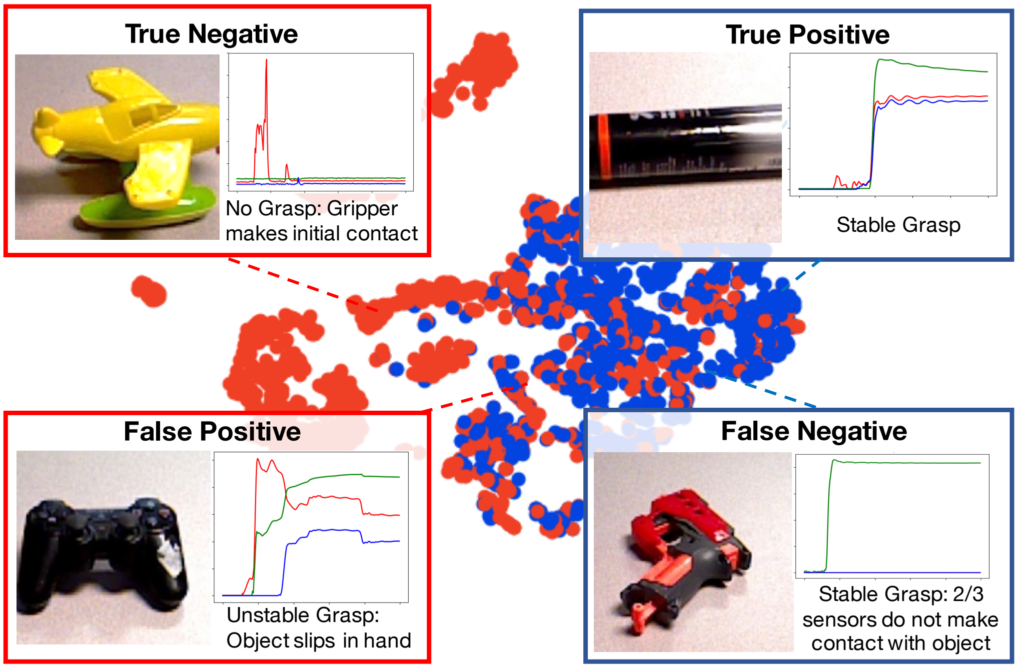

Figure 6.7: Visualization of learned haptic features using t-SNE Embedding. Red and blue dots correspond to failed and successful grasps respectively. We also plot four typical examples for grasp stability estimation.

The results are shown in Table 6.3. Grasp accuracy on Set B is always higher than Set A. For the full set, the baseline accuracy for chance grasping is 24.4%, where the first (and only) grasp is sampled from a random policy with no re-grasping. Interestingly, multiple random re-grasps slightly decreased the accuracy by 1.2%. And our re-grasping model get the best accuracy of 37.5%. This is 13.1% better than the baseline of multiple random re-grasps. This result demonstrates the effectiveness of our re-grasping module.

Grasping without Seeing: Going beyond re-grasping, we test our full pipeline of tactile based grasping, which includes touch based localization and re-grasping. In this case, we simplify our benchmark by only considering our test set B and use 5 trials per object. This is primarily limited by the run time of our experiments. Our results are show in Table 6.4. Our pipeline increases the baseline of random grasping by 14% and reaches an accuracy of 40% with only tactile sensing. This is one of the first results for a complete grasping of multiple novel objects using only the sense of touch.

Table 6.4: Grasping accuracy of our full method. We also present results of combining our re-grasping module with a vision based policy to further improve grasping.

Object Location	Initial Grasp	Re-grasp	Grasp Accuracy (% on Set B)
Touch	Random	-	26.0
Touch	Random	Ours ($\leq$4 trials)	**40.0**
Vision	Vision	-	51.3
Vision	Vision	Ours ($\leq$4 trials)	**61.9**

Visio-Haptic Grasping: Our last experiment combines the proposed re-grasping model with a vision based policy from [176]. The results are show in Table 6.4. Our framework can further benefit from a good initial grasp (+11.3%). And more importantly, combining vision based grasping with our tactile based re-grasping can largely improve the accuracy by 10.6%. These results provide a strong evidence for the need of combining visual and tactile sensing for robust grasping. Through this experiment, we also shows the flexibility of our re-grasping model, which can be readily plugged into existing grasp planning methods.

6.8 Conclusion

In this chapter, we demonstrate one of the first attempts of learning to grasp novel objects using only tactile sensing and without prior knowledge about the object. The core of our method lies in the combination of a) a simple method of touch based localization b) unsupervised learning of rich tactile features and c) a learning based method for re-grasping using haptic feedback. First, we created a large-scale dataset for visio-haptic grasping to evaluate our method and to facilitate future research. With this dataset, we used a auto-encoder to learn rich features from raw tactile signals. These features proved effective for both passive tasks like material recognition and active tasks like re-grasping, and displayed an improvement of around 4-9% over prior methods. Finally, we show that our novel re-grasping model can progressively improve the grasping, leading to significantly higher success rate even from a noisy initial grasp. Our method achieved a grasping accuracy of 40.0% using only tactile sensing for both localization and grasping. We also demonstrate that this re-grasping model can be combined with existing vision based grasping to further improve the accuracy by about 10%. We hope that our method together with our dataset could provide valuable insights for solving the challenging problem of autonomous grasping.

Our current method is limited in the sense that re-grasping has to start from a random initial grasp, which is far from optimal. Looking forward, tactile exploration could be used to build a representation of object shape (e.g., Gaussian Process Implicit Surfaces) followed by grasp planning [151]. Also, the major failure mode with our current hardware setup is one of partial observability - the regions of the robot's finger not covered by the sensor might come in contact and push the object. This in turns affects all stages of our pipeline - from feature learning, localization, grasp stability estimation to re-grasping. This could be mitigated by using novel skin/contact sensors and wrist force-torque sensors alongside incidental contact algorithms [31]. Furthermore, instead of adding symmetric Gaussian noise in the motion model of the particle filter, we can bias the model in the direction of the detected contact force. Finally, a joint learning of localization and re-grasping with reinforcement learning is interesting to explore. Staged learning or policy iteration on the learned policy would greatly improve its performance as in prior work [142, 176, 190].

Part IV

Generalization to Semantic Tasks

Chapter 7

Data and Semantic Knowledge for Task-Oriented Grasping

7.1 Introduction

We have seen tremendous progress in the fundamental task of robotic grasping in recent years. State-of-the-art grasping algorithms have shown generalization to object instances [120, 153, 190, 257], viewpoints [141], DOF constraints [173, 179, 232], unknown environments [99] and even adversarial objects [242]. The key reason for the success of these approaches is large-scale learning. Typically data is sampled from analytical approaches in simulation [153, 173] or using a self-supervised framework [141, 190]. Despite these recent successes, there is still a significant gap between how humans grasp objects and how robots perform picking. Most techniques plan for stable grasps assuming grasping to be the end goal. However, when humans grasp an object, we do so with a particular purpose in mind and grasping is just the first step as a means to that end. For example, when humans grasp a cup, we use the handle to drink from it though several other stable grasps exist. Humans also use objects creatively, such as scooping with a bowl or hammering with a heavy mug. Different tasks may require completely different grasps for the same object. To effectively operate in human homes and complete multiple tasks, a personal robot would have to learn from humans to generalize grasping to several tasks and skills beyond a tool's prototypical use. For instance, if the robot is cooking and needs to stir a pot of pasta but doesn't have a spoon at hand, it can use an alternate tool, such as a knife. To truly get to human-level grasping, we must study not just stable grasping or grasping for an object's primary use-case but rather how to grasp depending on both the task and the object.

What are the bottlenecks in task-oriented robotic grasping? The biggest hurdle is the need for human-labeled data. Unlike self-supervised or analytical approaches for which force sensing or contact models can provide labels for stable grasps, here we need humans to identify how an object can be grasped for multiple tasks. There has been a lot of recent work in this area, including [37, 81, 146]. Brahmbhatt et al. [37] used thermal imaging in a curated setup to study human grasping contacts on 50 3D printed objects for two tasks. Fang et al. [81] proposed to jointly learn a task-oriented grasping network and manipulation policy in simulation with reinforcement learning and demonstrated the framework on two-goal tasks with two object categories. Liu et al. [146] proposed a data-driven approach to learning the complex relationships between grasps, objects, tasks, and broadened semantic contexts. However, their approach required pixel-wise affordance segmentation [69] for a small set of known object categories, which is challenging to generalize and get supervision for. Despite this progress in learning from human grasping, there are still significant gaps, both from a data and methods perspective. On the data side, existing datasets are limited in terms of the number of object instances, but especially in the number of tasks and object classes collected. Yet, even if we scale the datasets, it is unclear if current approaches will generalize to new object categories and tasks in the real world. We tackle both problems: first, we collect a dataset that is diverse both in terms of objects and tasks and an order of magnitude larger than previous datasets. Second, we exploit the semantic knowledge of objects and tasks to present a system that can generalize to new object instances, classes, and new tasks. To the best of our knowledge, this work is one of the first efforts in demonstrating robust generalization in task-oriented grasping, especially with semantic knowledge.

More specifically, our first key contribution of this work is the collection of a large-scale dataset which we call TaskGrasp. We increase the number of real objects from the current best of 50 in prior works [37] to 191, and collect RGB-D point cloud observations and object-centric 6-DOF grasps for the task-oriented grasping problem. We also scale the number of object classes from 40 [37] to 75 and resolve each of these to the standard WordNet ontology [167]. And perhaps most importantly, we scale the number of tasks from $2-7$ in prior works [37, 81, 146] to 56. This expanded dataset both gives a better benchmark for task-oriented grasping and allows us to study generalization by expanding the number of object categories and tasks.

In order to generalize to a new object or task, we need to have some prior semantics about it. For instance, if we knew that mugs and bowls were both containers, we might infer that we should apply the scoop action in a similar way. To this end, and for our second main contribution, we propose a method, called GCNGrasp, that incorporates semantic knowledge into the end-to-end learning of task-oriented grasping from object point clouds. In particular, we use a Graph Convolutional Network (GCN) [127] to reason about a knowledge graph that encodes relations between objects and tasks, and further leverage word embeddings trained on large-scale language tasks to provide additional prior information. Our GCNGrasp model shows a significant improvement of 12% and

3.5% on held-out tasks and object categories, respectively, compared to baselines which do not incorporate semantics. We also show that our method and dataset are applicable for actual robots by executing task-oriented stable grasps on a 7-DOF Sawyer Robot on unknown objects.

