Learning to Grasp

Jacob Varley

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Learning to Grasp

Jacob Varley

Providing robots with the ability to grasp objects has, despite decades of research, remained a challenging problem. The problem is approachable in constrained environments where there is ample prior knowledge of the scene and objects that will be manipulated. The challenge is in building systems that scale beyond specific situational instances and gracefully operate in novel conditions. In the past, heuristic and simple rule based strategies were used to accomplish tasks such as scene segmentation or reasoning about occlusion. These heuristic strategies work in constrained environments where a roboticist can make simplifying assumptions about everything from the geometries of the objects to be interacted with, level of clutter, camera position, lighting, and a myriad of other relevant variables. With these assumptions in place, it becomes tractable for a roboticist to hardcode desired behaviour and build a robotic system capable of completing repetitive tasks. These hardcoded behaviours will quickly fail if the assumptions about the environment are invalidated. In this thesis we will demonstrate how a robust grasping system can be built that is capable of operating under a more variable set of conditions without requiring significant engineering of behavior by a roboticist.

This robustness is enabled by a new found ability to empower novel machine learning techniques with massive amounts of synthetic training data. The ability of simulators to create realistic sensory data enables the generation of massive corpora of labeled training data for various grasping related tasks. The use of simulation allows for the creation of a wide variety of environments and experiences exposing the robotic system to a large number of scenarios before ever operating in the real world. This thesis demonstrates that it is now possible to build systems that work in the real world trained using deep learning on synthetic data. The sheer volume of data that can be produced via simulation enables the use of powerful deep learning techniques whose performance scales with the amount of data available. This thesis will explore how deep learning and other techniques can be used to

encode these massive datasets for efficient runtime use. The ability to train and test on synthetic data allows for quick iterative development of new perception, planning and grasp execution algorithms that work in a large number of environments. Creative applications of machine learning and massive synthetic datasets are allowing robotic systems to learn skills, and move beyond repetitive hardcoded tasks.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I want to thank the great undergraduate and master's students I was given the chance to work and interact with while at Columbia; Kira Whitehouse, Jared Weiss, Alexandros Sigaras, Viabhav Malipani, Jamis Johnson, Robert Ying, Henrique Maia, Danfei Xu, Jiayan Hu, Boyuan Chen, Adam Richardson, Rui Wang, Caroline Weinberg, Chaiwen Chou, Minhaz Palasara, Jorge Guerra, Joaquín Ruales, Avinash Nair, Srihari Sridhar, Jonathan Koss, Aalhad Patankar, and Greg Kocher. I had fun trying out a wide variety of robotics ideas with all of you.

I also want to thank the other Ph.D. students who I overlapped with at Columbia. Jon Weisz, I owe you big time for all your help quickly getting me up to speed with the world of robotics starting from the moment I arrived at Columbia. Yinxiao Li, I really enjoyed traveling to IROS with you, organizing NEMS together, and bouncing ideas off one another throughout our time at Columbia. Fred Rabelo, I had fun organizing the Humanoids class with you. Chad DeChant, I couldn't ask for a better partner in the saga that was getting our shape completion work published. Caroline Yu, exploring the intersection of novel materials, EE, and CS while creating new tactile sensors with you made it abundantly clear to me that robotics is not just a software problem. Iretiayo Akinola and David Watkins, I hope I was able to provide a fraction of the help that was given to me when I entered the lab. I am excited to see what you guys do with your time here.

I also want to express my gratitude to the members of the Columbia University faculty and lecturers who have provided guidance over the last several years. Shree Nayer's lecture style and command of the room for his Computer Vision class has been something I have tried my best to emulate whenever I give a talk. Matei Ciocarli has been a role model whose work and career trajectory has provided an example to aspire to throughout my time here whenever trying to imagine where I would like to be in 5 to 10 years. I have also greatly benefited from my interactions with Eitan Grinspun, Paul Blaer, John Kymissis, Austin Reiter, Liangliang Cao, and James Fan.

I am also thankful for Junho Oh and his lab who hosted me for a summer at KAIST. I enjoyed the time spent with the collaborators I worked with on the BCI project, Alex Barszap, Kenneth Lyons, Sanjay Joshi, Joel Stein, and Ethan Rand. During my time at Clarifai, I learned from some of the best in the field about deep learning and had a front row seat to watch the startup transform into a successful company. I am grateful for the opportunity I had there to work with Matt Zeiler, Adam

Berenzweig, Po-Sen Huang, and David Eigen.

I was fortunate to have Peter Allen as my advisor and mentor. I don't think I can name a place in the world that Peter hasn't visited, or a person in robotics that he doesn't know. His experiences have been invaluable to me in many ways and particularly in guiding me towards exciting questions and areas of research. Peter was always ready for the next lab demo, and kept me working hard to produce results. Looking back, I appreciate his leadership style which was a good balance of oversight and freedom that provided me the opportunity to grow and explore ideas of my own while under his guidance. He has also been a good friend and is always in my corner. I am excited to be joining Peter's family of researchers and collaborators that he has been growing since before I was even born.

Finally I would like to thank my friends, family, parents, brothers, and especially my wife Zefa for all their love and support.

Jake Varley

New York, NY

January 9th, 2018

There are an endless number of things to discover about robotics. A lot of it is just too fantastic for people to believe.

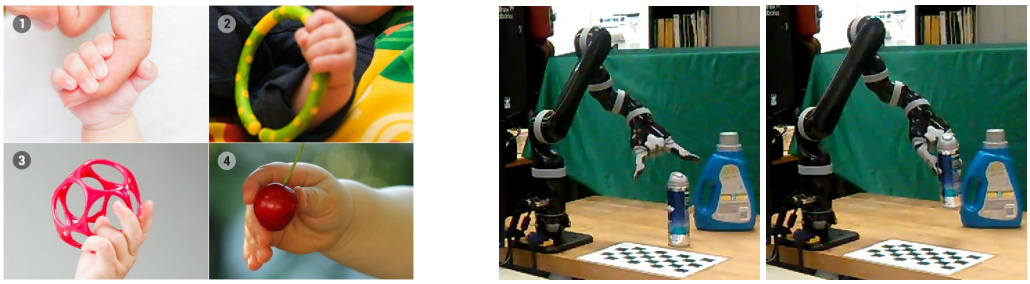
- Daniel H. Wilson

Chapter 1

Introduction

1.1 Robotic Grasping

The field of robotic grasping is easy to describe, it is study of making robots pick things up. One may think that endowing a robotic system with an ability to pick up objects off a table would be relatively easy. After all, this is a skill that young children are able to master within their first several months of life as shown with the robotic equivalent in Figure 1.1.



(a) baby grasps (b) robotic grasp

Figure 1.1: Examples of successful grasping, a task even a small child can accomplish. Despite the task's apparent simplicity, it has been deceptively difficult to engineer robotic grasping systems.

While it is possible to build useful repetitive robotic systems that execute the same motion repeatedly, building general purpose robotic grasping systems is still a difficult engineering challenge. It is hard in a manner that is shared with many other problems in robotics - the complexity of building robotic systems quickly spirals out of control due to interdependencies with a variety of other fields.

Robotic grasping requires competency in perception, planning, and control; with each one of these fields containing many open questions and challenges:

1. Perception - Robotic grasping requires an ability to interpret the rich stream of sensory data and determine what sorts of graspable things are in a robot's environment. How do you build a system that is capable of recognizing the millions of objects that exist in our environment, and even instances of objects the system has never seen before? People are capable of recognizing a new mug as being a mug even if they have never seen that specific brand of coffee mug before. Not only do we have an ability to recognize a vast array of objects, we are also very good at generalizing this knowledge to novel object instances.

2. Planning - After a robotic system has recognized an object that it wants to grasp out of its environment, a grasp plan must be formulated. The formulation of a desired plan for interacting with the world, requires a robotic system to be in possession of semantic information relevant to the task at hand, and have an ability to reference this information efficiently. What is that object I am picking up? What is that object used for? How will I use it? Why am I picking it up? It makes no sense to have a robot pick up a filled coffee cup by sticking its fingers inside of it. How do we tell a robot to be gentle with a wine glass? Or that it should hold a drill in a way that allows it to drill holes. If you have a robot in a factory that only picks up wine glasses and mugs, maybe you can hand craft a system that picks up wine glasses carefully, and mugs by the handle. The challenge is in building a robotic system that can operate in our messy unconstrained world and interact with the millions of objects that we do and possesses the semantics or common sense knowledge required to succeed in our world. A system that is aware of and able to utilize this common sense knowledge will be able to plan robotic grasps that are both stable and semantically useful.

3. Control - Sometimes even the best plan needs to be adjusted. There is always error in camera calibration, sensor models, encoder values, estimates for coefficients of friction, and object masses, among many other things. These errors accumulate, and mean even the best plan will often need to be modified to account for new observations of the environment while we are interacting with it. How can a robot incorporate feedback, and actually realize a planned grasp while making adjustments as new sensory data is collected? Tactile sensor data and

visual data seem like the right sensory modalities for robotic grasping, but visual information is often occluded by the hand during robotic grasping and tactile data is still often incredibly sparse and noisy. How exactly should a robot interpret and react to sensory information while executing a planned robotic grasp is an open question.

In the past roboticists have dealt with this complexity by limiting the scope in which the robotic system was expected to operate, and by encoding common sense knowledge in heuristics. To make perception easier, constraints are placed on what sort of objects will be interacted with, assumptions are made about where they will be placed, and object geometry is provided ahead of time. These constraints allow heuristic methods to work; when you can assume an object on the table is a cylinder of given dimensions, it is much easier to write code for a cylinder detector, than for a general purpose object recognition system. The heuristic, or short cut, is to assume everything is a cylinder. This heuristic works great, as long as the constraint, that everything on the table must actually be a cylinder, holds true. Even a single heuristic assumption such as this makes the design of the entire robotic system much simpler. With objects of known geometry, planning can be be done offline ahead of time and verified by a human. The robot's end-effector can even be specialized for the picking up cylinders task. Most control problems can be avoided by running open loop execution, and simply stopping if anything out of the ordinary occurs, rather than reacting to it by adjusting how the task is executed. This sort of setup has been incredibly valuable for automating warehouses and factories. We have made robotic systems complete useful work by constraining the scope of what they are designed to accomplish, and by encoding the relevant prior knowledge in the form of heuristics.

1.2 Problem Statement

Our problem statement can be formulated as follows:

Robotic grasping systems suffer from high dimensionality in terms of everything; workspace arrangements, lighting conditions, grasp postures, encountered objects, difficult to ground semantics, etc... To cope with this complexity, researchers have developed heuristics based hand crafted systems and apply strong constraints regarding what conditions a system can operate in. These heuristic methods are prohibiting robotic systems from operating outside of controlled lab and factory

conditions where the required hard constraints hold true.

Our Approach

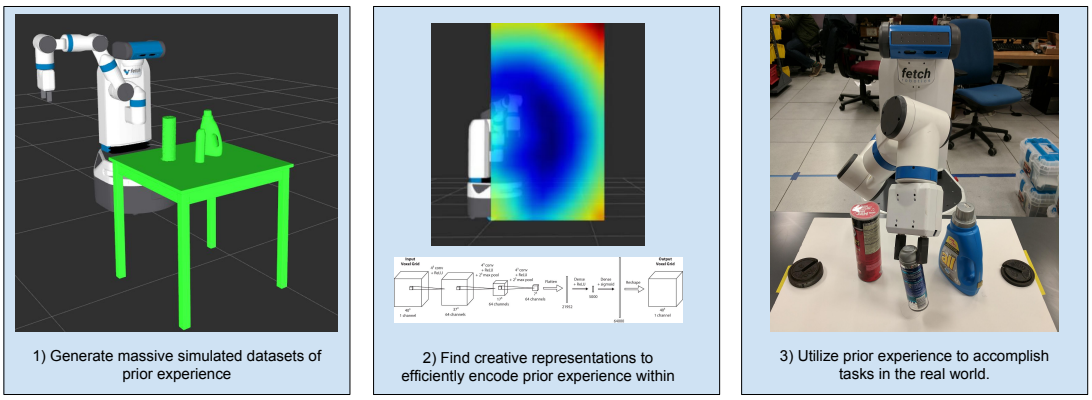


Figure 1.2: Overview of the general approach to robotic grasping from this thesis: Simulate large amounts of prior experience, find good representations for the simulated data, and utilize this stored prior experience at runtime.

There is a need to provide robotic systems with knowledge about a wide variety of environments and objects in order to allow them to operate outside of controlled environments. It is not feasible for roboticists to hand-craft code for every sort of situation a robotic system could find itself in as we currently do now. Rather than implicitly embedding common sense knowledge in handcrafted-heuristics, this thesis looks at how data driven systems can be developed to operate in a wide variety of situations and to modify behaviour based on the data the system was provided. This allows the behavior of the system to adapt when different training data is used rather than requiring code to be rewritten. Gathering training data by running a robotic system in large numbers of new environments is time consuming, expensive, and dangerous. To avoid these problems, the data driven systems in this thesis are all developed using simulated data. The use of simulation allows for the quick, easy creation of massive datasets with data for unique environments and many possible situations. This thesis also explores how to encode the information within this datasets into deep neural networks and other data structures to allow this information to be leveraged quickly and efficiently by the robotic system at runtime. An overview of our general approach to simulate, represent and utilize

prior experience is given in Figure 1.2.

Thesis Contributions

The purpose of this thesis is to create a robust framework for robotic grasping that is data driven and made effective by leveraging simulation to create massive datasets which are then distilled in an effective manner using tools such as a deep learning so that the information can be utilized at runtime. The goal is to aggressively avoid encoding assumptions about the environment and tasks to be accomplished with handcrafted heuristics. The main contributions of this thesis are broken into four chapters as overviewed below:

1.2.0.1 Multi-Fingered Robotic Grasp Planning:

A grasp planner trained with synthetic data whose prior experience is distilled into a CNN. This demonstrates that a grasp planner for a multi-fingered robotic hand can be trained using simulated data to detect good hand configurations based on prior experience. The CNN is used at runtime to generate affordance maps showing where good palm and fingertip locations exist in the scene. These affordance maps are used by a simulated annealing grasp planner to plan stable robotic grasps. Contributions from this section will include:

1. A framework for generating synthetic image patches from simulated robotic grasps that can be used to train a system to recognize good fingertip and palm locations.

2. A data driven grasp planning algorithm for GraspIt!'s simulated annealing planner that learns to recognize good palm and fingertip locations and detect good robotic grasps based on prior experience.

1.2.0.2 Shape Completion Enabled Robotic Grasping:

A novel shape completion algorithm to enable robotic grasping by reasoning about occlusion. In this chapter, nearly half a million synthetically rendered depth images are used to train a 3D CNN to reason about object geometry. This system is able to generalize these examples to reason about novel objects as well. The CNN can be used fill in occluded regions of a scene to enable grasp and motion planning. Contributions from this section will include:

1. A 3D CNN architecture, and method for training the CNN to reason about 3D geometry and occlusion.

2. A training data pipeline for generating voxel grids of partial views of mesh models aligned with voxel grids representing the ground truth occupancy of the respective mesh.

3. A framework for utilizing the trained CNN to fill in occluded regions of a robotic workspace and enable a grasp planner to act of the predicted geometry of an object to be grasped.

1.2.0.3 Fused Visual-Tactile Object Refinement:

A framework for the integration of depth and tactile sensory information into a unified hypothesis of an object's geometry. Here, synthetic depth renderings of objects are paired with simulated tactile contact information to train a CNN to use multi-modal sensory data to reason about 3D occupancy. This is challenging due to the sparse nature of tactile data. The contributions from this section will be:

1. A pipeline for generating synthetic depth and tactile information of robotic grasps.

2. A 3D CNN architecture and method for training the CNN to reason about 3D geometry and occlusion by utilizing both tactile and depth information.

3. A framework for the integration of tactile and depth sensory information into a unified hypothesis of an object's geometry during exploratory robotic actions or grasp execution.

1.2.0.4 Workspace Aware Online Grasp Planning:

A workspace representation that allows a grasp planner to be guided by a learned notion of what grasp configurations are reachable. Offline, in simulation, half a million inverse kinematic(IK) checks are made to determine workspace reachability. This data is used to build a signed-distance field representation of the workspace which can be used to speed up grasp planning. The contributions from this section will be:

1. A novel representation of reachability space using a signed-distance field (SDF) to provide a gradient field during grasp planning.

2. An online grasp planning system with an integrated notion of reachability.

3. A method for embedding obstacles into the reachability space online during grasp planning.

These four chapters highlight several different robotic subtasks where we are able to move from heuristic based methods to data driven approaches. The data driven approaches here are powered by simulated data efficiently embedded in priors, CNN's, and signed distance fields for efficient use at runtime. The robotic systems here learn to accomplish robotic grasping rather than being hardcoded to execute robotic grasps.

Chapter 2

Related Work

This chapter starts with an overview of several related works in the fields of simulation and deep learning. A full overview of these areas is well outside the scope of this thesis, but I will highlight several recent works and discuss how these fields are impacting the way robotic grasping systems are built. Our improved ability to simulate robotic systems makes it possible to produce large amounts of training data for various robotic tasks. Deep learning is becoming the mechanism to encode this simulated data in a manner that can be accessed and used efficiently at runtime.

Section 2.2 discusses various strategies for modeling a robot's self and its surroundings. A common initial step to robotic grasping is to model the object to be grasped, and to make predictions about it's full geometry in the face of occlusion. Several strategies for this geometric reasoning from visual observations are overviewed. In addition, this section covers notions of workspace reachability. Robotic systems can often be sped up if information about what areas of their workspace can be feasibly reached is cached in a useful way.

Section 2.3 overviews different grasp planning paradigms. These methods all produce a planning goal, or a desired hand object configuration. Most of these approaches only consider the relationship between the hand and the object, leaving reasoning about reachability to be done at a later time.

Finally, Section 2.4 covers related works for incorporating sensory feedback into the grasping process. This feedback often comes in the form of tactile and visual sensory observations.

2.1 Fast Moving Fields Related to Robotic Grasping

2.1.1 Simulation

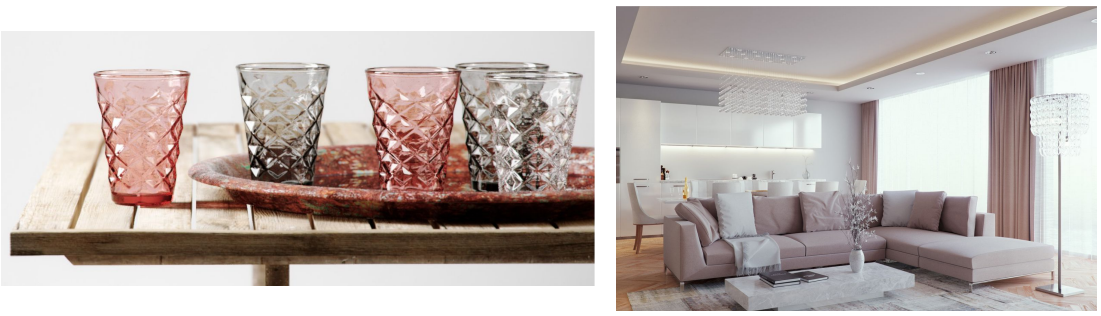


Figure 2.1: Renderings from the Embree high-performance ray tracing kernels, developed at Intel. It is becoming increasingly difficult to discern synthetic images from real ones.

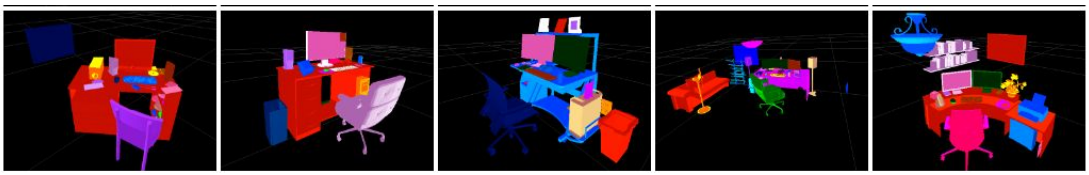


Figure 2.2: SceneNet desk layouts automatically generated via simulated annealing. Realistic simulated environments to train robotic systems can be generated without human hand crafting.

The ability to use synthetic images to train perception systems is quickly maturing due to improved renders such as Embree[Woop *et al.*, 2013], and algorithms for generating realistic scenes such as SceneNet[Handa *et al.*, 2015]. Several example images rendered using Embree are shown in Figure 2.1, and Figure 2.2 shows how environments can be automatically laid out using Scenenet. Simulation of visual data has been used to create perception systems for the Amazon Picking Challenge (APC) [Wurman and Romano, 2015][Mitash *et al.*, 2017][Rennie *et al.*, 2016]. With these systems, it is possible to render massive amounts of realistic labeled data offline that can be used to train a perception system to recognize objects, segment scenes, and reason about occlusions. The ability to leverage simulation in order to enable robotic vision systems is just getting interesting. Systems like OpenAI[Brockman *et al.*, 2016] and Microsoft's AirSim are allowing agents to stream data from virtual environments that at first glance is difficult to discern from "Real World" data.

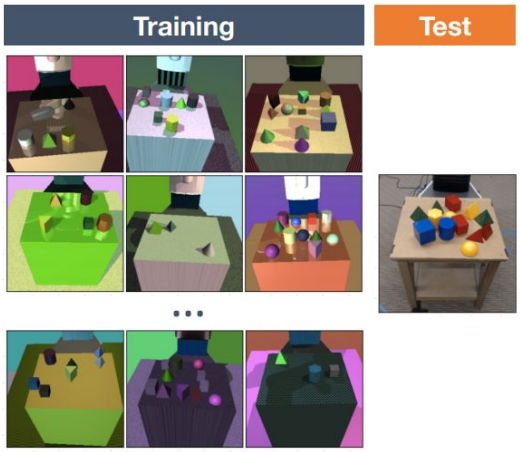


Figure 2.3: Domain randomization training and test renderings from [Tobin *et al.*, 2017]. The system is trained on such a wide variety of conditions, that at test time no additional training is required to work on real sensory data.

An ability to synthetically render large amount of realistic images is incredibly useful for developing robotic perception system. Despite its utility, these powerful rendering tools do not help with the development of new planning algorithms or the training of new grasp execution policies. These problems require a robotic simulator that is true enough to reality that a system trained within it also works well in the real world. This way the results from a new planner, and new policies for grasp execution can be evaluated at scale in simulation. Grasping simulators have been around since the early 2000s [Miller and Allen, 2004]. Current popular simulators include GraspIt! [Miller and Allen, 2004], Klampt[Rocchi *et al.*, 2016], Gazebo[Koenig and Howard, 2004], VREP[Rohmer *et al.*, 2013], OpenRave[Diankov and Kuffner, 2008], OpenGrasp[León *et al.*, 2010], MuJoCo, DartSim, and Bullet.

While robotics simulators and rendering engines have been incredibly successful in enabling vision-based robotic agents. It is just becoming possible simulate other sensory information especially related to sustained contact between rigid bodies at a level that is accurate enough to simulate robotic manipulation tasks. Until very recently, it was not possible to simulate tactile, or force torque sensory data as the underlying models of how contact occurs were not accurate enough in the robotic grasping situations we are interested in. The simulations would often become unstable, or exhibit large amounts of jitter whenever an object was caged by a robotic hand. As our simulators

have improved, different research groups continue to explore enabling grasping systems via more dynamics simulation with several examples including [Kappler *et al.*, 2015][Veres *et al.*, 2017][Dang *et al.*, 2011] [Johns *et al.*, 2016].

In addition, recent works have begin to further explore how to train systems that are agnostic to whether or not the data they are fed in is simulated or real. In [Tzeng *et al.*, 2015], a CNN is trained with multiple loss terms to predict an end effector's location. The additional loss terms ensure that the system, while trained in simulation, performs well on real data. The domain randomization approach taken in [Tobin *et al.*, 2017] and shown in Figure 2.3 ensures that the trained system has been exposed to such a wide variety of environmental conditions and textures, that real-world sensory data is nothing out of the ordinary to the trained system.

2.1.2 Deep Learning

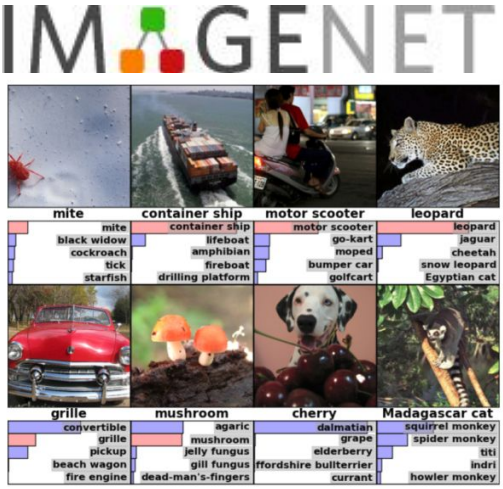


Figure 2.4: Imagenet Examples.

Imagenet is a challenge that evaluates algorithms for object localization and detection from images and videos at scale. Figure. 2.4 shows several example images from the classification competition. In 2012, Imagenet was won by the SuperVision team using a Convolutional Netural Network (CNNs) [LeCun and Bengio, 1995]. Again in 2013, the competition was won by Clarifai using a similar CNN based approach. Since then, deep learning has become an increasingly popular way to build powerful classifiers with applications to a wide variety of areas including computer vision, audio

processing, and natural language processing.

With an eye towards robotics many of the RGB image processing approach have been adapted to the RGB-D domain, for example the per pixel segmentation of [Couprie *et al.*, 2013], and more recently, this work was further improved with the Fully Convolutional Network (FCN) from [Long *et al.*, 2015].

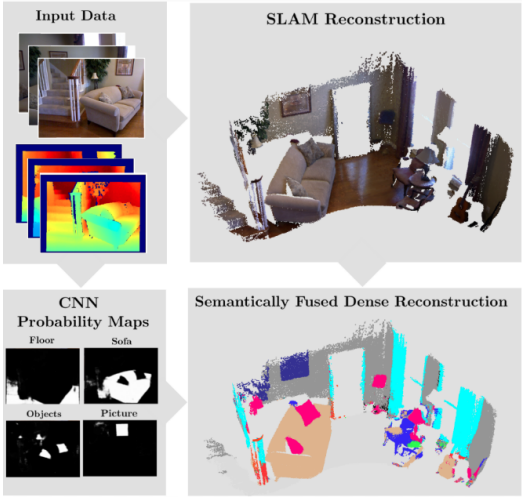


Figure 2.5: Semantic Fusion creates semantically labeled SLAM maps via an FCN.

These technique has already begin to be ported over to robotics. The per pixel classification algorithms are being paired with SLAM to create rich semantic maps for robot navigation [McCormac *et al.*, 2016]. These approaches are also used by teams involved in the Amazon Picking Challenge [Zeng *et al.*, 2016]. These convolutional methods are also being used to enable grasp planning and execution as discussed further in the sections below.

2.2 Representations of Robotic Workspaces

2.2.1 Reasoning about Object Geometry with Vision

Once a robotic system has segmented out the points related to the visible portion of an object, and it can be helpful to reason about the occluded regions of the object. There has been a large amount of work in object modeling. Prior approaches followed two main schools of thought. Either instance recognition which often involved a database of objects to be detected in the scene. These approaches

usually take advantage of some variant of RANSAC or ICP [Papazov and Burschka, 2010]. The second school of thought belongs to more general completion heuristics, these often involve finding axes of symmetry or axes of extrusion and using those to complete the object [Quispe *et al.*, 2015]. The instance based methods tended to be limited to at most tens of objects and fail to translate to novel objects. The general completion methods fail on objects more complicated than simple geometric primitives.

Since 2016, more advanced data driven techniques using machine learning have developed both to reason about occluded geometry, and to localize known objects. In [Firman *et al.*, 2016] a Random Forrest approach is used to generate a voxel grid of the full scene given a voxelized grid of the scene as shown from a single perspective. A similar approach was taken by [Song *et al.*, 2016] where a 3D CNN was used to decide whether occluded voxels were occupied or not. In [Zeng *et al.*, 2016], a 3D CNN is used to estimate the poses of APC objects.

