Building a Target Recognition Pipeline for Mechanical Search and Algorithmically Generating Adversarial Grasp Objects with Minimal Random Perturbations

David Wang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-94
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-94.html

May 22, 2019

Building a Target Recognition Pipeline for Mechanical Search and Algorithmically Generating Adversarial Grasp Objects with Minimal Random Perturbations

by

David Wang

A technical report submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Trevor Darrell

Spring 2019

Building a Target Recognition Pipeline for Mechanical Search and Algorithmically Generating Adversarial Grasp Objects with Minimal Random Perturbations

Abstract

Building a Target Recognition Pipeline for Mechanical Search and Algorithmically Generating Adversarial Grasp Objects with Minimal Random Perturbations

by

David Wang

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

Robots will become more prevalent in industry and our everyday lives as we continue on the current trend of automation. In a variety of settings, robots need to robustly interact with their environment to successfully accomplish various tasks. Towards this goal of robust robotic systems, I have worked on two projects during the course of my master's studies.

The first project is Mechanical Search, a class of tasks that requires a robot to locate and extract a target object. We implement a physical system for a particular instance of the Mechanical Search problem that involves retrieving a target object from a heap of objects in a bin by leveraging recent advancements in computer vision and using action primitives such as grasping and pushing. For this project, I worked on improving and evaluating the target recognition segment of the pipeline through experiments with varying Siamese network architectures and dataset augmentation techniques.

The second project is Adversarial Grasp Objects, in which we explore an analog of adversarial images in the domain of robust robot grasping to synthesize objects that are difficult for a robot to grasp, but appear similar in shape to existing objects. By doing so, we can analyze the failure modes of a grasp planner. We explore two algorithms for generating such objects: an analytical algorithm and a deep learning algorithm. For this project, I developed one variant of the analytical algorithm that minimally perturbs vertices on antipodal faces in randomly sampled directions subject to geometric constraints to maintain similarity to the input object. I also conducted experiments and analyses on the objects generated by both types of algorithms and evaluated adversarial grasp objects on a physical system.

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my research advisor, Professor Ken Goldberg, for the opportunity to conduct research in the AUTOLAB over the past two years. Through working on various projects, I have grown a lot, both as a student and as a person. I greatly admire Professor Goldberg's curiosity, creativity, and passion for research, and his insights, feedback, and guidance have helped me tremendously in my projects. I also thank him for pushing me to improve in other areas as well, such as developing better presentation and communication skills.

Much of the work in this technical report is the result of collaboration with other lab members. I would like to thank David Tseng, Jeff Mahler, Ashwin Balakrishna, Yiding Jiang, Pusong Li, Michael Danielczuk, Matthew Matl, Kate Sanders, and Menglong Guo for their helpful suggestions and their work on these projects. I would especially like to thank David Tseng, whom I have worked closely with on several projects over the past several years, for all the conversations and debates and for being a great friend. I would also like to thank other lab members I had the pleasure of working with in the past: Carolyn Matl, Jim Ren, and Michael Laskey. Thank you all for contributing to a friendly, collaborative, and fun research environment.

In addition, I would like to thank Professor Trevor Darrell for taking the time to serve as a second reader for my technical report.

I would also like to thank my friends for supporting me and for the fun memories over the last five years at Berkeley. Finally, I would like to thank my family for their continual love and support throughout my life. Mom, Dad, and Brian, thank you for always being by my side, celebrating accomplishments with me and helping guide me through the difficulties.

Chapter 1

Introduction

As we move forward in time, robots will play an increasingly larger role in a variety of environments, ranging from industrial warehouses to homes. In a variety of applications, robots need the ability to effectively interact with their surroundings to accomplish tasks, such as searching for or grasping objects, even in the presence of uncertainty.

Through the course of my master's studies, I have worked on making progress towards building robust robotic systems through two projects. The first involves the Mechanical Search problem [12], a class of tasks that requires a robot to locate and extract a known target object in a heap of objects. For this project, I worked on developing a robust computer vision system based on a Siamese neural network [32] to reliably identify the desired target object from an image of a heap of objects with possible occlusions.

The second project involves studying how to improve learning-based grasp planners by algorithmically synthesizing objects that are difficult to grasp, but still similar in shape to existing objects. We call such objects "adversarial grasp objects" [64], analogous to adversarial images in the computer vision domain, which are images that visually look similar to original images, but actually fool an image classifier. We study two variants of algorithms for generating adversarial objects: an analytical algorithm where we perform small perturbations to vertex locations of existing objects subject to geometric constraints to enforce shape similarity, and a deep-learning based where we use generative adversarial networks (GANs) [20] and the cross-entropy method (CEM) [14] to synthesize a distribution of adversarial objects using a set of objects (i.e., bottles or prisms) as a geometric prior.

These two projects have been the result of collaboration with many fellow lab members. In this report, I describe some of the greater context of the work in addition to my contributions to these projects.

Chapter 2

Mechanical Search

2.1 Overview

In unstructured settings such as warehouses or homes, robotic manipulation tasks are often complicated by the presence of dense clutter that obscures desired objects. Whether a robot is trying to retrieve a can of soda from a stuffed refrigerator or pick a customer's order from warehouse shelves, the target object is often either not immediately visible or not easily accessible for the robot to grasp. In these situations, the robot must interact with the environment to localize the target object and manipulate the environment to expose and plan grasps. *Mechanical Search* describes a class of tasks where the goal is to locate and extract the target object, and poses challenges in visual reasoning, task, motion, and grasp planning, and action execution.

Significant progress has been made in recent years on sub-problems relevant to Mechanical Search. Deep-learning methods for segmenting and recognizing objects in images have demonstrated excellent performance in challenging domains [24, 46, 59] and new grasp planning methods have leveraged convolutional neural networks (CNNs) to plan and execute high-quality grasps directly from sensor data [34, 21, 40]. By combining object segmentation and recognition methods with action selectors that can effectively choose between different motion primitives in long horizon sequential tasks, multi-step policies can search for a target object and extract it from clutter. In this project, we propose a framework that integrates perception, action selection, and manipulation policies to address a version of the Mechanical Search problem shown in Figure 2.1.

2.2 Individual Contributions

Several of these sections are adapted from our publication for the Mechanical Search project [12], which will appear in at the 2019 IEEE International Conference on Robotics and Automation (ICRA). This work is the result of collaboration with Michael Danielczuk, Andrey

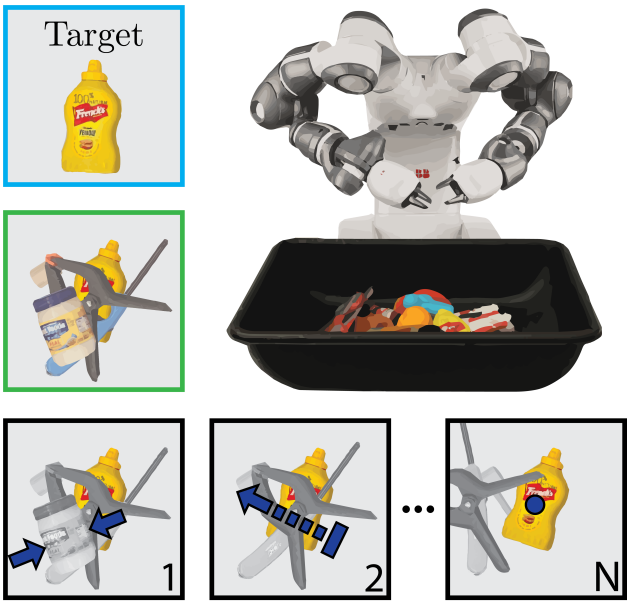

Figure 2.1: Mechanical Search Overview. To locate and extract the target object from the bin, the system selects between 1) grasping objects with a parallel-jaw gripper, 2) pushing objects, or 3) grasping objects with a suction-cup gripper until the target object is extracted, a time limit is exceeded, or no high-confidence push or grasp is available.

Kurenkov, Ashwin Balakrishna, Matthew Matl, Kate Sanders, Dr. Roberto Martín-Martín, Dr. Animesh Garg, Professor Silvio Savarese, and Professor Ken Goldberg.

My work on this project involved developing and evaluating the target recognition segment of the Mechanical Search pipeline through experimentation with varying architectures, dataset augmentation techniques, and training strategies. My contributions to this project are detailed in Sections 2.6 and 2.7. I also assisted in running the physical experiments in Section 2.8.

The core team leading the Mechanical Search project consists of Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, and Matthew Matl. They developed the formulation of the Mechanical Search problem in Section 2.4, much of the design and implementation of the pipeline described in Section 2.5 for a particular instance of Mechanical Search involving retrieval of a target object from a heap of clutter, and the infrastructure of the physical experiments in Section 2.8.

In particular, I would like to highlight several contributions that were integral to my part of the project. Matthew Matl developed the initial version of the target recognition pipeline that provided a starting point for me to improve upon. Ashwin Balakrishna provided lots of valuable suggestions in our weekly meetings for this project. Kate Sanders assisted in labeling data and conducting additional experiments on the target recognition pipeline.

Dr. Roberto Martín-Martín, Dr. Animesh Garg, Professor Silvio Savarese, and Professor Ken Goldberg were the advisors for this project. I would like to thank Professor Goldberg

in particular for providing suggestions and insights on my portion of the project.

2.3 Related Work

Perception for Sequential Interaction

Searching for an object of interest in a static image is a central problem in active vision [62, 54, 45]. There has also been work on optimizing camera positioning for improving visual recognition (i.e., *active perception* [4, 3]) and embodied interactions to explore (i.e., *interactive perception* [6, 22]). Mechanical Search differs from prior works in interactive perception in that it deals with long grasping sequences.

Recent deep learning based methods achieve remarkable success in segmentation of RGB [55, 51] and depth images [10], as well as in localizing visual templates in uncluttered [32, 63] and cluttered scenes [59, 46]. Furthermore, one-shot learning approaches using Siamese Networks for matching a novel visual template in images [32, 63] can translate well to pattern recognition in clutter [59, 46]. We build on Mask R-CNN [24] by training a variant for depth-image based instance segmentation and leverage a Siamese network for target template matching for localization.

Other Related Work

Mechanical search also touches other areas, such as grasping and manipulation in clutter, sequential decision making, and search-based methods. As this report primarily focuses on the perception portion of the Mechanical Search pipeline, we refer the reader to [12] for a more comprehensive survey of existing work in these other areas.

2.4 Problem Formulation

In Mechanical Search, the objective is to retrieve a specific target object (x^*) from a physical environment (E) containing a variety of objects X within task horizon H while minimizing time. The agent is initially provided with a specification of the target object in the form of images, text description, a 3D model, or other representation(s). We can frame the general problem of Mechanical Search as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$.