7.2 Related Work

Task-Oriented Grasping: Prior work in Task-Oriented Grasping can be grouped into analytic methods, data-driven approaches using object state information, and frameworks learning from observations. Early work in analytic grasping proposed task wrench spaces with task-oriented grasp quality metrics [34]. Data-driven approaches have been proposed to improve generalization, though a large body of work has relied on object state information. Song et al. [221] used generative Bayesian Networks to model the relations between objects, grasps and tasks; Antanas et al. [23] and Ardón et al. [24] leveraged probabilistic logic languages to reason about grasp regions affording different tasks through semantic relations. However, both methods require grounding geometric information about objects to semantic representations and can only reason about semantic knowledge alone. A related line of work has used object parts and affordance detection [67, 69, 131, 138]. Do et al. [69] leveraged the affordances of object parts to define the correspondences between affordances and grasp types (e.g., rim grasp for parts with contain or scoop affordance). Detry et al. [67] trained a separate affordance detection model using synthetic data to detect suitable grasp regions for each task. While we do not provide explicit supervision for object affordance, we demonstrate that our model achieves an implicit understanding.

More recent works have learned task-oriented grasping from just RGB-D observations of objects. Dang and Allen [58] proposed an example-based approach which learns task-oriented grasps by storing visual and tactile data of grasps. Hjelm et al. [107] proposed a discriminative model based on visual features of objects. Jang et al. [114] proposed an end-to-end learning method of grasping objects from specific categories in a bin. To accelerate learning from observations, there have been efforts in scaling datasets as discussed previously [37, 81, 146]. The computer vision community has also focused on annotating datasets for inferring human grasp pose estimation from visual data [112, 132, 218] with the aim that it could be adapted to robotic grasping with kinematic retargeting. In this work, we propose an expanded dataset in terms of the number of object categories and tasks to study generalization. We also present a unified framework that jointly learns from semantic knowledge and geometric observations.

Semantic Knowledge in Vision: The use of knowledge and knowledge graphs for visual reasoning has been well studied. Word embeddings from language has been used extensively [89]. Class hierarchies, such as WordNet [167], have often been used to aid in image recognition [259]. More generally, knowledge graphs have found extensive use in visual classification and detection [161], as well as zero-shot classification [243]. We draw on many of the ideas from these works in Computer Vision, especially those related to word embeddings and graphs, and apply them to a robotics task and to 3D point cloud data.

Semantic Knowledge in Robotics: In robotics, semantic knowledge has been used to help robots adapt to diverse and changing environments by providing abstractions that generalize across similar situations. Large-scale robotic knowledge bases, such as KnowRob [233], RoboBrain [215], and RoboCSE [63], aimed to provide robots with extensive knowledge about objects, spaces, tasks, actions, and agents. Other methods leveraged more specific knowledge in a variety of robotic tasks, such as affordance learning [170] and visual-semantic navigation [252]. Similar to Antanas et al. [23] and Ardón et al. [24], we reason about semantic knowledge for task-oriented grasping, but we leverage semantic knowledge for generalization to novel object classes and tasks.

7.3 Dataset

In this section we describe our dataset: TaskGrasp, specifically its properties, collection and annotation methodology. As shown in Table 7.1, TaskGrasp is the largest and most diverse dataset for task-oriented grasping to date with respect to number of objects, categories and tasks.

TaskGrasp contains 191 individual household and kitchen objects comprising 75 distinct object categories and varying in size, geometry, material, and visual appearance. Figure 7.2 shows the class of each object and its proportion in the dataset. We collect RGB-D pointclouds for each object, and automatically annotate $250K$ stable grasps. We also curate a list of 56 everyday tasks that impose different semantic constraints on grasping and annotate for each grasp whether that grasp is appropriate for each particular task.

7.3.1 Data Acquisition on a Robot

After selecting our 191 objects by browsing various homegoods stores, we scan the objects to acquire their point clouds. A Realsense D415 eye-in-hand camera mounted on a LoCoBot [177] is used for 3D scanning. The object is placed on a transparent mount in front of the robot, which is commanded to different poses along the object approach direction to capture point clouds from multiple viewpoints. This setup helps to capture more of the object geometry under self-occlusion, which in turn increases the coverage of grasp samples. The multi-view observations are registered using robot kinematics and further refined with the iterative closest point algorithm. After table plane segmentation, 600 object-centric stable grasps are then sampled [231] from the object point cloud. 25 grasps are selected with farthest point sampling (to maximize grasp coverage) for annotation. These grasps are chosen as a representative, albeit limited, grasp set for the object to trade off between dataset size and budget.

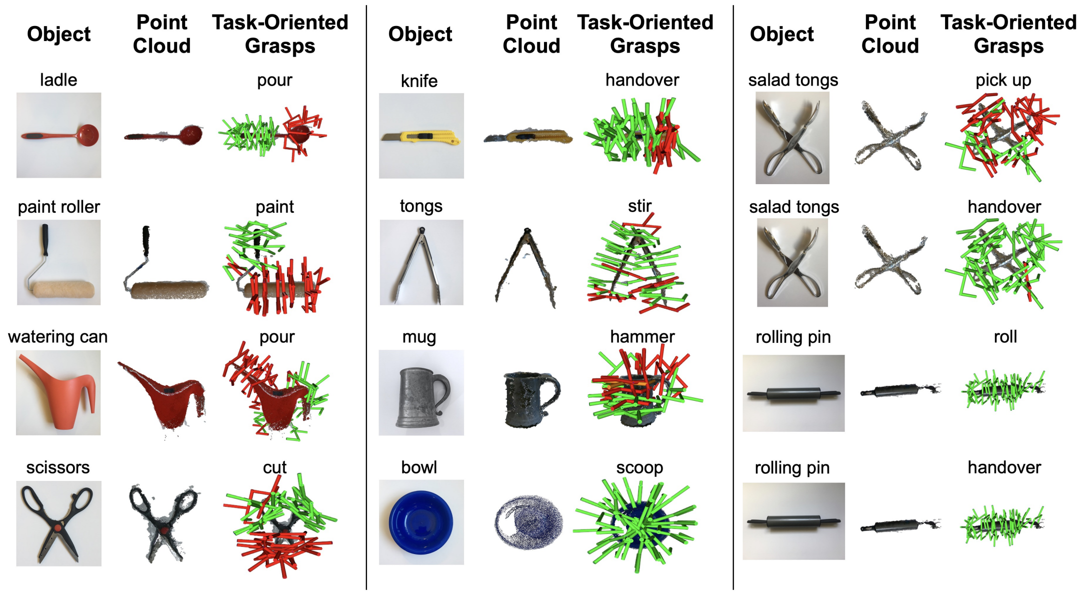

Figure 7.1: Example point clouds and grasps from our TaskGrasp dataset. Column 7-9 shows how grasps vary with tasks for a salad tongs (with higher diversity) and a rolling pin (with lower diversity). Green and Red means successful and incorrect task-oriented grasps respectively.

Table 7.1: Comparing recent Task-Oriented Grasping Datasets

	ContactDB [37]	SG14000 [146]	TOG-Net [81]	TaskGrasp (Ours)
Semantic Knowledge	✗	✗	✗	✓
Object Categories	40	5	2	75
Objects	50	44	18K (synthetic)	191
Tasks	2	7	2	56
Grasps	3750	14K	1.5M	250K
Grasp Type	Contact Map	*SE*(3)	Planar	*SE*(3)

7.3.2 Data Annotation by Crowdsourcing

We use Amazon Mechanical Turk (AMT) to crowdsource labels for the 250K stable grasps. Instead of exhaustively labelling each task-object combination ($\sim 10K$) , we reduce the annotation cost with a two-stage procedure. We use the insight that the pre-condition for a task-oriented grasp is that the object has to be capable of the task in the first place. First, we gather labels for whether a task is suitable for each object. Second, for this filtered subset of task-object combinations, we collect labels for the 25 task-oriented grasps per object. To ensure annotation quality, we assign each labeling task to three annotators and use gold standard questions (questions that we know the answers to) to filter annotators with low accuracy. For both stages, we take a majority vote between the annotators. We measure agreement with Randolph's free-marginal multirater kappa [199]. Kappa values for the two stages are 0.65 and 0.62 respectively (0.0 meaning agreement equal to chance, and 1.0 indicating perfect agreement above chance), which suggests good agreement between annotators.

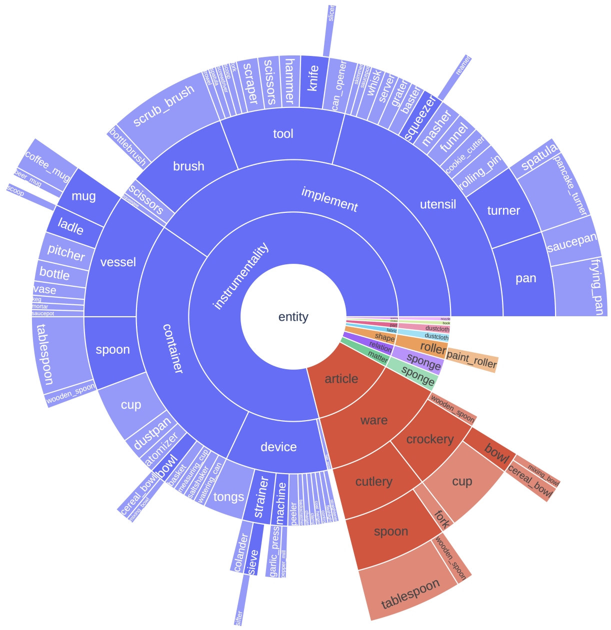

Figure 7.2: Semantic hierarchy of objects. Each level of the hierarchy is represented by one ring with the innermost circle as the root of the hierarchy. The angle of each segment is proportional to the number of objects.

7.3.3 Analysis

In Figure 7.1 we show prototypical examples from TaskGrasp. We provide additional examples in the supplementary materials.

Diversity of Grasps: As a result of the large number of objects and tasks, TaskGrasp contains a wide variety of task-oriented grasps. On average, each object is suitable for 7 tasks. As shown in Figure 7.1, these tasks involve both prototypical (a ladle for pouring) and creative use of objects (tongs for stirring), imposing drastically different semantic constraints on grasping. These examples also demonstrate the complex geometries presented in real world objects, which pose another challenge for generalization. We list all the tasks suitable for each object category in Table 7.5.