2.2.2 Reasoning about Workspace Reachability

A 2016 review of the Amazon Picking Challenge (APC) [Correll *et al.*, 2016] explicitly highlighted the fact that many teams opted against online grasp planning. This was due in part to the fact that grasp planning metrics typically ignore the notion of reachability, a large percentage of the planned grasps end up being unreachable and not useful for the robot. Hence, an extra computational step is required to check each of the planned grasps for reachability; a top performing APC team [Hernandez *et al.*, 2016] uses this post-check solution which shows that this limited partial solution is built into even competitive systems. This post-check solution has a number of drawbacks: first, the post-processing check for reachability might not contain any reachable grasps which implies that the entire grasp planning process has to be repeated. Another closely related problem with post-reachability checking is that more time is incurred when replanning grasps that fail the reachability test stage. Also, many systems, such as those in APC, are highly tuned to specific use-cases which do not generalize well especially when the grasping environment changes in small ways.

Many grasp planning platforms such as GraspIt![Miller and Allen, 2004], OpenRave[Diankov and Kuffner, 2008], and OpenGrasp[León *et al.*, 2010] only consider properties specific to the end-effector during grasp planning. Given the hand description and the object, these platforms compute grasp quality for different hand-object configurations using metrics such as force/form closure and

return the best quality grasps found. These approaches ignore reachability of the resulting grasp as often the arm is not even modeled in the simulator, and leaves the burden of verifying the grasp outputs to the subsequent stages of robotics applications. Miller etal. [Miller *et al.*, 2003] paper mentions the concept of arm reachability in grasp planning but does not discuss any details neither does it explicitly input reachability as an optimization objective in grasp planning.

An early example of a reachability space is demonstrated in [Berenson *et al.*, 2007]. In their work, they use an *"Environment Clearance Score"* to guide grasps away from obstacles in the robot workspace. The same work introduced the idea of *"Grasp-scoring function"* which uses an approximate kinematics of the robot to score a grasp.

The concept of offline reachability analysis exist in the literature [Vahrenkamp *et al.*, 2009] [Porges *et al.*, 2014]. However, most of this work uses offline precomputed grasps.

[Haustein *et al.*, 2017] has recently looked at combining grasp planning and motion planning. Their method uses a bidirectional search approach to randomly sample grasp candidates in one direction and sample arm trajectories in the opposite direction till a connection is made to form a hand grasp/arm trajectory pair.

2.3 Grasp Planning Paradigms

There is a significant body of work related to grasp planning with incomplete sensor data as well as different metrics for generating grasps for known objects for which we overview here while also referring the reader to [Bohg *et al.*, 2014]. Trade offs in these different grasp planning approaches include considerations such as planning speed, whether the object geometry must be known ahead of time, whether the target object's geometry will be modeled at all, the sophistication of the end-effector, and constraints on allowed approach directions.

2.3.1 Simulated Annealing Grasp Planning

One common approach to grasp planning is to anneal through a search space. A point in the space represents a specific hand-object configuration. Different heuristics related to force closure can be used to evaluate points within this space, and simulated annealing can be used to navigate towards strong local minima which represent stable grasp locations. This approach is well represented in

[Ciocarlie *et al.*, 2007] where the eigen-grasp dimensionality reduction technique is used to reduce the size of the search space and allow the planner to quickly arrive at good grasp configurations.

2.3.2 Data Driven Grasp Planning

In the approach taken by [Goldfeder *et al.*, 2009b][Goldfeder and Allen, 2011], a grasp planner such as the simulated annealing planning approach from the previous section plans grasps that maximize a given quality metric offline to generate grasp for known 3D objects. Then, when attempting to grasp an object in a scene, the scene is segmented, and the segmented object's closest model is found in the database and the precomputed grasps from the closest matching model are used on the new object. This approach works reliably when the object to be grasped can easily be segmented out of the scene and is not markedly different than models in the database, but it is difficult to fine tune a retrieved grasp with this system especially when the full geometry of the object is unknown. Other related works in this area include [Detry *et al.*, 2013][Kopicki *et al.*, 2014].

2.3.3 Deep Learning for Parallel Jaw Grasp Planning

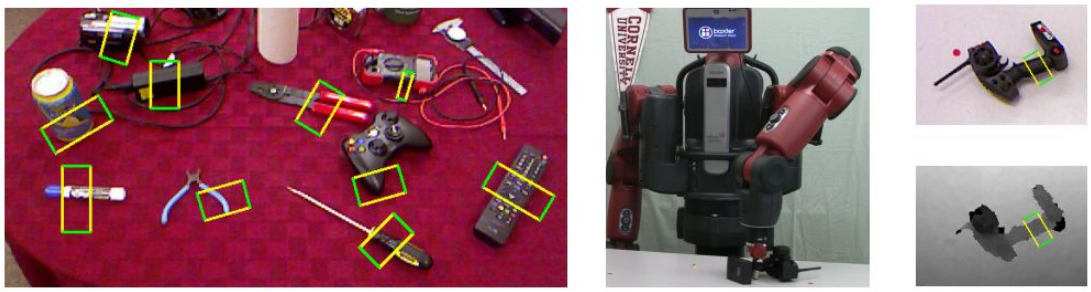


Figure 2.6: The grasping rectangles approach from [Lenz *et al.*, 2014]. A CNN is trained to recognize good grasping positions for a parallel jaw gripper approaching from above.

In another approach [Lenz *et al.*, 2014][Lenz *et al.*, 2015] [Jiang *et al.*, 2011] [Johns *et al.*, 2016] [Pinto and Gupta, 2015] the grasping system learns grasps for a parallel jaw gripper from labeled images. The labels consist of rectangles with two of the edges representing where the gripper should be placed as shown in Figure 2.6. The grasping rectangles approach was a successful adaptation of the CNNs from Imagenet to robotic grasping for the Baxter robot. This approach, while able to

learn arbitrary grasp types based on whatever training data is given, is not easily extensible to more complicated robotic hands and only allows grasps to be planned from above. This approach has been thoroughly researched, and more recently several groups have successfully explored generated training data for these systems entirely in simulation [Kappler *et al.*, 2015] [Johns *et al.*, 2016].

2.3.4 Fingertip Space and Precision Grasp Planning

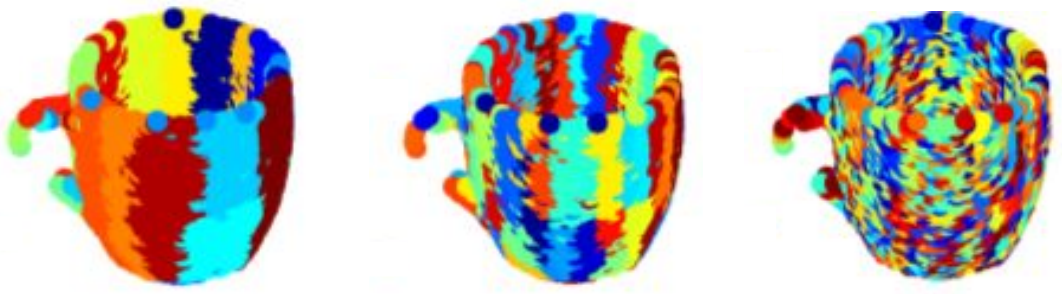


Figure 2.7: The hierarchical fingertip space from [Hang *et al.*, 2014]. A grasp planner anneals through the different levels of the fingertip space, from a coarse to a fine-tuned hand configuration.

The concept of a Fingertip Space and an efficient algorithm for producing stable grasps within this space has been proposed by a number of researchers including [Hang *et al.*, 2014][Rosales *et al.*, 2011][Lin and Sun, 2015]. For example, [Hang *et al.*, 2014] can synthesize precision grasps as long as the observed parts of the object are graspable as shown in Figure 2.7. This stands in contrast to work in this thesis where information about the observed parts of the object is supplemented with knowledge gained from prior grasps generated on known objects.

2.4 Incorporating Sensory Feedback

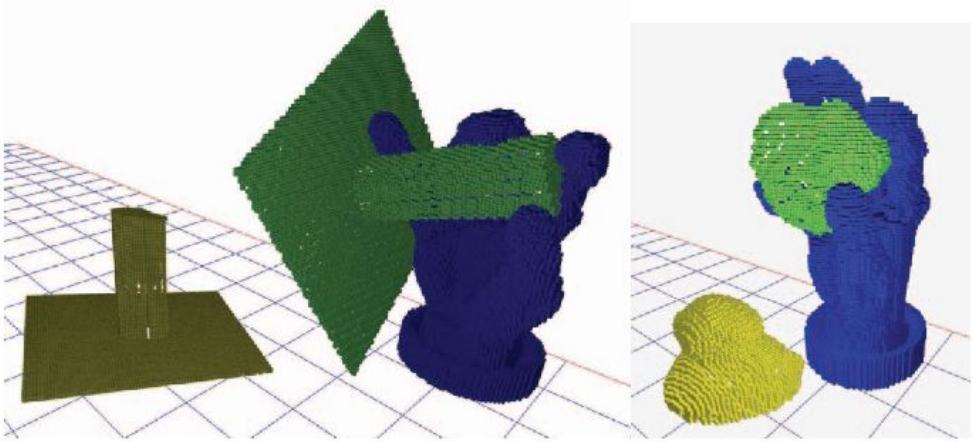
The idea of incorporating sensory information from vision, tactile and force sensors has been around for a long time [Miller and Leibowitz, 1999]. Despite the intuitiveness of using multi-modal data, there is still no agreed upon framework to best integrate multi-modal sensory information in a way that is useful for robotic manipulation tasks. In this thesis, we are interested in reasoning about object geometry and grasp stability in particular.

2.4.1 Tactile Geometric Refinement

Several recent uses of tactile to improve estimates of object geometry has focused on the use of Gaussian Process Implicit Surfaces(GPIS) [Williams and Fitzgibbon, 2007]. Several examples along this line of work include [Caccamo *et al.*, 2016][Yi *et al.*, 2016] [Bjorkman *et al.*, 2013][Dragiev *et al.*, 2011][Jamali *et al.*, 2016][Sommer *et al.*, 2014][Mahler *et al.*, 2015]. This approach is able to quickly incorporate additional tactile information and improve the estimate of the objects geometry local to the tactile contact or observed sensor readings. There has additionally been several works that incorporate tactile information to better fit planes of symmetry and super quadratics to observed pointclouds [Ilonen *et al.*, 2014][Ilonen *et al.*, 2013][Bierbaum *et al.*, 2008]. These approaches work well when interacting with objects that confirm to the heuristic of having clear detectable planes of symmetry or are easily modeled as super quadratics.

2.4.2 Continuous Geometric Refinement

There has been successful research in utilizing continuous streams of visual information similar to Kinect Fusion[Newcombe *et al.*, 2011] or SLAM[Thrun and Leonard, 2008] in order to improve models of 3D objects for manipulation. One example being [Krainin *et al.*, 2011b][Krainin *et al.*, 2011a] In this work, the authors develop an approach to building 3D models of unknown objects based on a depth camera observing the robot's hand while moving an object. The approach integrates both shape and appearance information into an articulated ICP approach to track the robot's manipulator and the object while improving the 3D model of the object. Similarly [Hermann *et al.*, 2016] attaches a depth sensor to a robotic hand, and plans grasps directly in the sensed voxel grid as shown in Figure 2.8. The use of a voxel grids rather than meshes significantly speeds up the grasp planning process and the representation lends itself more naturally to GPU programming and allows the system to quickly incorporate new observations. These approaches improve their models of the object using only a single sensory modality, but from many time points.



(a) Pinch-grasping a cylindrical object while regarding floor (b) Power-grasping a rubber duckie

Figure 2.8: The grasp planner from [Hermann *et al.*, 2016] operates directly on a voxel grid rather than mesh of the object, this allows for incredibly fast planning on a GPU.

2.4.3 Grasp Stability Estimation

There has been a massive amount of work related to producing analytical definitions of grasp stability which are well reviewed in [Roa and Suárez, 2015]. More recently the applicability of analytical approaches to stability analysis has been debated. With [Weisz and Allen, 2012] demonstrating uncertainty in the pose of the object in relation to the hand can cause planned grasp quality measures to correlate poorly with executed grasp quality. [Krug *et al.*, 2016] has shown that while the planned grasp quality measure may not correlate with the executed grasp quality, the executed grasp quality can be accurately estimated and predict the success of lifting an object when using tactile sensors to localize the true contact locations.

In addition to the analytical definitions of grasp stability, there has also been a large number of data-driven approaches utilizing sensory data to predict whether a grasp will result in a successful lift. These approaches do not explicitly reason about the hand-object configuration. Several recent notable examples of these works include [Cockbum *et al.*, 2017][Madry *et al.*, 2014][Dang and Allen, 2014][Levine *et al.*, 2016].

Chapter 3

Heuristic Grasping

In this chapter, the robotic grasping system used in the Columbia University Robotics Lab prior to the contributions of this thesis is overviewed, highlighting both some of the systems strengths and its weaknesses. This robotic system is highly representative of many robotic systems at various labs around the world. Most labs have an equivalent system in place, while research is conducted to improve a portion of the framework whether it is the introduction of a new perception algorithm, a new grasp planner, or a new control policy for grasp execution. A high level overview of the Columbia setup is shown in Figure 3.1, and discussed in the following sections.

3.1 Perception

Our workspace consists of a table on which objects to be manipulated are placed. Augmented Reality (AR) tags are fixed to the table at a known position and orientation relative to the base of the robot. When the tags are viewed by the camera, we are able to determine the series of transformations that relate the robot to the table, and the table to the camera. In order to perceive objects, the pointcloud from the camera was converted into the table's frame of reference, and the points that were part of or below the table were removed via a pass through filter. This process leaves all the points associated with objects sitting on the table in a new filtered cloud while removed all other points. This cloud was then passed to a RANSAC based method from [Papazov and Burschka, 2010]. This CUDA based method detects mesh models from pointclouds using a database of mesh models, it outputs the names and poses of detected meshes.

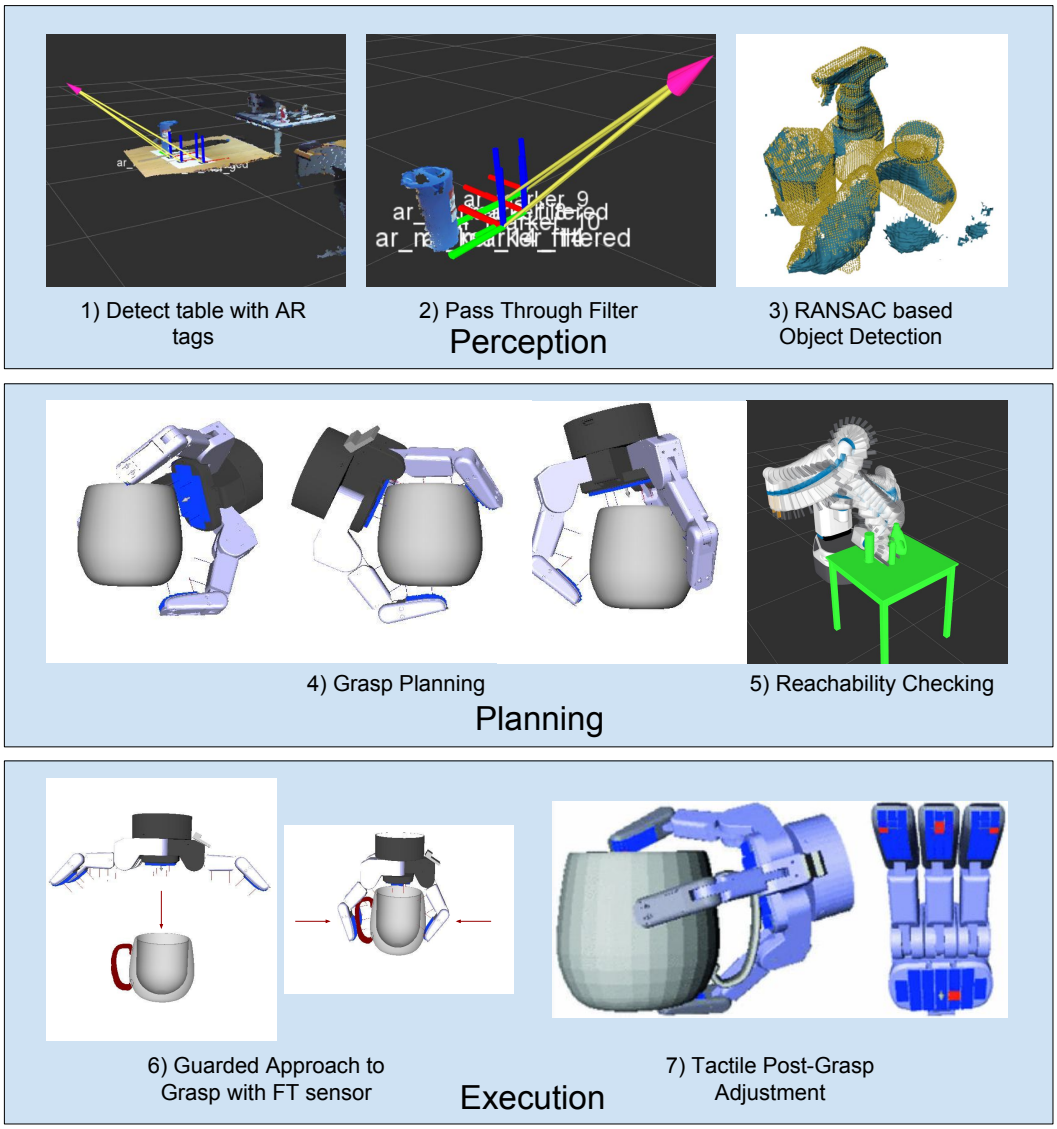


Figure 3.1: A robotic grasping framework representative of the setup in many labs. The Perception component segments out and reasons about the geometry of an object to be grasped. The Planning component includes both grasp and arm-motion planning. The Execution component realizes the planned grasp, and incorporates sensory feedback.

The method required a relatively uncluttered work environment resting on a table in order for the initial pass through filter to work. The RANSAC recognition database approach works incredibly well as long database contained around 5-10 mesh models. If more meshes were loaded in it, the system would either run out of memory or would quickly begin to hallucinate objects everywhere and provide nonsensical results. This meant that our system was unable to handle novel objects, and unable to handle more than a handful of objects even if they were known ahead of time.

3.2 Grasp Planning

Once the objects in the scene have been detected, all the detected objects, as well as a model of the table are loaded into GraspIt!. A target object to be grasped is then selected either programmatically, or by a human user. A set of precomputed grasps for the target object would be retrieved from the Columbia Grasp Database. These grasps would be ranked based on their ability to resist external forces and torques using Graspit. Then the list of grasps would be traversed in order checking for reachability using MoveIt!. If no reachable grasps were found, then GraspIt!'s Online Simulated Annealing planner would be run in order to plan additional grasps.

Another common problem with the RANSAC based approach for recognizing objects was that it would often detect roughly symmetrical objects as upside-down. This made it very difficult to precompute semantically meaningful grasps as objects were often detected in poses that, while aligned somewhat well with the pointcloud of the scene were very incorrect. For example a mug detected as upside down could still overlap a lot with a right-side up mug, but the way the robot should plan to interact with these two objects is very different. This flip in orientation would place any useful grasps that should be approaching the object from above as approaching the upside object from below and would be invalid due to collisions with the table. In addition, whenever the online planning method had to be used, it was significantly slower than other components of the grasping pipeline taking 20 seconds to plan a grasp. These grasps were planned using the Contact and Potential Energy Function within GraspIt!. This heuristic, while a reasonable proxy for grasp stability has no notion of semantics, meaning it often produced grasps that while potentially stable would prohibit the system from accomplishing any further task with the grasped object.

3.3 Grasp Execution

Once a reachable grasp was found, it would be executed. This grasp execution process would incorporate minimal feedback from the attached Force Torque Sensor, and tactile sensors on the hand. The move of the hand towards the object, was guarded, and would stop prematurely if contact was registered via the Force Torque Sensor. Additionally, work has been done to utilize the tactile sensor on the hand. Once a grasp was completed, the tactile sensor values would be used to determine if the current grasp was stable, and update the grasp posture if the current configuration was found to be unstable [Dang and Allen, 2014].

The largest weakness in this grasp execution process is the lack of feedback. The only use of the Force Torque sensor is a simple heuristic to stop the hand from directly knocking over the object. The only use of the tactile sensors is to evaluate each grasp after it has been executed and to determine if an adjustment needs to be made as opposed to incorporating tactile feedback during the grasping process. In addition, there is no visual feedback of any kind.

3.4 Where is the Common Sense?

We argue that this current system does has common sense, or access to prior experiences that it can utilize to assist in perceiving, planning for, and acting in the system's environment. Much of the system's common sense exists in hard-coded heuristics which only operate under certain conditions. For example, the perception system requires a constraint that the objects are all resting on a table. When this constraint is satisfied, it allows for the use of a heuristic to remove uninteresting points from the pointcloud, the pass through filter. For grasp planning, constraints include the assumption that the objects are rigid bodies with uniformly distributed mass. This constraint allows us to use GraspIt!'s Eigen Grasp planner which computes grasp quality metrics dependent on these assumptions using heuristic estimates of a given grasps ability to resist external disturbances.

Over the course of this thesis we will aggressively work to remove these hard-coded heuristics and move from the system shown here towards a grasping system that has common sense in the form of data-driven representations powered by simulation. This means that they are adaptable to new situations and can have their behaviour changed by introducing new data rather than having to rewrite any code. They also represent case studies in how large amounts of simulated data, which is

much cheaper to collect that real data, can be leveraged to make robotic systems work in the real world. These large bodies of simulated data distilled into CNN's, Signed Distance fields, and other representations compose the systems common sense, the prior experiences on which it relies when operating in new situations.

For example, we will show in later chapters how the RANSAC-based perception algorithm can be replaced with a CNN trained to accomplish shape completion. This new approach is able to handle novel objects, as well as hundreds of different objects rather than 5-10. Also, the performance of the CNN is modified by changing its training data rather than rewriting any code. We will demonstrate how we can use an additional CNN during the grasp planning stage to encode notions of stability as well as knowledge of what a good grasp looks like rather than explicitly computing heuristics related to stability. This thesis will also show how a learned notion of reachability can further speed up online planning time and move the planners attention to regions of the workspace that are feasible. In addition, this thesis will demonstrate how simulated tactile and depth information can be used to improve the systems model of the environment during grasp execution providing valuable feedback about the environment.

Chapter 4

Multi-Fingered Robotic Grasp Planning

This chapter presents a system relying on deep learning methods to generate grasps for objects that may or may not have been seen before and whose current full geometry is unknown. The proposed system uses a Convolutional Neural Network to quickly calculate grasp affordances such as force closure [Ferrari and Canny, 1992] directly from RGBD images. These methods make our system efficient, allow the use of a large context window for determining grasp quality, and enable the system to generalize precomputed grasps for known models to novel 3D objects.

The proposed approach involves training a deep learning network on thousands of quality grasps generated using known object models from BigBIRD[Singh *et al.*, 2014], the Berkeley Instance Recognition Dataset. Training grasps are generated in an environment where the full object geometry is known using GraspIt![Miller and Allen, 2004], a grasp simulator. For each grasp, the locations of the fingertip positions and palm are recorded along with the grasp quality score representing the stability of the given grasp for this known environment. Then, RGBD images of the grasp can be rendered, and RGBD image patches centered on the fingertip and palm locations can be extracted. These patches paired with the recorded grasp quality score are used to train a deep network to recognize stable fingertip and palm locations for different canonical grasp types directly from RGBD images.

At runtime, an RGBD image and a pointcloud for a scene are generated from a single point of view. The pointcloud is segmented and meshes are created for the segmented objects. Since the pointcloud was generated from a single view, these meshes are incomplete, and represent only the faces of the object that were visible to the camera. At the same time that the pointcloud is processed,

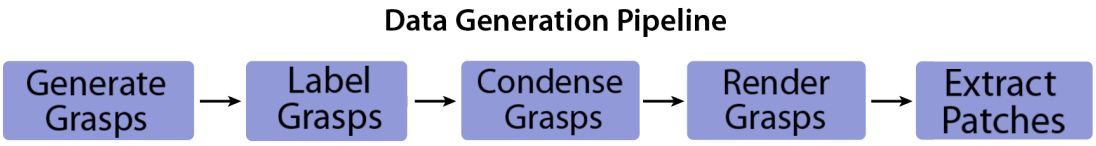


Figure 4.1: Overview of the grasp training data generation stages. Initially, a large number of grasps are generated offline using GraspIt!'s simulated annealing planner. These grasps, are then labeled with a grasp type. Only the 8 most common grasp types are kept. These canonical grasp examples are rendered in Gazebo, and image patches around the finger tip and palms are extracted.

the RGBD image passes through the deep learning network in order to generate heatmaps or dense per pixel labels revealing how well every location in the image would act as a fingertip or palm location for a number of different canonical hand configurations. The incomplete meshes are then imported into GraspIt!, and Simulated Annealing is run to find stable grasps that align well with the visible portions of the mesh model to be grasped, and whose fingertips and palm positions project into low energy locations in the heatmaps produced by the deep network.

4.1 Grasp Training Data Generation

The generation of training data is a multi-stage process. Grasps are generated in GraspIt! using complete object models. The locations of the palm and fingertips for grasps in prototypical hand configurations are saved along with the energy of the given grasp. RGBD images of these palm and fingertip locations are then captured in simulation, and patches around these locations are saved. An overview of the data generation process is shown in Figure 4.1.

4.1.1 Grasp Generation

Training of the grasp generation system required the generation of quality grasps on known objects. In order to generate these grasps, models from BigBIRD, the Berkeley Instance Recognition Dataset, were uploaded into GraspIt!. The dataset contains mesh models for approximately 125 everyday objects. Several of the models had large holes due to transparent regions and were not used as valid training grasps could not be generated for them. In GraspIt! each model was uploaded along with a Barrett Hand model as shown in Figure 4.2. From each object, points were chosen uniformly over

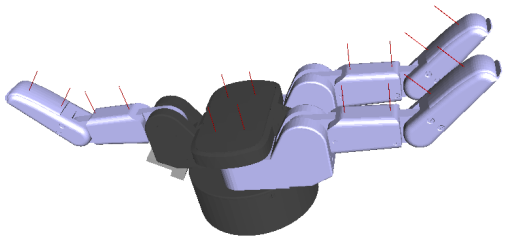


Figure 4.2: Barrett Hand with virtual contacts shown in red. The hand has 7 joints, but only 4 degrees of freedom. The virtual contact points are used within GraspIt!'s simulated annealing planner, where a grasp's quality is dependent on how close these points are to a target object, and how well aligned they are to the surface of the object.

the mesh surface and GraspIt!'s Simulated Annealing planner ran for 30 seconds at a time seeded over the chosen points on the object. For the grasp data generation, GraspIt!'s Contact and Potential Energy function was used to score the generated grasps. The Potential Energy function ensures that the generated grasps exhibit force closure. Only the grasps with low-energy quality scores below a threshold were kept. This generated 41,708 grasps with quality scores below our threshold to be used for training.

4.1.2 Grasp Type Labeling

The generated grasps were then categorized into different grasp types based on the 7 joint values and the roll of the wrist. Each joint value of the hand and the wrist roll were given $nBins = 5$ bins. This provided $nBins^{nJoints} = 5^8 = 390,625$ different grasp types. The grasp type is simply an integer between 0 and 390,624 that is determined by the coarse configuration of the hand. The exact formulation is given by:

$$grasp\_type = \sum_{i=0}^{nJoints-1} bin(joint[i]) * nBins^i \tag{4.1}$$

4.1.3 Grasp Type Reduction

The vast majority of the above grasp types had no grasp examples. For instance, the grasp type corresponding to the hand being completely open never results in a stable grasp. Only the $n = 8$

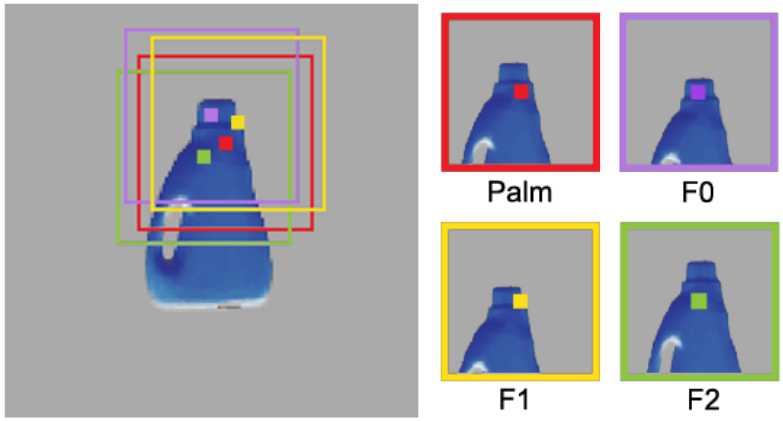


Figure 4.3: Image patches extracted from a grasp on an All brand detergent bottle. The center of each image patch corresponds to where the palm and fingertips from the grasp project into the image plane. These for images will be used to train a CNN to recognize good palm and image locations for the canonical grasp type associated with this grasp.