- **States ($\mathcal{S}$)**. A bounded environment E at time t containing N objects $\mathbf{s}_t = \{\mathcal{O}_{1,t}, ..., \mathcal{O}_{N,t}\}$. Each object state $\mathcal{O}_{i,t}$ includes a ground truth triangular mesh defining the object geometry and pose. Each state also contains the pose and joint states of the robot as well as the poses of the sensor(s).
- **Actions ($\mathcal{A}$)**. A fixed set of parameterized motion primitives.

- **Transitions ($\mathcal{T}$)**. Unknown transition probability distribution $P : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,\ \infty)$.
- **Rewards ($\mathcal{R}$)**. Function given by $R_t = R(\mathbf{s}_t, \mathbf{a}_t) \to \mathbb{R}$ at time t that estimates the change in probability of successfully extracting the target object $x^* \in X$ within task horizon H.
- **Observations ($\mathcal{Y}$)**. Sensor data, such as an RGB-D image, y_t from robot's sensor(s) at time t (see Figure 2.1).

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

- **Observations**. An RGB-D image from an overhead camera.
- **Actions.**
  - **Parallel Jaw Grasping**: A center point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ between the jaws, and an angle in the plane of the table $\varphi \in \mathbb{S}^1$ representing the grasp axis [41].
  - **Suction Grasping**: A target point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and spherical coordinates $(\varphi, \theta) \in \mathbb{S}^2$ representing the axis of approach of the suction cup [42].
  - **Pushing**: A linear motion of the robot end-effector between two points $\mathbf{p}$ and $\mathbf{p}' \in \mathbb{R}^3$.
- **Reward.** Let v_t, derived from y_t, denote the estimated grasp reliability on the target object. An intuitive reward function would be the increase in estimated grasp reliability on the target object:

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

  The policies used in this project do not directly optimize this reward function because it is difficult to compute; instead, they continue to remove and push objects via heuristic methods until the target object is extracted.

2.5 Perception and Decision System

We implement the system shown in Figure 2.2 for physical experiments for the specific instance of Mechanical Search described in Section 2.4 for retrieving a target object from a heap of clutter. The system consists of a perception system and a decision system.

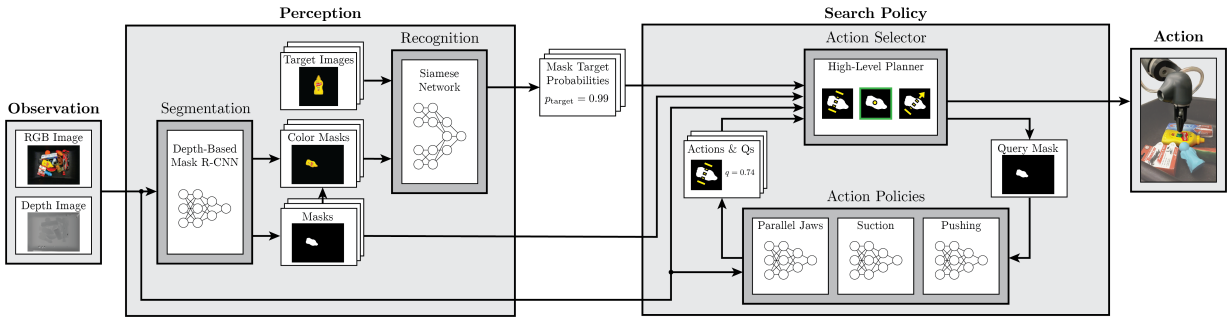

Figure 2.2: Mechanical Search System Architecture. At each timestep, the perception system attempts to segment and locate the target object, while the decision system decides which object to manipulate and which action primitive to use. This process continues until the target object is retrieved.

Perception

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

Object Instance Segmentation We first compute a mask for each object instance. Each mask is a binary image with the same dimensions as the input RGB-D image. These masks are computed with SD Mask R-CNN, a variant of Mask R-CNN trained exclusively on synthetic depth images [13]. It converts a depth image into a list of unclassified binary object masks, and generalizes well to arbitrary objects without retraining. Recent results suggest that depth cues alone may be sufficient for high-performance segmentation, and this network's generalization capabilities are beneficial in a scenario where only the target object is known and many unknown objects may be present.

Target Recognition Next, the set of masks is combined with the RGB image to create color masks of each object. Each of the m color masks is cropped, scaled, rotated, and compared to each of the k images in the target object image set using a Siamese network [32]. For each pair of inputs, the Siamese network outputs a recognition confidence value between 0 and 1, with a mask's recognition confidence score set to the maximum recognition confidence value over the k target object images. If the mask with the highest score has a score above recognition confidence threshold t_r, the mask is labeled as the target object. Otherwise, we report that no masks match the target object. The following sections of this technical report further detail this part of the pipeline.

Search Policy

Given the RGB-D image and the output of the perception pipeline, the system executes the next action in the search procedure by selecting the object to act on and the action to perform on it.

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

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

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

2.6 Target Recognition: Siamese Networks

In this section, we describe the approach we used to develop target recognition segment of the Mechanical Search pipeline. Figure 2.3 shows an illustration of the task: after obtaining color masks of the individual objects in the bin using SD Mask-RCNN [13], the goal is then to find the mask that best matches the target object. We utilize a Siamese network [32] that takes in two RGB images and outputs the probability that the two masks are of the same object.

Dataset

To create a dataset to train the Siamese network, we first take 5 RGB images of common household objects, such as toys and tools, in varying stable poses. For the target recognition task, we need to be able to match complete images of the target images to extracted masks of objects in the bin, which may be occluded. Thus, some positive pairs of images consist of an unoccluded image of an object and the same view of the object with a simulated occlusion: we randomly draw a line through the object and only keep the pixels on one side of the line, such that at least 30% of the pixels of the original object remain (so that the remaining object is still recognizable). Some negative pairs of images consist of an unoccluded image

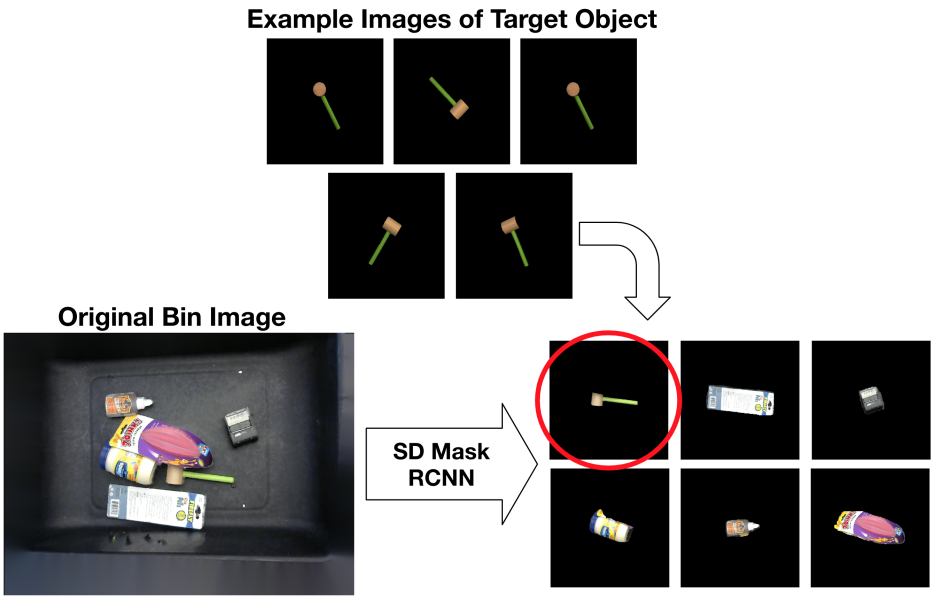

Figure 2.3: Target Recognition Overview. From the original bin image, we use SD Mask-RCNN [13] to obtain color masks of individual segmented objects in the bin. Given several images of varying viewpoints and stable poses of the target object, the goal is then to find the color mask that corresponds to the target object if it is currently visible in the bin.

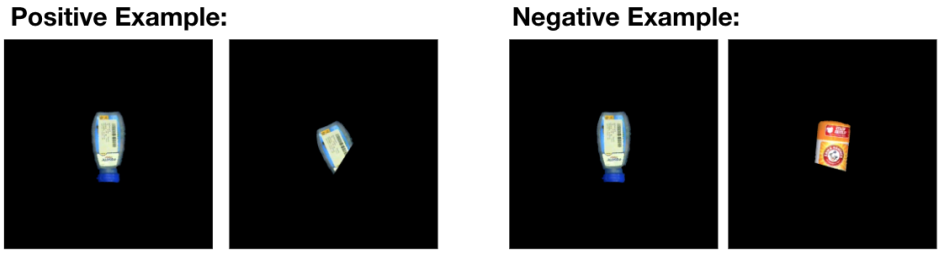

Figure 2.4: Siamese Network Dataset Examples. In the target recognition task, we are trying to match unoccluded images of a target object to a (possibly occluded) image of the target object in the bin. Thus, some examples in the dataset include rotations of the object and simulated occlusions to attempt to train a more robust neural network for object matching.

of an object and a occluded image of an entirely different object, using the same occlusion strategy as before. Finally, we also include positive and negative example pairs without any occlusion, as it is also possible for the target object to be fully visible in the bin. Figure 2.4 show both positive and negative dataset examples.

Siamese Network Architectures

We utilize a Siamese network for the target recognition task to output the probability that two given images are of the same object. The two images are passed through identical layers in the neural network, and the resulting activations are then combined in some manner to form the final prediction. As a starting point, we use ResNet [23] as a feature extractor for the images before further processing the features.

We consider several Siamese network architectures:

- **Baseline:** For a baseline architecture, after obtaining the ResNet featurizations of both images, we take the absolute value of the difference between corresponding elements in the feature vectors (element-wise L1 distance). Then, we have a fully connected layer with one output neuron for the final predicted probability. Figure 2.5 shows an illustration of the network.