We also quantitatively measure grasp diversity by analyzing the effect of tasks on grasps. Since different tasks provide different labels for the same set of stable grasps on each object, we compute Randolph's kappa [199] on these labels as a measure of agreement between tasks, i.e., how likely grasps for one task (e.g., stir) agree with grasps for another task (e.g., cut). Ranging from 0.19 to 0.93, kappa values of the objects suggest that the effect of tasks vary greatly for different objects. Column 7-9 in Figure 7.1 show how grasps vary with tasks for a salad tongs with a kappa value of 0.38 and a rolling pin with kappa value of 0.97. In TaskGrasp, 25% of the objects have kappa values lower than 0.5 and these objects require significantly different grasps for different tasks.

Semantic Knowledge of Objects and Tasks: We also provide semantic knowledge about objects and tasks in the dataset. Objects are manually mapped to WordNet synsets [167] which represent a semantic hierarchy, as shown in Figure 7.2. Each of the 75 leaf synsets in the hierarchy represents a distinct object class and is linked to 2.5 objects on average. Building on the hypernym paths from WordNet, the semantic hierarchy includes a rich set of object concepts interlinked by "Is-A" relations. This provides useful semantic knowledge for task-oriented grasping as objects in the same subtree of the hierarchy often share similar functionalities or geometric properties. For example, mug, ladle, and bottle are in the vessel subtree and can all be used to hold liquid. In addition, we connect a task to an object class through "Used-For" relations if any object in the class is considered suitable for the task from the first stage of our crowdsourcing. We provide a thorough breakdown of object counts, class hierarchies and used-for relations in the supplementary materials.

7.4 Task-Oriented Grasping with Semantic Knowledge

We consider the problem of generating grasps for task-oriented grasping given the object point cloud and task constraints. Specifically, we want to estimate the grasp distribution $P(G^*|X, \mathcal{T})$, where X is the point cloud input, $\mathcal{T}$ are the constraints imposed by goal tasks, and G^* is the space of successful grasps. Following convention in related work [173, 232], we represent grasps $g \in G^*$ as the grasp pose $(R, T) \in SE(3)$ of a parallel-jaw gripper with its fingers open which when closing will lead to a stable grasp. We further factorize the estimation of $P(G^*|X, \mathcal{T})$ into 1) task-agnostic grasp sampling $P(G^*|X)$ and 2) task-oriented grasp evaluation $P(S|X, \mathcal{T}, g)$. The primary benefit of this factorization is that it allows us to leverage prior work in stable grasp generation.

In this section, we describe our method (GCNGrasp) for Task-Oriented grasping. Our method is composed of: (1) a Shape Encoder built on a PointNet++ architecture [197] to encode the object point cloud, (2) a Graph Convolutional Network [127] which takes the encoded shape as input as well as a knowledge graph $\mathcal{G}$ encoding the semantic relationships between object categories, tasks and hierarchies and (3) a Grasp Evaluator which outputs the final grasp prediction. See Figure 7.3.

Grasp and Object Shape Encoder: Our input observations are object point clouds and we want to reason about $SE(3)$ grasps. Qi et al. [197] proposed the PointNet++ architecture to efficiently represent 3D data which we use to learn a representation for the object point cloud and 6-DOF grasp poses. The grasp g is defined in the object frame and six control points are selected on the gripper to form a gripper point cloud X_g. Similar to Mousavian et al. [173], X_g is concatenated with the object point cloud X with an extra latent indicator vector to represent whether a point is part of the gripper or the object. The PointNet layer reasons about the relative spatial information between the grasp and the object and outputs a geometric embedding vector.

Graph Convolutional Network: We use the standard Graph Convolutional Network (GCN) model from Kipf and Welling [127], which is a neural network structured on the shape of the input

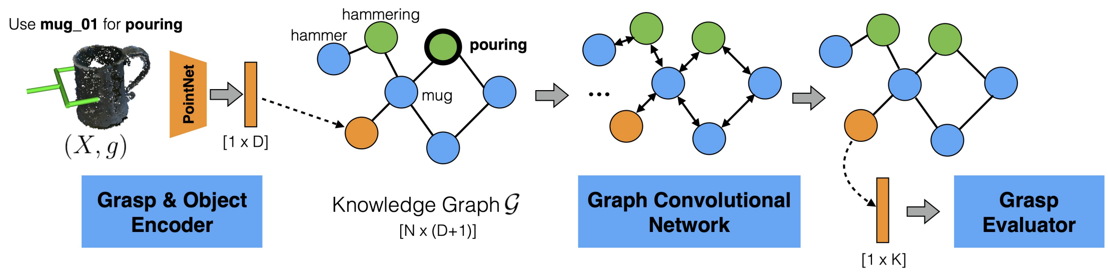

Figure 7.3: Overview of our Task-Oriented grasping framework using semantic knowledge graphs.

graph. By structuring a neural network to pass information between adjacent nodes, we use the input graph to correctly reason about the relationship between the object classes and the target task.

The first input of a GCN is the graph itself $\mathcal{G} = (V, E)$. In our application, we use a knowledge graph constructed from two sources: the task-object class relationships in our dataset and the object hierarchy from WordNet [167]. The graph is represented as a binary adjacency matrix A, which we normalize to obtain $\hat{A}$ following [127]. The next input to each node of the GCN is a $D-$dimensional embedding vector. The target tasks are specified using an extra indicator latent variable that is concatenated with this embedding to get the vector of size $D + 1$. The embedding vectors are stacked across nodes to get the input matrix $\mathcal{X} \in \mathcal{R}^{|V| \times (D+1)}$. We initialize the matrix with the word embeddings corresponding to each concept in the knowledge graph (e.g. "mug"). We use ConceptNet numberbatch [222] for the word embeddings. The grasp and shape encoder nodes are added online to the existing knowledge graph $\mathcal{G}$ by connecting edges to the corresponding object class nodes.

The output of the GCN are K-dimensional embeddings for each node $\mathcal{Z} \in \mathcal{R}^{|V| \times K}$. The node embeddings are propagated to their neighbours using message passing in each convolutional layer:

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}) \tag{7.1}$$

where σ is the ReLU activation function, $H^{(0)} = \mathcal{X}$ and $H^{(L)} = \mathcal{Z}$ where L is the number of layers.

Grasp Evaluator: After the GCN, we are left with a node-level embedding $\mathcal{Z}$. We use the embedding corresponding to the grasp node z_g to train the final grasp evaluator $P(S|z_g)$, where S is the grasp score. This module has three fully connected layers with K units and a final sigmoid layer. The entire model, including the shape encoder, GCN and grasp evaluator, is optimized with ADAM using a binary cross entropy loss.

Implementation Details: The point clouds were downsampled to 4096 points during training. They were also mean centered and unit-scaled. The PointNet module consists of three set abstraction layers and the number of points sampled are 512, 128 and all points. The set abstraction layers are followed by three fully connected layers with sizes [1024, 512, D]. Each set abstraction layer has three fully connected layers to learn features. The point clouds were perturbed with random rotations, jitter and dropout for data augmentation and to build robustness when testing on novel objects in unknown poses. We choose D=300 and K=128, and L=6 as the parameters for our GCN network.

7.5 Experimental Evaluation

7.5.1 Zero-Shot Generalization

A central goal of both our dataset and our method is to show that we can learn task-oriented grasping models which generalize to novel objects, classes and tasks. In an ideal robotics system, we should be able to correctly grasp a novel object from a novel object class, or even grasp for a novel task. To test this, we measure our system and baselines in three different held-out test settings: held-out object instances, held-out object categories, and held-out tasks.

These held-out settings are of increasing difficulty in terms of zero-shot generalization. For each setting, we perform k-fold cross validation (k=4), such that each category (a task, object class, or object instance, based on the setting) will be held out exactly once. In each fold, grasps from 25% of the categories will be used for testing while remaining grasps will be used for training and validation.

In all experiments, we only evaluate tasks that are valid for a given input object class. This makes sense from an evaluation perspective as it separates the problem of predicting applicable tasks for objects from task-driven grasping. It also makes the comparison to methods using object-task information fair since the models do not have to decide whether the object-task pair is valid.

Evaluation Metrics: Since k-fold cross validation in any held-out setting will evaluate all grasps in the dataset, we can compute Average Precision (AP) scores for any category, i.e., any object instance, object class, or task. We then compute an mAP averaged over object instances, mAP averaged over object classes, and mAP over tasks. We show all three metrics for each of our three settings in Tables 7.2a,7.2b,7.2c, but emphasize the mAP metric that corresponds to what category is being held out.

Baselines: We compare our approach to the following models: (1) Random, which represents grasping strategies that focus on grasp stability and ignore task constraints. Results are averaged over five random seeds. (2) Semantic Grasp Network (SGN), which learns to reason about context of grasps (e.g., constraints imposed by objects and tasks) from data. This model is adapted from [146], with the difference that the input to the model is replaced with geometric embedding from our shape encoder and word embeddings of the task and the object class. Note that embeddings of

Table 7.2: Results on TaskGrasp

(a) Object Instance Generalization

Model	Test Performance (mAP)		
	Instances	Classes	Tasks
Random	59.75	60.28	54.76
SGN [146]	78.51	75.08	68.8
SGN + word embedding	79.74	77.91	**74.36**
GCNGrasp (ours)	**80.25**	**77.94**	73.71

(b) Object Class Generalization

Model	Test Performance (mAP)		
	Instances	Classes	Tasks
Random	59.32	58.73	52.27
SGN [146]	74.2	72.95	62.55
SGN + word embedding	77.21	75.51	**63.73**
GCNGrasp (ours)	**78.81**	**76.57**	57.36

(c) Task Generalization

Model	Test Performance (mAP)		
	Instances	Classes	Tasks
Random	59.06	58.24	52.37
SGN [146]	75.17	71.59	63.35
SGN + word embedding	78.06	74.49	70.55
GCNGrasp (ours)	**81.5**	**79.56**	**76.01**

Table 7.3: Ablation on Semantic Knowledge

Model	Graph		Held-out Setting		
	Nodes	Edges	Task	Class	Instance
GCN + tasks + WordNet	345	989	76.01	**76.57**	80.25
GCN + tasks	131	693	**77.54**	75.86	**81.46**
GCN + WordNet	155	106	71.77	70	78.66

tasks and object classes are both learned from training data. (3) SGN + *word embedding*, which uses ConceptNet [222] numberbatch as pretrained word embeddings for object classes and tasks.

7.5.2 Analysis

First, to get context for our results in Table 7.2, we see that random grasp prediction achieves approximately 50-60% accuracy, establishing a floor for the other methods. Because the number of positive and negative grasps in the dataset is about even, random guessing is able to achieve a seemingly high mAP. In a dataset with more negatives we would expect this number to be much lower.