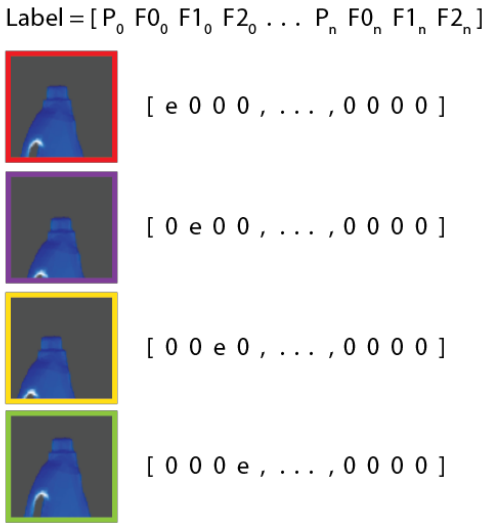


Figure 4.4: The four image patches and associated label vectors for a single grasp with energy e. The enery is used in the label vector as a one-hot encoding.

Layer	1	2	3	4	5	6	7	8
Stage	conv + pool	conv + pool	conv	conv	conv	full	full	full
# channels	96	256	256	256	256	1000	100	$f * n$
Filter Size	11x11	3x3	3x3	3x3	3x3	-	-	-
Filter Stride	2x2	-	-	-	-	-	-	-
Pool Size	2x2	2x2	-	-	-	-	-	-
Pool Stride	2x2	2x2	-	-	-	-	-	-
Input Size	480x640	118x158	58x78	56x76	54x74	52x72	52x72	52x72

Table 4.1: Architecture for the model. The final number of channels in the final stage is the number of grasp types (n) multiplied by the number of interest points on the hand (f) i.e the palm and fingertips. Pooling refers to max pooling.

most common grasps types were kept as the set of canonical grasps. The set of 41,708 quality grasps from above contained 3,770 grasps of the 8 most common grasp types. The number of examples per grasp type was no longer sufficient past the 8th most common grasp type.

4.1.4 Grasp Rendering

For each grasp, the target object was placed into Gazebo[Koenig and Howard, 2004]. The position of the palm with relation to the object was calculated and a simulated Kinect sensor was positioned 2 meters backed off from the object along the approach direction of the hand and an RGBD image was captured.

4.1.5 Training Patch Extraction

From each RGBD image, patches around the location of the palm and fingertips are extracted. These training patches allow the deep learning model to learn what good fingertip locations look like for specific grasp types, for a known object, from a specific viewing angle. Figure 4.3 demonstrates how 4 patches are generated from a single grasp of a known model.

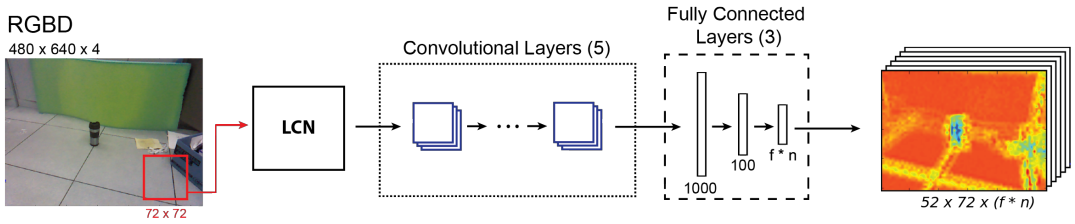


Figure 4.5: Deep Model with local contrast normalization (LCN), 5 convolutional layers, and 3 fully connected layers. The input to this network is a 480x640 RGBD image. The output is a set of 52x72 heatmaps representing good palm and fingertip locations within the scene for each of the canonical grasp types.

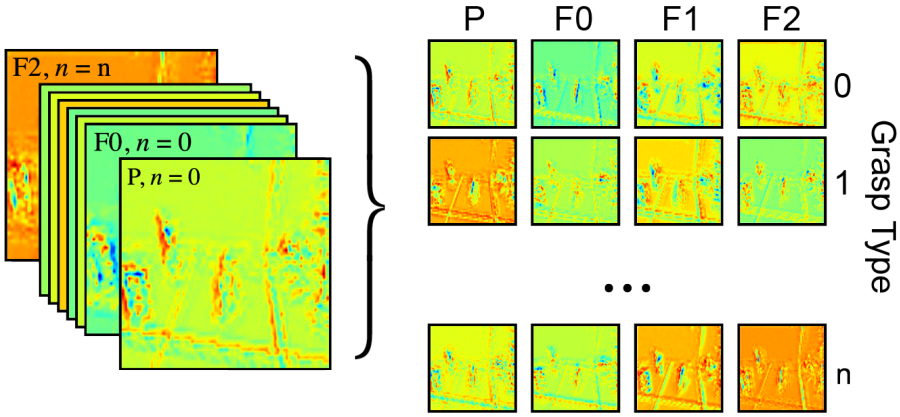


Figure 4.6: The output of the Deep Network can be thought of as a set of heatmaps representing how well each location in the image acts as a fingertip (F0, F1, F2) or palm (P) location for each of the n grasp types. At runtime, the set of four heatmaps for the canonical grasp configuration closest to the current grasp posture will be used to determine the current hand configuration's energy.

4.2 Deep Learning Model

4.2.1 Training Data Labels

For each training patch, a label vector the length of the number of canonical grasp types (n) multiplied by the number of palm and finger tip locations (f) is created, i.e. $len(\text{label\_vec}) = n*f = 8*4 = 32$ The vector contains zeros except at the entry corresponding to the grasp and finger type of the patch. This entry is filled with the grasp quality score for the grasp that this patch was extracted from. Figure 4.4 shows how the training labels for a set of patches from the same grasp are composed.

4.2.2 Model Description

Our model was implemented using Pylearn2 [Goodfellow *et al.*, 2013] and consists of a local contrast normalization stage(LCN) [Jarrett *et al.*, 2009] followed by five Convolutional Rectified Linear layers and then by three fully connected Linear layers. The model was trained using an Nvidia GTX 780 GPU. The parameters of the model are shown in Table 4.1, and the architecture is shown in Figure 4.5. This network is small compared to networks built for the ImageNet challenge [Russakovsky *et al.*, 2014], which have 1000 output categories compared to our 32.

4.2.3 Heatmaps

While the model was trained on image patches of 72x72, when running it takes as input an entire RGBD image (480x640) and outputs a label vector for every pixel in the image with the exception of the border pixels where a full 72x72 patch is not available. The output matrix is downsampled due to the Filter Strides and Max Pooling in the first two Convolutional Layers. This output matrix can be thought of as a set of potentials showing how well every pixel would work as a fingertip location for every fingertip type for each of the canonical grasp types. Figure 4.6 demonstrates how the output of the deep learning model can be interpreted.

4.2.4 Training

The model is trained using stochastic gradient descent (SGD) with minibatches of size 20, for 151 epochs. The learning rate was .1, and an L1 term was used to induce sparsity in the weights. In addition, small amounts of Gaussian noise was randomly added to the training patches.

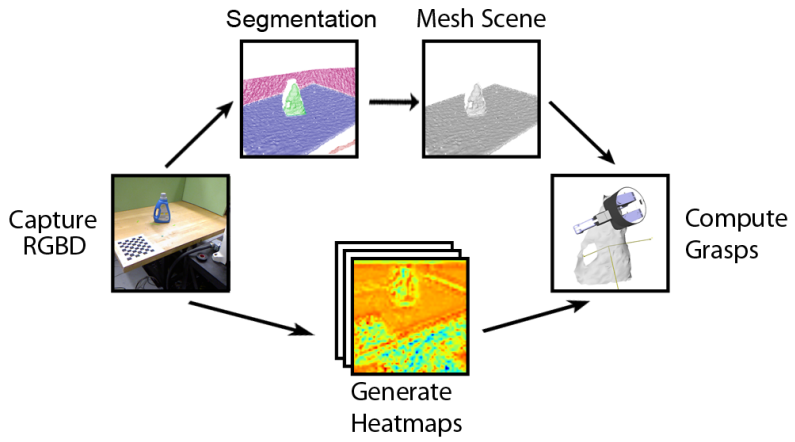


Figure 4.7: Overview of the Grasp Generation System. After an image is captured, the scene is segmented and meshed while the heatmaps are generated in parallel. The meshed scene and the heatmaps are used within GraspIt! to plan grasps.

4.3 Grasping from Partial Views

In order to realize a grasp of an object using the model above, several additional steps must be taken. An RGBD image must be captured, the scene must be segmented, and meshes must be generated for the segmented objects in the scene. Then Simulated Annealing can be run within GraspIt! on the mesh of the scene in order to find an optimal grasp that is well aligned with the visible portions of the object and is stable according to the deep network. An overview of this process is shown in Figure 4.7.

4.3.1 Scene Segmentation

The system segments the point cloud associated with the RGBD image using Euclidean Cluster Extraction [Rusu and Cousins, 2011][Rusu, 2010]. The largest plane in the pointcloud is found using RANSAC and then removed. Euclidean clustering is then run on the remaining points, and any objects that were previously resting on the largest plane in the scene now form separate clusters.

4.3.2 Mesh Generation

The segmented clusters from the previous step are then meshed and the set of meshed partial objects is returned.

4.3.3 Heatmap Generation

In parallel with the generation of the meshes, the RGBD image is sent through the deep learning network. A given input RGBD image creates a set of $f * n$ heatmaps, one for each of the f finger and the palm locations for each of the n canonical grasp types.

4.3.4 Grasp Planning

Within GraspIt!, the meshes are loaded into a planning scene, a partial mesh is selected for grasp planning, and GraspIt!'s Simulated Annealing planner is started [Ciocarlie and Allen, 2009]. At each step, the planner calculates the energy associated with the current hand configuration as a weighted sum of a Contact Energy term and the Heatmap Energy term, both of which are described in the following section. The grasp planner stores a list of the top grasps ranked by energy that are calculated over a pre-specified number of iterations of the planner. At this point, either the lowest energy grasp that is reachable can be executed automatically or a user can select a preferred grasp from the list generated by the planner.

4.4 Grasp Energy Calculation

4.4.1 Contact Energy

The Contact Energy function makes use of Virtual Contacts, or user-specified points of interest on the hand as shown in Figure 4.8. In our case, we spread a total of 16 virtual contacts uniformly over the Barrett Hand. The Contact Energy metric penalizes grasps whose virtual contacts are either distant from the object or whose normals do not align well with the object. The algorithm is fully described in Algorithm 1. The two components come from the distance between the closest point on the object to be grasped, and the dot product of the normalized vector extending from the virtual contact location to the closest point on the model with the normal of the virtual contact and the hand.

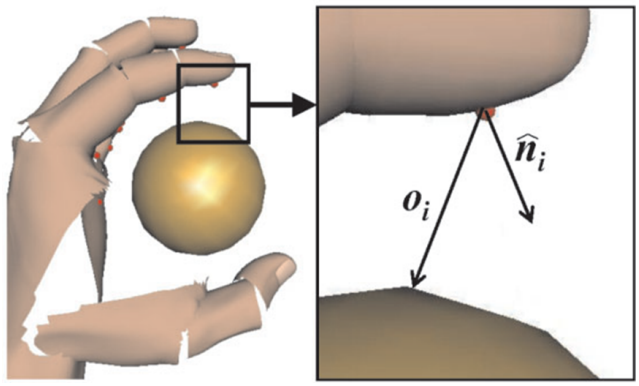


Figure 4.8: Minimizing Contact Energy is equivalent to minimizing both $o_i \cdot n_i$ and the length of o_i to stay close to and aligned with the object.

γ is used to scale how large of a contribution the alignment term makes relative to the distance term. Two versions of the isVisible function were explored. For forward facing grasps using only the 4 virtual contacts on the palm was sufficient, while for all cases, the orientation of the vector extending from the virtual contact to the nearest point on the mesh was compared to the vector extending from the camera location to the same point. The sign of the dot product was used to ensure that contact was being made on a visible face of the mesh. The Contact Energy term encourages grasps to align well with the visible portions of the meshed object.

4.4.2 Heatmap Energy

The second component of the energy function comes from the heatmaps generated by the deep learning network. The distance from the current hand configuration to the nearest canonical grasp type is computed. If there are no nearby grasp types, then the Heatmap Energy is returned as a large value implying that the current hand configuration is not well associated with any of the learned grasp types. If there is a nearby grasp type, then the corresponding heatmaps are used to determine the Heatmap Energy. The current locations of the palm and finger tips of the hand are projected back into the plane of the image captured by the Kinect, and the heatmap values for the individual fingers and the palm for the current grasp type are added to the energy score for the current hand configuration. In addition, the distance to the nearest grasp type is multiplied by μ and added to the energy. This penalizes grasps for moving away from the canonical hand configurations. Higher μ values keep the hand in configurations close to the canonical grasp hand configuration. A full

Algorithm 1 Grasp Energy Calculations

```
procedure CONTACTENERGY()
    numVisible = 0
    energy = 0
    energy = 0
    for vc in virtualContacts do
        ip = getIntersectionPoint(vc, body)
        if isVisible(ip) then
            numVisible ++
            energy += dist(vc, ip)
            energy += γ∗ (1 - (vc.norm · ip.norm))
    return energy / numVisible
procedure HEATMAPENERGY()
    energy = 0
    graspType, distance = getGraspType()
    if distance > α then
        return MAX_ENERGY
    for vc in virtualContacts do
        fingerType = getFingerType(vc)
        if fingerType in [Palm, F0, F1, F2] then
            u,v = contactToImagePlane(vc.x, vc.y, vc.z)
            index = getIndex(graspType, fingerType)
            energy += heatmaps[index][u][v]
            energy += μ∗distance
    return energy
```

description of this algorithm is shown in Algorithm 1. Overall, the Heatmap Energy function favors hand configurations where the palm and fingertips project into low energy areas of the heatmaps, areas that the deep model has associated with stable grasps.

Real World Grasps							
Scene	Target	Success	Scene Seg. (s)	Mesh Gen. (s)	Heatmap Gen. (s)	Sim. Ann. (s)	Total Time (s)
All bottle	All bottle	yes	0.168	3.014	3.122	10.982	14.164
water bottle	water bottle	yes	0.159	2.977	3.141	10.471	13.612
drill	drill	yes	0.157	2.991	3.119	12.745	15.893
shampoo	shampoo	yes	0.339	3.277	3.122	12.253	15.869
tool case	tool case	no	0.175	3.026	3.242	14.328	17.570
crowded	shampoo	yes	0.181	3.307	3.127	14.328	17.816
crowded	drill	no	0.181	3.167	3.118	15.940	19.288
crowded	All bottle	yes	0.188	3.338	3.118	15.507	19.033
	Success:	6/8				Average:	16.655

Table 4.2: Results of grasp attempts for varying scenes and objects. $\rho = .1$, $\mu = 10$, and $\gamma = 500$

4.4.2.1 Combined Energy

The Contact Energy and Heatmap Energy terms are combined using a mixing parameter ρ as shown in the following equation:

$$\rho * heatmapEnergy + (1 - \rho) * contactEnergy \tag{4.2}$$

Higher values of ρ lead to grasps where the fingertips are able to move into lower energy positions at the expense of increasing the contact energy. It is a balance for how much a grasp should be rewarded for aligning nicely with the visible portions of the object against how the current hand configuration compares to the learned idea of what a stable grasp should look like.

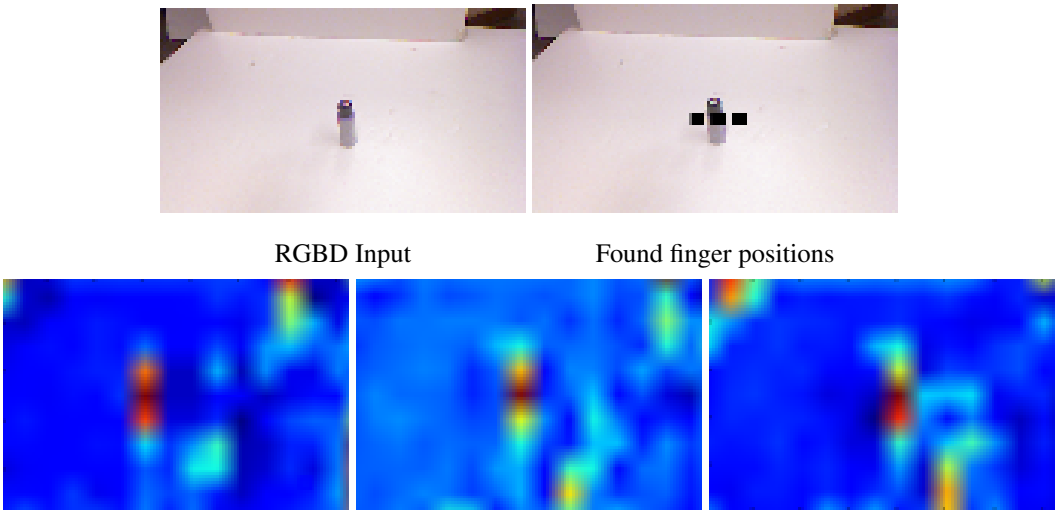


Figure 4.9: Scene from the Grasping Rectangles Dataset. Positive grasp examples were scored +1 and negative examples -1. Red corresponds to good gripper or palm positions, and the argmax for each heatmap is displayed in the overlaid image.

4.5 Experimental Results

4.5.1 Grasping Rectangle Dataset

In order to demonstrate the ability of the system to learn arbitrary grasp quality functions, we trained our system on the Grasping Rectangle Dataset from [Lenz *et al.*, 2014]. In this dataset, two of the edges of a rectangle represent where to place the gripper. Image patches were extracted from the center of each edge of the rectangle to represent each finger, and from the center for the palm. The grasps were discretized based on the rotation of the rectangle, so that grasp types related to the rotation of the wrist. The system was able to reproduce results similar to those found by[Lenz *et al.*, 2014]. Figure 4.9 shows an example scene with a set of heatmaps produced by the deep learning network and the best fingertip positions found by our system.

4.5.2 BigBird Generated Dataset

Using the deep learning model trained on the BigBird dataset, grasps were generated for several different real world scenes, and executed using a Barrett Hand attached to a StaubliTX60 robotic arm. None of the objects in the scenes were present in the training data for the deep learning model.

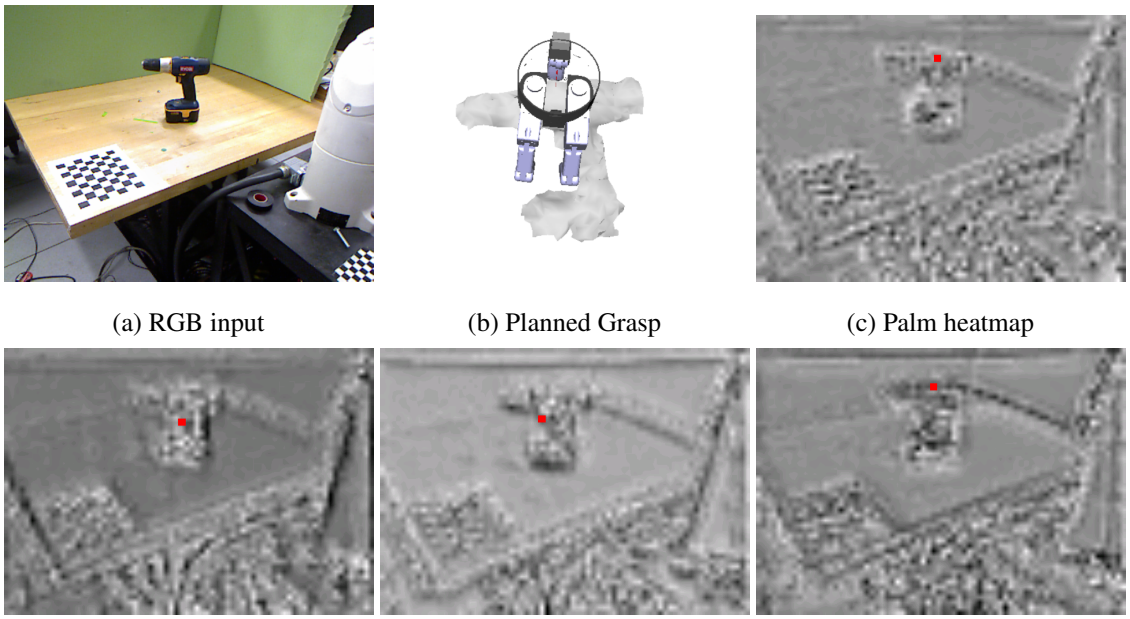
For each trial, a single RGBD image and pointcloud of the scene was captured. The Simulated Annealing Planner was run for 10,000 steps using the Combined Energy function and the lowest energy reachable grasp generated within that time was executed by the arm. Success was determined by whether or not the object was contained in the hand and lifted off of the table. The total time is generated with the following formula as the heatmap generation and scene segmentation are run in parallel:

$$t_{total} = max(t_{segment} + t_{mesh}, t_{heatmap}) + t_{sim.ann.} \tag{4.3}$$

The results for the experiments are shown in Table 4.2. While $\rho = .1$ sets more weight to the Contact Energy, the unweighted Contact Energy is smaller than the unweighted Heatmap Energy, so this value balances the two. The experimental results demonstrate that our system is able to generalize grasps generated on the BigBird models to these new objects. Several example grasps are shown in Figures 4.10, 4.11, 4.12, 4.13. These experiments reveal not only that our system generalizes well, but that it is effective in crowded scenes and with semi transparent objects. As long as some portion of the object is visible and able to be meshed, then this system is able to generate grasps around that portion of the object. While the current system failed to generate a stable grasps for the tool case, we attribute this partially to the fact that very few of the objects in the training set have handles, and none of them had horizontal handles like the tool case. For the failure case involving the drill, when the partial mesh of the drill was created, it was missing portions of the handle. While closing, the hand collided with a portion of the drill that had not been meshed, this is difficult to avoid, as we can only plan trajectories and generate hand configurations that avoid collisions with portions of objects that are meshed.

4.5.3 Comparison of Grasp Energy Components

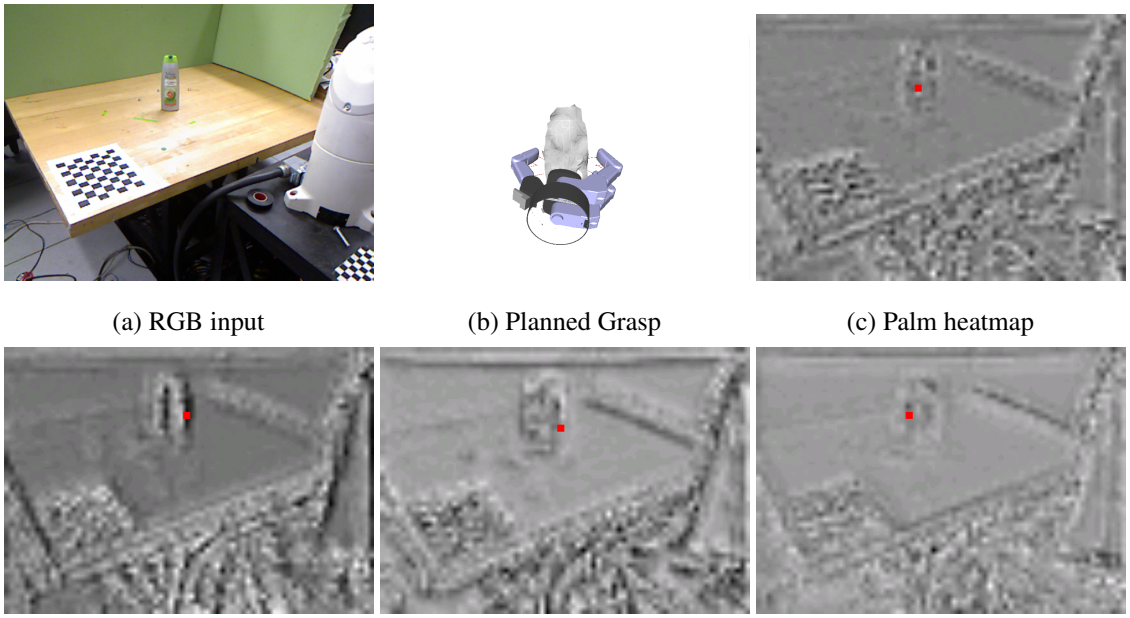
Using a single scene, grasps were planned using Contact Energy, Heatmap Energy, and Combined Energy to demonstrate how the different components of the Combined Energy function work in isolation. The Contact Energy function aligns the palm to the object to be grasped, but does not place the fingers into any specific configuration. The Heatmap Energy alone places the fingers into positions that project into low energy locations within the heatmaps, but often finds low energy regions of the heatmaps that are for other objects, or may be either to far into the scene, or too close to the camera. The Combined Energy function uses the Contact Energy to ensure that the hand is near



(a) RGB input (b) Planned Grasp (c) Palm heatmap

(d) F0 heatmap (e) F1 heatmap (f) F2 heatmap

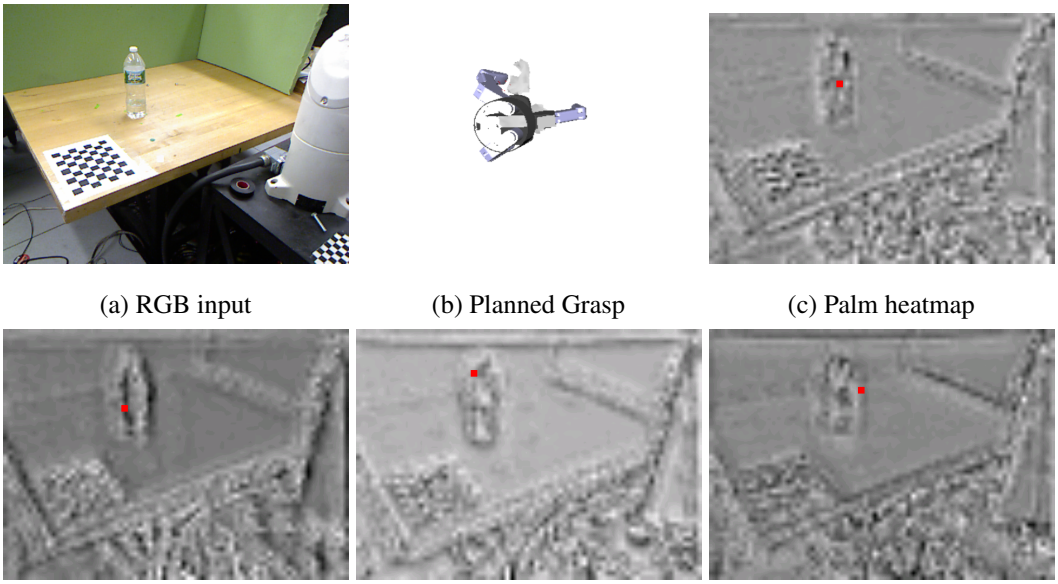
Figure 4.10: Input scene, resultant grasp, and associated heatmaps.



(a) RGB input (b) Planned Grasp (c) Palm heatmap

(d) F0 heatmap (e) F1 heatmap (f) F2 heatmap

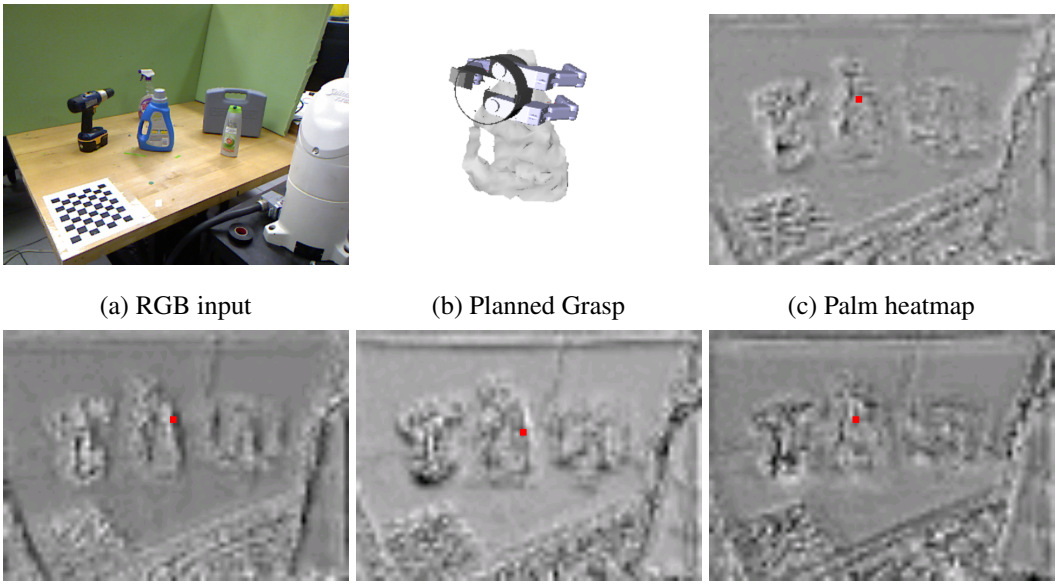
Figure 4.11: Input scene, resultant grasp, and associated heatmaps.