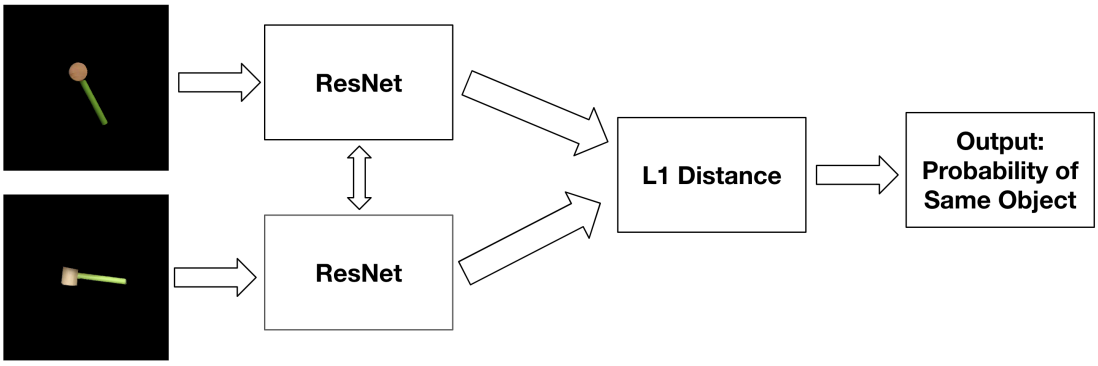

Figure 2.5: Siamese Network Baseline Architecture. We take the element-wise L1 distance between ResNet featurization vectors of the two images and learn a single fully connected layer to the output probability that the two images are of the same object. The vertical double arrows represent shared weights in the neural network.

- **Learned Metric:** Similar to the baseline architecture, we first obtain the ResNet featurizations of both images. Then, the featurizations are concatenated and followed by two fully connected layers, one with 1024 neurons and one with 1 neuron for the final output. This method is motivated by Zagoruyko et al. [67] to allow the network to learn its own distance metric between the featurizations through the fully connected layers. Figure 2.6 shows an illustration of the network.

- **Learned Embedding:** Similar to the previous two architectures, we first obtain the ResNet featurizations of both images. Then, the featurizations are passed through a custom embedding network consisting of two fully connected layers with 128 and 64 neurons. This network is trained separately using the triplet loss with semi-hard negative mining [58], and Figure 2.7 shows an example of a triplet used to train the

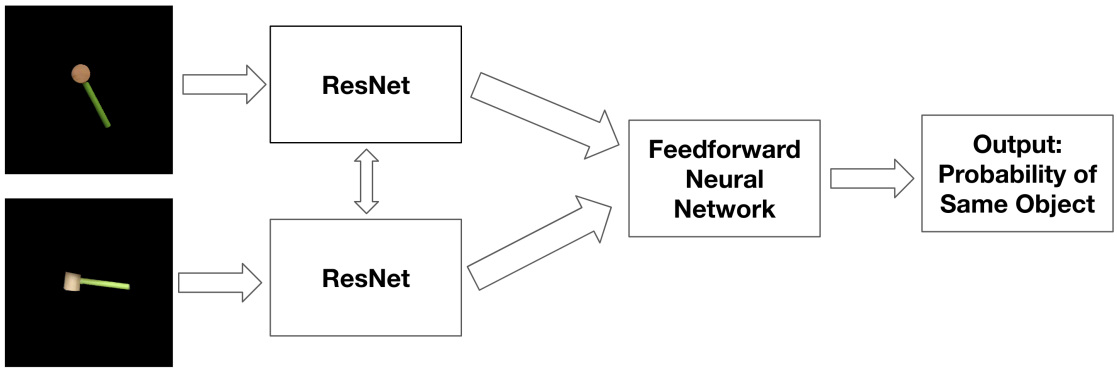

Figure 2.6: Siamese Network Learned Metric Architecture. We use several fully connected layers after concatenating the ResNet feaurizations of the two input image, giving the neural network sufficient capacity to learn its own distance metric between featurizations to come up with an output probability. The vertical double arrows represent shared weights in the neural network.

embedding. Afterwards, the featurizations are concatenated and the rest of the architecture is the same as the Learned Metric architecture. Figure 2.8 shows an illustration of the network.

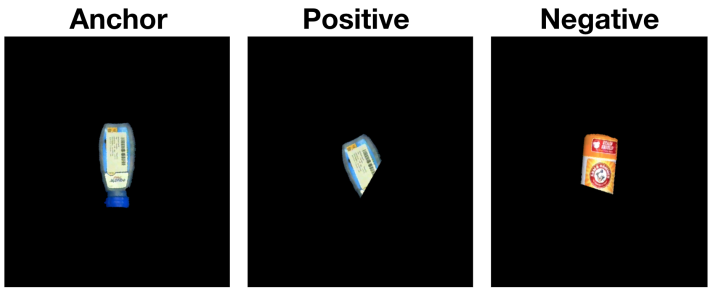

Figure 2.7: Triplet Example for Training Embedding Network. A triplet consists of an anchor, positive, and negative images, where the anchor and positive are of the same object, and the negative is a different object. The goal is to learn an embedding such that the Euclidean distance between the embeddings of the anchor and the positive is closer than those of the anchor and the negative.

2.7 Target Recognition Experiments

We trained the Siamese network architectures in Section 2.6 using the following experimental setup. Each Siamese network architecture involves first passing each input 512×512 RGB

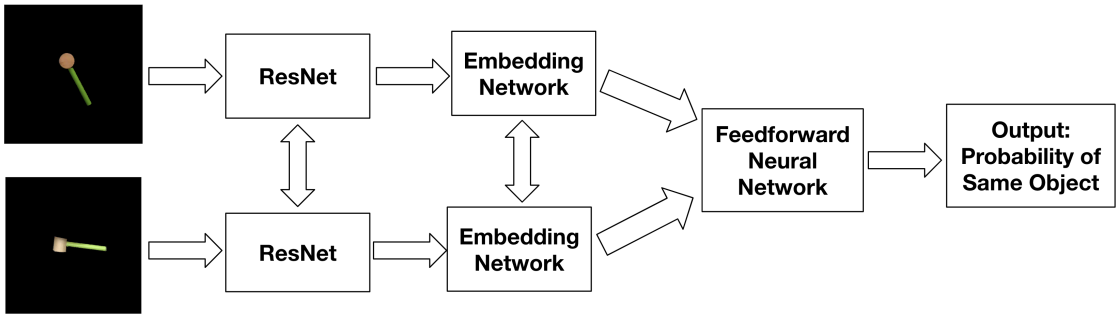

Figure 2.8: Siamese Network Learned Embedding Architecture. We use a custom embedding network that converts the ResNet featurizations of the two input images to custom featurizations. Then, we use fully connected layers after concatenating the featurizations of the two input image to allow the network to learn a metric on the custom embeddings. The vertical double arrows represent shared weights in the neural network.

image through a ResNet-50 architecture pretrained on ImageNet. During training of the Siamese network, these weights remained fixed. In all subsequent layers besides the output layer, the ReLU activation function is used. In the output layer, the sigmoid activation function is used to obtain an output between zero and one. The training dataset for the Siamese network consists of 5 views of each of the objects used in physical experiments. For each view, we generated a total of 10 additional images: 5 randomly rotated versions of the original image as well as 5 rotated versions that are partially occluded using the linear slicing strategy. For training, we sampled 10,000 positive and 10,000 negative image pairs. Each network is then trained with a contrastive loss function for 10 epochs using a batch size of 64 and the Adam optimizer with a learning rate of 0.0001. We use several methods to measure the performance of the resulting networks.

Held-Out Validation

When generating the training dataset for the Siamese network, we hold out 20% of the objects. From these objects, we can generate 1000 positive and 1000 negative example pairs of images using the same process used to construct the training dataset. We evaluated the classification accuracy using on the validation set using a probability threshold of 0.5 as well as the AUC-ROC to show the accuracy over many threshold values. The results are shown in Table 2.1. We observe the Learned Metric architecture performs the best on both metrics. One possible explanation is that the embedding network for the Learned Embedding architecture overfits to the objects in the training dataset and does not generalize well to the held-out objects in the validation dataset.

Architecture	Accuracy	AUC-ROC
Baseline	93.20%	0.9739
Learned Metric	**96.37%**	**0.9940**
Learned Embedding	94.15%	0.9844

Table 2.1: Siamese Network Results on Held-Out Objects. The Learned Metric architecture performs best on objects not seen in the training dataset.

Evaluation on WISDOM

As a validation set comprised of held-out objects but constructed in the same way as the training set may not be indicative of performance on real heaps of objects, we also evaluated the trained Siamese network architectures on a separate dataset. We used WISDOM (Warehouse Instance Segmentation Dataset for Object Manipulation), the dataset used to train SD Mask-RCNN in [13]. WISDOM contains 400 images of heaps of objects and ground truth segmentations for the individual objects in each heap. However, as the SD Mask-RCNN is used primarily for segmentation and not for classification of the individual objects, there are no ground truth labels of the identity of the objects. We labeled all of the objects in the WISDOM dataset for evaluation of the Siamese network architectures. Then, for each of 100 heaps in WISDOM, we treat each visible object as the target object and attempt to find the best corresponding mask from the data. In particular, given k images of the target object and m masks of individual objects in the heap, we run all km combinations through the Siamese network and see if the pair resulting in the highest probability of being the same object is correct. The results of repeating this process 481 times (481 total objects in the 100 heaps we evaluated) are shown in Figure 2.2. Again, we observe that Learned Metric architecture performs best, and the discrepancy in performance between the different architectures is much higher.

Architecture	Number of Successful Matches	Accuracy
Baseline	298	61.95%
Learned Metric	**437**	**90.85%**
Learned Embedding	348	72.34%

Table 2.2: Siamese Network Architecture Results on WISDOM. On WISDOM [13], a dataset to better simulate the target recognition task in Mechanical Search, the Learned Metric architecture performs significantly better than the others.

Other Experiments

We attempted several other strategies in an effort to increase the performance of the Siamese network on the WISDOM dataset. First, we examined several data augmentation techniques

to increase the number of images in the training set:

- **Histogram Equalization:** Scales the range of pixel values within each color channel to occupy the whole 0 to 255 range to increase contrast.
- **Gaussian Noise:** Adds i.i.d. zero-mean Gaussian noise with standard deviation of 15 to each individual pixel, clipping values to be between 0 and 255.
- **Salt and Pepper Noise:** Randomly selects 0.4% of the pixels within the image and sets these pixels to either white or black.
- **Lookup-Table Transformations:** Deterministic mappings of pixel values that can alter lighting effects.

Examples of these data augmentation techniques are shown in Figure 2.9.