Our method outperforms baselines in all three settings. This confirms that our method can effectively leverage the knowledge graph to generalize to novel object instances, object classes, and tasks. SGN + *word embedding* also outperforms SGN, suggesting that implicit distributional knowledge provides a prior that is useful for generalization. Despite the benefit of distributional knowledge, it still only represents semantic similarities between concepts. In contrast, the knowledge graph directly stores relations between the relevant objects and tasks, and exploiting this additional structured knowledge allows our model to achieve better zero-shot generalization than SGN + *word embedding*.

When comparing our method with SGN and SGN + *word embedding*, we observe increasingly

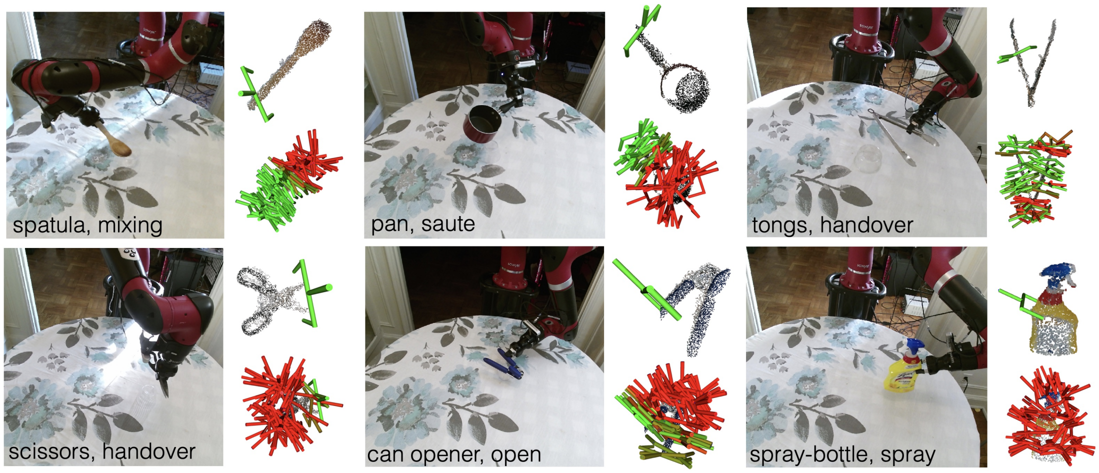

Figure 7.4: Robot executions of example task-oriented grasps on unknown objects. For each execution, the top 3D visualization shows the grasp that was executed (which had the best evaluator score) and the bottom shows all the stable grasp candidates colored by their scores (green is higher).

larger margins in performance from the held-out instance to the held-out class setting. As objects from different classes have more variance in terms of geometric and visual features than objects from the same class, semantic knowledge becomes more important in unifying these objects. The difference in performance between our method and these two baselines on the held-out task setting reached 12.6% and 5.46% respectively, affirming that semantic knowledge is especially crucial for generalizing disparate constraints from different tasks.

Ablations on Knowledge Graph: We investigated how performance is affected by changing the knowledge graph used in our model. Specifically, we compared the default knowledge graph with a knowledge graph containing only the semantic hierarchy of objects and a knowledge graph containing only the relations between object classes and tasks. The results from the three held-out settings are summarized in Table 7.3 (we only show the mAP metrics corresponding to the held-out category). From these results, we observe that edges between object classes and tasks were the most important knowledge for generalizing to novel tasks and instances, though every task we tested was valid for the target object class. This suggests that knowledge about which objects could generally be used for which tasks provide important information for discovering similarities between tasks. In the held-out object class setting, additional knowledge from the object hierarchy helped generalize to novel object classes by associating known classes and novel classes through the WordNet hierarchy.

7.5.3 Real Robot Evaluation

We run experiments to show that our approach and dataset transfer to a real robot. We test our approach on novel objects not from the dataset and in unknown poses. We place each object (without clutter) on a table in front of the robot. After table plane segmentation to obtain the object point cloud, 600 stable grasps are sampled and 50 candidates are selected using farthest point sampling for evaluation. We evaluate the grasps on our best performing GCNGrasp model from the held-out task ablations (Table 7.2). Our hardware setup comprises of a 7-DOF Sawyer Robot with a 2-fingered Robotiq gripper and a Intel Realsense D415 RGB-D camera mounted on the gripper wrist. Inference for the 50 grasps takes around $3s$ on a desktop with an NVIDIA GTX 1080 Ti GPU and the grasp with the best score is executed on the robot. Fig 7.4 shows the executed task-oriented grasps on unknown objects. Even though our dataset objects were collected only in one canonical pose, our approach is able to generalize to new grasps and in unknown poses due to data augmentation during training. Based on the grasp evaluator scores from Fig 7.4, our model is also able to interpolate between modes in the continuous $SE(3)$ space to reason about task-oriented grasping. One failure mode of our work is that it does not generalize to categories (like the spray bottle in Fig 7.4 in the bottom right) with limited training data. A future work is to balance the dataset in terms of object categories.

7.5.4 Comparison to SG14000

We want to demonstrate that grasping models trained on our GCNGrasp dataset generalize to other task-oriented grasping datasets, namely SG14000 proposed by Liu et al. [146]. We show transfer learning results on SG14000, since it has the most similar setting by providing objects with their corresponding point clouds and grasps in $SE(3)$. Since SG14000 does not come with any semantic knowledge, we use the Semantic Grasping Network (SGN) + *word embedding* as the backbone model instead of GCNGrasp. SG14000 is significantly smaller and less diverse with 14K grasps. The five object categories and seven tasks were resolved to WordNet synsets to have complete overlap with TaskGrasp. The test dataset was held-out based on grasps, hence may include known object classes and tasks during evaluation. The model trained on SG14000 performed well when tested on itself. However, it failed to generalize to the more diverse TaskGrasp with only a 17% increase over a random baseline. It is not surprising that the model trained on TaskGrasp was able to generalize to the held-out test set in TaskGrasp. It also performed well when tested on the SG14000 test set though it did not outperform the model trained on SG14000. This is owing to several reasons. First, the point clouds in SG14000 were incomplete with a lot of self-occlusions (since objects were scanned from just a single view) whereas our point clouds are constructed based on scans from multiple view points. This could affect the performance of the Object and Grasp Encoder based on PointNet [197]. Second, SG14000 has a dataset bias since it models the effects of material and object state on grasps, while we focus on object geometry. Another reason could be dataset imbalance in TaskGrasp as we do not have sufficient quantities of certain categories (bowls, bottles) in comparison. Lastly, SG14000 has some grasps in free space (which we filtered out in our dataset) where our model predicts a high score. This can be corrected by adding unstable grasps as hard negatives during training, similar to prior work [173, 179].

Table 7.4: Cross generalization on TaskGrasp and SG14000

Train Dataset	**Held-out Test Grasps (mAP)**	
	TaskGrasp	SG14000 [146]
TaskGrasp	76.2	52.3
SG14000	25.1	62.7
Random	7.9	24.8

7.5.5 Analysis on GCNGrasp Predictions

Next we visualize AP scores for each task from GCNGrasp predictions trained on TaskGrasp in Fig 7.5. The AP scores for all tasks were computed with cross validation as detailed earlier. The red bar corresponds to AP score with predictions from a random model (averaged over five seeds) while the red and blue bar cumulatively represents the model AP score. Overall, GCNGrasp performed better than random predictions, though some tasks are more challenging than others. For instance, juice, saute and screw are harder tasks (with low random prediction scores) compared to handover and poke. Tasks that represent more creative than prototypical uses of an object are typically more ambiguous and challenging to label. Yet, our model is able to improve over random predictions even in these challenging tasks.

7.6 Conclusion

We present the TaskGrasp dataset to study generalization in Task-Oriented grasping. The dataset is diverse and an order of magnitude larger than previous datasets. We also present a framework for jointly learning from geometric observations and semantic knowledge to generalize to new object instances, classes and even new tasks. Future work could explore recent techniques in automatic knowledge graph generation [35] for grasping tasks. While we collected real point cloud data of objects, we could convert the point clouds to meshes or acquire shape models from large online repositories to use in physics simulators. This could expand the scope of the dataset for sim2real transfer and to even learn task policies in simulation conditioned on the task-oriented grasps like in prior work [81].

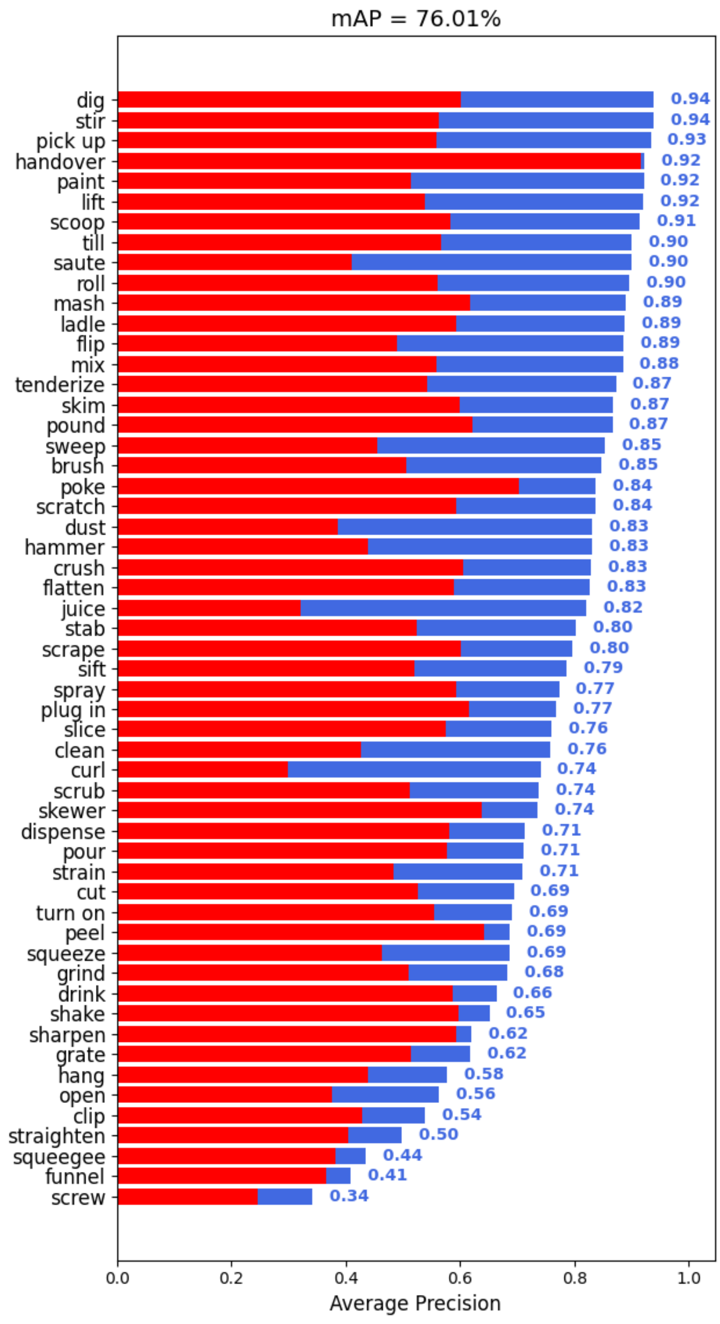

Figure 7.5: mAP across tasks for GCNGrasp predictions. The red bar is for AP predictions by a random model while the red and blue cumulatively represents the model prediction.