(a) RGB input (b) Planned Grasp (c) Palm heatmap

(d) F0 heatmap (e) F1 heatmap (f) F2 heatmap

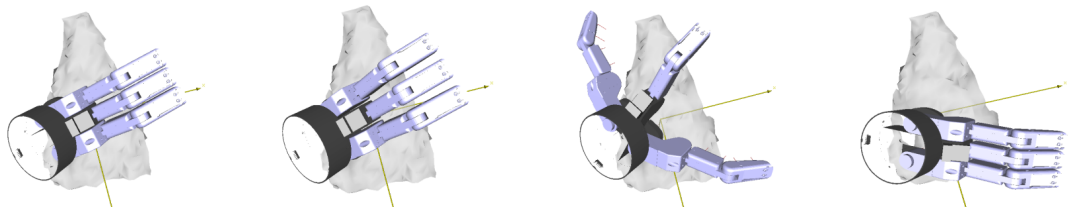
Figure 4.12: Input scene, resultant grasp, and associated heatmaps. The RGB input component allows for a grasp to be planned on a partially transparent object.



(a) RGB input (b) Planned Grasp (c) Palm heatmap

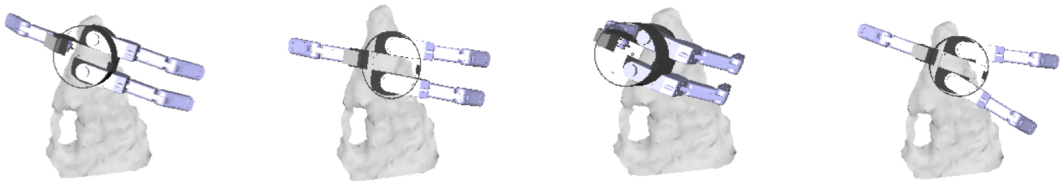
(d) F0 heatmap (e) F1 heatmap (f) F2 heatmap

Figure 4.13: Input scene, resultant grasp, and associated heatmaps. The Contact Energy term keeps the hand close to the object of interest, while the heatmaps pull the fingers into a stable configuration.



(a) Contact Energy Grasps

(b) Heatmap Energy Grasps



(c) Combined Energy Grasps

Figure 4.14: Grasps calculated for the same scene using different energy functions. **Top-Contact Energy**: These grasps stay close to the target object, but the energy function does nothing to encourage any sort of grasp stability. **Middle-Heatmap Energy:** The energy function here pulls the hand into configurations that often result in stable grasps, but there is nothing in the energy function to keep the hand near the particular target object in the scene we are interested in grasping. **Bottom-Combined Energy:** This energy function uses the Contact Energy term to stay local to the object of interest, while the Heatmap Energy term pulls the hand into locally optimal configurations similar to the training grasps.

the target object, while the heatmap energy refines the fingertip and palm locations locally. Figure 4.14 shows several low energy grasps computed with the different energy functions.

4.6 Discussion

We have shown that traditional methods for calculating grasps on known 3D objects can be paired with a deep learning architecture in order to efficiently estimate arbitrary grasp affordances from RGBD images. The deep architecture can learn grasp qualities that may take arbitrarily long to calculate on the training models enabling the use of thorough and meaningful grasp quality scores to generate training grasps. The generated grasps can then be used to train a deep model to quickly recognize quality grasps without explicitly calculating the grasp quality or even requiring a full 3D model of the object to be grasped.

This chapter has shown that heuristic grasp quality functions can be replaced with a learned notion of grasp stability encoded in a CNN. The training grasps can be produced in a variety of ways, for example a large number of grasps can be generated offline using GraspIt! as was done here, and going further, Mechanical Turk could be used to filter the training grasps. Where grasps that semantically do not make sense can be removed. We have replaced a heuristic-based grasp planner with a simulated data-driven planner whose behavior can be modified by changing the training data rather than writing additional code.

Chapter 5

Shape Completion Enabled Robotic Grasping

One limitation of the heatmap-based approach from the previous chapter is that the resultant grasps must come from roughly the direction that the camera is viewing the object from. This is because the Contact Energy term encourages contact with the visible portion of the mesh, and these portions of the mesh are only available from the approach direction of the camera. In this chapter, we propose a method to fill in occluded portions of the target object's geometry so that a planner like the one from the previous chapter can generate grasps approaching the object from a wider variety of approach directions.

Grasp planning based on raw sensory data is difficult due to occlusion and incomplete information regarding scene geometry. This chapter utilizes a 3D CNN to enable stable robotic grasp planning via shape completion. The 3D CNN is trained to do shape completion from a single pointcloud of a target object, essentially filling in the occluded portions of objects. This ability to infer occluded geometries can be applied to a multitude of robotic tasks. It can assist with path planning for both arm motion and robot navigation where an accurate understanding of whether occluded scene regions are occupied or not results in better trajectories. It also allows a traditional grasp planner to generate stable grasps via the completed shape.

The proposed framework consists of two stages: a training stage and a runtime stage. During the training stage, the CNN is shown occupancy grids created from thousands of synthetically rendered

depth images of different mesh models. Each of these occupancy grids is captured from a single point of view, and occluded portions of the volume are marked as empty. For each training example, the ground truth occupancy grid (the occupancy grid for the entire 3D volume) is also generated for the given mesh. From these pairs of occupancy grids the CNN learns to quickly complete mesh models at runtime using only information from a single point of view. Several example completions are shown in

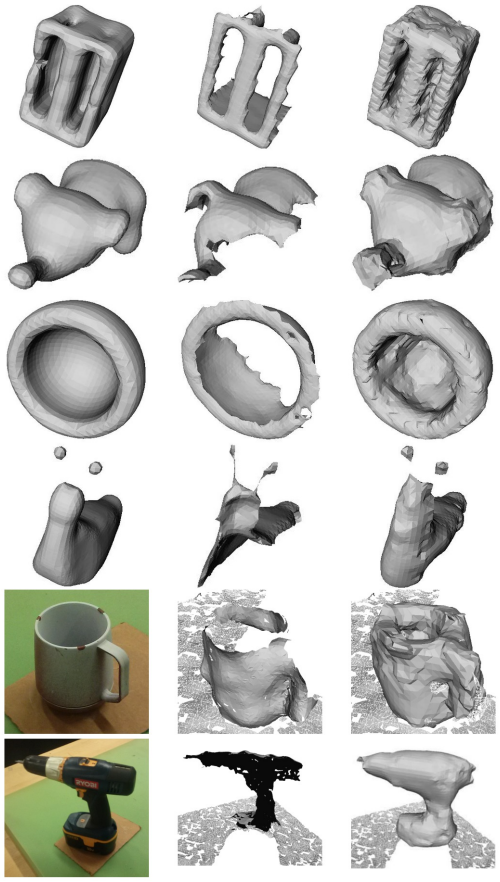


Figure 5.1: Ground Truth, Partials, and Completions (L to R). The top four are completions of synthetic depth images of Grasp Dataset holdout models. The mug and drill show completions of Kinect-captured depth images of physical objects.

Figure 5.1. This setup is beneficial for robotics applications as the majority of the computation time takes place during offline training, so that at runtime an object's partial-view pointcloud can be run through the CNN and completed in under a tenth of a second on average and then quickly meshed.

During the runtime stage, a pointcloud is captured using a depth sensor. A segmentation algorithm is run, and regions of the pointcloud corresponding to graspable objects are extracted from the scene. Occupancy grids of these regions are created, where all occluded regions are labeled as empty. These maps are passed separately through the trained CNN. The outputs from the CNN are occupancy grids, where the CNN has labeled all the occluded regions of the input as either occupied or empty for each object. These new occupancy grids are either run through a fast marching cubes algorithm[Lorensen and Cline, 1987], or further post-processed if they are to be grasped. Whether the object is completed or completed and post-processed results in either 1) fast completions suitable for path planning and scene understanding or 2) detailed meshes suitable for grasp planning, where the higher resolution visible regions are incorporated into the reconstruction. This framework is extensible to crowded scenes with multiple objects as each object is completed individually. It is also applicable to different domains because it can learn to reproduce objects from whatever dataset it is trained on, and further shows the ability to generalize to unseen views of objects or even entirely novel objects. This applicability to multiple domains is complemented by the thousands of 3D models available from datasets such as ShapeNet[Chang *et al.*, 2015] and the rapidly increasing power of GPU processors.

The contributions of this work include: 1) A novel CNN architecture for shape completion; 2) A fast mesh completion method, resulting in meshes able to quickly fill the planning scene; 3) A second CUDA enabled completion method that creates detailed meshes suitable for grasp planning by integrating fine details from the observed pointcloud; 4) A large open-source dataset of over 440,000 40^3 voxel grid pairs used for training. This dataset and the related code are freely available at **http://shapecompletiongrasping.cs.columbia.edu**. In addition, the website makes it easy to browse and explore the thousands of completions and grasps related to this work; 5) Results from both simulated and live experiments comparing our method to other approaches and demonstrating its improved performance in grasping tasks.

5.1 Previous Approaches to General Shape Completion

General shape completion to enable robotic grasping has been studied in robotics. Typical approaches [Bohg *et al.*, 2011][Quispe *et al.*, 2015][Schiebener *et al.*, 2016] use symmetry and extrusion heuristics for shape completion, and they are reasonable for objects well represented by geometric

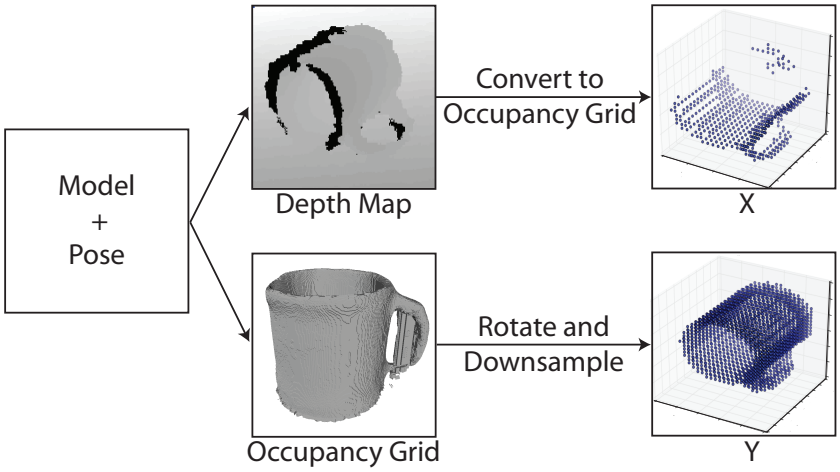


Figure 5.2: Training Data; In X, the input to the CNN, the occupancy grid marks visible portions of the model. Y, the expected output, has all voxels occupied by the model marked.

primitives. Our approach differs from these methods in that it learns to complete arbitrary objects based upon a large set of training exemplars, rather than requiring the objects to conform to heuristics.

A common alternative to general shape completion in the robotics community is object recognition and 3D pose detection [Rennie *et al.*, 2016][Papazov and Burschka, 2010][Hinterstoisser *et al.*, 2011]. In these approaches objects are recognized from a database of objects and the pose is then estimated. These techniques fill a different use case: the number of encountered objects is small, and known ahead of time often in terms of both texture and geometry. Our approach differs in that it extends to novel objects.

The computer vision and graphics communities have become increasingly interested in the problem of shape completion. Some examples include [Wu *et al.*, 2014][Wu *et al.*, 2015], which use a deep belief network and Gibbs sampling for 3D shape reconstruction, and [Firman *et al.*, 2016], which uses Random Forests. In addition, work by [Rock *et al.*, 2015] uses an exemplar based approach for the same task. Others [Tulsiani *et al.*, 2016][Choy *et al.*, 2016] have developed algorithms to learn 3D occupancy grids directly from 2D images. [Li *et al.*, 2015] uses a database of pre-existing models to do completion.

It is difficult to apply many of these works directly to robotic manipulation as no large dataset of renderings of handheld objects needed for robotic manipulation tasks existed until now. Also, [Tulsiani *et al.*, 2016][Choy *et al.*, 2016] use pure RGB rather than the RGBD images prevalent in robotics, making the problem more difficult as the completed shape must somehow be positioned

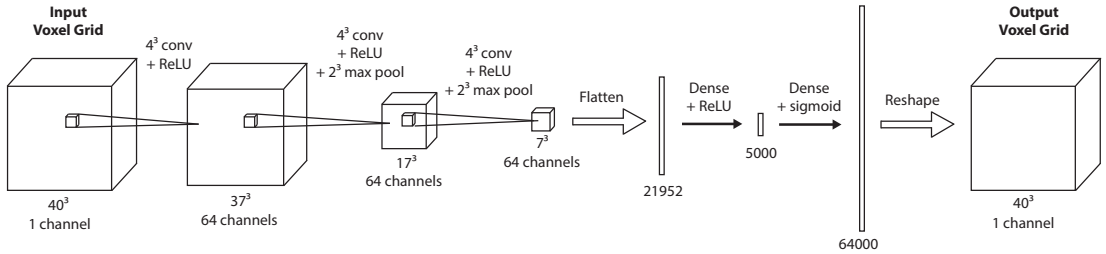
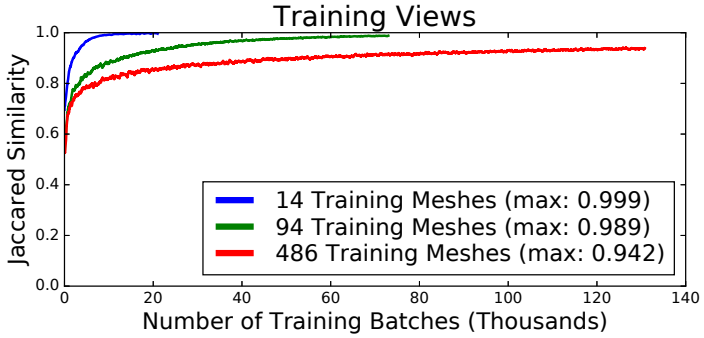


Figure 5.3: CNN Architecture. The CNN has three convolutional and two dense layers. The final layer has 64000 nodes, and reshapes to form the resulting 40^3 occupancy grid. The numbers on the bottom edges show the input sizes for each layer. All layers use ReLU activations except for the last dense layer, which uses a sigmoid.

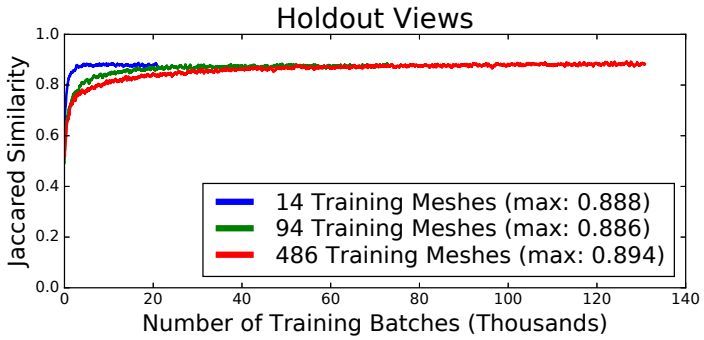
in the scene and the process does not utilize available depth information. Most create results with resolutions too low for use with current grasp planners which require meshes. Our work creates a new dataset specifically designed for completing objects useful for manipulation tasks using the 2.5-D range sensors prevalent in robotics, and provides a technique to integrate the high resolution observed view of the object with our relatively high resolution CNN output, creating a completion suitable for grasp planning.

Our work differs from [Mahler *et al.*, 2016] and our own related work [Goldfeder *et al.*, 2009a], both of which require a complete mesh to query a model database and retrieve grasps used on similar objects. These approaches could be used in tandem with our framework where the completed model would act as the query mesh. While grasps can be planned using partial meshes where the object is not completed (see [Varley *et al.*, 2015]), they still have their limitations and issues. Shape completion can be used to alleviate this problem.

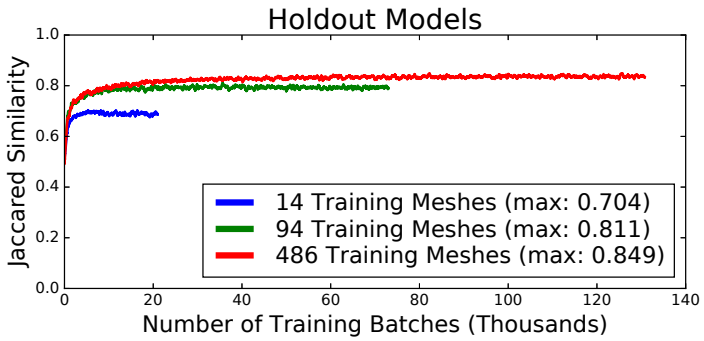
While many mesh model datasets exist such as [Chang *et al.*, 2015], [Wu *et al.*, 2014], and [Xiang *et al.*, 2014], this framework makes heavy use of the YCB[Calli *et al.*, 2015] and Grasp Database[Kappler *et al.*, 2015] mesh model datasets. We chose these two datasets as many robotics labs all over the world have physical copies of the YCB objects, and the Grasp Database contains objects specifically geared towards robotic manipulation. We augmented 18 of the YCB objects whose provided meshes were of high quality with models from the Grasp Database which contains 590 mesh models.



(a)



(b)



(c)

Figure 5.4: Jaccard similarity for three CNNs, one (shown in blue) trained with 14 mesh models, the second (green) trained with 94 mesh models, and the third (red) trained with 486 mesh models. For each plot, while training, the CNNs were evaluated on inputs they were being trained on (Training Views, plot a), novel inputs from meshes they were trained on (Holdout Views, plot b) and novel inputs from meshes they have never seen before (Holdout Models, plot c).

5.2 Training

5.2.1 Data Generation

In order to train a network to reconstruct a diverse range of objects, meshes were collected from the YCB and Grasp Database. The models were run through binvox[Min, 2004] in order to generate 256^3 occupancy grids. In these occupancy grids, both the surface and interior of the meshes are marked as occupied. In addition, all the meshes were placed in Gazebo[Koenig and Howard, 2004], and 726 depth images were generated for each object subject to different rotations uniformly sampled (in roll-pitch-yaw space, 11*6*11) around the mesh. The depth images are used to create occupancy grids for the portions of the mesh visible to the simulated camera, and then all the occupancy grids generated by binvox are transformed to correctly overlay the depth image occupancy grids. Both sets of occupancy grids are then down-sampled to 40^3 to create a large number of training examples. The input set (X) contains occupancy grids that are filled only with the regions of the object visible to the camera, and the output set (Y) contains the ground truth occupancy grids for the space occupied by the entire model. An illustration of this process is shown in Figure 5.2.

5.2.2 Model Architecture and Training

The architecture of the CNN is shown in Figure 5.3. The model was implemented using Keras[Chollet, 2015], a Theano[Bergstra *et al.*, 2010][Bastien *et al.*, 2012] based deep learning library. Each layer used rectified linear units as nonlinearities except the final fully connected (output) layer which used a sigmoid activation to restrict the output to the range $[0, 1]$. We used the cross-entropy error $E(y, y')$ as the cost function with target y and output y':

$$E(y, y') = -\big(y \log(y') + (1 - y) \log(1 - y')\big)$$

This cost function encourages each output to be close to either 0 for unoccupied target voxels or 1 for occupied. The optimization algorithm Adam[Kingma and Ba, 2014], which computes adaptive learning rates for each network parameter, was used with default hyperparameters (β_1=0.9, β_2=0.999, ϵ=10^{-8}) except for the learning rate, which was set to 0.0001. Weights were initialized following the recommendations of [He *et al.*, 2015] for rectified linear units and [Glorot and Bengio, 2010] for the logistic activation layer. The model was trained with a batch size of 32. Each of the 32 examples in a batch was randomly sampled from the full training set with replacement.

We used the Jaccard similarity to evaluate the similarity between a generated voxel occupancy grid and the ground truth. The Jaccard similarity between sets A and B is given by:

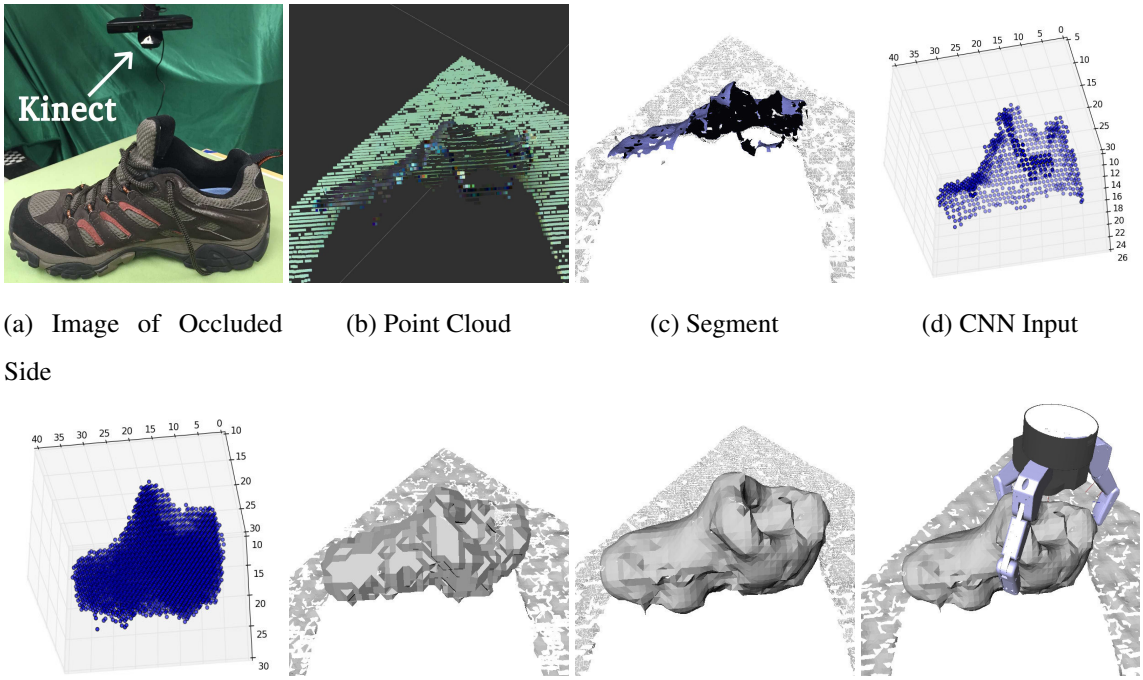
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard similarity has a minimum value of 0, where A and B have no intersection and a maximum value of 1 where A and B are identical. During training, this similarity measure is computed for input meshes that were in the training data (**Training Views**), meshes from objects within the training data but from novel views (**Holdout Views**), and for meshes of objects not in the training data (**Holdout Models**). The CNNs were trained with an NVIDIA Titan X GPU.

5.2.3 Training Results

Figure 6.7 shows how the Jaccard similarity measures vary as the networks' training progresses. In order to explore how the quality of the reconstruction changes as the number of models in the training set is adjusted, we trained three networks with identical architectures using variable numbers of mesh models. One was trained with partial views from 14 YCB models, another with 94 mesh models (14 YCB + 80 Grasp Database), and the third with 486 mesh models (14 YCB models + 472 Grasp Database). Each network was allowed to train until learning plateaued; for the CNN trained on 486 objects, this took over a week. The remaining 4 YCB and 118 Grasp Dataset models were kept as a holdout set. Results are shown in Figure 6.7. We note that the networks trained with fewer models perform better shape completion when they are tested on views of objects they have seen during training than networks trained on a larger number of models. This suggests that the network is able to completely learn the training data for the smaller number of models but struggles to do so when trained on larger numbers. Conversely, the models trained on a larger number of objects perform better than those trained on a smaller number when asked to complete novel objects. Because, as we have seen, the networks trained on larger numbers of objects are unable to learn all of the models seen in training, they may be forced to learn a more general completion strategy that will work for a wider variety of objects, allowing them to better generalize to objects not seen in training.

Figure 5.4(a) shows the performance of the three CNNs on training views. In this case, the fewer the mesh models used during training, the better the completion results. Figure 6.7(b) shows how the CNNs performed on novel views of the mesh objects used during training. Here the CNNs all did



(a) Image of Occluded Side (b) Point Cloud (c) Segment (d) CNN Input

(e) CNN Output (f) Fast Mesh (g) Detailed Mesh (h) Grasp Planning

Figure 5.5: Stages to the Runtime Pipeline. These images are not shown from the angle in which the data was captured in order to visualize the occluded regions. (a): An object to be grasped is placed in the scene. (b): A pointcloud is captured. (c): The pointcloud is segmented and meshed. (d): A partial mesh is selected by the user and then voxelized and passed into the 3D shape completion CNN. (e): The output of the CNN. (f): The resulting occupancy grid can be run through a marching cubes algorithm to obtain a mesh quickly. (g): Or, for better results, the output of the CNN can be combined with the observed pointcloud and preprocessed for smoothness before meshing. (h): Grasps are planned on the smoothed completed mesh. Note: this is a novel object not seen by the CNN during training.

approximately the same. Figure 5.4(c) shows the completion quality of the CNNs on objects they have not seen before. In this case, as the number of mesh models used during training increases, performance improves as the system has learned to generalize to a wider variety of inputs.

5.3 Runtime

At runtime the pointcloud for the target object is acquired from a 3D sensor, scaled, voxelized and then passed through the CNN. The output of the CNN, a completed voxel grid of the object, goes through a post processing algorithm that returns a mesh model of the completed object. Finally, a grasp can be planned and executed based on the completed mesh model. Figure 5.5 demonstrates the full runtime pipeline on a novel object never seen before.

1) Acquire Target Pointcloud: First, a pointcloud is captured using a Microsoft Kinect, then segmented using PCL's[Rusu and Cousins, 2011] implementation of euclidean cluster extraction. A segment corresponding to the object to be completed is selected either manually or automatically and passed it to the shape completion module.

2) Complete via CNN: The selected pointcloud is then used to create an occupancy grid with resolution 40^3. This occupancy grid is used as input to the CNN whose output is an equivalently sized occupancy grid for the completed shape. In order to fit the pointcloud to the 40^3 grid, it is scaled down uniformly so that the bounding box of the pointcloud fits in a 32^3 voxel cube, and then centered in the 40^3 grid such that the center of the bounding box is at point (20, 20, 18) in the voxel grid. Finally all voxels occupied by points from this scaled and transformed pointcloud are marked as such. Placing the pointcloud slightly off-center in the z dimension leaves more space in the back half of the grid for the network to fill.

3a) Create Fast Mesh: At this point, if the object being completed is not going to be grasped, then the voxel grid output by the CNN is run through the marching cubes algorithm, and the resulting mesh is added to the planning scene, filling in occluded regions of the scene.

3b) Create Detailed Mesh: Alternatively, if this object is going to be grasped, then post-processing occurs. The purpose of this post-processing is to integrate the points from the visible portion of the object with the output of the CNN. This partial view is of much higher density than the 40^3 grid and captures significantly finer detail for the visible surface. This merge is made difficult by the large

disparity in point densities between the captured cloud and 40^3 CNN output which can lead to holes and discontinuities if the points are naively merged and run through marching cubes.