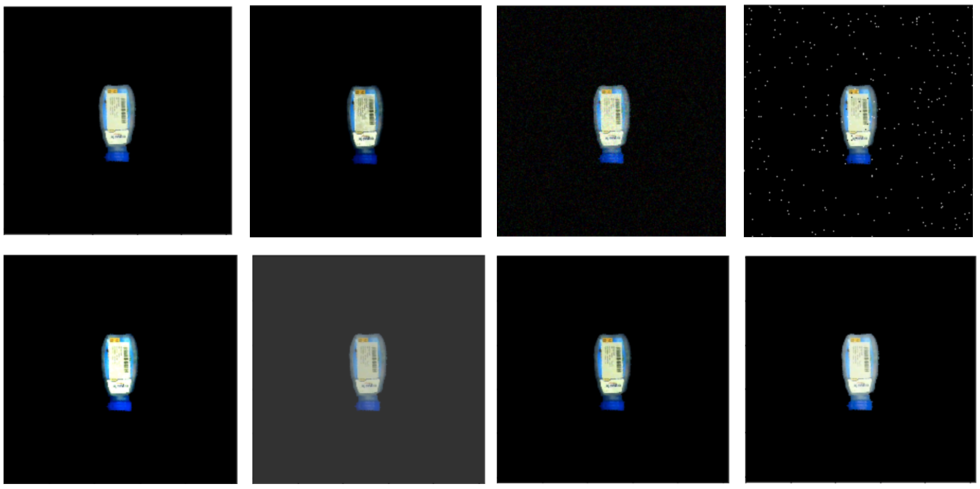

Figure 2.9: Data Augmentation Examples. Top Row: original image and images resulting from applying histogram equalization, Gaussian noise, and salt and pepper noise. Bottom Row: four different lookup-table transformations that simulate variations in lighting and color.

Another strategy we used was to simulate object occlusions in a more realistic manner compared to the linear slicing strategy. To do so, we took binary masks of objects from a synethetic dataset, randomly scaled and rotated the masks, and overlay them on the images in the dataset. We only used occlusions that covered at least 20% and at most 80% of the original pixels of the object. An examples is shown in Figure 2.10.

We evaluated these additional techniques on the Learned Metric Siamese architecture, which performed best on WISDOM. The results are shown in Table 2.3. We observe that the data augmentation actually hurts performance, possibly due to the augmentation techniques

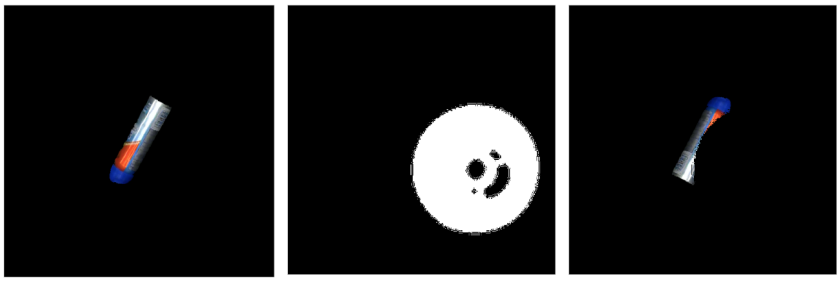

Figure 2.10: Example of More Realistic Object Occlusions. Left: original image. Middle: binary mask of an object in a synthetic dataset. Right: resulting image after overlaying the binary mask of the original image to simulate occlusion.

not matching the distribution of images seen in WISDOM. However, the alternative strategy for simulating occlusions appears to be helpful, and this was used in the final implementation of the Mechanical Search pipeline.

Strategy	Number of Successful Matches	Accuracy
Original	437	90.85%
Data Augmentation	429	89.19%
Realistic Object Occlusions	**439**	**91.27%**

Table 2.3: Learned Metric Siamese Network Architecture Results with Additional Strategies. The data augmentation strategies actually hurt performance, but using the more realistic object occlusions slightly increases performance.

2.8 Physical Experiments

We evaluate the Mechanical Search pipeline by exploring several different action select policies, which determine which type of action (parallel jaw grasp, suction grasp, or push) to take and which object in the bin to interact with.

Action Selection Policies

All action selection methods use input from the perception system to generate a specific object priority list. Each action selection method generates a priority list in a different way but all have the same action execution criteria.

Each action selection method iterates through its priority list, queries the grasping action policies for each object mask, and executes the returned action with the highest quality metric among the two grasping policies if it satisfies the action execution criteria. If the

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

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

1. **Random Search:** Prioritizes objects randomly, with no preference for the target object mask (o_T).
2. **Preempted Random Search (with and without pushing):** Always prioritizes o_T and prioritizes other objects randomly.
3. **Largest-First Search (with and without pushing):** Always prioritizes o_T and ranks the other objects by their visible area. If the target object isn't visible, this strategy will increase the likelihood of removing objects that may be occluding the target object.

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

Physical Experiments Setup

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

Results

Figure 2.11 shows results on the physical system. A total of 300 physical experiments were conducted over all policies. All policies retrieved the target object within the given number of timesteps at least 90% of the time, and success rates were not statistically different between policies. The largest-first policies successfully extract the target object within 5 or fewer actions on on 50% of the heaps, while the preempted random and random policies only do so for 40% and 10% of heaps respectively.

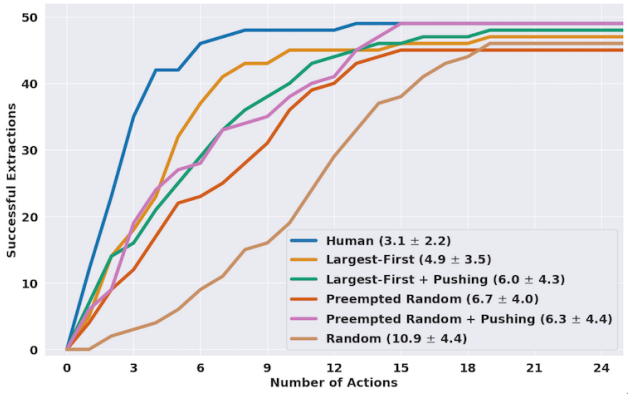

Figure 2.11: Mechanical Search Physical Experiment Results. Performance of policies on real heaps of 15 objects. The largest-first search policies are the most efficient, and are able to extract the target object in the least number of actions. All policies have similar reliability, although pushing shows potential to avoid more failures in simulation. The human was allowed to look at the RGBD image inputs and choose an object to push or grasp. Means and standard deviations for successful extractions are shown in parentheses for each policy.

93% of failure cases on the physical heaps arise from the policy being unable to plan an action. Failure to plan actions is almost always due to the target object lying flat on the bottom of the bin (e.g., the dice, sharpie pens, or another blister-pack object), making it difficult to obtain accurate segmentation. Another common reason for failure to plan actions is when no mask is identified as the target object, which often occurs for 3D printed objects.

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

2.9 Discussion

In this chapter, we presented the Mechanical Search problem and discussed the implementation of a specific instance, in which we attempt to retrieve a target object from a heap of clutter using grasping and pushing action primitives. We evaluate several types of action selection policies on a physical robotic system.

Throughout the course of working on the target recognition portion of the Mechanical Search pipeline, I learned several lessons. First, it is important to develop software infrastructure for easily iterating on different dataset generation techniques and network architectures. This makes conducting experiments and presenting results much more efficient. In addition, finding a way to realistically simulate the physical target recognition task by labeling and using the WISDOM dataset [13] was helpful. By doing so, we were able to see significant performance differences among the different attempted Siamese network architectures, which all performed relatively well on held-out data from the initial training dataset.

Chapter 3

Adversarial Grasp Objects

3.1 Overview

In the computer vision domain, adversarial images [60, 49, 33, 2] are images with minimal added perturbation in pixel-space that drastically alter the prediction made by a classifier. This project defines "adversarial grasp objects," an analog of adversarial images in the domain of robust robot grasping. Similar to adversarial images, adversarial grasp objects reduce graspability while retaining geometric similarity to input objects.

Robust robot grasping of a large variety of objects can benefit a diverse range of applications, such as the automation of industrial warehousing and home decluttering. Recent research suggests that robot policies based on deep learning can grasp a variety of previously unseen objects [35, 37, 52, 42], but can be prone to failures on objects that may not be encountered during training [43].

Adversarial image generation techniques involve performing constrained gradient-based optimization algorithms on the image classification loss [60]. However, a central challenge in applying these algorithms to deep grasping policies is that grasping performance is not a differentiable function of the network output. Instead, the grasp planned by a policy is the result of scoring, ranking, and pruning a set of grasp candidates for each object.

We present two algorithms for synthesizing adversarial objects: an algorithm that modifies objects by perturbing vertices on antipodal faces subject to geometric constraints to maintain similarity to the input object, and an algorithm for synthesizing adversarial 3D object models using 3D Generative Adversarial Networks (GANs) [20] and the Cross Entropy Method (CEM) for derivative-free optimization. Examples of objects generated by both algorithms are shown in Figure 3.1. We conduct experiments in simulation studying adversarial grasp objects of several categories (bottles, intersected cylinders, and intersected prisms) generated by the two algorithms for the Dexterity Network (Dex-Net) 1.0 robust grasp planner, which plans parallel-jaw grasps based on a robust quasi-static point contact model [39]. We also conduct some physical experiments to evaluate adversarial grasp objects using an ABB YuMi arm.

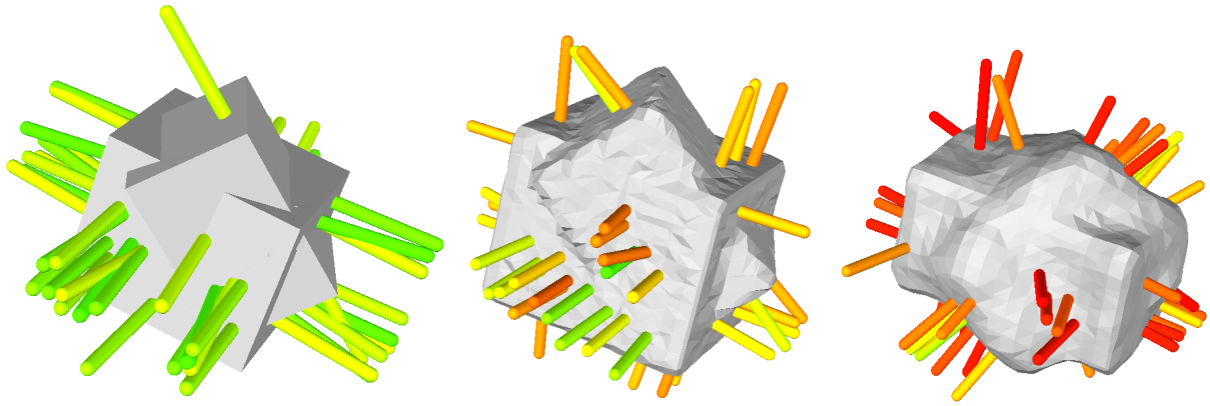

Figure 3.1: Adversarial Grasp Objects Overview. The most robust 25 of 100 parallel-jaw grasps sampled on each object are displayed as grasp axes colored by relative reliability on a linear gradient from green to red. Left: original object from a synthetic intersected prisms dataset. Middle: adversarial object generated by an analytical algorithm. Right: adversarial object generated by a deep learning algorithm.