Table 7.5: Object Task Combinations

Object Class	Suitable Tasks
atomizer.n.01	pour, clean, squeegee, dispense, handover, spray
backscratcher.n.02	pick up, turn on, shake, scrape, handover, scoop, scratch, mix
basket.n.01	pick up, dispense, lift, handover, scoop, hang
baster.n.03	poke, dispense, squeeze, handover, drink, spray
beer_mug.n.01	pour, scoop, ladle, flatten, dispense, hammer, lift, mash, handover, crush, drink, pound
book.n.02	pound, crush, handover
bottle.n.01	pour, clean, juice, squeegee, poke, shake, flip, flatten, dispense, hammer, straighten, handover, pound, drink, spray, mix
bottlebrush.n.01	clean, dust, scrub, scrape, straighten, brush, mash, handover, crush, scratch
brush.n.02	handover, brush, sweep, paint
can_opener.n.01	cut, open, squeegee, poke, pick up, squeeze, screw
cereal_bowl.n.01	pour, scoop, ladle, pick up, dispense, handover, drink, mix
charger.n.02	plug in, turn on
clamp.n.01	clip, straighten, lift, squeeze, crush, hang
coat_hanger.n.01	hang, straighten, pick up, handover
coffee_mug.n.01	pour, skim, clean, scoop, ladle, pick up, grind, shake, flatten, dispense, dig, lift, handover, drink, pound
colander.n.01	pour, skim, juice, poke, dig, funnel, handover, crush, sift, scoop, strain, pound
control.n.09	turn on
cookie_cutter.n.01	slice, cut
cup.n.01	pour, skim, clean, scoop, ladle, till, saute, poke, pick up, shake, tenderize, flatten, dispense, dig, lift, crush, handover, pound, drink, mix
dustcloth.n.01	clean, dust, brush, sweep
dustpan.n.02	clean, dust, pick up, flip, dispense, lift, handover, scoop, sweep
flashlight.n.01	turn on, handover
fork.n.01	skewer, juice, ladle, poke, stir, pick up, handover, stab, flip, grate, dig, scrape, curl, lift, funnel, mash, scoop, scratch, strain, mix
fork.n.04	till, stir, stab, dig, scrape, handover, scratch
frying_pan.n.01	pour, saute, stir, pick up, flip, tenderize, flatten, dispense, hammer, lift, mash, handover, crush, pound, scoop, mix
funnel.n.02	pour, scoop, pick up, stab, dispense, scrape, squeeze, funnel, roll, strain, mix
garlic_press.n.01	grind, flatten, hammer, squeeze, mash, handover, crush, scratch
grater.n.01	cut, slice, grind, tenderize, grate, scrape, scratch, strain
hair_spray.n.01	roll, spray, handover
hammer.n.02	tenderize, flatten, hammer, straighten, mash, crush, pound
keg.n.02	flatten, dispense, drink, pour
knife.n.01	cut, peel, poke, slice, stab, clip, scrape, sharpen, scratch
ladle.n.01	pour, skim, scoop, ladle, poke, saute, stir, pick up, dispense, hammer, scrape, lift, handover, sift, roll, drink, strain, sweep, mix
masher.n.02	cut, juice, poke, stir, grind, tenderize, flatten, hammer, mash, handover, crush, pound, mix
measuring_cup.n.01	pour, scoop, ladle, pick up, dispense, dig, lift, handover, drink
mixing_bowl.n.01	pour, dispense, pick up, mix
mortar.n.03	pour, stir, grind, tenderize, flatten, pound, mash, handover, crush, sift, scoop, mix
mug.n.04	pour, drink, scoop, handover
nozzle.n.01	dispense, spray
paint_roller.n.01	squeegee, paint, tenderize, flatten, dispense, brush, handover, roll, pound
pancake_turner.n.01	cut, skim, ladle, pick up, crush, scoop, saute, turn on, flatten, scrape, handover, mix, pound, pour, stir, poke, flip, dig, lift, mash, sift
peeler.n.03	peel, slice, grate, scrape, scratch
pepper_mill.n.01	grind, crush, handover
pitcher.n.02	pour, scoop, ladle, stir, pick up, shake, flatten, dispense, lift, handover, drink, mix
reamer.n.01	plug in, juice, scrape, mash, handover
roller.n.04	roll, clean, lift, handover
rolling_pin.n.01	poke, tenderize, flatten, hammer, straighten, mash, handover, crush, roll, pound
saltshaker.n.01	pour, dispense, crush, handover, strain, shake
saucepan.n.01	pour, scoop, ladle, saute, stir, pick up, shake, flatten, dispense, dig, lift, mash, handover, crush, drink, mix
saucepot.n.01	pour, saute, dispense, lift, mash, handover, crush, drink, mix
scissors.n.01	cut, open, poke, slice, handover, stab, clip, scrape, curl, straighten, sharpen, scratch
scoop.n.05	clean, pick up, flip, dig, lift, handover, sift, scoop, strain, mix
scoop.n.06	clean, stir, pick up, dig, lift, handover, scoop, pound
scraper.n.01	peel, clean, squeegee, slice, stir, stab, flatten, dig, scrape, straighten, lift, handover, scoop, scratch
screwdriver.n.01	skewer, open, poke, stab, dig, screw, hang, scratch, mix
scrub_brush.n.01	clean, dust, paint, poke, stir, scrub, shake, stab, flip, tenderize, flatten, scrape, straighten, mix, funnel, handover, brush, scratch, sweep, pound
server.n.04	ladle, stir, pick up, curl, lift, handover, sift, scoop, hang, mix
sieve.n.01	sift, dispense, strain, skim
sifter.n.01	sift, dispense, strain
skimmer.n.02	skim, ladle, saute, stir, pick up, flip, scrape, handover, sift, scoop, strain
slicer.n.03	cut, peel, open, juice, slice, saute, grate, screw, mix
spatula.n.01	skim, poke, saute, stir, pick up, scrub, flip, dig, lift, crush, handover, scoop, scratch, mix
spatula.n.02	skim, stir, pick up, flip, flatten, scrape, lift, mix
sponge.n.01	skim, clean, squeegee, dust, poke, scrub, scrape, brush, handover, drink, scratch, sweep
squeegee.n.01	clean, squeegee, scrub, scrape, handover
squeezer.n.01	juice, flatten, squeeze, mash, handover, crush, drink, pound
straightener.n.01	flatten, pick up, straighten
strainer.n.01	skim, stir, pick up, shake, flip, dispense, lift, funnel, handover, crush, sift, scoop, strain, sweep, mix
tablespoon.n.02	skim, ladle, pick up, dispense, curl, scoop, scratch, saute, turn on, stab, flatten, scrape, squeeze, handover, mix, pound, pour, stir, poke, flip, dig, lift, mash, sift, drink, strain
tongs.n.01	squeegee, pick up, clip, dispense, crush, scoop, scratch, saute, turn on, stab, scrape, squeeze, funnel, handover, shake, skewer, mix, stir, straighten, poke, flip, lift, roll
trowel.n.01	till, slice, stir, poke, stab, flip, flatten, hammer, dig, scrape, lift, crush, scoop, scratch, mix
vase.n.01	pour, scoop, tenderize, dig, straighten, lift, handover, drink, shake
watering_can.n.01	pour, scoop, poke, dispense, funnel, drink, shake
whisk.n.01	mix, stir, brush, handover
wooden_spoon.n.02	skim, ladle, poke, saute, stir, pick up, flatten, dig, scrape, lift, mash, handover, pound, scoop, mix

Chapter 8

Conclusion

8.1 Overview

Robotic grasping has seen tremendous advancements in generalization in recent years. Yet, the current paradigm of manipulation research is typically some form of table-top manipulation in constrained setups or in simulation. In this thesis, we explored several directions in scaling data-driven grasping to the diversity and constraints imposed by the real world.

We now summarize the key contributions in the thesis. In Part I, we showed how we can generalize beyond picking individual objects to 6-DOF grasping in structured clutter from just the raw partial point cloud observations. In Part II, we discussed challenges with scaling robot learning on diverse hardware systems. We presented an empirical attempt in collecting a large-scale self-supervised grasping dataset using several low-cost robots interacting in unstructured environments like human homes. We also proposed a framework for factoring out robot-specific noise which improves performance of grasping models during testing. Apart from discussing the algorithmic problem of transferring policies between different robots, we also introduced the Pyrobot software framework as an attempt to write hardware-agnostic code for manipulation. In Part III, we showed how we can improve robustness by closing the loop on grasp execution with data-driven tactile re-grasping. Lastly, in Part IV, we strive to go beyond robotic pick-and-place and generalize to diverse semantic manipulation tasks. We do so by scaling task-oriented grasping datasets with crowdsourcing and learning from semantic information like knowledge graphs.

8.2 Limitations

We will briefly discuss the overall failure modes in the approaches taken in this thesis. We refer to the individual chapters for more detailed discussion of limitations for each project.

First, a shortcoming in Part I was that we focused on collisions between just the gripper and the cluttered scene. Though this worked well in practice, several collisions could be prevented by considering the arm trajectory in the grasp generation process as well. The larger issue here is that the grasping literature has evolved separately from motion planning. Though this abstraction has worked well in table-top settings, it is unclear if this is sufficient in constrained setups like in homes, kitchens or when working in close proximity to humans. There is less free space in such constrained environments and only a handful of feasible grasps could be executed without collisions.

Data-driven grasping papers typically demonstrate generalization to unknown object instances but the generalization to domain shift is understudied. In Part II, we studied this problem in terms of data from out-of-distribution visual environments and robots. In Part IV, we collected a large grasp dataset with diverse object categories and found it challenging to generalize to point cloud data from unknown categories. This was partly because of our imbalanced dataset - common categories like mugs had more data than more specialized ones like spray bottles.