Algorithm 2 Shape Completion

```
1:  procedure MESH(cnn_out, observed_pc)
2:      //cnn_out: 40^3 voxel output from CNN
3:      //observed_pc: captured pointcloud of object
4:      if FAST then
5:          mesh ← marchingCubes(cnn_out)
6:      else
7:          d_ratio ← densityRatio(observed_pc, cnn_out)
8:          upsampled_cnn ← upsample(cnn_out, d_ratio)
9:          vox ← merge(upsampled_cnn, observed_pc)
10:         vox_no_gap ← fillGaps(vox)
11:         vox_weighted ← CUDA_QP(vox_no_gap)
12:         mesh ← marchingCubes(vox_weighted)
13:     return mesh
```

Alg. 2 shows how we integrated the dense partial view with our 40^3 voxel grid via the following steps. (Alg.2:L7) In order to merge with the partial view, the output of the CNN is converted to a point cloud and its density is compared to the density of the partial view point cloud. The densities are computed by randomly sampling $\frac{1}{10}$ of the points and averaging the distances to their nearest neighbors. (Alg.2:L8) The CNN output is up-sampled by d_ratio to match the density of the partial view. This is performed by examining each cube of 8 adjacent original low resolution voxels, with the centers of the voxels as the corners. The new voxels are uniformly distributed inside the cube. For each new voxel, the L1 distance to each original voxel is computed and the 8 distances are summed, weighted by 1 if the original voxel is occupied and -1 otherwise. The new voxel is occupied if its weighted sum is nonnegative. This has the effect of creating piecewise linear separating surfaces similar to the marching cubes algorithm and mitigates up-sampling artifacts. (Alg.2:L9) The upsampled output from the CNN is then merged with the point cloud of the partial view and the combined cloud is voxelized at the new higher resolution of $(40 * d\_ratio)^3$. For most objects

d_ratio tends to be either 2 or 3, depending on the physical size of the object, resulting in a voxel grid of either 80^3 or 120^3. (Alg.2:L10) Any gaps in the voxel grid between the upsampled CNN output and the partial view cloud are filled. This is done by finding the first occupied voxel in every z-stack. If the distance to the next occupied voxel is less than $d\_ratio + 1$ the intermediate voxels are filled. (Alg.2:L11) The voxel grid is smoothed using our own CUDA implementation of the convex quadratic optimization problem from [Lempitsky, 2010]. This optimization re-weights the voxels, minimizing the Laplacian on the boundary of the embedding function F, i.e.:

$$\int \left(\frac{\partial^2 F}{\partial x^2}\right)^2 + \left(\frac{\partial^2 F}{\partial y^2}\right)^2 + \left(\frac{\partial^2 F}{\partial z^2}\right)^2 dV \rightarrow \text{min.}$$

subject to the hard constraint:

$$\forall i, j, k \qquad v_{ijk} \cdot f_{ijk} \geq 0$$

The constraint means that for all input voxels v and output weighted voxels f all occupied voxels prior to the optimization stays occupied or on the boundary, and all unoccupied voxels remain unoccupied or on the boundary. Again, for further details see [Lempitsky, 2010]. (Alg.2:L12) The weighted voxel grid is then run through marching cubes.

4) Grasp completed mesh: The reconstructed mesh is then loaded into GraspIt![Miller and Allen, 2004] where a grasp planner is run using a Barrett Hand model. The reachability of the planned grasps are checked using MoveIt![Sucan and Chitta, 2013], and the highest quality reachable grasp is then executed.

5.4 Experimental Results

We created a test dataset by randomly sampling 50 training views (**Training Views**), 50 holdout views (**Holdout Views**), and 50 views of holdout models (**Holdout Models**). The Training Views and Holdout Views were sampled from the 14 YCB training objects. The Holdout Models were sampled from holdout YCB and Grasp Dataset objects. We used three metrics to compare the accuracy of the different completion methods: Jaccard similarity, Hausdorff distance, and geodesic divergence.

5.4.1 General Completion Results

We first compared several general completion methods: passing the partial view through marching cubes and then Meshlab's Laplacian smoothing (**Partial**), mirroring completion[Bohg *et al.*, 2011] (**Mirror**), our method (**Ours**). Our CNN was trained on the 484 objects from the YCB + Grasp Dataset and the weights come from the point of peak performance on holdout models (the peak of the red line in Figure 6.7.(c)).

The Jaccard similarity was used to guide training, as shown in Figure 5.4. We also used this metric to compare the final resulting meshes from several completion strategies. The completed meshes were voxelized at 80^3, and compared with the ground truth mesh. The results are shown in Table 5.1. Our proposed method results in higher similarity to the ground truth meshes than the partial and mirroring approaches for all tested views.

The Hausdorff distance is a one-directional metric computed by sampling points on one mesh and computing the distance of each sample point to its closest point on the other mesh. The mean value of a completion is the average distance from the sample points on the completion to their respective closest points on the ground truth mesh. The symmetric Hausdorff distance was computed by running Meshlab's[Cignoni *et al.*, 2008] Hausdorff distance filter in both directions. Table 5.2 shows the mean values of the symmetric Hausdorff distance for each completion method. In this metric, the CNN completions are significantly closer to the ground truth than are the partial and the mirrored completions.

The completions are also compared using a measure of geodesic divergence[Hamza and Krim, 2003]. A geodesic shape descriptor is computed for each mesh. A probability density function is then computed for each mesh by considering the shape descriptor as a random distribution and approximating the distribution using a Gaussian mixture model. The probability density functions for each completion are compared with that of the ground truth mesh using the Jenson-Shannon divergence. Table 5.3 shows the mean of the divergences for each completion method. Here, our method outperforms all other completion methods.

Across all metrics, our method results in more accurate completions than the other general completion approaches.

View Type	Partial	Mirror	Ours
Training Views	0.1182	0.2325	**0.7771**
Holdout Views	0.1307	0.2393	**0.7486**
Holdout Models	0.0931	0.1921	**0.6496**

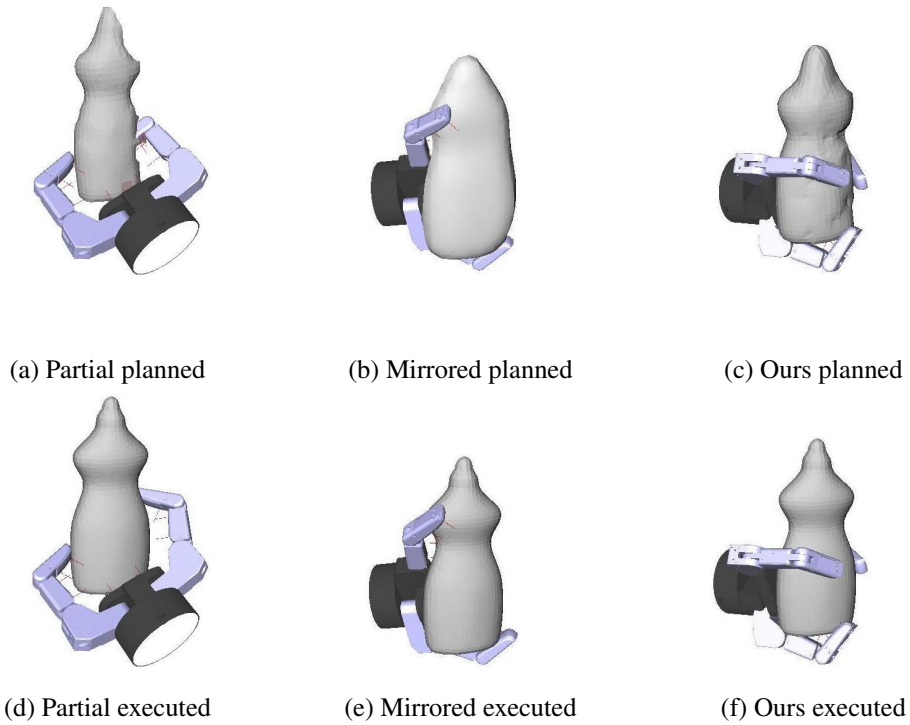
Table 5.1: Jaccard Similarity Results (Larger is better). This measures the intersection over union of two voxelized meshes as described in Section 5.4.1.

View Type	Partial	Mirror	Ours
Training Views	11.4	7.5	**3.6**
Holdout Views	12.3	8.2	**4.0**
Holdout Models	13.6	10.7	**5.9**

Table 5.2: Hausdorff Distance Results (Smaller is better). This measures the mean distance in millimeters from points on one mesh to another as described in Section 5.4.1.

View Type	Partial	Mirror	Ours
Training Views	0.3770	0.2905	**0.0867**
Holdout Views	0.4944	0.3366	**0.0934**
Holdout Models	0.3407	0.2801	**0.1412**

Table 5.3: Geodesic Divergence Results (Smaller is better). This measures the Jenson-Shannon probabilistic divergence between two meshes as described in Section 5.4.1.



(a) Partial planned (b) Mirrored planned (c) Ours planned

(d) Partial executed (e) Mirrored executed (f) Ours executed

Figure 5.6: Top Row: Planned Grasps using variety of completions methods. Bottom Row: Grasps from the top row executed on the Ground Truth object. Notice both the partial and mirrored completions' planned and executed grasps differ whereas our method shows fidelity between the planned and executed grasps.

5.4.2 Comparison to Database Driven Methods

In addition, we evaluated a RANSAC-based approach[Papazov and Burschka, 2010] on the Training Views of the YCB dataset using the same metrics. This corresponds to a highly constrained environment containing only a very small number of objects which are known ahead of time. It is not possible to load 484 objects into the RANSAC framework, so a direct comparison to our method involving the large number of objects we train on is not possible. In fact, the inability of RANSAC-based methods to scale to large databases of objects is one of the motivations of our work. However, we compared our method to a very small RANSAC using only 14 objects, and our method performs comparably to the RANSAC approach even on objects in its database, while having the additional abilities to train on far more objects and generalize to novel objects: Jaccard (Ours: 0.771, RANSAC: **0.8566**), Hausdorff (Ours: 3.6, RANSAC: **3.1**), geodesic (Ours: **0.0867**, RANSAC: 0.1245). Our approach significantly outperforms the RANSAC approach when encountering an object that neither method

View	Error	Completion Type			
		Partial	Mirror	Ours	RANSAC
Training View	Joint (°)	6.09°	4.20°	**1.75**°	1.83°
	Pose (mm)	16.0	11.5	**4.3**	7.3
Holdout View	Joint (°)	6.27°	4.05°	1.80°	**1.69**°
	Pose (mm)	20.8	15.6	**6.7**	7.4
Holdout Model	Joint (°)	7.59°	5.82°	**4.56**°	6.86°
	Pose (mm)	18.3	15.0	**13.2**	29.25

Table 5.4: Results from simulated grasping experiments. Joint Err. is the mean difference between planned and realized grasps per joint in degrees. Pose Err. is the mean difference between planned and realized grasp pose in millimeters. For both metrics smaller is better.

Completion Method	Grasp Success Rate (%)	Joint Error (degrees)	Completion Time (s)
Partial	71.43	9.156°	**0.545**
Mirror	73.33	8.067°	1.883
Ours	**93.33**	**7.276**°	2.426

Table 5.5: Grasp Success Rate shows the percentage of succesful grasp attempts. Joint Error shows the mean difference in degrees between the planned and executed grasp joint values. Completion Time shows how long specified metric took to create a mesh of the partial view.

has seen before (Holdout Models): Jaccard (Ours: **0.6496**, RANSAC: 0.4063), Hausdorff (Ours: **5.9**, RANSAC: 20.4), geodesic (Ours: **0.1412**, RANSAC: 0.4305). The RANSAC based approach's performance on the Holdout Models is also worse than that of the mirrored or partial completion methods on both the geodesic and Hausdorff metrics.

5.4.3 Simulation Based Grasp Comparison

In order to evaluate our framework's ability to enable grasp planning, the system was tested in simulation using the same completions from Sec 5.4.1, allowing us to quickly plan and evaluate over 24,000 grasps. GraspIt! was used to plan grasps on all of the completions of the objects by uniformly sampling different approach directions. These grasps were then executed, not on the completed object, but on the ground truth meshes in GraspIt!. In order to simulate a real-world grasp execution, the completion was removed from GraspIt! and the ground truth object was inserted in its place. Then the hand was placed 20cm backed off from the ground truth object along the approach direction of the grasp. The spread angle of the fingers was set, and the hand was moved along the approach direction of the planned grasp either until contact was made or the grasp pose was reached. At this point, the fingers closed to the planned joint values. Then each finger continued to close until either contact was made with the object or the joint limits were reached. Figure 5.6 shows several grasps and their realized executions for different completion methods. Visualizations of the simulation results for the entire YCB and Grasp Datasets are available at **http://shapecompletiongrasping.cs.columbia.edu**

Table 6.3 shows the differences between the planned and realized joint states as well as the difference in pose of the base of the end effector between the planned and realized grasps. Using our method caused the end effector to end up closer to its intended location in terms of both joint space and the palm's cartesian position.

5.4.4 Performance on Real Hardware

In order to further evaluate our framework, the system was used in an end-to-end manner using actual robotic hardware to execute grasps planned via the different completion methods described above. The 15 objects used are shown in Figure 5.7. For each object, we ran the arm once using each completion method. The results are shown in Table 6.4. Our method enabled a 20% improvement over the general shape completion methods in terms of grasp success rate, and resulted in executed

grasps closer to the planned grasps as shown by the lower joint error.

5.4.5 Crowded Scene Completion

A scene often contains objects that are not to be manipulated and only require completion in order to be successfully avoided. In this case, the output of our CNN can be run directly through marching cubes without post-processing to quickly create a mesh of the object. Figure 5.8(a) shows a grasp planned using only the partial mesh for the object near the grasp target. Figure 5.8(b) and Figure 5.8(c) show the robotic hand crashing into one of the nearby objects when attempting to execute the grasp. The failure is caused by an incomplete planning scene. Figure 5.8(d) shows the scene with the nearby objects completed, though without smoothing. With this fuller picture, the planner accurately flags this grasp as unreachable. The time requirement for the scene completion is:

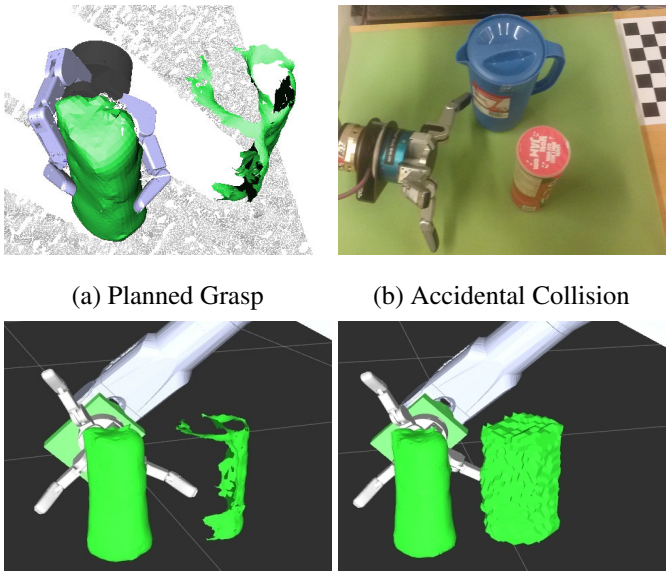
$$T_{completion} = T_{segment} + T_{target} + T_{non\_target} * n$$

with Segmentation Time ($T_{segment}$), Target Completion Time (T_{target}), Non Target Completion Time(T_{non_target}) and Number of Non Target Objects (n). Our system has the ability to quickly fill in occluded regions of the scene, and selectively spend more time generating detailed completions on specific objects to be manipulated. Average completion times in seconds from 15 runs are $T_{segment} = 0.119$, $T_{target} = 2.136$, and $T_{non\_target} = 0.142$.

5.5 Discussion

This chapter presents a framework to train and utilize a CNN to complete and mesh an object observed from a single point of view, and then plan grasps on the completed object. The completion system is fast, with completions available in a matter of milliseconds, and post processed completions suitable for grasp planning available in several seconds. The dataset and code are open source and available for other researchers to use. It has also been demonstrated that our completions are better than more naive approaches in terms of a variety of metrics including those specific to grasp planning. In addition, grasps planned on completions generated using our method are more often successful and result in executed grasps closer to the intended hand configuration than grasps planned on completions from the other methods.

Figure 5.7: Barrett Hand(BH8-280), StaubliTX60 Arm, and experiment objects.



(a) Planned Grasp (b) Accidental Collision

(c) Unfilled Scene (d) Filled Planning Scene

Figure 5.8: The system can be used to quickly complete obstacles that are to be avoided. The arm fails to execute the planned grasp (a), resulting in collision shown in (b). The collision with the non-target object occurred due to a poor planning scene as shown in (c), but the CNN without post-processing can be used to fill the planning scene allowing the configuration to be correctly marked as invalid as shown in (d).

The framework in this chapter allows robotic grasping systems to handle workspaces with hundreds of different known object geometries, and to even reason about the geometry of novel objects. Like the heatmap grasp planner from the previous chapter, the system in this chapter is also powered by a CNN trained on simulated renderings. With this approach, we are able to do away with another heuristic, the symmetry based completion methods. We can also improve on the performance of previously cutting edge RANSAC based approaches for geometric reasoning.

Chapter 6

Visual-Tactile Shape Completion

6.1 Introduction

Grasp planning based on raw sensory data is difficult due to occlusion and incomplete information regarding scene geometry. Often one sensory modality does not provide enough context to enable reliable planning. For example a single depth sensor image cannot provide information about occluded regions of an object, and tactile information is incredibly sparse spatially. This work utilizes a 3D convolutional neural network to enable stable robotic grasp planning by incorporating both tactile and depth information to infer occluded geometries. This multi-modal system is able to utilize both tactile and RGBD information to form a more complete model of the space the robot can interact with and also to provide a complete object model for grasp planning.

At runtime, a point cloud of the visible portion of the object is captured, and multiple guarded moves are executed where the hand is then moved towards the object, stopping when contact with the object occurs. The newly acquired tactile information is combined with the original partial view, voxelized, and sent through the CNN to create a hypothesis of the object's geometry.

Depth information from a single point of view often does not provide enough information to accurately predict object geometry. There is often unresolved uncertainty about the geometry of the occluded regions of the object. To alleviate this uncertainty, we utilize tactile information to generate a new more accurate hypothesis of the object's 3D geometry incorporating both visual and tactile information. Figure 6.1 demonstrates several real life example where the understanding of the object's 3D geometry is significantly improved by the additional sparse tactile data using our

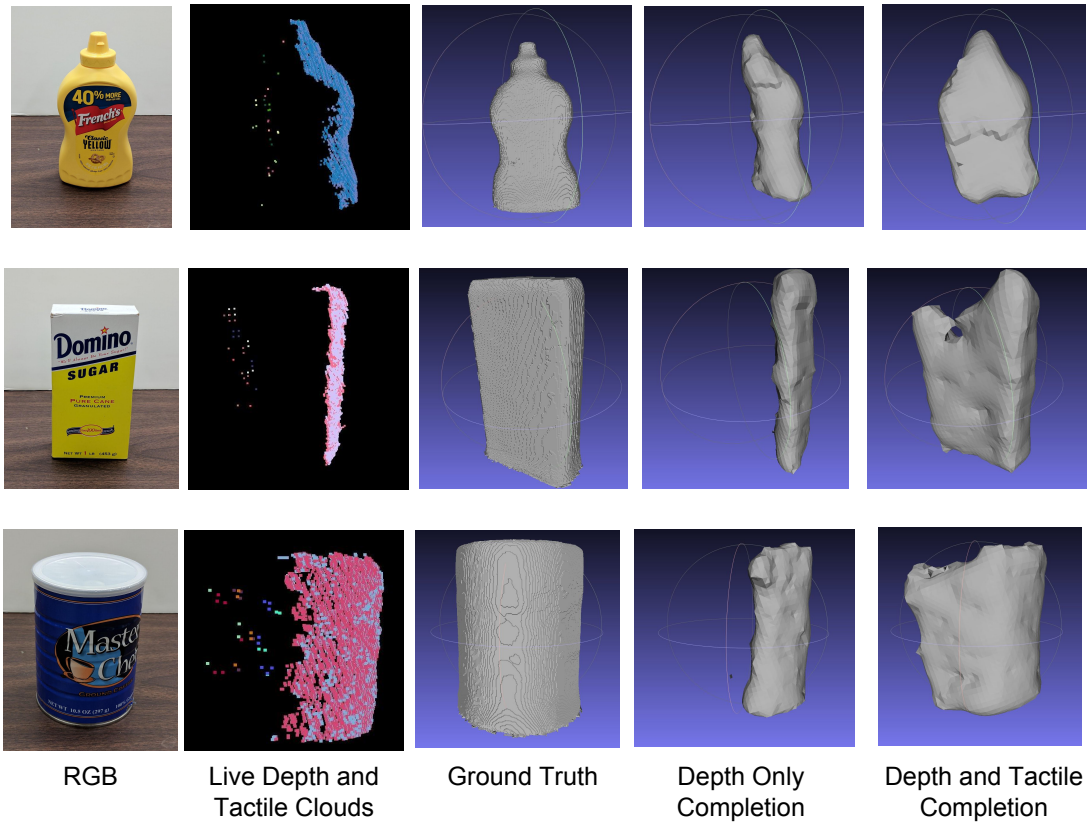


Figure 6.1: Completion examples from live tactile and depth data for YCB objects. These completions demonstrate that small amounts of additional tactile sensory data can significantly improve the systems ability to reason about 3D object geometry. The Depth Only completions for the mustard bottle looks similar to a mesh of a banana used during training while the Depth Only completion of the domino sugar box looks like a cell phone mesh from the training set. For these examples the small amounts of additional tactile information allows the CNN to differentiate from these incorrect training meshes.

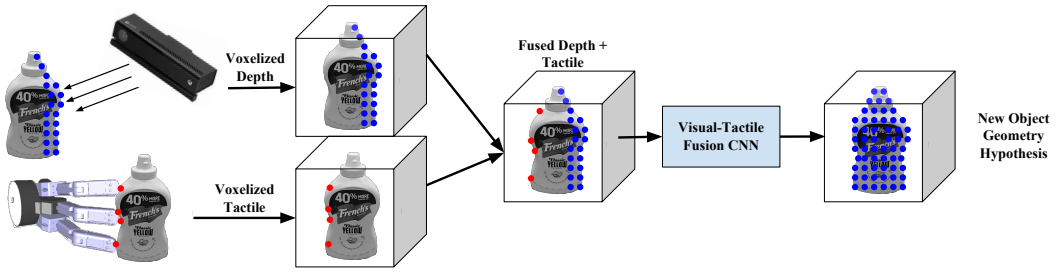


Figure 6.2: Both tactile and depth information are independently captured and voxelized into 40^3 grids. These are merged into a shared occupancy map which is fed into a CNN to produce a hypothesis of the object's geometry.

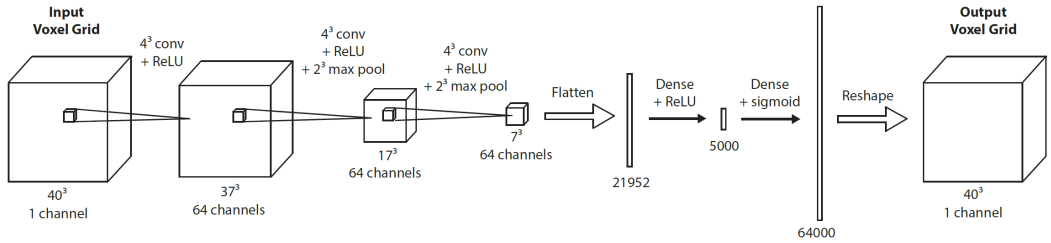


Figure 6.3: CNN Architecture. The CNN has three convolutional and two dense layers. The final layer has 64000 nodes, and reshapes to form the resulting 40^3 occupancy grid. The numbers on the bottom edges show the input sizes for each layer. All layers use ReLU activations except for the last dense layer, which uses a sigmoid.

framework. An overview of our sensory fusion architecture is shown in Figure 6.2.

The contributions of this work include: 1) an open source dataset for training a shape completion system using both tactile and depth sensory information, 2) a framework for integrating multi-modal sensory data to reason about object geometry, 3) open source code for alternative visual-tactile general completion methods, and 4) results comparing the completed object models using depth only, the combined depth-tactile information, and other visual-tactile completion methods.

6.2 Prior Visual-Tactile Geometric Reasoning Approaches

The idea of incorporating sensory information from vision, tactile and force sensors is not new [Miller and Leibowitz, 1999]. Despite the intuitiveness of using multi-modal data, there is still no

agreed upon framework to best integrate multi-modal sensory information in a way that is useful for robotic manipulation tasks. In this work, we are interested in reasoning about object geometry in particular.

Several recent uses of tactile information to improve estimates of object geometry has focused on the use of Gaussian Process Implicit Surfaces(GPIS) [Williams and Fitzgibbon, 2007]. Several examples along this line of work include [Caccamo *et al.*, 2016][Yi *et al.*, 2016] [Bjorkman *et al.*, 2013][Dragiev *et al.*, 2011][Jamali *et al.*, 2016][Sommer *et al.*, 2014][Mahler *et al.*, 2015]. This approach is able to quickly incorporate additional tactile information and improve the estimate of the objects geometry local to the tactile contact or observed sensor readings. There has additionally been several works that incorporate tactile information to better fit planes of symmetry and super quadrics to observed point clouds [Ilonen *et al.*, 2014][Ilonen *et al.*, 2013][Bierbaum *et al.*, 2008]. These approaches work well when interacting with objects that confirm to the heuristic of having clear detectable planes of symmetry or are easily modeled as super quadrics.

There has been successful research in utilizing continuous streams of visual information similar to Kinect Fusion[Newcombe *et al.*, 2011] or SLAM[Thrun and Leonard, 2008] in order to improve models of 3D objects for manipulation. One example being [Krainin *et al.*, 2011b][Krainin *et al.*, 2011a] In this work, the authors develop an approach to building 3D models of unknown objects based on a depth camera observing the robot's hand while moving an object. The approach integrates both shape and appearance information into an articulated ICP approach to track the robot's manipulator and the object while improving the 3D model of the object. Similarly [Hermann *et al.*, 2016] attaches a depth sensor to a robotic hand, and plans grasps directly in the sensed voxel grid. These approaches improve their models of the object using only a single sensory modality, but from many time points.

In previous work, we created a shape completion method using single depth images . The work provides an architecture to enable robotic grasp planning via shape completion. Shape completion is accomplished through the use of a 3D convolutional neural network (CNN). The network is trained on an open source dataset of over 440,000 3D exemplars captured from varying viewpoints. At runtime, a 2.5D point cloud captured from a single point of view is fed into the CNN, which fills in the occluded regions of the scene, allowing grasps to be planned and executed on the completed object. Runtime shape completion is very rapid because most of the computational costs of shape completion are borne during offline training. This prior work explored how the quality of completions

vary based on several factors. These include whether or not the object being completed existed in the training data and how many object models were used to train the network, and the ability of the network to generalize to novel objects allowing the system to complete previously unseen objects at runtime.

6.3 Visual-Tactile Geometric Reasoning

Our framework utilizes a trained CNN to produce a mesh of the target object incorporating both depth and tactile information.

6.3.1 Training

The architecture of the CNN is shown in Figure 6.3. The model was implemented using Keras[Chollet, 2015], a Theano[Bergstra *et al.*, 2010][Bastien *et al.*, 2012] based deep learning library. Each layer used rectified linear units as nonlinearities except the final fully connected (output) layer which used a sigmoid activation to restrict the output to the range $[0, 1]$. We used the cross-entropy error $E(y, y')$ as the cost function with target y and output y':

$$E(y, y') = -\big(y \log(y') + (1 - y) \log(1 - y')\big)$$

This cost function encourages each output to be close to either 0 for unoccupied target voxels or 1 for occupied. The optimization algorithm Adam[Kingma and Ba, 2014], which computes adaptive learning rates for each network parameter, was used with default hyperparameters (β_1=0.9, β_2=0.999, ϵ=10^{-8}) except for the learning rate, which was set to 0.0001. Weights were initialized following the recommendations of [He *et al.*, 2015] for rectified linear units and [Glorot and Bengio, 2010] for the logistic activation layer. The model was trained with a batch size of 32.