3.2 Individual Contributions

Parts of this chapter are adapted from our paper for the Adversarial Grasp Objects project [64], which will appear at the 2019 IEEE International Conference on Automation Science and Engineering (CASE), and this work is the result of collaboration with David Tseng, Yiding Jiang, Pusong Li, Menglong Guo, Michael Danielczuk, Dr. Jeffrey Mahler, and Professor Ken Goldberg.

My work on this project involved developing and implementing the analytical algorithm described in Section 3.5, conducting experiments and analyses on the objects generated by both the analytical and deep-learning based (described in Section 3.6) methods, and running physical experiments (described in Section 3.8).

Many others also made significant contributions to this work. David Tseng worked on developing alternative versions of the analytical algorithm described in Section 3.5 that considers convexity of the resulting shapes and rotational perturbations, ran experiments to characterize the analytical algorithms on simple shapes, ran many experiments to train the GAN in Section 3.6 on a variety of datasets, and helped me run physical experiments.

Yiding Jiang and Pusong Li initially started this project before David Tseng and I joined. They developed the initial versions of the problem formulation and the CEM + GAN algorithm in Section 3.6 through lots of experimentation with different datasets and architectures. We also thank them for their continued support and advice on the project even after already graduating from Berkeley.

Michael Danielczuk and Menglong Guo assisted us tremendously on the physical experiments. Michael Danielczuk helped develop the software used to conduct physical experiments

and also spent lots of time helping debug various hardware issues on the real robot. Menglong Guo designed several versions of a new gripper for us to help simulate point contacts on a physical system and also provided lots of 3D printing of objects that we used in physical trials.

Finally, we were fortunate to have been advised by Dr. Jeffrey Mahler and Professor Ken Goldberg throughout this project. They helped steer us in the right direction when we were lost or stuck and provided countless insights and helpful suggestions.

3.3 Related Work

Adversarial Images

Adversarial images [60, 49, 33, 2] are images with a small added perturbation that can change the output of an image classifier. The problem of finding adversarial images is typically formulated as a constrained optimization problem that can be approximately solved using gradient-based approaches [60]. Yang et al. developed a method to perturb the texture maps of 3D shapes such that their projections onto 2D image space can fool classifiers [66]. We build on this line of research by studying adversarial examples in the context of generating adversarial 3D objects for robotic grasping.

Grasp Planning

Grasp planning considers the problem of finding a gripper configuration that maximizes the probability of grasp success. Approaches generally fall into one of three categories: analytic [53], empirical [5], and hybrid methods.

Analytic approaches typically assume knowledge of the object and gripper state, including geometry, pose, and material properties, and consider the ability to resist external wrenches [53] or constrain the object's motion [56], possibly under perturbations to model robustness to sensor noise. Examples include GraspIt! [19], OpenGRASP [36], and the Dexterity Network (Dex-Net) 1.0 [39]. To satisfy the assumption of known state, analytic methods typically assume a perception system based on registration: matching sensor data to known 3D object models in the database [8, 11, 18, 25, 27, 31].

Empirical approaches use machine learning to develop models that map from robotic sensor readings directly to success labels from humans or physical trials. Research in this area has largely focused on associating human labels with graspable regions in RGB-D images [35, 26, 30] or using self-supervision to collect labels from successes and failures on a physical system [37, 52]. A downside of empirical methods is that data collection may be time-consuming and prone to errors.

Hybrid approaches make use of analytic models to automatically generate large training datasets for machine learning models [29, 50]. Recent results suggest that these methods can be used to rapidly train grasping policies to plan grasps on point clouds that generalize

well to novel objects on a physical robot [42, 43, 7]. In this project, we consider synthesizing adversarial 3D objects for the analytic supervisor used to train these hybrid grasp planning methods.

Generative Adversarial Networks (GANs)

Deep generative models map a simple distribution, such as a multivariate Gaussian distribution, to a much more complex distribution, such as natural images. In this work, we use a GAN as a generative model to learn distributions of 3D objects that are difficult to grasp. During training of a GAN, a discriminator tries to distinguish the generated samples apart from the samples from the real data while a generator tries to generate samples to confuse the discriminator. At the optimum of the optimization, the discriminator is unable to distinguish the generated samples from the real data. However, the optimization is difficult and can have problems such as mode collapse [61]. Training GANs involves searching for a saddle point rather than a local minimum due to the 2-player nature of the architecture, and such an optimization is inherently more difficult.

Applications of deep generative models to 3D data are relatively under-explored. Some notable works in this area include the 3D GAN work by Wu et al. [65], which uses a GAN on the latent code learned by a variational autoencoder to generate 3D reconstruction from an image, and the signed distance-based, higher-detail object generation by Jiang et al. [28], where the low frequency components and high frequency components are generated by two separate networks. We expand upon previous efforts in this direction by incorporating recent advances in GANs for 2D image data to synthesize 3D objects.

3.4 Problem Statement

Adversarial Grasp Objects

Let $\mathcal{X}$ be the set of all 3D objects. Let π be a robot grasping policy mapping a 3D object $\mathbf{x} \in \mathcal{X}$ specified as a 3D triangular mesh to a grasp action $\mathbf{u}$. In this work, we consider a parallel-jaw grasping policy. We assume that the policy can be represented as:

$$\pi(\mathbf{x}) \triangleq \arg\max_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} Q(\mathbf{x}, \mathbf{u}) \tag{3.1}$$

where $\mathcal{U}(\mathbf{x})$ denotes the set of all reachable grasp candidates on $\mathbf{x}$, and Q is a quality function measuring the reliability or probability of success for a candidate grasp $\mathbf{u}$ on object $\mathbf{x}$.

We define the graspability $g(\mathbf{x}, \pi)$ of $\mathbf{x}$ with respect to π as a measure of how well the policy can robustly grasp the object. We measure graspability by the γ-percentile of grasp quality [44]:

$$g(\mathbf{x}, \pi) \triangleq \mathbb{P}_{\gamma}(Q(\mathbf{x}, \mathbf{u})) \tag{3.2}$$

We then consider the problem of generating an adversarial grasp object: a 3D object that systematically reduces graspability under a grasping policy with constrained changes to the input geometry. Let $\sigma(A, B)$ for subsets $A, B \subset \mathcal{X}$ be a binary-valued shape similarity constraint between the two subsets of objects. We study the following optimization problem, which defines an adversarial grasp object $\mathbf{x}^*$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathcal{X}} g(\mathbf{x}, \pi) \text{ subject to } \sigma(\{\mathbf{x}\}, S) = 1, \tag{3.3}$$

where $S \subset \mathcal{X}$ is a subset of objects that the generated object should be similar in shape to.

Robust Grasp Analysis

In this paper, we optimize adversarial examples with respect to the Dexterity Network (Dex-Net) 1.0 grasping policy [39]. In this setting, the action set $\mathcal{U}(\mathbf{x})$ is a set of antipodal points on the object surface that correspond to a reachable grasp, where a pair of opposite contact points v_1, v_2 are antipodal if the line between the v_1, v_2 lie entirely within the friction cones [39]. The quality function Q measures the robust wrench resistance, or the ability of a grasp to resist a target wrench under perturbations to the object pose, gripper pose, friction, and wrench under a soft-finger point contact model [43].

When calculating g, both the reward and policy are based on the Dex-Net 1.0 robust grasp quality metric and the associated maximal quality grasping policy. Within the Dex-Net 1.0 robust quality metric, $Q(\mathbf{x}, \mathbf{u})$ is defined as:

$$Q(\mathbf{x}, \mathbf{u}) \triangleq \mathbb{E}_{\mathbf{u}' \sim p(\cdot|\mathbf{u}), \mathbf{x}' \sim p(\cdot|\mathbf{x})}[R(\mathbf{x}', \mathbf{u}')]$$

where $p(\mathbf{u}'|\mathbf{u})$ and $p(\mathbf{x}'|\mathbf{x})$ denote distributions over possible perturbations conditioned on $\mathbf{x}$ and grasp $\mathbf{u}$, and R represents a measure of grasp quality if the grasp is executed exactly as given; that is, executed with zero uncertainty in object and gripper pose. In this case, we use the epsilon metric by Ferrari and Canny with a soft-finger point contact model [15].

To calculate $g(\mathbf{x}, \pi)$ in practice, both the expected value over the distributions of object and grasp pose $p(\mathbf{x}'|\mathbf{x})$ and $p(\mathbf{u}'|\mathbf{u})$ and the γ-percentile are calculated using sample estimates [17]. To do this, we first uniformly sample a constant number of antipodal grasps across the surface of the object. We then approximate the robustness for each grasp by sampling perturbations in object and gripper pose and taking the average grasp quality over all sampled configurations.

The empirical robust grasp quality is:

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

where $\{\mathbf{u}_i\}_{i=1}^N$, $\{\mathbf{x}_i\}_{i=1}^N$ are i.i.d. samples drawn from $p(\mathbf{u}'|\mathbf{u})$ and $p(\mathbf{x}'|\mathbf{x})$ respectively.

The empirical graspability $\hat{g}(\mathbf{x}, \pi)$ is estimated by taking the discrete γ-percentile of $\hat{Q}(\mathbf{x}, \mathbf{u})$ for all sampled grasps.

3.5 Analytical Algorithm: Minimal Random Vertex Perturbations

We consider an analytical approach for modifying an existing 3D triangular mesh $\mathbf{x} \in \mathcal{X}$ to decrease the graspability of $\mathbf{x}$. Let the mesh $\mathbf{x}$ be specified by a set of vertices $\mathcal{V} = \{v_1, v_2, \dots v_n\} \subset \mathbb{R}^3$, a set of faces $\mathcal{F} = \{f_1, f_2, \dots f_m\}$, where each face f_i is the triangle defined by three distinct elements of $\mathcal{V}$, and a set of face normals $\mathcal{N} = \{n_1, n_2, \dots n_m\}$, where $\mathbf{n_i} \in \mathbb{R}^3$ is the unit normal of face f_i. Finally, let the antipodality angle φ between two faces be defined as $\varphi(f_i, f_j) = \arccos(-\mathbf{n_i}^T \mathbf{n_j})$.