Third, a failure mode in Part III was partial observability in terms of sensing. Our tactile sensors were only mounted on the finger-tip and hence a lot of contact on other parts of the gripper were unaccounted. Without any touch feedback, the robot accidentally pushed objects or slippage went undetected. Contact estimation is an active area of research including skin sensors with more spatial coverage[31], Finite-Element Models (FEM) of tactile sensors [182], etc.

Lastly, all the approaches presented in the thesis suffer from the common challenges in depth sensing (transparent, specular objects, etc.) that affect the quality of the object geometry.

8.3 Directions for Future Research

After reading this thesis and related work, you may still be wondering about the comment on whether "grasping is solved" which we discussed in the introduction. I hope that this thesis has convinced the reader otherwise and shown the vast space of problems to be solved before we can have reliable manipulation in-the-wild. We will conclude this thesis by discussing and speculating on some open problems.

Developments in Hardware: A major bottleneck for democratizing grasping is that we do not yet have a mass market manipulation platform. The collaborative robotics industry has not had its equivalent of the Ford Model T moment (in 1908) for automobiles or what the iPhone did for smartphones in 2008. Fortunately, the price of robots have been falling for decades [235] and recent low-cost mobile manipulators like LoCoBot [177], Blue [91], HelloRobot [18] have shown great promise in working in unstructured environments like homes. We may truly have a mass market manipulator platform in the near future. Apart from the manipulator itself, the hardware for depth perception and 3D scanning is also getting more robust and affordable, such as the Microsoft Kinect

[3] and Realsense line of products [123]. Cheaper tactile and force-torque sensors would also go a long way to improving robustness for grasping systems [255].

Faster and Robust Software: From a vision standpoint, segmentation is an important building block of grasping systems and is an active area of research in terms of data [100], algorithms [104, 248] and applications in grasping [22, 148, 179]. Grasping models are also getting very complex to train due to domain randomization and the need for curating appropriate simulations for sim2real transfer [173]. The vision community has started emphasizing system performance (low-power, real-time detection) and better training procedures. For instance, ImageNet classification training took 15mins in 2018 [253] compared to 6 days in 2012 [136]). Unfortunetaly, the same emphasis on systems performance cannot be said of the robotics community. A focus on these factors could benefit more challenging applications of data-driven grasping, such as dynamic grasping and manipulating deformable objects like cloth. This will require us to tightly integrate the vision + action loop ($>$100 Hz) from the current status quo of very slow inference (3-10s per grasp execution) that papers have reported [173, 179]. Safety in the context of robotic grasping is also an understudied problem and could benefit from faster inference times.

Benchmarking, Problem Definition and Harder Tasks: It may also be helpful as a community to come to an understanding as to when “grasping will be solved”. Establishing metrics and boundaries for the problem would be really beneficial for everyone to follow. Unlike machine learning and computer vision, this can be challenging for the robotics community though there have been initial efforts such as the YCB dataset [41] and Mean-Picks-Per-Hour (MPPH) metric for bin-picking [154]. By defining the problem better, we can also apply grasping to harder tasks. Overall, robotic skill learning and its recent incarnation of deep reinforcement learning has largely evolved separately from grasping. Despite its promise, skill learning has mainly been confined to known objects in simulation why grasping systems have shown industrial grade performance in the real world and on unknown objects. Could we perhaps bridge this gap?

Bibliography

[1] Dobot Magician. https://www.dobot.cc/dobot-magician/specification.html.

[2] Makerblock Gripper. http://store.makeblock.com/robot-gripper.

[3] Microsoft Azure Kinect. https://azure.microsoft.com/en-us/services/kinect-dk/#industries.

[4] Kobuki Base. http://kobuki.yujinrobot.com/about2/.

[5] On Board Computer. https://www.acer.com/ac/en/US/content/model/NX.GTSAA.005.

[6] Sawyer technical specifications. https://www.rethinkrobotics.com/sawyer/tech-specs/, 2016.

[7] Ur5 technical specifications. https://www.universal-robots.com/media/50588/ur5_en.pdf, 2016.

[8] Trossen widow x robotic arm. https://www.trossenrobotics.com/widowx-200-robot-arm-mobile-base.aspx, 2018.

[9] Battery pack. https://www.maxoak.net/laptop-power-bank/show/11.html, 2018.

[10] Create2 mobile base. https://www.irobot.com/about-irobot/stem/create-2, 2018.

[11] Franka emika specification. https://www.franka.de/, 2018.

[12] Open manipulator project. http://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/, 2018.

[13] YACS – yet another configuration system. https://github.com/rbgirshick/yacs, 2018.

[14] ICRA workshop on benchmarks for robotic manipulation. http://www.ycbbenchmarks.com/ICRA2019_workshop, 2019.

[15] ICRA workshop on dataset generation and benchmarking of slam algorithms for robotics and vr/ar. https://sites.google.com/view/icra-2019-workshop/home, 2019.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,

Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[18] Evan Ackerman. Ex-googler's startup comes out of stealth with beautifully simple, clever robot design. 2020. URL https://spectrum.ieee.org/automaton/robotics/home-robots/hello-robots-stretch-mobile-manipulator.

[19] Wisdom C Agboh and Mehmet R Dogar. Real-time online re-planning for grasping under clutter and uncertainty. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2018.

[20] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[21] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Josh Tobin Szymon Sidor, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[22] Shuran Song Daniel Suo Ed Walker Alberto Rodriguez Jianxiong Xiao Andy Zeng, Kuan-Ting Yu. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[23] Laura Antanas, Plinio Moreno, Marion Neumann, Rui Pimentel de Figueiredo, Kristian Kersting, José Santos-Victor, and Luc De Raedt. Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach. *Autonomous Robots*, pages 1–26, 2018.

[24] Paola Ardón, Èric Pairet, Ronald PA Petrick, Subramanian Ramamoorthy, and Katrin S Lohan. Learning grasp affordance reasoning through semantic relations. *IEEE Robotics and Automation Letters*, 4(4):4571–4578, 2019.

[25] Karl J Åström. *Introduction to stochastic control theory*. Courier Corporation, 2012.

[26] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y Matsuoka. Human-guided grasp measures improve grasp robustness on physical robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, page 22942301. IEEE, 2010.

[27] Yasemin Bekiroglu, Janne Laaksonen, Jimmy Alison Jorgensen, Ville Kyrki, and Danica Kragic. Assessing grasp stability based on learning and haptic data. *IEEE Transactions on Robotics*, 2011.

[28] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

[29] Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. *International Conference on Humanoid Robots*, 2007.

[30] Ankit Bhatia, Aaron Johnson, and Matthew Mason. Direct drive hands: Force-motion transparency in gripper design. *Robotics Science and Systems (RSS)*, 2019.

[31] T. Bhattacharjee, A. Kapusta, Jim Rehg, and Charles Kemp. Rapid categorization of object properties from incidental contact with a tactile sensing robot arm. *Humanoids*, 2013.

[32] A Bicchi and V Kumar. Robotic grasping and contact: a review. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[33] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesisa survey. *IEEE Transactions on Robotics*, 2014.

[34] Ch Borst, Max Fischer, and Gerd Hirzinger. Grasp planning: How to choose a suitable task wrench space. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 319–325. IEEE, 2004.

[35] Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. 2019. URL https://arxiv.org/abs/1906.05317.

[36] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[37] Samarth Brahmbhatt, Cusuh Ham, Charlie Kemp, and James Hays. Contactdb: Analyzing and predicting grasp contact via thermal imaging. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[38] Rodney Brooks. A robust layered control system for a mobile robot. *AI Memo 864*, 1985.

[39] Herman Bruyninckx. Open robot control software: the orocos project. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2001.

[40] Roberto Calandra, Andrew Owens, Manu Upadhyaya, Wenzhen Yuan, Justin Lin, Edward Adelson, and Sergey Levine. The feeling of success: Does touch sensing help predict grasp outcomes? *Conference on Robot Learning*, 2017.

[41] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 1070(9932/15):36, 2015.

[42] Yevgen Chebotar, Karol Hausman, Oliver Kroemer, Gaurav Sukhatme, and Stefan Schaal.

Generalizing regrasping with supervised policy learning. *ISER*, 2016.

[43] Yevgen Chebotar, Karol Hausman, Zhe Su, Artem Molchanov, Oliver Kroemer, Gaurav Sukhatme, and Stefan Schaal. Bigs: Biotac grasp stability dataset. In *ICRA Workshop on Grasping and Manipulation Datasets*, 2016.

[44] Yevgen Chebotar, Karol Hausman, Zhe Su, Gaurav S Sukhatme, and Stefan Schaal. Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[45] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *Nueral Information Processing Systems*, 2018. URL https://arxiv.org/abs/1811.09864.

[46] Xinlei Chen and Abhinav Gupta. Webly supervised learning of convolutional networks. *International Conference on Computer Vision*, 2015.

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

[48] Changhyun Choi, Wilko Schwarting, Joseph DelPreto, and Daniela Rus. Learning object grasping for soft robot hands. In *IEEE Robotics and Automation Letters*, pages 2370 – 2377. IEEE, 2018.

[49] Vivian Chu, Ian McMahon, Lorenzo Riano, Craig G McDonald, Qin He, Jorge Martinez Perez-Tejada, Michael Arrigo, Naomi Fitter, John C Nappo, Trevor Darrell, et al. Using robotic exploratory procedures to learn the meaning of haptic adjectives. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[50] Rico Jonschkowski Roberto Martin-Martin Arne Sieverling Vincent Wall Oliver Brock Clemens Eppner, Sebastian Hofer. Lessons from the amazon picking challenge: Four aspects of building robotic systems. *Robotics Science and Systems (RSS)*, 2016.

[51] Alvaro Collet, Dmitry Berenson, Siddhartha S Srinivasa, and Dave Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[52] Alexis Conneau and Douwe Kiela. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*, 2018.

[53] Hao Dang Peter Allen Corey Goldfeder, Matei Ciocarlie. The columbia grasp database. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[54] Nikolaus Correll, Kostas Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph Romano, and Peter Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 2017.

[55] Akansel Cosgun, Tucker Hermans, Victor Emeli, and Mike Stilman. Push planning for object placement on cluttered table surfaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4627–4632. IEEE, 2011.

[56] Erwin Coumans. Bullet physics simulation. *ACM SIGGRAPH*, 2015.

[57] George Dahl, Dong Yu, Li Deng, and Alex Acero. Context dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2010.

[58] Hao Dang and Peter K Allen. Semantic grasping: Planning robotic grasps functionally suitable for an object manipulation task. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1311–1317. IEEE, 2012.

[59] Hao Dang and Peter K Allen. Grasp adjustment on novel objects using tactile experience from similar local geometry. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[60] Hao Dang and Peter K Allen. Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 2013.