We used the Jaccard similarity to evaluate the similarity between a generated voxel occupancy grid and the ground truth. The Jaccard similarity between sets A and B is given by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard similarity has a minimum value of 0, where A and B have no intersection and a maximum value of 1 where A and B are identical. During training, this similarity measure is computed for input

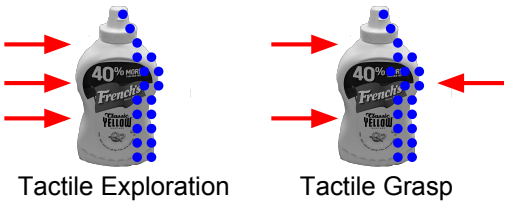
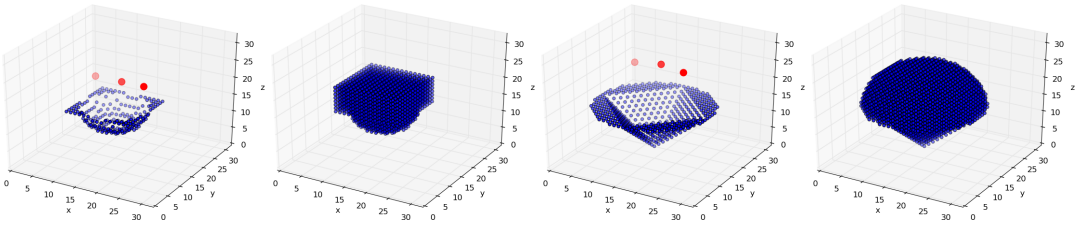


Figure 6.4: Red arrows show how the fingers approach the object for the tactile exploration case and for the tactile grasp case. Blue dots show points in the depth image captured by the camera.



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

Figure 6.5: Example training pairs from simple shape dataset. The red dots represent the tactile readings from tactile exploration. The blue dots on (a) and (c) represent to occupancy map gathered from the depth image. The blue points in (b) and (d) represent the ground truth 3d geometry.

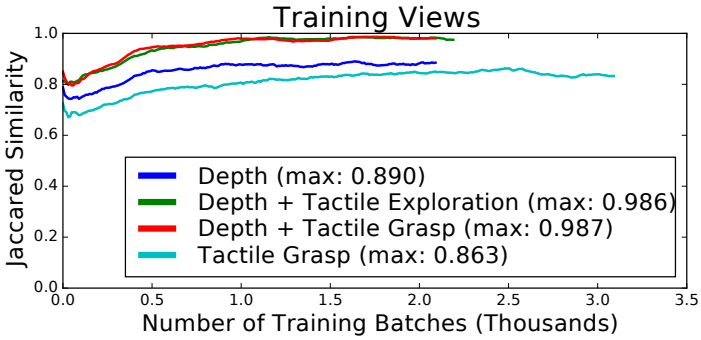


Figure 6.6: Different runs of the shape completion system where input is provided from: Depth, Depth and Tactile Exploration, Depth and Tactile from Grasp, and only from a Tactile Grasp. When using both tactile and grasp information, the system is able to complete the object almost 100% of the time. While depth or tactile alone are not sufficient to successfully reason about object geometry in all cases.

meshes that were in the training data (**Training Views**), meshes from objects within the training data but from novel views (**Holdout Views**), and for meshes of objects not in the training data (**Holdout Models**). The CNNs were trained with an NVIDIA Titan X GPU.

6.4 Completion of Simulated Geometric Shapes Dataset Objects

6.4.1 Geometric Shape Dataset

In order to evaluate our system's ability to utilize additional tactile sensory information to reason about 3D geometry, initial experimentation was done on a geometric shape dataset. This dataset consisted of conjoined half shapes. Both front and back halves of the objects were randomly chosen to be either a sphere, cube, or diamond. The front and back halves do match in size. Several example shapes are shown in Figure 6.5 (b) half cube half sphere and (d) half sphere half diamond. Next, synthetic sensory data was generated for these example shapes. Depth information was captured from a fixed camera location, and tactile information was collected using both a tactile exploration, and a tactile grasp as shown in Figure 6.4. The sensory data for two shapes is shown in Figure 6.5 (a) and (c).

6.4.2 Depth and Tactile Completion

Four networks with the exact same architecture from [Varley *et al.*, 2016] as shown in Figure 6.3 were trained on this dataset using different sensory data as input. The results are shown in Figure 6.6. One network was only provided the tactile grasp information during training, and performed poorly. A second network was given only the depth information during training, and performed better than the first network, but still encountered many situations where it did not have enough information to accurately complete the back half of the object. The other two networks were given the depth and tactile information. One in the form of a tactile grasp and the other from a tactile exploration. These networks were able to learn the task to completion. They successfully utilized the tactile information to differentiate between plausible geometries of occluded regions. This task demonstrated that a CNN could be trained to leverage incredibly sparse tactile information in order to decide between multiple object geometry hypothesis. For example, when the correct answer had sharp edges, the system would use the tactile information to generate a completion with sharp edges, not simply a

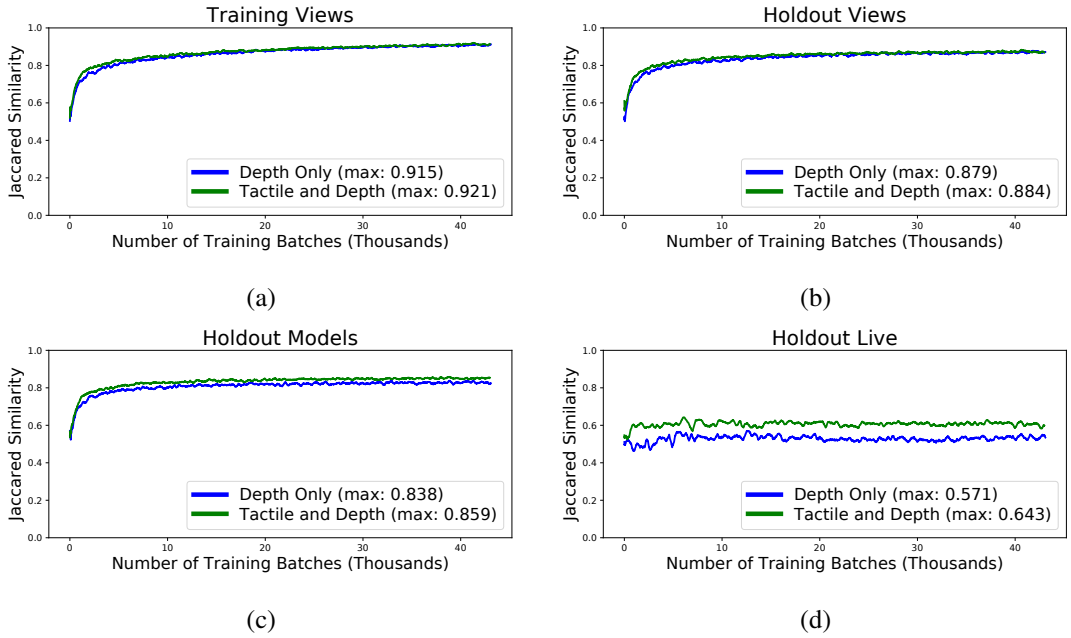


Figure 6.7: Jaccard similarity for two identical CNN models. Depth Only was trained using only depth information, while Tactile and Depth was trained with both tactile and depth information. While training, the two CNNs were evaluated on examples from the training data (Training Views), holdout views of objects from the training data (Holdout Views), views of novel objects not used during training (Holdout Models), and challenging views of of real objects where the depth information was acquired head on (Holdout Live). This sequence of plots shows that as the completion problem becomes more difficult from more challenging depth information, the tactile sensor data is able to provide a larger performance benefit.

completion that is accurate near the sparse tactile observations.

6.5 Completion of YCB and Grasp Dataset Objects

After demonstrating on the geometric shape dataset, we trained two additional models using 486 of the grasp and YCB dataset objects, the remaining models were kept for a holdout set. One model was again trained using only the depth information, while a second model was trained using both depth and tactile information provide from a tactile exploration performed in a similar manner as with the simple shape dataset.

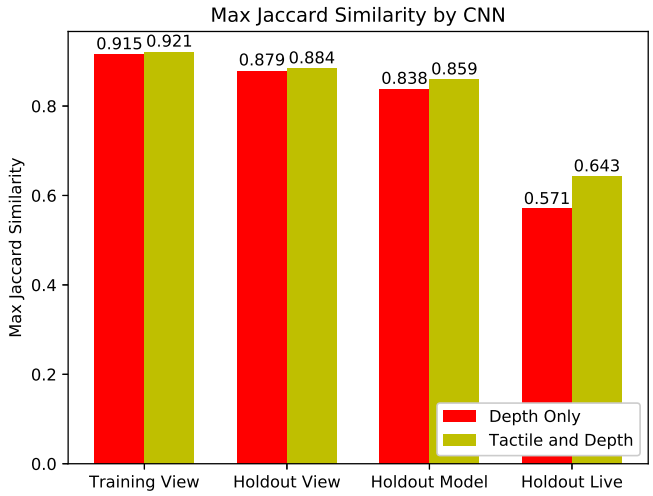


Figure 6.8: As the difficulty of the datasets increases, the max Jaccard Similarity reached by a given CNN decreases, but the difference between the results from the **Depth Only** CNN and the **Tactile and Depth** CNN increases. The additional tactile information is more useful on more difficult completion problems.

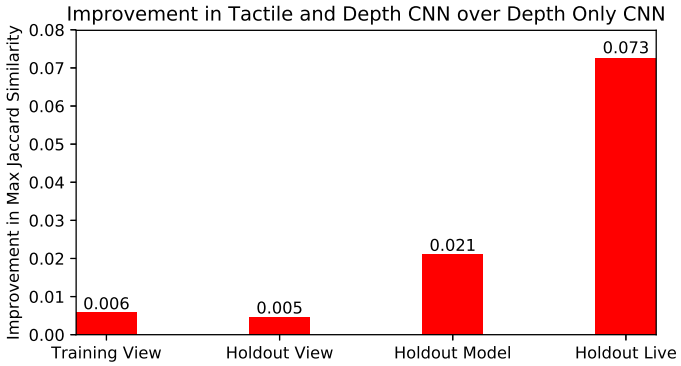
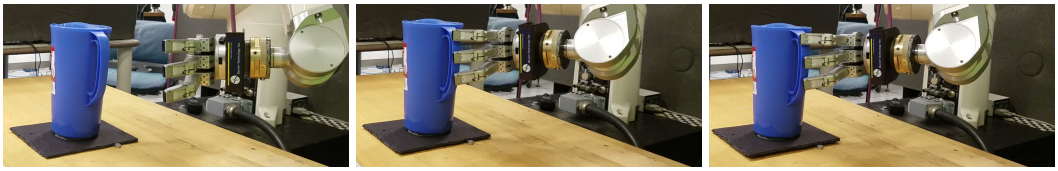


Figure 6.9: As the difficulty of the datasets increases,the difference between the results from the **Depth Only** CNN and the **Tactile and Depth** CNN increases. The additional tactile information is more useful on more difficult completion problems.



(a) Hand approach (b) Finger contact (c) Finger curl

Figure 6.10: Barrett hand showing contact with a fixed object. The hand is first brought manually brought to an approach position shown in a), followed by and approach as shown in b) and then the fingers are curled towards the object to collect tactile information in c). This process is repeated 6 times over the occluded surface of the object.

The dataset consists of approximately half a million pairs of oriented voxel grids. Where one grid's voxels are marked as occupied if visible to a camera, and the second grid's voxels are marked as occupied if the object intersects a given voxel, independent of perspective. This dataset was augmented with tactile information. To generate tactile data, the high resolution model was reoriented to align with the captured partial view, and then 40 rays randomly distributed within the bounding box of the object were traced, and any collisions with the object were marked as tactile contact locations.

Again, two identical CNNs were trained where one CNN was provided only the depth information (**Depth Only**) and a second was provided both the tactile and depth information **Tactile and Depth**. During training, performance was evaluated on views of meshes within the training data (*Training Views*), novel views of meshes in the training data (**Holdout Views**), novel views of meshes not in the training data (*Holdout Models*). Additionally performance was evaluated on a live dataset also not used for training (*Holdout Live*) consisting of depth information captured from a real Kinect and tactile information captured from a real Barrett Hand attached to a Staubli Arm. The object was fixed in place during the tactile data collection process. While collecting the tactile data, the arm was manually moved to place the end effector behind the object, and 6 exploratory guarded motions were made where the fingers closed towards the object, and each finger stopped independently when contact was made as shown in Fig. 6.10. Completions using various methods described below for the entire *Holdout Live* dataset are shown in Figure 6.11.

Figure 6.7 shows the performance of the two CNNs on the different datasets. Notice the performance of the **Depth Only** CNN nearly matches the performance of the **Tactile and Depth**

View Type	Convex Hull	Partial	Ours
Sim. Training Views	32.7	7.8	**5.8**
Sim. Holdout Views	45.1	7.0	**5.8**
Sim. Holdout Models	49.1	7.6	**6.2**
Live Holdout	11.6	11.9	**7.4**

Table 6.1: Hausdorff Distance Results (Smaller is better). This measures the mean distance in millimeters from points on one mesh to another as described in Section 6.6.

View Type	Convex Hull	Partial	Ours
Sim. Training Views	.50	.01	**.69**
Sim. Holdout Views	.51	.02	**.66**
Sim. Holdout Models	.46	.01	**.65**
Live Holdout	.43	.01	**.64**

Table 6.2: Jaccard Similarity Results (Larger is better). This measures the intersection over union of two voxelized meshes as described in Section 6.6.

CNN on the training views. This is because these views are used during training, so at runtime in both cases, the network is provided enough information to generate a reasonable completion. Moving from *Holdout Views* to *Holdout Models* to *Holdout Live*, the completion problems move farther away from the examples experienced during training. As the problems become harder, the **Tactile and Depth** network begins outperforming the **Depth Only** network as it is able to utilize the sparse tactile information to differentiate between various possible completions.

Figure 6.8 and Figure 6.9 show that as the CNNs are exposed to harder completion problems the predicted completions have lower Jaccard Similarity scores. Figure 6.8 also demonstrates that the difference between the **Depth Only** CNN completion and the **Tactile and Depth** CNN completion becomes larger on more difficult completion problems. This shows that the network is able to make more use of the tactile information when the depth information alone is insufficient.

6.6 Comparison to Alternative Visual-Tactile Completion Methods

6.6.1 Alternative Visual-Tactile Completion Methods

In our prior work [Varley *et al.*, 2016] we were able to compare our depth only completion method with both a RANSAC based approach[Papazov and Burschka, 2010], and a mirroring approach [Bohg *et al.*, 2011], both of these approaches make assumptions about the visibility of observed points, and do not directly work with data from tactile contacts that occur in unobserved regions of the workspace.

In our current work we benchmark our framework against the following:

Partial Completion: the set of points captured from the Kinect is concatenated with the tactile data points. The combined cloud is run through marching cubes, and the resulting mesh is then smoothed using Meshlab's[Cignoni *et al.*, 2008] implementation of Laplacian smoothing. These completions are incredibly accurate where the object is directly observed, but make no predictions in unobserved areas of the scene.

Convex Hull Completion: The set of points captured from the Kinect is concatenated with the tactile data points. The combined cloud is run through QHull to create a convex hull. The hull is then run through Meshlab's implementation of Laplacian smoothing as well. These completions are reasonably accurate near observed regions, and fills regions of the unobserved space with the hull.

6.6.2 Comparison Metrics

The Jaccard similarity was used to guide training, as shown in Figure 6.7. We also used this metric to compare the final resulting meshes from several completion strategies. The completed meshes were voxelized at 80^3, and compared with the ground truth mesh. The results are shown in Table 6.2. Our proposed method results in higher similarity to the ground truth meshes than the Partial and Convex Hull approaches for all tested views. In this case, the Partial Completion method performs poorly as it's completions are accurate near to observed regions of the surface, but poorly model the unobserved regions of the object.

The Hausdorff distance is a one-directional metric computed by sampling points on one mesh and computing the distance of each sample point to its closest point on the other mesh. The mean value of a completion is the average distance from the sample points on the completion to their

respective closest points on the ground truth mesh. The symmetric Hausdorff distance was computed by running Meshlab's Hausdorff distance filter in both directions. Table 6.1 shows the mean values of the symmetric Hausdorff distance for each completion method. In this metric, the CNN completions are significantly closer to the ground truth compared to the Partial and the Convex Hull completions. Here the Partial Completion method outperforms the QHull approach as the Partial Completion results are incredibly accurate near the observed surface, but still penalized by the bi-directional nature of the Hausdorff distance, as any unobserved points on the ground truth mesh are far from the surface of the Partial Completion.

6.6.3 Simulation Based Grasp Comparison

In order to evaluate our framework's ability to enable grasp planning, the system was tested in simulation using the same set of completions. The use of simulation allowed for the quick planning and evaluation of over 12,000 grasps. GraspIt! was used to plan grasps on all of the completions of the objects by uniformly sampling different approach directions. These grasps were then executed, not on the completed object, but on the ground truth meshes in GraspIt!. In order to simulate a real-world grasp execution, the completion was removed from GraspIt! and the ground truth object was inserted in its place. Then the hand was placed 20cm backed off from the ground truth object along the approach direction of the grasp. The spread angle of the fingers was set, and the hand was moved along the approach direction of the planned grasp either until contact was made or a maximum approach distance was traveled. At this point, the fingers closed to the planned joint values. Then each finger continued to close until either contact was made with the object or the joint limits were reached.

Table 6.3 shows the differences between the planned and realized joint states as well as the difference in pose of the base of the end effector between the planned and realized grasps. Using our method caused the end effector to end up closer to its intended location in terms of both joint space and the palm's cartesian position.

6.6.4 Performance on Real Hardware

In order to further evaluate our framework, the grasps were planned and executed on the Holdout Live views using actual robotic hardware to execute grasps planned via the different completion

View	Error	Completion Type		
		Partial	Convex Hull	Ours
Training View	Joint (°)	57.4°	24.7°	**17.4°**
	Pose (mm)	19.9	13.9	**7.7**
Holdout View	Joint (°)	47.0 °	28.1 °	**23.8°**
	Pose (mm)	21.1	16.1	**13.9**
Holdout Model	Joint (°)	61.5°	**32.1°**	34.7°
	Pose (mm)	16.6	14.1	**13.6**
Holdout Live	Joint (°)	53.7°	26.4°	**17.0°**
	Pose (mm)	18.6	10.5	**6.2**

Table 6.3: Results from simulated grasping experiments. Joint Error is the mean L2 distance between planned and realized grasps in degrees. Pose Error is the mean L2 distance between planned and realized grasp pose in millimeters. For both metrics smaller is better.

Completion Method	Grasp Success Rate (%)	Joint Error (degrees)
Partial	62.5%	44.6°
Convex Hull	62.5%	42.4°
Ours	**87.5%**	**32.7°**

Table 6.4: Grasp Success Rate shows the percentage of successful grasp attempts. Joint Error shows the mean difference in degrees between the planned and executed grasp joint values.

methods described above. For each of the 8 objects, we ran the arm once using each completion method. The results are shown in Table 6.4. Our method enabled a 25% improvement over the general visual-tactile shape completion methods in terms of grasp success rate, and resulted in executed grasps closer to the planned grasps as shown by the lower joint error.

6.7 Discussion

We have developed a framework for geometric reasoning utilizing a CNN trained from a large simulated dataset of rendered depth and tactile data. The CNN uses both the dense depth information and the sparse tactile information to fill in occluded regions of an object. Experimental results show that using both tactile and visual data provides more accurate completion than using either vision or tactile data alone. In addition, our approach outperforms other general visual-tactile completion methods.

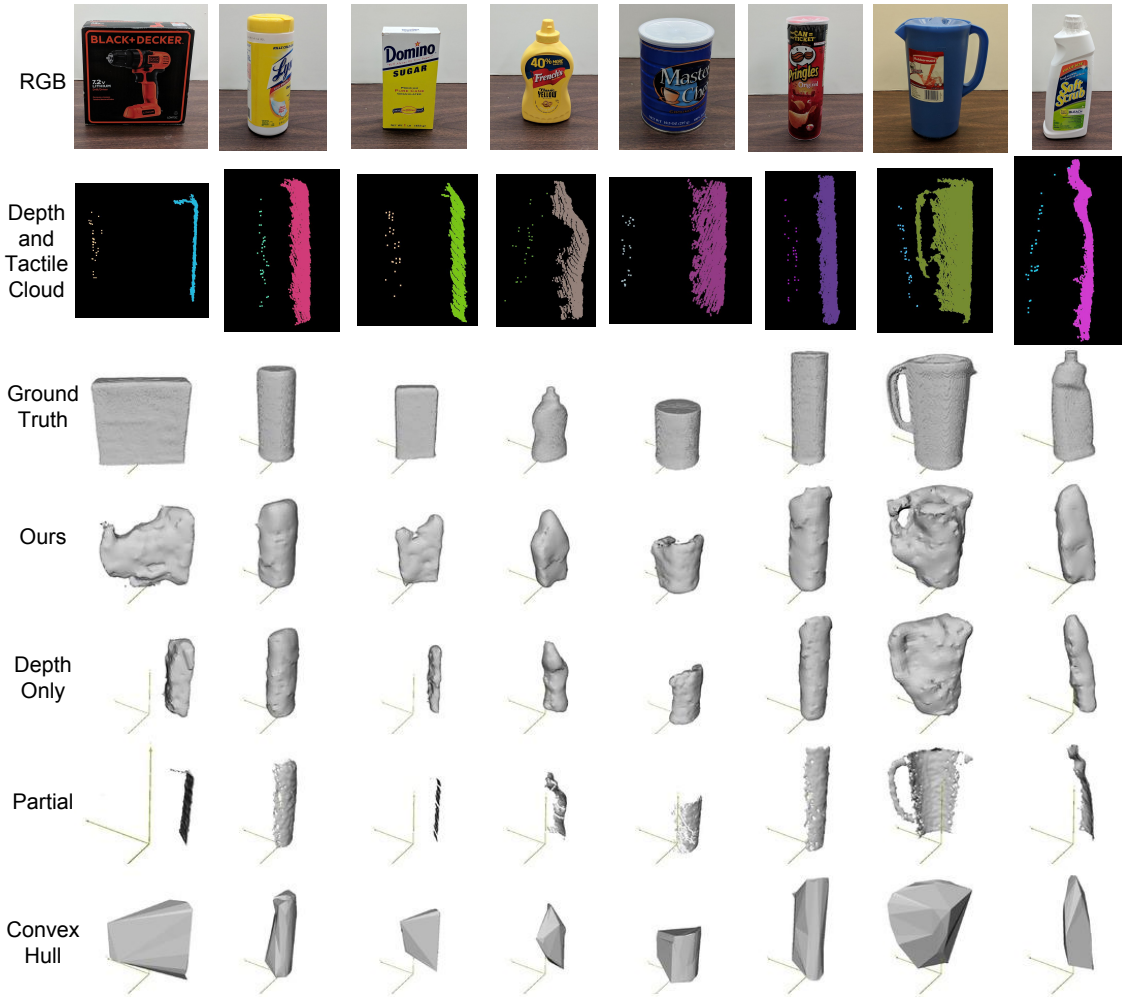


Figure 6.11: The entire *Holdout Live* dataset. These completion were all created from data captured from a real Kinect and a real Barrett Hand attached to a Staubli Arm. Notice many of the **Depth Only** completions do not extend far enough back or look like other objects that were in the training data (ex: cell phone, banana). Our method outperforms the **Depth Only**, **Partial**, and **Convex Hull** methods in terms of Hausdorff Distance and Jaccard Similarity.

Chapter 7

Workspace Aware Online Grasp Planning

7.1 Faster Online Grasp Planning

With the shape completion approach from the previous chapter, our grasping system can now handle novel objects, but using a database of precomputed grasps is not straightforward, as every mesh is now new as shown in Figure 7.1. This means that we must rely upon an online grasps planner that does not assume the object's geometry is known ahead of time. Our current Simulated Annealing based planner becomes a bottle neck in our grasp planning pipeline when the results cannot be cached for later use. One approach for reducing the planning time has been to reduce the dimensionality of the search space used in the annealing process. This was done both by [Ciocarlie *et al.*, 2007] where the concept of an Eigen Grasp was introduced, and the annealing could occur in a reduced dimensionality subspace of the original hand-object configuration space. A similar approach was taken in [Hang *et al.*, 2014] where the annealing was accomplished in a hierarchical manner in order to reduce the search space size. In this chapter we explore how the search space of the annealing process can be reduced by incorporating prior information about what areas of the space the robot can actually reach. There is no reason for the grasp planner to spend time annealing through portions of the workspace for which no feasible IK solutions exist. Here we propose a method to quickly move the annealing process out of these infeasible regions of the workspace.

Grasp planning and motion planning are two fundamental problems in the research of intelligent

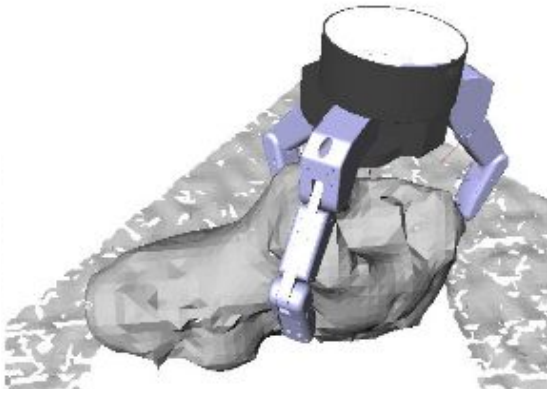


Figure 7.1: The use of general shape completion allows for planning on novel objects, but this makes it impossible to use a precomputed database of grasps since every mesh given to our planner is new. In fact every time the same object is completed the observed pointcloud changes slightly resulting in a slightly different mesh.

robotic manipulation systems. These problems are not new, and many reasonable solutions have been developed to approach them individually. Most of the research has treated these problems as distinct research areas focusing either on: 1) how to get a set of high quality candidate grasps, ex: [Ciocarlie *et al.*, 2007][Hang *et al.*, 2014] or 2) how to generate viable trajectories to bring the gripper to the desired hand configuration, ex: [Sucan *et al.*, 2012][Schulman *et al.*, 2014][Diankov and Kuffner, 2008]. While it is often convenient to treat grasp and motion planning as distinct problems, in this chapter we demonstrate that solving them jointly can improve performance and reduce computation time. Online grasp planners unaware of the robot workspace and without a notion of reachability often spend significant time and resources evaluating grasps that may simply be unreachable. For example, a reachability unaware planner is equally likely to return impossible grasps that assume the robot will approach the object from the object's side furthest from the robot. Our planner avoids unreachable areas of the workspace dramatically reducing the size of the search space. This lowers online planning time, and improves the quality of the planned grasps as more time is spent refining quality grasps in reachable portions of the workspace, rather than planning grasps that may be stable but are unreachable.

In order to generate grasps which are stable and reachable, we propose a new energy function for use with simulated annealing grasp planners [Ciocarlie *et al.*, 2007][Hang *et al.*, 2014]. This energy function is a weighted combination of a grasp stability term, and a grasp reachability term. Our grasp

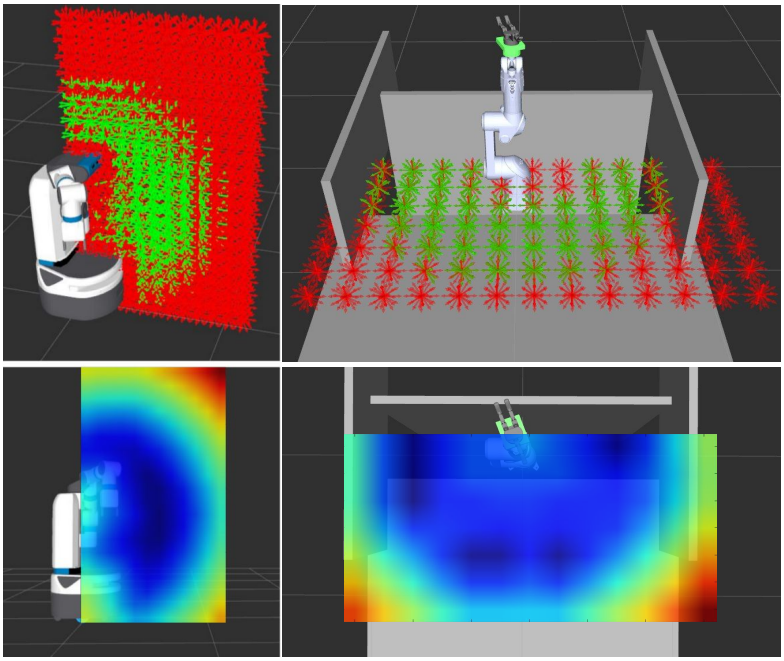


Figure 7.2: Top Row: Visualization of cross sections of the precomputed reachable space for a Fetch Robot and Staubli Arm with Barrett Hand. The green arrows represent reachable end-effector poses, meaning that a, collision free, inverse kinematic solution exists placing the end-effector at the specified pose. The red arrows are not reachable. This space is computed offline, once for a given robot. Bottom Row: Signed Distance Field generated from the above reachability spaces. This representation is incorporated into our grasp planning process.

reachability term is inspired by the idea of a precomputed reachability space which has demonstrated utility in other robotics tasks [Porges *et al.*, 2014][Vahrenkamp *et al.*, 2009]. In our work, we generate a densely sampled reachability space for our robot as shown in Fig. 7.2(Top Row). This offline computation checks whether an Inverse Kinematic (IK) solution exists for a given pose. This database in and of itself has utility for filtering lists of precomputed-grasps and quickly removing unreachable grasps, leaving only reasonable grasps as possible results. We further postprocess our reachability space and produce a compute a Signed Distance Field (SDF) representation Fig. 7.2(Bottom Row). The SDF is used in our online grasp planning energy function to guide the hand towards reachable regions of the robot's workspace.