Dex-Net 1.0's graspability metric specifically considers the robustness of a parallel jaw grasp, which requires antipodal point pairs and can be susceptible to small pose variations. Consider a mesh $\mathbf{x} \in \mathcal{X}$. We want to perturb vertices while constraining the movement such that the surface normals of adjacent faces do not deviate by more than some angle α from the original object. We define this to be the shape similarity constraint σ in Equation 3.3, and in this case, $S = \{\mathbf{x}\}$, the original object itself.

Sampling-Based Algorithm

For an object with many vertices and faces, it is computationally expensive to keep track of all antipodal face pairs: those whose antipodality angle is less than some threshold angle φ corresponding to the friction angle. Furthermore, we also are not guaranteed to be able to perturb vertices in such a way that all pairs of antipodal faces are eliminated. Thus, we consider a sampling-based algorithm perturb vertices to increase the antipodality angle as much as possible between two faces while satisfying the shape similarity constraint σ. We also want the perturbations to be minimal to further preserve similarity in shape to the original object.

Pseudocode for the analytical minimal random perturbation algorithm is given in Algorithm 1. Let N be the number of iterations we want to run and ϵ to be the mimimal perturbation amount to apply. Denote the sets of vertices, faces, and face normals of the adversarial version of the object by $\mathcal{V}'$, $\mathcal{F}'$, and $\mathcal{N}'$, respectively. In each iteration, we sample a pair of antipodal faces f_i' and f_j' from $\mathcal{F}'$, where $i, j \in \{1, 2, \dots m\}$. We then randomly sample one of the vertices $v_k' \in \mathcal{V}'$ of f_i' and f_j'. Let $\mathcal{I} \subset \{1, 2, \dots m\}$ denote the set of indices of the faces adjacent to v_k'.

We consider a set of 6 possible perturbation directions $\mathcal{W}$, which consists a set of unit vectors $\{\mathbf{w_1}, -\mathbf{w_1}, \mathbf{w_2}, -\mathbf{w_2}, \mathbf{w_3}, -\mathbf{w_3}\}$, where $\mathbf{w_1}$, $\mathbf{w_2}$, and $\mathbf{w_3}$ are randomly selected and orthogonal, forming a basis for $\mathbb{R}^3$. The intuition is to search along all three directions by adding both a positive and negative perturbation. For each direction $\mathbf{w} \in \mathcal{W}$, we compute the perturbation $\delta_w \in \mathbb{R}^+$ such that the antipodality angle φ between faces f_i' and f_j' is maximized subject to the constraints that $\cos^{-1}(\mathbf{n_i}^T \mathbf{n_i'}) < \alpha$ for all $i \in \mathcal{I}$, where $\mathbf{n_i'} \in \mathbb{R}^3$ denotes the unit surface normal of face f_i' after moving vertex v_k' to $v_k' + \delta_w \mathbf{w}$. This amount is computed via an approximation: we start with a perturbation amount of ϵ and continue

Algorithm 1 Minimal Random Perturbation Algorithm. The algorithm takes in the sets of vertices, faces, and face normals for the original object: $\mathcal{V}$, $\mathcal{F}$, $\mathcal{N}$. It also takes in the number of iterations N, α for the shape similarity constraint, and $\epsilon > 0$ for the minimal perturbation amount.

procedure MIN_RANDOM_PERTURBATION($\mathcal{V}$, $\mathcal{F}$, $\mathcal{N}$, N, α, ϵ)
    $\mathcal{V}' \leftarrow \mathcal{V}$
    $\mathcal{F}' \leftarrow \mathcal{F}$
    $\mathcal{N}' \leftarrow \mathcal{N}$
    diam $\leftarrow$ DIAMETER($\mathcal{V}$)
    for $i \in 1, 2, \ldots N$ **do**
        $f_i', f_j' \leftarrow$ SAMPLEANTIPODALFACEPAIR($\mathcal{F}'$)
        $v_k' \leftarrow$ SAMPLEVERTEXFROMFACES(f_i', f_j')
        $\mathcal{I} \leftarrow$ COMPUTEADJACENTFACEINDICES($\mathcal{F}', v_k'$)
        $\mathcal{W} \leftarrow$ SAMPLEPERTURBATIONDIRECTIONS()
        best_dir $\leftarrow$ None
        min_overall_perturb $\leftarrow \infty$
        curr_angle $\leftarrow$ ANTIPODALITYANGLE(f_i', f_j')
        for $\mathbf{w} \in \mathcal{W}$ **do**
            $d \leftarrow \epsilon$
            best_antipodality_angle_change $\leftarrow 0$
            best_perturbation_amount $\leftarrow 0$
            while $d <$ diam **do**
                $v_k' \leftarrow v_k' + d\mathbf{w}$
                if not NORMALCONSTRAINTSSATISFIED($\mathcal{N}, \mathcal{N}', \alpha$) **then**
                    $v_k' \leftarrow v_k' - d\mathbf{w}$
                    break
                antipodality_angle $\leftarrow$ ANTIPODALITYANGLE(f_i', f_j')
                if antipodality_angle $-$ curr_angle $>$ best_angle_change **then**
                    best_angle_change $\leftarrow$ antipodality_angle $-$ curr_angle
                    best_perturbation_amount $\leftarrow d$
                $d \leftarrow 2d$
            if best_perturbation_amount $<$ min_overall_perturb **then**
                min_overall_perturb $\leftarrow$ best_perturbation_amount
                best_dir $\leftarrow \mathbf{w}$
        $v_k' \leftarrow v_k' +$ min_overall_perturb $\cdot$ best_dir
    return $\mathcal{V}'$

doubling it until either the shape similarity constraint is violated or d is no longer less than the diameter of the original object (the maximum Euclidean distance between any pair of vertices). Since we ensure that the shape similarity constraint is satisfied for the resulting object at the end of each iteration, for $\epsilon > 0$ sufficiently small, applying an ϵ-perturbation in any direction should yield an object that still satisfies the constraint. For computational efficiency, we aggressively double the tested perturbation amount to quickly find out how much perturbation is needed to violate the constraint.

After computing the perturbation necessary to apply along each direction to maximize the antipodality angle change of f_i' and f_j', we move in the direction corresponding to the minimal perturbation. By constraining the perturbations, the algorithm attempts to maintain local similarity of the region of perturbation while decreasing the graspability.

3.6 Deep Learning Algorithm: CEM + GAN

An alternative approach for the problem of generating adversarial grasp objects is to use a data-driven approach to learn a distribution over objects $\mathcal{X}$ and extract adversarial grasp objects by sampling from it. As opposed to the analytical algorithm, which generates an adversarial version of an existing object, the CEM + GAN algorithm takes as input a set $S \subset \mathcal{X}$ of objects and can output of a set of generated objects similar to those in S.

One challenge in performing the optimization in Equation 3.3 is that the graspability function $g(\mathbf{x}, \pi)$ is not differentiable; therefore, we need to perform the derivative-free optimization by querying the function with different inputs and adjust the model parameters based on the responses of the function. Let $p_\theta(\mathbf{x})$ be a probability distribution over $\mathcal{X}$ parameterized by some $\theta \in \Theta$. Then, we can formulate a similar objective to Equation 3.3, but instead optimizing for a distribution of objects that we want to be similar to some prior subset $S \subset \mathcal{X}$:

$$\theta^*(\pi) = \arg\min_{\theta \in \Theta} \mathbb{E}_{\mathbf{x} \sim p_\theta(\cdot)}[g(\mathbf{x}, \pi)] \text{ subject to } \mathrm{D}_{\mathrm{KL}}(P_S || P_\theta) < \epsilon, \tag{3.4}$$

where P_θ is the distribution over $\mathcal{X}$ induced by the generative model with parameter θ, P_S is the uniform distribution over objects in S, and D_{KL} is the Kullback-Leibler divergence between two probability distributions.

We propose a deep learning method using the cross-entropy method (CEM) and generative adversarial networks (GANs) to approach this optimization problem.

Cross-Entropy Method (CEM) for Optimization

The cross-entropy method (CEM) [57] is an adaptive derivative-free optimization algorithm. We are interested in finding the distribution of objects that minimize the real-valued graspability function $g(\mathbf{x}, \pi)$ over $\mathcal{X}$.

As a starting point, the GAN is initialized with a prior distribution of objects $S \subset \mathcal{X}$ so that it generates objects similar in shape. We start by training the GAN on this prior set of

objects. Then, in a resampling step, we use the GAN to generate objects and take a subset of the objects with the lowest graspability to use as training data to retrain the GAN. We continue alternating between training and resampling steps for a number of iterations.

Signed Distance Function (SDF) Generative Adversarial Network (GAN)

Generative adversarial networks (GANs) [20] are a family of implicit generative models that can generate high-quality samples with relatively low inference complexity. We use the Signed Distance Function (also known as signed distance field or SDF) [48] as a representation for generating 3D geometry. The SDF of a closed object $\mathbf{x}$ with a well-defined inside and outside at point v can be given as the Euclidean distance from the closest boundary point to v.

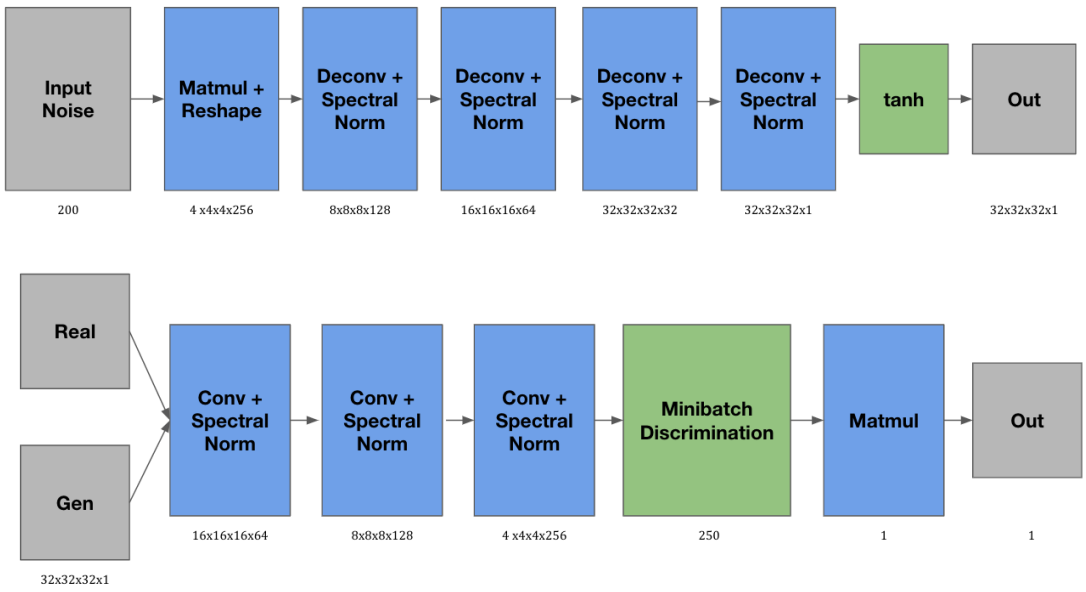

Figure 3.2: GAN Architecture. The generator takes in a random noise vector of length 200 and outputs an SDF representation of a 3D object of resolution $32 \times 32 \times 32$. The discriminator takes in batches of SDF representations of 3D objects from either the training dataset or samples from the the generator network and attempts to distinguish between the two.