[61] Hao Dang, Jonathan Weisz, and Peter K Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[62] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Robert Martn-Martn, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical search: Multi-step retrieval of a target object occluded by clutter. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[63] Angel Daruna, Weiyu Liu, Zsolt Kira, and Sonia Chetnova. Robocse: Robot common sense embedding. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9777–9783. IEEE, 2019.

[64] Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *RSS*, 2011.

[65] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[66] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking. *Robotics Science and Systems (RSS)*, 2019.

[67] Renaud Detry, Jeremie Papon, and Larry Matthies. Task-oriented grasping with semantic and geometric scene understanding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3266–3273. IEEE, 2017.

[68] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

[69] Thanh-Toan Do, Anh Nguyen, and Ian Reid. Affordancenet: An end-to-end deep learning approach for object affordance detection. In *International Conference on Robotics and Automation (ICRA)*, pages 1–5. IEEE, 2018.

[70] Mehmet Dogar and Siddhartha Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[71] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems VII*, 1, 2011.

[72] M McTaggart R Smith N Kelly-Boxall S Wade-McCue J Erskine R Grinover A Gurman T Hunn D Lee Anton Milan Trung Pham G Rallos A Razjigaev T Rowntree K Vijay Zheyu Zhuang C Lehnert I Reid Peter Corke Jrgen Leitner Douglas Morrison, Adam W Tow. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[73] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Uncertainty aware grasping and tactile exploration. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[74] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*, 2016.

[75] J.L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition, 48, 781799*, 1993.

[76] Clemens Eppner. Robot grasping by exploiting compliance and environmental constraints. *Doctoral Thesis*, 2018.

[77] Zackory Erickson, Sonia Chernova, and Charles C Kemp. Semi-supervised haptic material recognition for robots using generative adversarial networks. *Conference on Robot Learning*, 2017.

[78] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. Density-based spatial clustering of applications with noise. In *Int. Conf. Knowledge Discovery and Data Mining*, volume 240, 1996.

[79] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[80] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3516–3523. IEEE, 2018.

[81] Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *Robotics Science and Systems*, 2018.

[82] Ronald Fearing. Simplified grasping and manipulation with dextrous robot hands. *IEEE Journal on Robotics and Automation*, 1986.

[83] Javier Felip, Jose Bernabe, and Antonio Morales. Contact-based blind grasping of unknown objects. *IEEE-RAS International Conference on Humanoid Robots*, 2012.

[84] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[85] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, 2016.

[86] David Fischinger, Astrid Weiss, and Markus Vincze. Learning grasps with topographic features. *The International Journal of Robotics Research*, 34(9):1167–1194, 2015.

[87] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[88] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[89] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129, 2013.

[90] Yasuyuki Futagi, Yasuhisa Toribe, and Yasuhiro Suzuki. The grasp reflex and moro reflex in infants: Hierarchy of primitive reflex responses. *International Journal of Pediatrics*, 2012.

[91] David Gealy, Stephen McKinley, Brent Yi, Philipp Wu, Phillip Downey, Greg Balke, Allan Zhao, Menglong Guo, Rachel Thomasson, Anthony Sinclair, Peter Cuellar, Zoe McCarthy, and Pieter Abbeel. Quasi-direct drive for low-cost compliant robotic manipulation. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[92] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

[93] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation tech report. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[94] Corey Goldfeder and Peter K Allen. Data-driven grasping. *Autonomous Robots*, 31(1):1–20, 2011.

[95] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter Allen. The columbia grasp database. In *International Conference on Robotics and Automation (ICRA)*, pages 1710–1716. IEEE, 2009.

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

[97] Robin R Murphy Howie Choset-Henrik Christensen Steven H Collins Paolo Dario Ken Goldberg Koji Ikuta Neil Jacobstein Danica Kragic Russell H Taylor Marcia McNutt Guang-Zhong Yang, Bradley J Nelson. Combating covid-19the role of robotics in managing public health and infectious diseases. *Science Robotics*, 5, 2020.

[98] Erico Guizzo and Evan Ackerman. Gill pratt discusses toyotas ai plans and the future of robots and cars. 2015. URL `https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/gill-pratt-on-toyota-robot-plans`.

[99] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning

in homes: Improving generalization and reducing dataset bias. In *Advances in neural information processing systems*, 2018.

[100] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[101] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[102] Nick Hawes, Christopher Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrova, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Kortner, et al. The strands project: Long-term autonomy in everyday environments. *IEEE Robotics & Automation Magazine*, 24(3):146–156, 2017.

[103] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[104] Kaiming He, G. Gkioxari, Piotr Dollr, , and Ross Girshick. Mask r-cnn. *International Conference on Computer Vision*, 2017.

[105] Jon Herman and Will Usher. An open-source python library for sensitivity analysis. *Journal of Open Source Software*, 2017.

[106] Alexander Herzog, Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, Jeannette Bohg, Tamim Asfour, and Stefan Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, 36(1-2):51–65, 2014.

[107] Martin Hjelm, Carl Henrik Ek, Renaud Detry, and Danica Kragic. Learning human priors for task-constrained grasping. In *International Conference on Computer Vision Systems*, pages 207–217. Springer, 2015.

[108] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[109] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2017.

[110] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Variational information maximizing exploration. *arXiv preprintarXiv:1605.09674*, 2016.

[111] Kaijen Hsiao, Sachin Chitta, Matei Ciocarlie, and E Gil Jones. Contact-reactive grasping of objects with partial shape information. In *International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[112] De-An Huang, Minghuang Ma, Wei-Chiu Ma, and Kris Kitani. How do we use our hands? discovering a diverse set of common grasps. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[113] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged

robots. In *Science Robotics*, volume 4, 2019.

[114] Eric Jang, Sudheendra Vijayanarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[115] Shervin Javdani, Matthew Klingensmith, J Andrew Bagnell, Nancy S Pollard, and Siddhartha S Srinivasa. Efficient touch based localization through submodularity. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[116] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[117] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-paced curriculum learning. *AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Pages 2694-2700*, 2015.

[118] Roland S Johansson and J Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature reviews. Neuroscience*, 2009.

[119] Mohsen Kaboli, Di Feng, Kunpeng Yao, Pablo Lanillos, and Gordon Cheng. A tactile-based framework for active object learning and discrimination using multimodal robotic skin. *IEEE Robotics and Automation Letters*, 2017.

[120] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *Conference on Robot Learning*, 2018.

[121] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311. IEEE, 2015.

[122] Alexander Kasper, Zhixing Xue, and Rudiger Dillman. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *International Journal of Robotics Research*, 2012.

[123] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. *arXiv preprint arXiv:1705.05548*, 2017.

[124] Jennifer E King, Joshua A Haustein, Siddhartha S Srinivasa, and Tamim Asfour. Nonprehensile whole arm rearrangement planning on physics manifolds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2508–2515. IEEE, 2015.

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

[126] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014.

[127] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.

[128] Nikita Kitaev, Igor Mordatch, Sachin Patil, and Pieter Abbeel. Physics-based trajectory

optimization for grasping in cluttered environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3102–3109. IEEE, 2015.

[129] Ellen Klingbeil, Deepak Rao, Blake Carpenter, Varun Ganapathi, Andrew Y Ng, and Oussama Khatib. Grasping with application to an autonomous checkout robot. In *2011 IEEE international conference on robotics and automation*, pages 2837–2844. IEEE, 2011.

[130] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[131] Mia Kokic, Johannes A Stork, Joshua A Haustein, and Danica Kragic. Affordance detection for task-specific grasping using deep learning. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 91–98. IEEE, 2017.

[132] Mia Kokic, Danica Kragic, and Jeneatte Bohg. Learning task-oriented grasping from human activity datasets. In *IEEE Robotics and Automation Letters*, 2020.

[133] Michael C Koval, Mehmet R Dogar, Nancy S Pollard, and Siddhartha S Srinivasa. Pose estimation for contact manipulation with manifold particle filters. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[134] Michael C Koval, Nancy S Pollard, and Siddhartha S Srinivasa. Pre-and post-contact policy decomposition for planar contact manipulation under uncertainty. *The International Journal of Robotics Research*, 2016.

[135] Greg Kragten, Aris Kool, and Just Herder. Ability to hold grasped objects by underactuated hands: Performance prediction and experiments. In *International Conference on Robotics and Automation (ICRA)*, pages 2493– 2498. IEEE, 2009.

[136] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012.

[137] M. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Neural Information Processing Systems*, 2010.

[138] Safoura Rezapour Lakani, Antonio J Rodríguez-Sánchez, and Justus Piater. Exercising affordances of objects: A part-based approach. *IEEE Robotics and Automation Letters*, 3(4): 3465–3472, 2018.

[139] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *International Journal of Robotics Research*, 2015.

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

[141] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *ISER*, 2016.

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

[143] L.-J. Li and Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. *IJCV*, 2010.

[144] Hongzhuo Liang, Xiaojian Ma, Shuang Li, Michael Gorner, Song Tang, Bin Fang, Fuchun Sun, and Jianwei Zhang. Pointnetgpd: Detecting grasp configurations from point sets. In *IEEE International Conference on Robotics and Automation*, 2019.

[145] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[146] Weiyu Liu, Angel Daruna, and Sonia Chernova. Cage: Context-aware grasping engine. *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[147] Qingkai Lu, Kautilya Chenna, Balakumar Sundaralingam, and Tucker Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network. *arXiv preprint arXiv:1804.03289*, 2018.

[148] S. Gupta A. Li A. Lee J. Mahler M. Danielczuk, M. Matl and K. Goldberg. Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[149] Marianna Madry, Liefeng Bo, Danica Kragic, and Dieter Fox. St-hmp: Unsupervised spatio-temporal feature learning for tactile data. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[150] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. *Conference on Robot Learning*, 2017.

[151] Jeffrey Mahler, Sachin Patil, Ben Kehoe, Jur van den Berg, Matei Ciocarlie, Pieter Abbeel, and Ken Goldberg. GP-GPIS-OPT: Grasp planning with shape uncertainty using gaussian process implicit surfaces and sequential convex programming. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[152] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. 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 *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1957–1964. IEEE, 2016.

[153] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *Robotics Science and Systems (RSS)*, 2017.

[154] Jeffrey Mahler et al. Guest editorial open discussion of robot grasping benchmarks, protocols, and metrics. In *Transactions on Automation Science and Engineering*, volume 15. IEEE, 2018.

[155] Alexei Makarenko, Alex Brooks, and Tobias Kaupp. On the benefits of making robotic software frameworks thin. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[156] Abhijit Makhal, Federico Thomas, and Alba Perez Gracia. Grasping unknown objects in clutter by superquadric representation. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 292–299. IEEE, 2018.