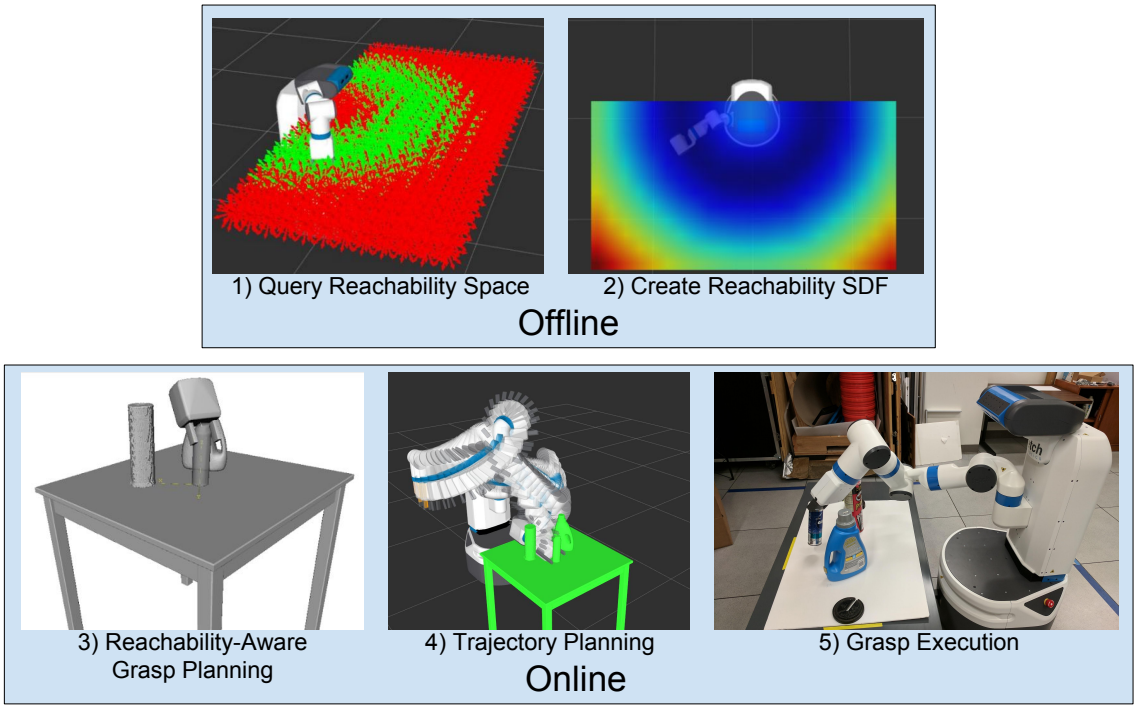


Figure 7.3: Workspace Aware Online Grasping Framework - **Offline**: 1) the robot's reachability space is queried for IK solutions that are free of collisions with the robot itself and static objects such as walls and tables. 2) An SDF is created from the reachability space. **Online**: 3) Grasp planning is quickly accomplished utilizing the reachability space SDF. 4) A motion plan is found for one of the planned grasps. 5) The trajectory is executed by the robot resulting a in a stable grasp.

Experiments in both simulation and physical environments have been performed to assess the performance of our method on multiple objects and under various poses. We demonstrate that our reachability aware grasp planner results in a higher fraction of reachable grasps, a higher fraction of successful grasp executions, and a reduced planning time compared to other online grasp planning methods.

Our method for workspace aware online grasp planning is overiewed in Fig. 7.3. This framework has a number of advantages. First, it combines two closely coupled processes into one; rather than going back and forth between grasp planning and trajectory planning until a feasible grasp is found, we solve for a fully reachable grasp at once. It increases the probability of finding feasible grasps quickly since the optimization process is guided by our energy function within reachable spaces.

Also, our method is well suited for online planning which in general is more adaptable and robust than the traditional offline grasp planning approach and works in the case of novel objects for which precomputed grasps do not exist.

In summary, we take a closer look at the problem of grasp quality and reachability analysis of grasps in an integrated and structured manner. The contributions of this work include: 1) A novel representation of reachability space that uses signed-distance field (SDF) to provide a gradient field during grasp planning; 2) An online grasp planning system with an integrated notion of reachability; 3) A formulation that ranks a list of grasps (e.g. from a database) based on their combined reachability and grasp quality; 4) An embedding of obstacles into the reachability space during grasp planning; 5) An analysis of our framework compared to other grasp planning solutions evaluated in multiple scenarios.

While our work is most similar to [Berenson *et al.*, 2007] [Vahrenkamp *et al.*, 2009] in being workspace aware, ours differs in a number of ways: (1) we focus on solving online grasp planning, (2) we compute the actual reachability map, not an approximation as in [Berenson *et al.*, 2007] (3) we use a novel representation of the reachability space and (4) we use a different optimization technique, simulated annealing.

7.2 Offline Reachability Space Generation

A grasp is reachable if a motion plan can be found to move the arm from its current configuration to a goal configuration that places the hand at desired grasp location. This is not always possible for a number of reasons typically because no inverse kinematics(IK) solution exists to place the hand at the desired grasp pose, self collision with other parts of the robot, or collision with obstacles in the planning scene. Our method addresses these issues by incorporating the notion of reachability during grasp planning.

Algorithm 3 Reachability Space Generation

```
1: procedure GENERATEREACHABILITYSPACE
2:     poses = uniformSampleWorkspace()
3:     pose2Reachable = {}
4:     for pose in poses do
5:         reachable = hasCollisionFreeIK(pose)
6:         pose2Reachable[pose] = reachable
7:     SDF = computeSDF(pose2Reachable)
8:     return SDF
```

7.2.1 Reachability Space Representation

We observe that this definition of the original reachability space is binary, either 1 (reachable), or 0 (not reachable), and has no gradient to indicating the direction from non-reachable regions to reachable portions of the workspace. In the context of Simulated Annealing for grasp planning, it is beneficial to provide a energy function that provides clear cues about how to move to more desired regions of the annealing space. This observation informed a novel signed-distance-field (SDF) representation of the reachability space that is amenable to optimization formulations. Our SDF maps a pose to a value representing the distance to the manifold or boundary between the unreachable poses and reachable poses and can be interpreted as follows $d_{sdf} = SDF(pose)$:

- $d_{sdf} = 0$: pose lies exactly on boundary between reachable and unreachable grasp poses.
- $d_{sdf} < 0$: pose lies withing the reachable region of the workspace, and is distance d_{sdf} away from the boundary.
- $d_{sdf} > 0$: pose lies outside the reachable region of the workspace, and is distance d_{sdf} away from the boundary.

This field can then serve as criteria for classifying a grasps as either lying in a reachable space or not. And two nearby hand configurations can be ordered based on d_{sdf}, their distance from this boundary.

Table 7.1: Discretization of robot workspace for SDF generation for the Fetch robot. Values are in meters (x, y, z) and radians (roll, pitch, yaw). The origin of the space is centered at the robot base.

	X	Y	Z	Roll	Pitch	Yaw
min	0.0	-1.1	0.0	-PI	-PI	-PI
max	1.2	1.1	2.0	PI	PI	PI
step	0.1	0.1	0.1	PI	PI/4.0	PI/4.0

7.2.2 Reachability Space Generation

Reachable-space generation is a one-time process for a given robot. Algorithm 3 details the process for generating the Reachable Space. We search a discretization of the 6D pose space of the robot to determine which configurations will be reachable by the hand. For the Fetch Robot, we sampled the workspace using the values in meters centered at the robot's base shown in Table 7.1. In total 675,840 unique poses where queried for reachability. 102,692 were reachable and 573,148 where unreachable. This was computed using Moveit!'s IK solver which checks for a valid collision free IK solution for a given pose. The poses and IK query results were used to generate an SDF using sci-kit-fmm[1]. Once this computation is done, the SDF can be queried with any pose inside the space to determine both its binary reachability and distance to the boundary separating reachable and non reachable poses.

7.3 Online Reachability-Aware Grasp Planning

Grasp planning is the process of finding "good grasps" for an object that can be executed using an articulated robotic end effector. A grasps is typically classified as "good" based on how well it can resist internal and external forces which largely determines the success of the grasp. The Ferrari-Canny method [Ferrari and Canny, 1992] is a common way to evaluate robot grasps. Other techniques are typically built around this metric as shown in this review [Roa and Suárez, 2015]. Grasp planning frameworks such as GraspIt! use these grasp metrics of force and form closure as objective functions for optimization. Since this formulation has no notion of reachability, the grasp

[1] https://github.com/scikit-fmm/scikit-fmm

results, while stable, might require the end effector to be placed in a pose that is impossible to reach given the robot's current location.

7.3.1 Simulated Annealing for Grasp Planning

Grasp planning can be thought of as finding low energy configurations within the hand object space. This search is done in a multidimensional space where grasps are sampled and evaluated. The 6 dimensional (x, y, z, roll, pitch, yaw) space we generated with our SDF is a subspace of the 6 + N dimensional grasp search space. The additional N dimensions represent the EigenGrasps or eigen vectors of the end effector Degree of Freedom (DOF) space as described in [Ciocarlie *et al.*, 2007]. For the Fetch (1 DOF), N=1, and for the Barrett Hand (4 DOF), N=2. These 6 + N dimensions describe both the pose and DOF values of the end effector. The goal of grasp planning is to find low energy points in this 6 + N dimensional space. These points represent the pose and hand configuration of "good grasps".

Simulated annealing (SA) [Ingber, 1989] – a probabilistic technique for approximating global optimums for functions lends itself nicely for our formulation. SA presents a number of advantages in grasp planning[Allen *et al.*, 2014]. For example, it does not need an analytical gradient which can be computationally infeasible; it handles the high nonlinearity of grasp quality functions; and it is highly adaptable to constraints. Our approach implicitly guides the annealing process ensuring that sampling is done in regions of reachable space which increases the probability of getting useful grasp solutions. While online grasping is typically avoided due to time cost of exploring a larger search space, this work makes online grasping more feasible since the majority of the space which is unreachable need not be searched. With our new energy formulation, the annealing process will quickly drive the hand towards reachable grasp locations. Section 7.4 below shows that this new energy function increases the success probability of online grasping significantly.

7.3.2 Novel Grasp Energy Formulation

In our work, we augment the *Contact and Potential* grasp energy function from [Ciocarlie *et al.*, 2007]. The *Contact and Potential* energy function is described in Algorithm 4 and consists of a potential energy term measuring the grasps potential to resist external forces and torques and a contact energy term that brings the hand closer to the object. Our method (*Reachability Aware*)

Algorithm 4 Contact And Potential Energy (SA-CP)

```
1: procedure CONTACTANDPOTENTIALENERGY
2:     e_p = potentialEnergy()
3:     stable = e_p < 0
4:     if stable then
5:         return e_p
6:     else
7:         e_contact = contactEnergy()
8:         return e_contact
```

Algorithm 5 Reachability-Aware Energy (SA-OURS)

```
1: procedure REACHABILITYAWAREENERGY
2:     e_p = potentialEnergy()
3:     e_reach = reachabilityEnergy()
4:     stable = e_p < 0
5:     reachable = e_reach < 0
6:     if reachable && stable then
7:         return e_p + α_1 · e_reach
8:     else if reachable && !stable then
9:         e_contact = contactEnergy()
10:        return e_contact + α_2 ·(reach_max - e_reach)
11:    else //!reachable
12:        e_contact = contactEnergy()
13:        return e_contact + α_3 · e_reach
```

energy augments the *Contact and Potential* with a *Reachability* term that encodes the kinematic constraints of the robot. In this cost function for grasp planning optimization, we use the reachability SDF described previously as a regularizing term for our search energy to drive the hand towards reachable hand configurations. As shown in [Oleynikova *et al.*, 2016] discretized SDFs still model and behave like a continuous function so it can be used during the simulated annealing optimization process to drive the search towards reachable hand configurations. In terms of implementation, we build on the GraspIt! simulator and combine the existing energy metrics with the new reachable energy term. The energy function E is

$$E = G + \alpha R \tag{7.1}$$

where G is a metric of grasp quality (e.g. force closure). R is a measure of reachability of a given grasp pose. To optimize the overall grasp energy, we use the simulated annealing search according to [Ciocarlie and Allen, 2009].

The grasp quality term G contains two terms: contact energy ($e_{contact}$) and potential energy (e_p). Contact Energy defines the proximity of the hand to the object being grasped while the potential energy captures the quality of grasp in terms of force closure analysis. The reachability energy (e_{reach}) is the new term that we use to guide the optimization process to come up with grasps in reachable regions. In order to sample points from our discrete reachability space, we use a multi-linear extension of bilinear interpolation to get a continous reachability value for any point in space. Each query pose during the optimization is situated in the robot's reachable space and the reachability value for that point is given as the multi-linear interpolation (weighted sum) of the values for the 2^N ($N = 6$) corners of the hypervoxel where the query point falls in the pre-computed SO(6) reachable space. The 2^N corner values are looked up in the pre-computed rechability space.

$$R[p_{query}] = \sum_{i=1}^{2^N} w_i R[p_i] \tag{7.2}$$

where $p_{query} = [x_q, y_q, z_q, r_q, p_q, y_q]$ is the query pose, $p_i = [x_i, y_i, z_i, r_i, p_i, y_i]$ for $i = 1...2^N$ correspond to the neighbouring corners in the reachability space grid and w_i is the proportion of the grid occupied by creating a volume from connecting the query point and the corner p_i. See [Wagner, 2008] for more details.

The α value in equation 7.1 defines the tradeoff between the conventional grasp metric and the reachability value. Algorithm. 5 shows specifically how these three energy terms are combined.

Given the scale of the original grasp function, we chose the α that punishes non-reachable grasps in a way that provides a gradient from bad to good quality (force-closure) grasps. α values were set to $\alpha_1 = -0.1$, $\alpha_2 = 1$, $\alpha_3 = -10$ making a continuous function where negative low energy values correspond to reachable stable grasps, and positive high energy values correspond to unreachable and unstable grasps.

7.3.3 Embedding Obstacles in the Reachability Space

Obstacles whose poses are not expected to change in relation to the robot can be placed in the scene while generating the initial reachability space. This approach is applicable to modeling tables, walls and fixtures around fixed base robotic arms, or static room environments for mobile manipulators. This is useful for the Staubli Arm and attached Barrett Hand as shown in Fig. 7.2. This robot is fixed to the table, next to the walls, making it reasonable to incorporate these obstacles during the initial reachability space creation.

In order to incorporate obstacles into our precomputed reachability space, we mask out regions in the robot reachability space that overlap with an obstacle i.e. locations that were otherwise reachable but now overlap with an obstacle are marked as unreachable. This corresponds to modifying the reachable space in the first block of Fig. 7.3. Then we can then regenerate an SDF representation of the resulting reachable space. The SDF generation is fast, taking 2-3 seconds on average, and this step is only required when the workspace changes.

7.4 Experiments

We describe different experiments in both simulation and physical environments performed to assess the performance of our method on multiple objects and under various poses. We use the Fetch mobile manipulator from Fetch Robotics as our robot platform (both simulated and real experiments).

7.4.1 Grasp Planners

For experiments we compared the following grasp planning methods:

Sim. Ann. Contact and Potential(SA-C&P): Simulated Annealing in GraspIt! using the Contact and Potential energy function from Algorithm 4.

Sim. Ann. Reachability-Aware(SA-Ours): Simulated Annealing in GraspIt! using the newly proposed energy function from Algorithm 5.

Uniform Sampling(Uniform): Uniformly sample 20 grasps around the z-axis of the object as shown in Fig. 7.4

Grasp Database(Grasp DB): Retrieve 60 pre-computed grasps from a database for the given object sorted by energy These grasps were computed by running the (SA-C&P) planner ahead of time, 3x per object for 80,000 steps each run, this is significantly longer than what is done at runtime.

7.4.2 Evaluation Metrics

The metrics used for evaluating our methods include:

Percent Reachable Grasps: The number of reachable grasps divided by the total number of grasps generated from a run of a given grasp planner.

Number Required Plan Attempts: The grasp planner returns a list of grasps ordered according to quality. We record how many of these grasps we have to go through until a valid path can be planned.

Lift Success: Here we execute the first valid grasp from the set returned by the planner. A grasp is successful if the gripper placed at the grasp pose can successful lift the object off the ground. We use the Klamp't simulator [Hauser, 2016] to test this in simulation. For the real world experiments, we executed the grasps on the real robot and checked if the object was picked up.

7.4.3 Uniform-Sampled Grasp Ranking Experiment

Here we set-up a typical grasp planning pipeline; the **Uniform** planner is used to generate a set of 20 possible grasps, and the robot goes through the list of grasps attempting to create a valid motion plan. The first grasp that a valid trajectory can be planned for is then executed. This experiment was done for 13 unique objects as shown in Fig 7.4. Each object was run 24 times at different poses. This resulted in 6240 unique grasps. These grasps were sorted using two energy functions Ours(Alg. 5) and Contact and Potential(Alg. 4); We check the reachability of all grasps in the sorted lists and evaluate the fraction of reachable grasps in each ranked position. In other words, what fraction of the time is the first grasp in the list reachable when sorted by our Reachability Aware energy function, what fraction of the time is the second grasp in the list reachable, and the third grasp, etc.

Figure 7.5 shows the mean fraction of reachable grasps at each position with grasps ranked using

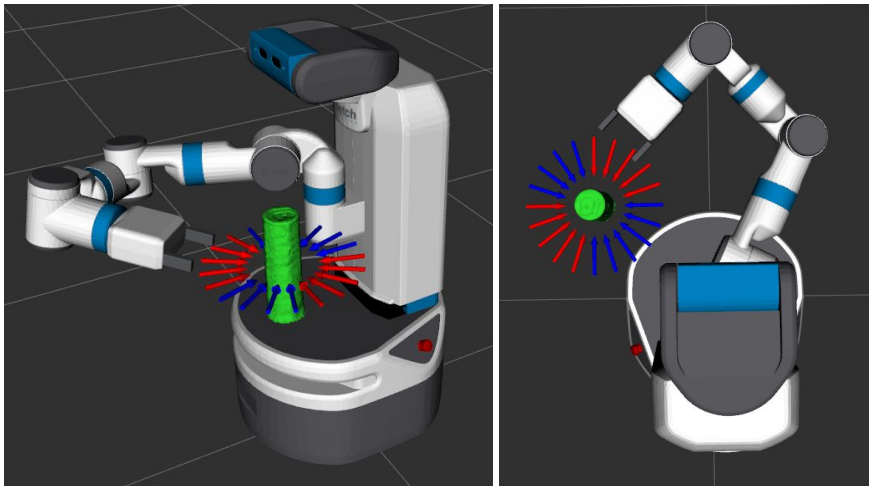


Figure 7.4: Reachability results for uniformly sampled grasps. Red arrows are reachable, blue are unreachable. This visualization shows that even for objects well within the bounds of the robot workspace, there are many invalid approach directions which are not easily modeled by simple heuristics.

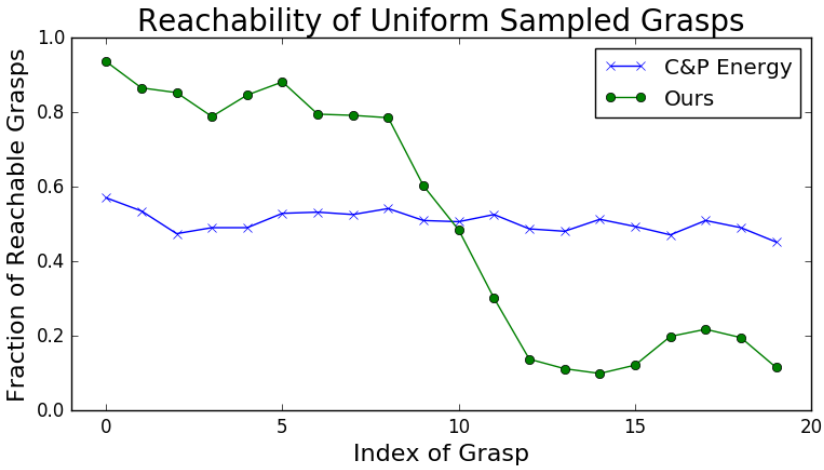


Figure 7.5: Mean fraction of reachable grasps at each position with grasps ranked using different energy functions. X-axis is the index of the grasp in the list of 20, so 0 for the best grasp in the list, 19 for the worst. Y-axis is the percentage of reachable grasps with a given index. For our energy the 0th index or best grasp is reachable 95% of the time. Without reachability, a top ranked grasp has an equal chances of being reachable as the 19th index grasp.

different energy functions. Without the reachability term, a top ranked grasp has equal chances of being reachable than a lowly ranked one. With reachability information, the probability distribution is adjusted to make top-ranked grasp have a high chance of being reachable. In this test environment, the first grasp returned by our method is reachable 95% of the time, while *any* grasp returned by the Contact and Potential function is reachable less than 60% of the time.

7.4.4 Online Planning Challenge Scenes Experiment

Here, one of 4 target objects was placed on a table in a virtual scene in 9 very difficult to reach poses that at were either at the extent of the robot's workspace, or extremely close to the robot. Both of these situations drastically limit the number of valid approach directions that the robot can use to grasp the object. For each pose, GraspIt!'s Simulated Annealing planner was run for 40,000, 10,000 and 2,500 steps using both the **SA-Ours** and **SA-C&P** energies. The results are shown in Table 7.2. The table shows that since the reachability term guides the sim-ann process to reachable regions, we can run the **SA-Ours** for signicifantly fewer steps than **SA-C&P**. From the table, **SA-Ours** returns double the percentage of reachable grasps compared to **SA-C&P** regardless of the number of steps the planner is run for. The reachability term implicitly shrinks the search space and guides the search to occur only in reachable regions. This improves the planner performance and allows for a significant reduction in the number of search steps required.

7.4.5 Comparison to Grasp Database Approach

Using the same simulated challenge scenes setup as above, we compared our method to a **Grasp DB** approach with the following result: % Reachable Grasps: (Ours: **86.23%**, GraspDB: 19.95%), # Required Plan Attempts: (Ours: **0.08**, GraspDB: 8.83), Lift Success: (Ours: 88.89%, GraspDB: **94.44%**). The **Grasp DB** approach is excluded from the Table 7.2 because it is not an online grasp planner, it requires knowledge of the object models ahead of time so that grasps can be precomputed. Our method can be paired with geometric reasoning systems such as [Varley *et al.*, 2016] and plan for never before seen objects. Using only 10,000 annealing steps our method is able to come within 6% of the Lift Success performance of the **Grasp DB** that had a-priori information about the scene objects.

Table 7.2: **Klampt Simulation Results:** *% Reachable Grasps*: fraction of grasps returned by GraspIt! that are reachable. Once GraspIt! returns a list of grasps, we iterate through the grasps and attempt to plan a valid path for them in order of grasp quality. *# Required Plan Attempts* is how many grasps we have to go through before finding a valid grasp that a path can be planned for. *Lift Success*: percentage of the time that the valid grasp results in a successful 10 cm lift of the object.

Simulation Results on Online Planning Challenge Scenes			
	% Reachable Grasps	# Required Plan Attempts	Lift Success
Simulated Annealing with 40,000 Planning Steps			
SA-Ours	78.38 %	1.083	86.11 %
SA-C&P	34.53 %	3.639	69.44 %
Simulated Annealing with 10,000 Planning Steps			
SA-Ours	**86.23** %	**0.08**	**88.89** %
SA-C&P	33.27 %	5.11	66.67 %
Simulated Annealing with 2,500 Planning Steps			
SA-Ours	75.46 %	0.16	63.89 %
SA-C&P	36.51 %	4.17	41.67 %

7.4.6 Grasp Planning with Dynamic Obstacles Experiment

The experimental setup as Sec. 7.4.4 is modified to include two additional distractor obstacles representing new objects that have entered the workspace after the reachability space has already been generated. The following different methods of handling new obstacles that are introduced at runtime are evaluated below:

Unmodified SDF: Load target object and new obstacles into the grasp planner: This provides a hard constraint that the hand cannot interpenetrate the objects during grasp planning. The full arm may still interpenetrate the object as only the hand is modeled in the grasp planner. 75.8% percent of grasp results by this method were reachable. (10% drop from 86.23% case in Table 7.2 with no additional obstacles.)

Dynamic Obstacles Updated SDF: Load target object and obstacles into the grasp planning scene and also embed the obstacles in the reachability space. The SDF can then be regenerated, a process that takes 2-3 seconds on average. This updated SDF guides the annealing process both away from the obstacles and towards reachable regions of the space increasing the % Reachable Grasps to 89.5%.

By embedding the new obstacles into the SDF, we are able to plan grasps that are 89.5% reachable compared to 75.8% reachable when the SDF only avoids self collision and static objects such as tables and walls.

7.4.7 Real Robot Crowded Scene Experiment

To verify our planner and demonstrate that it works outside of simulation, the simulated annealing based grasp planner was run with a variable number of annealing steps and the success rate was reported in Table 7.3. This experiment compares two grasp energy formulations: **SA-Ours** and **SA-C&P**.

Multiple objects were placed in the planning scene as shown in Fig 7.6. The additional objects reduce the range of possible grasps since as the obstacles make many grasp poses infeasible. Note that both methods are aware of the obstacles during grasp planning and avoid grasps in collision with obstacles. We are able to show that our method, **SA-Ours**, is able to achieve grasp success – picking the object three times out of three within the limited planning time. Conversely, **SA-C&P** fails to produce a reachable grasp 2/3 times when run for 10,000 steps, and fails once even when it is

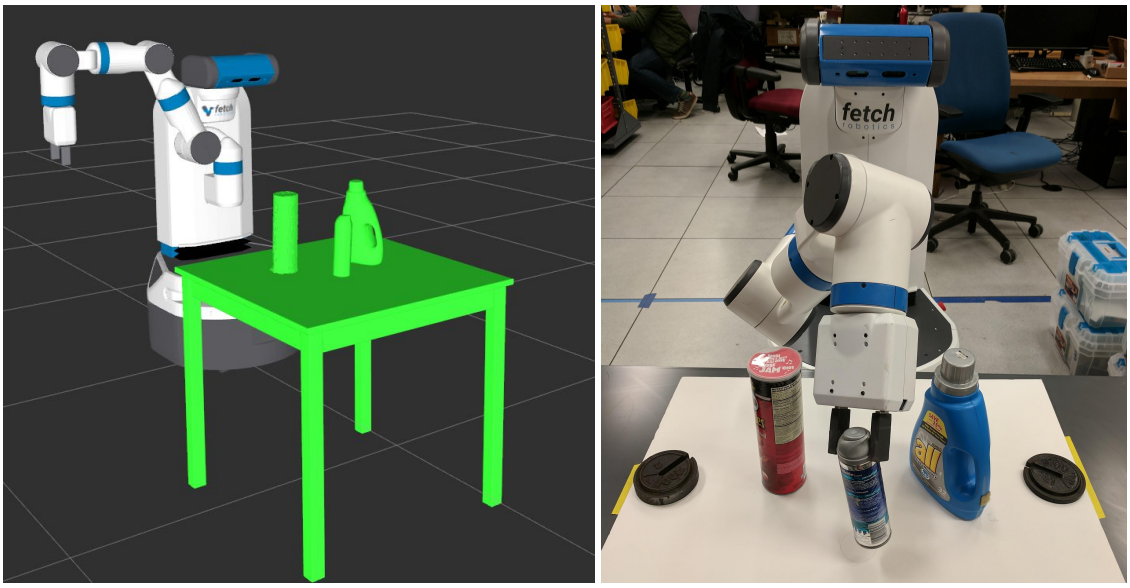


Figure 7.6: Crowded experiment scene for real world experiments. Our Reachability-Aware planner (SA-Ours) was able to successfully grasp the shaving cream bottle 3/3 times, while annealing without the reachability space (SA-C&P) failed 2/3 times.