We draw on techniques used in Spectral-Normalization GAN (SNGAN) [47], which can generate high-fidelity images, and apply them to SDFs. We denote the standard Gaussian noise vector as $\mathbf{z} \in \mathbb{R}^{200}$ drawn from p_z, the empirical distribution defined by training data as p_{data}, the Generator as $G : \mathbb{R}^{200} \to \left[-1, 1\right]^{32\times32\times32}$, and the Discriminator as $D : \left[-1, 1\right]^{32\times32\times32} \to \mathbb{R}$. For the training objective, we use the hinge version of adversarial loss

[38] as we empirically found that it stabilizes training. The GAN objective is then

$$\mathcal{L}_D^{data} = -\mathbb{E}_{\mathbf{x}\sim p_{data}(\cdot)}[\min(0, -1 + D(\mathbf{x}))]$$

$$\mathcal{L}_D^{gen} = -\mathbb{E}_{\mathbf{z}\sim p_z(\cdot)}[\min(0, -1 - D(G(\mathbf{z})))]$$

$$\mathcal{L}_D = \mathcal{L}_D^{data} + \mathcal{L}_D^{gen}$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim p_z(\cdot)}[D(G(\mathbf{z}))],$$

where $\mathcal{L}_D$ corresponds to the loss function for the discriminator, and $\mathcal{L}_D$ corresponds to the loss function for the generator. More in-depth visualizations of the GAN architecture are shown in Figure 3.2. This GAN loss function implicitly enforces this shape similarity constraint in Equation 3.4 as it has been shown that at the global optimum, the KL-divergence between the generated distribution and the original distribution is zero [38]. The ϵ in the shape similarity constraint accounts for the fact that GANs do not usually reduce the loss to 0 in practice and that we use multiple resampling iterations.

3.7 Simulation Experiments

We run the two algorithms to minimize overall graspability on two synthetic datasets as well as on the ShapeNet [9] bottles category. To allow a fair comparison between our two algorithms, we converted all three datasets to SDFs. For the synthetic datasets, we used the process presented by Bousmalis et al. [7] where they generated objects to grasp in simulation by randomly attaching rectangular prisms of varying sizes together at varying angles. The intersected cylinders dataset consists of one large central cylinder with two smaller cylinders randomly grafted onto it. The intersected prisms dataset is similar to previous dataset but uses prisms instead: it consists of one central rectangular prism with two other rectangular prisms randomly grafted onto it. All three prisms have a wide distribution of sizes. The bottle, cylinder, and prism datasets have averages of 1,391 vertices and 2,783 faces, 1,202 vertices and 2,400 faces, and 2731 vertices and 4739 faces, respectively, and have 479, 1000, and 1000 total objects, respectively. Examples from each of these datasets are shown in Figure 3.4.

In the following experiments, we set the angle of the friction cone to be $\arctan(0.5)$. For the graspability metric $g(\mathbf{x}, \pi)$, we chose $\gamma = 75\%$: often, one of the top 25% of grasps is accessible, so we choose to look at the worst case from this set. Consider a set of generated objects $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\} \subset \mathbf{X}$ from a prior dataset of objects. We define mean normalized graspability as $c \cdot \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x}_i, \pi)$, where c is a normalizing constant. We note that the objects in the figures in the section have been smoothed for visual clarity to demonstrate the behavior of the algorithms, but the metrics represent the results of the objects without smoothing. Meshes in all datasets have large numbers of vertices and faces, and displaying all of them makes it difficult to distinguish differences within and between algorithms.

Analytical Algorithm

We run the analytical algorithm for local perturbations of vertices on antipodal faces on 100 objects from each of the three datasets. We experimented with α values of 10, 15, and 20 degrees for the shape similarity constraint for maximum deviation in surface normals described in Section 3.5. We find that the analytical algorithm decreases the graspability metric for all datasets. With a value of $\alpha = 10$ degrees, the mean normalized graspability is decreased by 32% on the intersected cylinders dataset, 12% on the intersected prisms dataset, and 32% on the ShapeNet bottles dataset. At each level of α, we observe that the objects from the prism dataset have the highest graspability; we conjecture that it is difficult to decrease the antipodality of large, flat prism surfaces with only local perturbations. The full graspability results for each dataset and level of α are shown in Table 3.1, and the distributions of graspabilities are shown as histograms in Figure 3.3. Sample object examples along with their adversarial versions for varying levels of α are shown in Figure 3.4. Increasing α decreases the graspability at the cost of similarity to the original object, corresponding to an increasingly relaxed shape similarity constraint.

Dataset	Original	$\alpha = 10$	$\alpha = 15$	$\alpha = 20$
Intersected Cylinders	1.00	0.68	0.53	0.42
Intersected Prisms	1.00	0.88	0.75	0.65
ShapeNet Bottles	1.00	0.68	0.54	0.42

Table 3.1: Analytical Algorithm Graspability Results. We report the mean normalized graspibility for 100 objects on each of the three datasets for shape similiarity constraints using $\alpha = 10$, $\alpha = 15$, and $\alpha = 20$. The analytical algorithm is able to decrease graspability on objects from all three datasets.

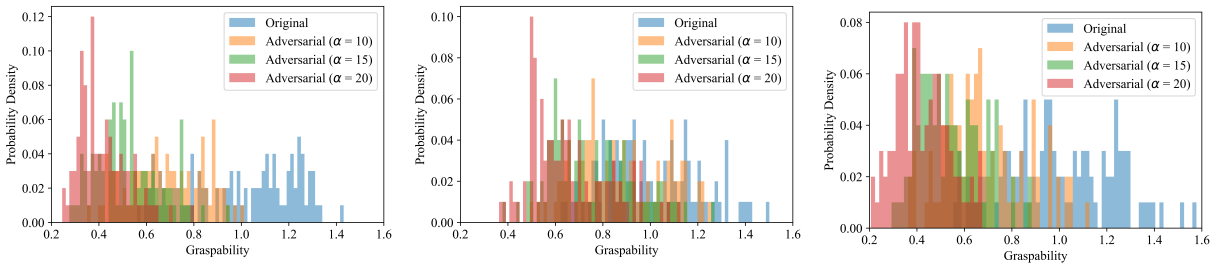

Figure 3.3: Analytical Algorithm Graspability Distributions. We show the distribution of graspabilities for 100 objects from each of the three datasets for the original versions as well as adversarial versions using $\alpha = 10$, $\alpha = 15$, and $\alpha = 20$ for the shape similarity constraint. Left: intersected cylinders dataset. Middle: intersected prisms dataset. Right: ShapeNet bottles dataset.

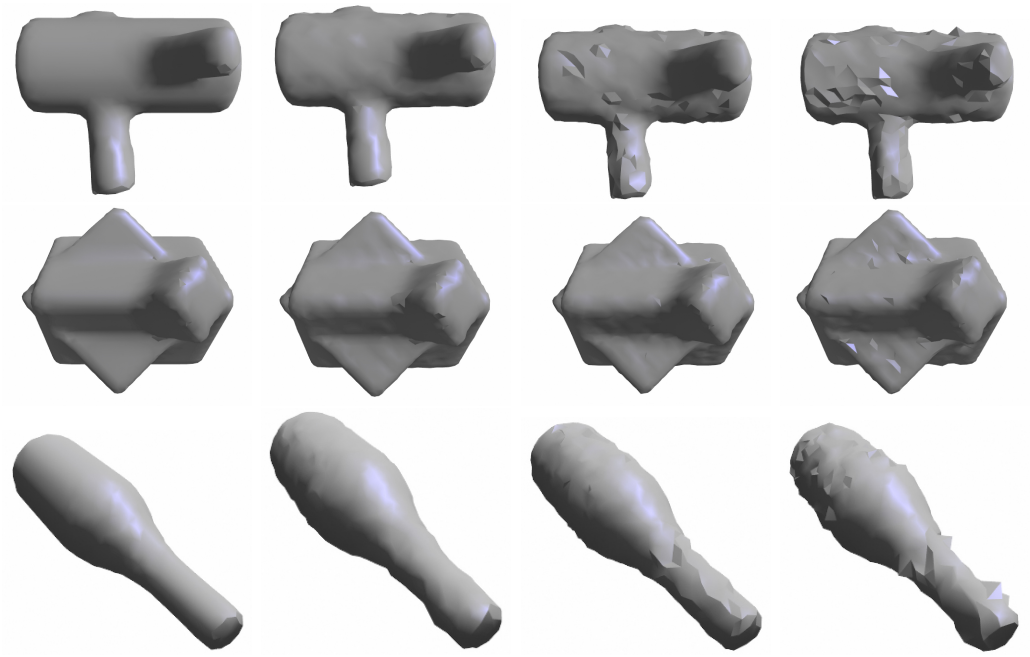

Figure 3.4: Analytical Algorithm Examples. We show the progression of an example from each dataset as we increase the surface normal constraint angle α: each row (from left to right) shows the original object and then the perturbed versions using the surface normal constraint with $\alpha = 10$, $\alpha = 15$, and $\alpha = 20$, respectively. The objects have been smoothed for visualization purposes with OpenGL smooth shading. Top Row: example from intersected cylinders dataset. Middle Row: example from intersected prisms dataset. Bottom Row: example from ShapeNet bottles dataset.

CEM + GAN Algorithm

We train the resampling GAN on the previously described intersected prisms and cylinders datasets, as well as the ShapeNet bottles category. All three datasets are preprocessed into signed distance field format with stride 0.03125 after being scaled such that the entire set has bounding boxes of approximately $1 \times 1 \times 1$.

For all three datasets, we sample 2500 new objects and keep 500, and train the GAN for 16000 iterations between resampling steps. Resampling in all experiments rejects output grids that produce non-watertight meshes, as producing meshes with non-orientable faces, gaps, self-intersection, or disjoint pieces is not desirable when generating a distribution of 3D objects. Such outputs are possible because the GAN does not explicitly enforce such constraints.