[157] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[158] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*, 2019.

[159] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[160] E Marder-Eppstein. move base, a ros package that lets you move a robot to desired positions using the navigation stack.

[161] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[162] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[163] Martin Meier, Matthias Schopfer, Robert Haschke, and Helge Ritter. A probabilistic approach to tactile shape reconstruction. *IEEE Transactions on Robotics*, 2011.

[164] Nuttapong Chentanez Miles Macklin, Matthias Muller and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 2014.

[165] Andrew Miller. Graspit!: A versatile simulator for robotic grasping. *Doctoral Thesis*, 2001.

[166] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[167] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

[168] Ishan Misra, Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. Seeing through the human reporting bias:visual classifiers from noisy human-centric labels. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[169] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[170] Bogdan Moldovan, Plinio Moreno, Martijn Van Otterlo, José Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In

International Conference on Robotics and Automation (ICRA), pages 4373–4378. IEEE, 2012.

[171] A. Morales, T. Asfour, P. Azad, S. Knoop, , and R. Dillmann. Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 56635668. IEEE, 2006.

[172] Hans Moravec. *Mind children*. Harvard University Press, 1988.

[173] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DOF GraspNet: Variational grasp generation for object manipulation. *International Conference on Computer Vision*, 2019.

[174] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017. doi: 10.1109/TRO.2017.2705103.

[175] Adithyavairavan Murali, Yin Li, Dhiraj Gandhi, and Abhinav Gupta. Learning to grasp without seeing. *International Symposium on Experimental Robotics (ISER)*, 2018.

[176] Adithyavairavan Murali, Lerrel Pinto, Dhiraj Gandhi, and Abhinav Gupta. CASSL: Curriculum accelerated self-supervised learning. *IEEE International Conference on Robotics and Automation*, 2018.

[177] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. 2019. URL `https://arxiv.org/abs/1906.08236`.

[178] Adithyavairavan Murali, Weiyu Liu, Kenneth Marino, Sonia Chernova, and Abhinav Gupta. Same object, different grasps: Data and semantic knowledge for task-oriented grasping. In *Submitted*, 2020.

[179] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-dof grasping for target-driven object manipulation in clutter. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.

[180] Richard Murray, Zexiang Li, and Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[181] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *International Conference on Robotics and Automation*, 2017.

[182] Yashraj Narang, Karl Van Wyk, Arsalan Mousavian, and Dieter Fox. Interpreting and predicting tactile signals via a physics-based and data-driven framework. *Robotics Science and Systems (RSS)*, 2020.

[183] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[184] Van-Duc Nguyen. Constructing force-closure grasps. *The International Journal of Robotics Research*, 1988.

[185] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[186] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017. URL `http://arxiv.org/abs/1705.05363`.

[187] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*, 2017.

[188] Anna Petrovskaya and Oussama Khatib. Global localization of objects via touch. *IEEE Transactions on Robotics*, 2011.

[189] Zachary Pezzementi, Caitlin Reyda, and Gregory D Hager. Object mapping, recognition, and localization from tactile geometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

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

[191] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. *International Conference on Robotics and Automation*, 2017.

[192] Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer, 2016.

[193] Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1601–1608. IEEE, 2017.

[194] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *Robotics Science and Systems*, 2018.

[195] Florian Pokorny and Danica Kragic. Classical grasp quality evaluation: New algorithms and theory. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[196] Domenico Prattichizzo and Jeffrey Trinkle. Grasping. *Springer handbook of robotics*, pages 671–700, 2008.

[197] Charles Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Neural Information Processing Systems (NeurIPS)*, 2017.

[198] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.

[199] Justus J Randolph. Free-marginal multirater kappa (multirater k [free]): An alternative to

fleiss' fixed-marginal multirater kappa. *Online submission*, 2005.

[200] Peter D Brook Joshua R Smith Yoky Matsuoka Ravi Balasubramanian, Ling Xu. Physical human interactive guidance: Identifying grasping principles from human-planned grasps. *Transactions on Robotics*, (99):1–12, 2012.

[201] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[202] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[203] Colin Rennie, Rahul Shome, Kostas Bekris, and Alberto De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. In *Robotics and Automation Letters*. IEEE, 2016.

[204] Jean-Philippe Roberge, Samuel Rispal, Tony Wong, and Vincent Duchaine. Unsupervised feature learning for classifying dynamic tactile events using sparse coding. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[205] Alberto Rodriguez, Matthew Mason, and Steve Ferry. From caging to grasping. *Robotics Science and Systems (RSS)*, 2011.

[206] Joseph M Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katherine J Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 2011.

[207] Jacob Rosen, Diana Friedman, Hawkeye King, Phillip Roan, Lei Cheng, Daniel Glozman, Ji Ma, Sina Kosari, and Lee White. Raven-ii: An open platform for surgical robotics research. In *Transactions on Biomedical Engineering*. IEEE, 2012.

[208] Stephane Ross, Geoffrey Gordan, and Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *arXiv preprint arXiv:1011.0686*, 2010.

[209] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. In *arXiv preprint arXiv:1409.0575*, 2014.

[210] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[211] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications*, 2010.

[212] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications 181 259270*, 2010.

[213] Brad Saund, Shiyuan Chen, and Reid Simmons. Touch based localization of parts for high precision manufacturing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[214] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, Chioma Osondu, and Andrew Y Ng. Learning to grasp novel objects using vision. In *International Symposium on Experimental Robotics (ISER)*, 2006.

[215] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula. Robobrain: Large-scale knowledge engine for robots. *arXiv preprint arXiv:1412.0691*, 2014.

[216] Alexander Schneider, Jrgen Sturm, Cyrill Stachniss, Marco Reisert, Hans Burkhardt, and Wolfram Burgard. Object identification with tactile sensors using bag-of-features. *International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[217] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

[218] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[219] David Silver, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016.

[220] B.F. Skinner. Reinforcement today. *American Psychologist 13, 9499*, 1958.

[221] Dan Song, Kai Huebner, Ville Kyrki, and Danica Kragic. Learning task constraints for robot grasping using graphical models. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1579–1585. IEEE, 2010.

[222] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. pages 4444–4451, 2017. URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972.

[223] Adam Spiers, Minas Liarokapis, Berk Calli, and Aaron Dollar. Single-grasp object classification and feature extraction with simple robot hands and tactile sensors. *IEEE Transactions on Haptics*, 9, 2016.

[224] Ioan Sucan, Mark Moll, and Lydia Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4), 2012.

[225] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, and Arthur Szlam. Intrinsic motivation and automatic curricula via asymmetric self-play. *International Conference on Learning Representations (ICLR)*, 2017.

[226] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *International Conference on Computer Vision*,

2017.

[227] Jaeyong Sung, J Kenneth Salisbury, and Ashutosh Saxena. Learning to represent haptic feedback for partially-observable tasks. *arXiv preprint arXiv:1705.06243*, 2017.

[228] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 1996.

[229] Maxwell Svetlik, Leonetti Matteo, Sinapov Jivko, Rishi Shah, Nick Walker, and Peter Stone. Automatic curriculum graph generation for reinforcement learning agents. *AAAI'17 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2017.

[230] Johan Tegin and Jan Wikander. Tactile sensing in intelligent robotic manipulation–a review. *Industrial Robot: An International Journal*, 2005.

[231] Andreas ten Pas and Robert Platt. Using geometry to detect grasp poses in 3d point clouds. In *Robotics Research*, pages 307–324. Springer, 2018.

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

[233] Moritz Tenorth and Michael Beetz. Representations for robot knowledge in the knowrob framework. *Artificial Intelligence*, 247:151–169, 2017.

[234] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 2001.

[235] Jonathan Tilley. Automation, robotics, and the factory of the future. 2017. URL https://www.mckinsey.com/business-functions/operations/our-insights/automation-robotics-and-the-factory-of-the-future.

[236] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. 2017. URL https://arxiv.org/abs/1703.06907.

[237] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[238] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *Conference on Robot Learning*, 2018.

[239] Emmanouil Tsardoulias and Pericles Mitkas. Robotic frameworks, architectures and middleware comparison. *arXiv preprint arXiv:1711.06842*, 2017.

[240] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Tom Kollar, Cetin Mericli, Mehdi Samadi, Susana Brandao, and Rodrigo Ventura. Cobots: Collaborative robots servicing multi-floor buildings. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5446–5447. IEEE, 2012.

[241] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.

[242] David Wang, David Tseng, Pusong Li, Yiding Jiang, Menglong Guo, Michael Danielczuk, Jeffrey Mahler, Jeffrey Ichnowski, and Ken Goldberg. Adversarial grasp objects. In *Conference on Automation Science and Engineering*. IEEE, 2019.

[243] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[244] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Neural Information Processing Systems*, 2015.

[245] Melonee Wise, Michael Ferguson, Derek King, Eric Diehr, and David Dymesich. Fetch & freight: Standard platforms for service robot applications. *Workshop on Autonomous Mobile Service Robots, IJCAI*, 2016.

[246] Fei Xia, Amir R. Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: real-world perception for embodied agents. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[247] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[248] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2019.

[249] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[250] Xinchen Yan, Jasmined Hsu, Mohammad Khansari, Yunfei Bai, Arkanath Pathak, Abhinav Gupta, James Davidson, and Honglak Lee. Learning 6-dof grasping interaction via deep geometry-aware 3d representations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[251] Brian Yang, Jesse Zhang, Vitchyr Pong, Sergey Levine, and Dinesh Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019.

[252] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[253] Cho-Jui Hsieh James Demmel Kurt Keutzer Yang You, Zhao Zhang. Imagenet training in minutes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[254] Zhengkun Yi, Roberto Calandra, Filipe Veiga, Herke van Hoof, Tucker Hermans, Yilei Zhang, and Jan Peters. Active tactile object exploration with gaussian processes. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[255] Wenzhen Yuan, Siyuan Dong, and Edward Adelson. Gelsight: High-resolution robot tactile

sensors for estimating geometry and force. *Sensors*, 17(12):2762, 2017.

[256] Wenzhen Yuan, Chenzhuo Zhu, Andrew Owens, Mandayam A Srinivasan, and Edward H Adelson. Shape-independent hardness estimation using deep learning and a gelsight tactile sensor. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[257] Andy Zeng, Shuran Song, Kuan-Ting Yu, Elliott Donlon, Francois R Hogan, Maria Bauza, Daolin Ma, Orion Taylor, Melody Liu, Eudald Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[258] Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. *arXiv preprint arXiv:1810.06045*, 2018.

[259] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *European Conference on Computer Vision (ECCV)*, 2014.