Table 7.3: Grasp success results on real robot with a crowded scene. Each method was given 3 attempts to plan and execute a grasp on the shaving cream bottle.

Crowded Scene Grasp Planning			
Search Energy	# Planning Steps	Success Rate	Mean Grasp Planning Time (s)
SA-Ours	10K	**100.0%**	8.706
SA-C&P	10K	33.3%	8.360
SA-C&P	40K	66.6%	32.31

allowed to run for 40,000 steps. Despite being allowed to run for this long duration, the naive planner spends much of its time exploring the back half of the bottle which is completely unreachable to the Fetch robot.

7.5 Discussion

Several future research directions include: utilizing our computed reachability workspace to help mobile robots navigate to optimal locations for manipulating objects and improvements for speeding up the process of incorporating dynamic objects into our notion of reachability to further assist the grasp planner.

This work provides a framework for a workspace aware grasp planner. This planner has greatly improved performance over standard online grasp planning algorithms because of its ability to incorporate a notion of reachability into the online grasp planning process. This improvement is accomplished by leveraging a large precomputed database of over 675,84 unique end-effector poses which have been tested for reachability. At runtime, our grasp planner uses this database to bias the hand towards reachable end effector configurations. This bias allows the grasp planner to generate grasps where a significantly higher percentage of grasps are reachable, a higher percentage result in successful grasp executions, and the planning time required is reduced.

Chapter 8

Conclusion

In this thesis, a variety of algorithms and techniques were introduced to improve robotic grasping systems. Chapters 4 and 7 overviewed how simulated data could be leveraged to improve grasp planners. Chapter 5 looked at how a CNN can be used reason about object geometry, and produce meshes useful for robotic grasp planning. In Chapter 6 a CNN was trained to provided an updated understanding of object geometry using both depth and tactile information. All of these ideas are overviewed in Figure 8.1 and were designed to address the problem statement of this thesis.

Providing robots with the ability to grasp objects has remained a challenging problem despite decades of research. Historically, robotics has been made approachable by constraining the systems environment and encoding ample prior knowledge about how to deal with the scene and objects that will be manipulated in heuristics or hard-coded rules. The problem this thesis looks to address is how can we replace these hard-coded heuristics and required constraints in order to build systems that scale beyond specific situational instances and gracefully operate in a wide variety of novel conditions. The heuristic rule based strategies of the past that were used to accomplish tasks such as scene segmentation or reasoning about occlusion do not scale. These heuristic strategies only work in constrained environments where a roboticist can make simplifying assumptions about everything from the geometries of the objects to be interacted with, level of clutter, camera position, lighting, and a myriad of other relevant variables.

This thesis provides a strategy to move robots into more general environments and move beyond heuristic based strategies. We demonstrate how a grasping system can be built that is robust to changes in environmental variables. This robustness is enabled by a new found ability to empower

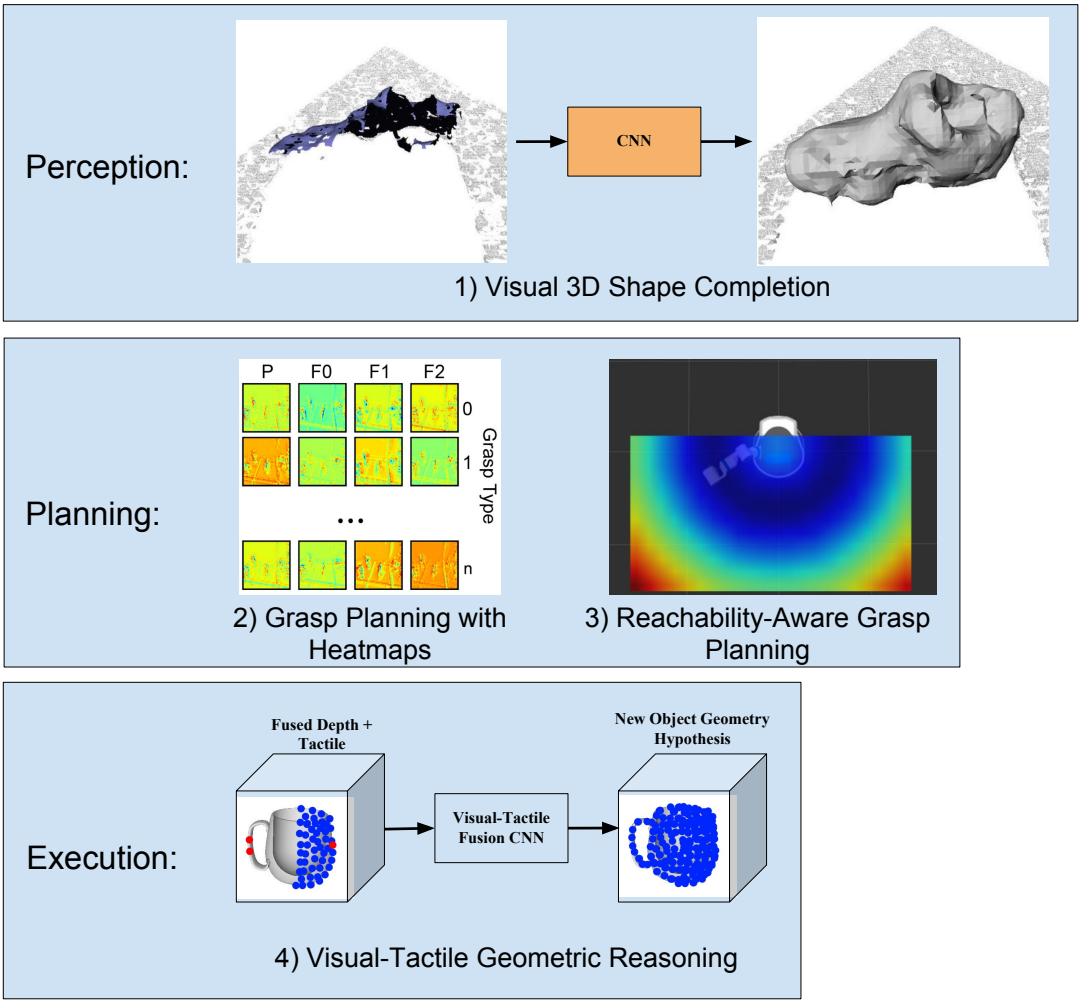


Figure 8.1: Overview of contributions from this thesis and where they fit within a perception, planning and execution breakdown of robotic grasping. The contributions include: The shape completion work from Chapter 5. Novel approaches to grasp planning with heatmaps (Chapter 4), and reachability aware planners (Chapter 7). We have also covered how visual and tactile information can be combined to improve our understanding of object geometry (Chapter 6).

novel machine learning techniques with massive amounts of synthetic training data. The ability of simulators to create realistic sensory data enables the generation of massive corpora of labeled training data for various grasping related tasks. We show that it is now possible to build data driven systems that works in the real world trained on synthetic data. These datasets are embedded in effective mediums such as neural networks or signed distance fields so that the information can be leveraged at runtime. In addition, this ability to train and test on synthetic data allows for quick iterative development of new perception, planning and grasp execution algorithms that work in a large number of environments. In summary the purpose of these thesis has been to aggressively avoid encoding assumptions about the environment and tasks to be accomplished within hand crafted heuristics. Instead, we create a robust framework for robotic grasping that is data driven and made effective by leveraging simulation to create massive datasets embedded in CNNs and other useful representations that are efficiently used at runtime.

8.1 Thesis Summary

The initial work in this thesis towards accomplishing this goal have been the creation of a grasp planner that has learned what stable grasps look like in RGBD images. The planner has distilled a dataset of prior experience into a CNN that can be used at runtime to produce heatmaps useful for annealing. This learned measure of grasp stability meant that the grasp planner could learn to reproduce semantically meaningful grasps, or any other desired grasps by providing the system with training data representative of the desired behavior.

Next we demonstrated a novel shape completion algorithm for enabling a robotic grasping system to reason about occlusion. We showed that this skill improves the system's ability to plan and execute stable grasps compared to both heuristic and other data driven methods for reasoning about object geometry. Again the data for this system was generated in simulation and embedded into a CNN for easy use at runtime. New training data could be simulated to adjust the system's behaviour rather than writing additional code. This 3D CNN allowed are grasping system to cope with 100s of meshes viewed from hundreds of angles and to handle novel objects gracefully. This offered a huge improvement compared to the RANSAC based approach which could only handle tens of objects or the symmetry based approaches which required strong assumptions to be satisfied about object

geometry.

Additionally, we built upon the initial shape completion work, and built a system capable of utilizing a large dataset of both tactile and depth data to improve a model of an object using multiple sensory modalities. Again, the data used in this work was generated in simulation, and the architecture used here allowed us to do away with heuristics and constraints related to symmetry and axes of extrusion.

Finally, we showed how to leverage a large precomputed dataset of IK checks done in simulation. The reachability results from the IK checks were embedded into a signed distance field, and algorithms were developed in order to speed up a simulated annealing based grasp planner by using this SDF. Not only did the use of the reachability space speed up grasp planning, it also improved the resultant planned grasps as more time was spent in feasible regions of the workspace.

8.2 Current Limitations and Future Work

One of the many exciting things about robotics research is that there is always more to be done. One avenue of future research is the further integration of several of the works within this thesis. Particularly, the shape completion work with the Heatmaps for Grasp planning work and the reachability space. It should be feasible to extend the 2D heatmaps into 3D heatmaps so that the locations of the palm and fingers do not have to be projected back into an image plane. Then it will be possible to run a simulated annealing planner while utilizing 3 sets of 3D grids, the reachability SDF, an occupancy voxel grid from the shape completion work, and a grasp affordance voxel grid from an extended version of the heatmap work. This should make an incredibly fast planner, and these maps may also be interesting to use as a rich 3D internal representation of a workspace for the utilization of a Reinforcement Learning agent. It would also be interesting to modify the heapmap CNN to directly produce a grasp location and posture.

Several future avenues of research for the shape completion work include the use of Generative Adversarial Networks (GANs)[Goodfellow *et al.*, 2014] for training, migrating to a larger object dataset such as ShapeNet[Chang *et al.*, 2015], and using the completed object as a query to retrieve grasps planned on similar objects as done with the Columbia Grasp Database[Goldfeder *et al.*, 2009a] and DexNet[Mahler *et al.*, 2016]. It would also be exciting to more closely coupling the CNN trained

to reason about object geometry with grasp planning, in similar spirit to the work done here [Yan *et al.*, 2017].

For the Depth and Tactile refinement work, the next generation of tactile sensors such as GelSight will provide significantly more detailed information about the contacts occuring between the object and hand compared to the built in capacitive sensors on the Barrett hand. This improved sensors should continue to make research in tactile sensing more tractable as many current sensors are incredibly noisy, expensive, and exhibit a large amount of hysteresis. We can also look forward to the continued improvement of our ability to simulate contact sensor data in simulation so that we are able to build larger and more accurate synthetic datasets.

From the Reachability work, It should be possible to extend the SDF to model both the arm and the hand rather than just the arm. It would also be interesting to use the SDF annealing framework that was setup with other information such as priors for how likely the current hand configuration is to be stable.

8.3 Learning To Grasp: A Stepping Stone

Robotic grasping is a stepping stone in building functional robotic systems capable of operating outside of constrained laboratory, factory, and warehouse conditions. Building systems capable of interacting within our homes, work environments, outdoors, and other locations requires abilities to; Perceive and organize the rich stream of sensory data these systems are exposed to. Plan how to execute complicated motions that are motivated by and included the rich corpus of semantic common sense knowledge that we posses. And an ability to realize a desired plan while incorporating additional sensory feedback. These 3 problems can all be explored in the test bed of robotic grasping. Grasping is a precursor to manipulating and interacting with our environment, and once we can pick things up in a meaningful and robust way, I don't think it will be much more difficult to figure out how to reorient grasped objects, or use them as tools to further affect changes in the environment. All of these tasks will tap into the same core skill set that is required to accomplish robotic grasping. Once we can solve grasping, we will quickly soon after be able to build robotic systems that can do many things.

Part I

Bibliography

Bibliography

[Allen *et al.*, 2014] Peter K Allen, Matei Ciocarlie, and Corey Goldfeder. Grasp planning using low dimensional subspaces. In *The Human Hand as an Inspiration for Robot Hand Development*, pages 531–563. Springer, 2014.

[Bastien *et al.*, 2012] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I., A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio. Theano: new features and speed improvements. *arXiv:1211.5590*, 2012.

[Berenson *et al.*, 2007] Dmitry Berenson, Rosen Diankov, Koichi Nishiwaki, Satoshi Kagami, and James Kuffner. Grasp planning in complex scenes. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, pages 42–48. IEEE, 2007.

[Bergstra *et al.*, 2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4. Austin, TX, 2010.

[Bierbaum *et al.*, 2008] Alexander Bierbaum, Ilya Gubarev, and Rüdiger Dillmann. Robust shape recovery for sparse contact location and normal data from haptic exploration. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3200–3205. IEEE, 2008.

[Bjorkman *et al.*, 2013] Marten Bjorkman, Yasemin Bekiroglu, Virgile Hogman, and Danica Kragic. Enhancing visual perception of shape through tactile glances. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3180–3186. IEEE, 2013.

[Bohg *et al.*, 2011] Jeannette Bohg, Matthew Johnson-Roberson, Beatriz León, Javier Felip, Xavi Gratal, Niklas Bergström, Danica Kragic, and Antonio Morales. Mind the gap-robotic grasping under incomplete observation. In *ICRA*, pages 686–693. IEEE, 2011.

[Bohg *et al.*, 2014] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *Robotics, IEEE Transactions on*, 30(2):289–309, 2014.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[Caccamo *et al.*, 2016] Sergio Caccamo, Yasemin Bekiroglu, Carl Henrik Ek, and Danica Kragic. Active exploration using gaussian random fields and gaussian process implicit surfaces. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 582–589. IEEE, 2016.

[Calli *et al.*, 2015] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 510–517. IEEE, 2015.

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Chollet, 2015] Francois Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

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

[Cignoni *et al.*, 2008] Paolo Cignoni, Massimiliano Corsini, and Guido Ranzuglia. Meshlab: an open-source 3d mesh processing system. *Ercim news*, 73:45–46, 2008.

[Ciocarlie and Allen, 2009] Matei T Ciocarlie and Peter K Allen. Hand posture subspaces for dexterous robotic grasping. *The International Journal of Robotics Research*, 28(7):851–867, 2009.

[Ciocarlie *et al.*, 2007] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dimensionality reduction for hand-independent dexterous robotic grasping. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3270–3275. IEEE, 2007.

[Cockbum *et al.*, 2017] Deen Cockbum, Jean-Philippe Roberge, Alexis Maslyczyk, Vincent Duchaine, et al. Grasp stability assessment through unsupervised feature learning of tactile images. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2238–2244. IEEE, 2017.

[Correll *et al.*, 2016] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 2016.

[Couprie *et al.*, 2013] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013.

[Dang and Allen, 2014] Hao Dang and Peter K Allen. Stable grasping under pose uncertainty using tactile feedback. *Autonomous Robots*, 36(4):309–330, 2014.

[Dang *et al.*, 2011] Hao Dang, Jonathan Weisz, and Peter K Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5917–5922. IEEE, 2011.

[Detry *et al.*, 2013] Renaud Detry, Carl Henrik Ek, Marianna Madry, and Danica Kragic. Learning a dictionary of prototypical grasp-predicting parts from grasping experience. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 601–608. IEEE, 2013.

[Diankov and Kuffner, 2008] Rosen Diankov and James Kuffner. Openrave: A planning architecture for autonomous robotics. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 79, 2008.

[Dragiev *et al.*, 2011] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2845–2850. IEEE, 2011.

[Ferrari and Canny, 1992] Carlo Ferrari and John Canny. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2290–2295. IEEE, 1992.

[Firman *et al.*, 2016] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Structured prediction of unobserved voxels from a single depth image. In *CVPR*, pages 5431–5440, 2016.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th AISTATS*, pages 249–256, 2010.

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

[Goldfeder *et al.*, 2009a] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The columbia grasp database. In *ICRA*, pages 1710–1716. IEEE, 2009.

[Goldfeder *et al.*, 2009b] Corey Goldfeder, Matei Ciocarlie, Jaime Peretzman, Hao Dang, and Peter K Allen. Data-driven grasping with partial sensor data. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1278–1283. IEEE, 2009.

[Goodfellow *et al.*, 2013] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. in neural information processing systems*, pages 2672–2680, 2014.

[Hamza and Krim, 2003] A Ben Hamza and Hamid Krim. Geodesic object representation and recognition. In *International conference on discrete geometry for computer imagery*, pages 378–387. Springer, 2003.

[Handa *et al.*, 2015] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. Scenenet: Understanding real world indoor scenes with synthetic data. *arXiv preprint arXiv:1511.07041*, 2015.

[Hang *et al.*, 2014] Kaiyu Hang, Johannes A Stork, and Danica Kragic. Hierarchical fingertip space for multi-fingered precision grasping. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 1641–1648. IEEE, 2014.

[Hauser, 2016] Kris Hauser. Robust contact generation for robot simulation with unstructured meshes. In *Robotics Research*, pages 357–373. Springer, 2016.

[Haustein *et al.*, 2017] Joshua A Haustein, Kaiyu Hang, and Danica Kragic. Integrating motion and hierarchical fingertip grasp planning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3439–3446. IEEE, 2017.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[Hermann *et al.*, 2016] Andreas Hermann, Felix Mauch, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Eye in hand: Towards gpu accelerated online grasp planning based on pointclouds from in-hand sensor. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 1003–1009. IEEE, 2016.

[Hernandez *et al.*, 2016] Carlos Hernandez, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, et al. Team delft's robot winner of the amazon picking challenge 2016. *arXiv preprint arXiv:1610.05514*, 2016.

[Hinterstoisser *et al.*, 2011] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV), 2011*, pages 858–865. IEEE, 2011.

[Ilonen *et al.*, 2013] Jarmo Ilonen, Jeannette Bohg, and Ville Kyrki. Fusing visual and tactile sensing for 3-d object reconstruction while grasping. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3547–3554. IEEE, 2013.

[Ilonen *et al.*, 2014] Jarmo Ilonen, Jeannette Bohg, and Ville Kyrki. Three-dimensional object reconstruction of symmetric objects by fusing visual and tactile sensing. *The International Journal of Robotics Research*, 33(2):321–341, 2014.

[Ingber, 1989] Lester Ingber. Very fast simulated re-annealing. *Mathematical and computer modelling*, 12(8):967–973, 1989.

[Jamali *et al.*, 2016] Nawid Jamali, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Active perception: Building objects' models using tactile exploration. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 179–185. IEEE, 2016.

[Jarrett *et al.*, 2009] Kevin Jarrett, Koray Kavukcuoglu, M Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

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

[Johns *et al.*, 2016] Edward Johns, Stefan Leutenegger, and Andrew J Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. *arXiv preprint arXiv:1608.02239*, 2016.

[Kappler *et al.*, 2015] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. Leveraging big data for grasp planning. In *ICRA*, pages 4304–4311. IEEE, 2015.

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

[Koenig and Howard, 2004] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IROS*, volume 3, pages 2149–2154. IEEE, 2004.

[Kopicki *et al.*, 2014] Marek Kopicki, Renaud Detry, Florian Schmidt, Christopher Borst, Rustam Stolkin, and John L Wyatt. Learning dexterous grasps that generalise to novel objects by combining hand and contact models. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5358–5365. IEEE, 2014.

[Krainin *et al.*, 2011a] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5031–5037. IEEE, 2011.

[Krainin *et al.*, 2011b] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *The International Journal of Robotics Research*, 30(11):1311–1327, 2011.

[Krug *et al.*, 2016] Robert Krug, Achim J Lilienthal, Danica Kragic, and Yasemin Bekiroglu. Analytic grasp success prediction with tactile feedback. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 165–171. IEEE, 2016.

[LeCun and Bengio, 1995] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In *Handbk. of brain theory & neural networks*, 1995.

[Lempitsky, 2010] Victor Lempitsky. Surface extraction from binary volumes with higher-order smoothness. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1197–1204. IEEE, 2010.

[Lenz *et al.*, 2014] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *IJRR*, 2014.

[Lenz *et al.*, 2015] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[León *et al.*, 2010] Beatriz León, Stefan Ulbrich, Rosen Diankov, Gustavo Puche, Markus Przybylski, Antonio Morales, Tamim Asfour, Sami Moisio, Jeannette Bohg, James Kuffner, et al.

Opengrasp: a toolkit for robot grasping simulation. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 109–120. Springer, 2010.

[Levine *et al.*, 2016] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199*, 2016.

[Li *et al.*, 2015] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015.

[Lin and Sun, 2015] Yun Lin and Yu Sun. Robot grasp planning based on demonstrated grasp strategies. *The International Journal of Robotics Research*, 34(1):26–42, 2015.

[Long *et al.*, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

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

[Madry *et al.*, 2014] Marianna Madry, Liefeng Bo, Danica Kragic, and Dieter Fox. St-hmp: Unsupervised spatio-temporal feature learning for tactile data. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2262–2269. IEEE, 2014.

[Mahler *et al.*, 2015] 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. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[Mahler *et al.*, 2016] 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 *ICRA, 2016*, pages 1957–1964. IEEE, 2016.

[McCormac *et al.*, 2016] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. *arXiv preprint arXiv:1609.05130*, 2016.

[Miller and Allen, 2004] Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004.

[Miller and Leibowitz, 1999] Peter K Allen Andrew T Miller and Paul Y Oh Brian S Leibowitz. Integration of vision, force and tactile sensing for grasping. 1999.

[Miller *et al.*, 2003] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.

[Min, 2004] Patrick Min. Binvox, a 3d mesh voxelizer, 2004.

[Mitash *et al.*, 2017] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. *arXiv preprint arXiv:1703.03347*, 2017.

[Newcombe *et al.*, 2011] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[Oleynikova *et al.*, 2016] Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart. Signed distance fields: A natural representation for both mapping and planning. In *RSS 2016 Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*. University of Michigan, 2016.

[Papazov and Burschka, 2010] Chavdar Papazov and Darius Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In *Asian Conference on Computer Vision*, pages 135–148. Springer, 2010.

[Pinto and Gupta, 2015] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *arXiv preprint arXiv:1509.06825*, 2015.

[Porges *et al.*, 2014] Oliver Porges, Theodoros Stouraitis, Christoph Borst, and Maximo A Roa. Reachability and capability analysis for manipulation tasks. In *ROBOT2013: First Iberian Robotics Conference*, pages 703–718. Springer, 2014.

[Quispe *et al.*, 2015] Ana Huamán Quispe, Benoît Milville, Marco A Gutiérrez, Can Erdogan, Mike Stilman, Henrik Christensen, and Heni Ben Amor. Exploiting symmetries and extrusions for grasping household objects. In *ICRA*, pages 3702–3708, 2015.

[Rennie *et al.*, 2016] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[Roa and Suárez, 2015] Máximo A Roa and Raúl Suárez. Grasp quality measures: review and performance. *Autonomous Robots*, 38(1):65–88, 2015.

[Rocchi *et al.*, 2016] Alessio Rocchi, Barrett Ames, Zhi Li, and Kris Hauser. Stable simulation of underactuated compliant hands. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4938–4944. IEEE, 2016.

[Rock *et al.*, 2015] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In *CVPR*, pages 2484–2493, 2015.

[Rohmer *et al.*, 2013] Eric Rohmer, Surya PN Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1321–1326. IEEE, 2013.

[Rosales *et al.*, 2011] Carlos Rosales, Lluís Ros, Josep M. Porta, and Raúl Suárez. Synthesizing grasp configurations with specified contact regions. *The International Journal of Robotics Research*, 30(4):431–443, 2011.

[Russakovsky *et al.*, 2014] 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, 2014.

[Rusu and Cousins, 2011] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, Shanghai, China, May 9-13 2011.

[Rusu, 2010] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[Schiebener *et al.*, 2016] David Schiebener, Andreas Schmidt, Nikolaus Vahrenkamp, and Tamim Asfour. Heuristic 3d object shape completion based on symmetry and scene context. In *IROS*, pages 74–81. IEEE, 2016.

[Schulman *et al.*, 2014] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[Singh *et al.*, 2014] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 509–516. IEEE, 2014.

[Sommer *et al.*, 2014] Nicolas Sommer, Miao Li, and Aude Billard. Bimanual compliant tactile exploration for grasping unknown objects. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6400–6407. IEEE, 2014.

[Song *et al.*, 2016] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *arXiv preprint arXiv:1611.08974*, 2016.

[Sucan and Chitta, 2013] Ioan A Sucan and Sachin Chitta. Moveit! *http://moveit.ros.org*, 2013.

[Sucan *et al.*, 2012] Ioan A Sucan, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[Thrun and Leonard, 2008] Sebastian Thrun and John J Leonard. Simultaneous localization and mapping. In *Springer handbook of robotics*, pages 871–889. Springer, 2008.

[Tobin *et al.*, 2017] 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. *arXiv preprint arXiv:1703.06907*, 2017.

[Tulsiani *et al.*, 2016] Shubham Tulsiani, Abhishek Kar, Joao Carreira, and Jitendra Malik. Learning category-specific deformable 3d models for object reconstruction. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[Tzeng *et al.*, 2015] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015.

[Vahrenkamp *et al.*, 2009] Nikolaus Vahrenkamp, Dmitry Berenson, Tamim Asfour, James Kuffner, and Rüdiger Dillmann. Humanoid motion planning for dual-arm manipulation and re-grasping tasks. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2464–2470. IEEE, 2009.

[Varley *et al.*, 2015] Jacob Varley, Jonathan Weisz, Jared Weiss, and Peter Allen. Generating multi-fingered robotic grasps via deep learning. In *IROS*, pages 4415–4420, 2015.

[Varley *et al.*, 2016] Jacob Varley, Chad DeChant, Adam Richardson, Avinash Nair, Joaquín Ruales, and Peter Allen. Shape completion enabled robotic grasping. *arXiv preprint arXiv:1609.08546*, 2016.

[Veres *et al.*, 2017] Matthew Veres, Medhat Moussa, and Graham W Taylor. An integrated simulator and dataset that combines grasping and vision for deep learning. *arXiv preprint arXiv:1702.02103*, 2017.

[Wagner, 2008] Rick Wagner. Multi-linear interpolation. *Beach Cities Robotics*, 2008.

[Weisz and Allen, 2012] Jonathan Weisz and Peter K Allen. Pose error robust grasping from contact wrench space metrics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 557–562. IEEE, 2012.

[Williams and Fitzgibbon, 2007] Oliver Williams and Andrew Fitzgibbon. Gaussian process implicit surfaces. *Gaussian Proc. in Practice*, pages 1–4, 2007.

[Woop *et al.*, 2013] Sven Woop, Louis Feng, Ingo Wald, and Carsten Benthin. Embree ray tracing kernels for cpus and the xeon phi architecture. In *ACM SIGGRAPH 2013 Talks*, page 44. ACM, 2013.

[Wu *et al.*, 2014] Zhirong Wu, Shuran Song, Aditya Khosla, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets for 2.5D object recognition and next-best-view prediction. *arXiv preprint arXiv:1406.5670*, 2014.

[Wu *et al.*, 2015] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[Wurman and Romano, 2015] Peter R Wurman and Joseph M Romano. The amazonpicking challenge 2015. *IEEE Robotics and Automation Magazine*, 22(3):10–12, 2015.

[Xiang *et al.*, 2014] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014.

[Yan *et al.*, 2017] Xinchen Yan, Mohi Khansari, Yunfei Bai, Jasmine Hsu, Arkanath Pathak, Arbhinav Gupta, James Davidson, and Honglak Lee. Learning grasping interaction with geometry-aware 3d representations. *arXiv preprint arXiv:1708.07303*, 2017.

[Yi *et al.*, 2016] Zhengkun Yi, Roberto Calandra, Filipe Veiga, Herke van Hoof, Tucker Hermans, Yilei Zhang, and Jan Peters. Active tactile object exploration with gaussian processes. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4925–4930. IEEE, 2016.

[Zeng *et al.*, 2016] Andy Zeng, Kuan-Ting Yu, Shuran Song, Daniel Suo, Ed Walker Jr, Alberto Rodriguez, and Jianxiong Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. *arXiv preprint arXiv:1609.09475*, 2016.