Table 3.2 shows the graspability results of 100 objects sammpled from the learned generative models at different points of the algorithm. After 3 resampling iterations on the intersected cylinders dataset, the mean normalized graspability is reduced by 22% relative to objects in the original dataset. Similarly, graspability is reduced by 36% on the intersected

prisms datset after 4 resampling iterations and by 17% on the ShapeNet bottles dataset after 5 resampling iterations. Histograms showing the overall distribution of graspability over resampling episodes for each of the three datasets are shown in Figure 3.5. Finally, examples from the GAN output distributions at the beginning and end of training for the intersected prisms dataset are shown in Figure 3.6.

Dataset	Original	Before Resampling	End of Training
Intersected Cylinders	1.00	0.98	0.78
Intersected Prisms	1.00	0.95	0.64
ShapeNet Bottles	1.00	1.02	0.83

Table 3.2: CEM + GAN Algorithm Graspability Results. We report the mean normalized graspibility for 100 objects generated by the learned model for each of the three datasets at various stages in the algorithm. The "Before Resampling" column shows the graspability after fitting a generative model to the geometric prior, and the "End of Training" column shows the graspability after several alternating training and resampling iterations. The CEM + GAN Algorithm is able to learn a distribution of objects with reduced graspability for all three datasets.

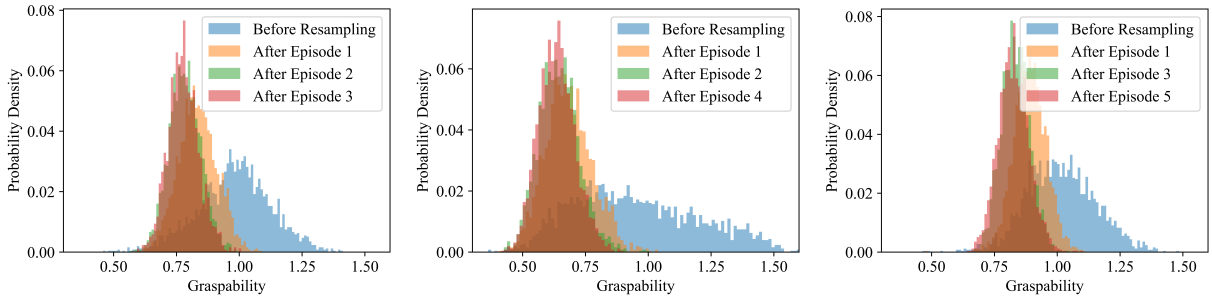

Figure 3.5: CEM + GAN Algorithm Graspability Distributions. We show the distribution of graspabilities for 100 objects from each of the three datasets after varying number of training and resampling iterations. As the algorithm progresses through the episodes, the probability mass shifts towards lower graspability. Left: intersected cylinders dataset. Middle: intersected prisms dataset. Right: ShapeNet bottles dataset.

Shape Similarity

Experiments suggest that both the analytical and the CEM + GAN algorithms decrease the graspability metric, but in different manners. The analytical algorithm maintains local shape similarity through the constraints on surface normal changes, while the GAN introduces geometric changes that decrease graspability while maintaining shape similarity at a more

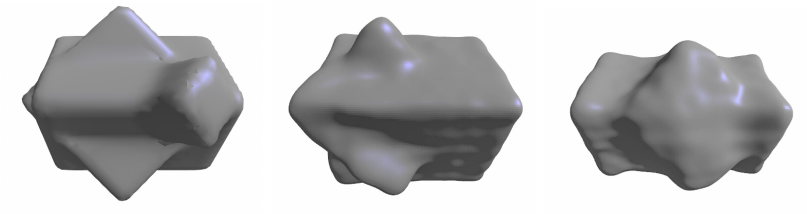

Figure 3.6: CEM + GAN Algorithm Examples. Left: sample object from the original intersected prisms dataset. Middle: sample from the GAN output distribution after fitting the geometric prior. Right: sample from the GAN output distribution after several resampling iterations. The CEM + GAN algorithm tends to reduce graspability of objects through larger structural changes. The objects have been smoothed for visualization purposes with OpenGL smooth shading.

global level (e.g., generates a tapered bottle that resembles a bottle but was not in the original dataset).

To quantify shape similarity, we use 1 iteration of Laplacian smoothing on the generated objects from each of the three datasets by both algorithms to minimize surface roughness and measure the effect of smoothing on object graspability. The objects generated by the analytical algorithm use $\alpha = 10$ degrees for the surface normal deviation constraint. The full results are shown in Table 3.3. Before smoothing, the mean normalized graspability for objects from the analytical algorithm is 10% lower, 30% higher, and 15% lower than objects from the GAN on the intersected cylinders, intersected prisms, and bottles datasets, respectively. After smoothing, the mean normalized graspability of objects generated by the GAN are lower by 10%, 18%, and 8% on the same datasets. The mean graspability of smoothed objects from the analytical algorithm is at least 95.9% of the original datasets in all cases, suggesting that surface roughness accounts for almost all of the decrease in graspability. Although the GAN also introduces surface roughness (smoothing still increases graspability in all cases), it appears to learn more global geometric changes to decrease graspability.

Alternative GAN Architecture

GANs are prone to mode collapse [61], the phenomenon where a GAN can learns to only outputs one distinct object regardless of the input. Furthermore, since resampling decreases diversity of objects in the dataset due to similar generated objects tending to have similar metric scores, complete mode collapse tends to occur after enough resampling episodes.

We experimented with several variations of the GAN architecture and observed that removing spectral normalization can lead to more diverse objects on the intersected cylinders dataset. In this experiment, mode collapse does not occur before the metric quality mean stops improving, reaching a decrease of 83% from the original dataset. However, these generated objects deviate quite significantly from the prior dataset.

Dataset	Graspability Before Smoothing		Graspability After Smoothing	
	Analytical	CEM + GAN	Analytical	CEM + GAN
Intersected Cylinders	$\mathbf{0.677 \pm 0.031}$	0.783 ± 0.011	0.959 ± 0.048	$\mathbf{0.862 \pm 0.031}$
Intersected Prisms	0.876 ± 0.036	$\mathbf{0.577 \pm 0.012}$	0.961 ± 0.047	$\mathbf{0.777 \pm 0.037}$
ShapeNet Bottles	$\mathbf{0.682 \pm 0.034}$	0.827 ± 0.012	0.980 ± 0.038	$\mathbf{0.899 \pm 0.033}$

Table 3.3: Smoothing Experiment Graspability Results. Comparison of the normalized mean graspability (reported with 95% confidence intervals) of objects generated by both the analytical algorithm and the GAN algorithm before and after Laplacian smoothing. After smoothing, the objects generated by the GAN have lower graspability metrics than the corresponding objects generated by the analytical algorithm for all three datasets, which suggests that the GAN generates more global adversarial geometries, whereas the analytical algorithm uses local surface roughness to reduce graspability.

3.8 Physical Experiments

While we showed that both the analytical algorithm and the CEM + GAN algorithm can systematically reduce the computed graspability of different types of objects, we also wanted to develop a physical system to evaluate the graspability of objects using a real robot.

Objects

We consider several types of objects to study in physical experiments.

First, we consider a unit cube, which is highly graspable with its three pairs of parallel faces. To manually develop an adversarial version of the cube with minimal perturbation such that any pair of faces of the resulting object have an antipodality angle of at least θ degrees, we considered raising the midpoint of three adjacent faces of the cube by $\frac{\tan\theta}{2}$ in the direction of the surface normal of the original face. We considered other variations without adding additional vertices, but observed that such strategies tend to add require increased volume change to the cube, increasing the visual distortion. The cubes we studied in physical experiments are shown in Figure 3.7.

We also consider a cuboctahedron, a polyhedron with 6 square faces and 8 triangular faces. As it is more difficult to manually design an adversarial version satisfying the property that all pairs of faces have an antipodality angle of at least θ, we used a modificaiton of the analytical algorithm described in Section 3.5. We apply random perturbations until the property above is satisfied, rejecting perturbations if they introduce concavities in the objects, as these can introduce feasible areas to grasp on a physical system. We find that this algorithm can converge on simpler objects. The resulting cuboctahedrons are shown in Figure 3.8.

Courtesy of HP, we have some 3D prints shown in Figure 3.9 of the adversarial intersected cylinders objects generated using the alternative GAN architecture described at the end of

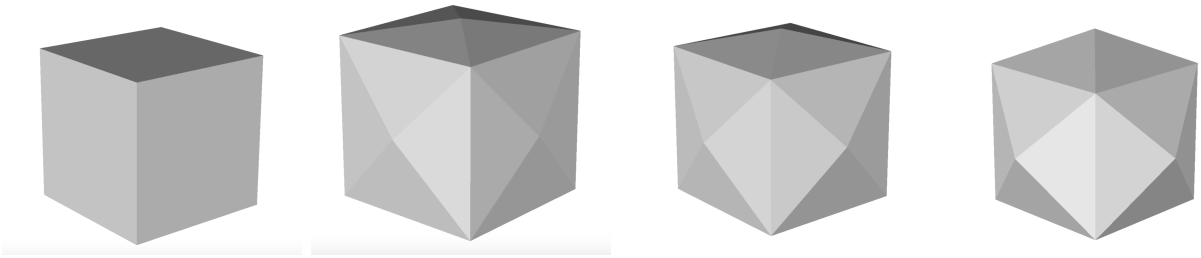

Figure 3.7: Cube Objects for Physical Experiments. Left to Right: original cube and then adversarial versions of 10, 15, and 26 degrees for θ. For the adversarial cubes, we only show the distorted faces; the remaining faces are unchanged.

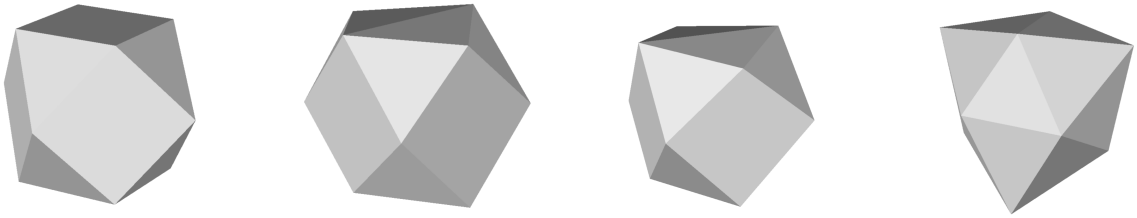

Figure 3.8: Cuboctahedron Objects for Physical Experiments. Left to Right: original cuboctahedron and then adversarial versions of 10, 15, and 26 degrees for θ. These were generated by a version of the analytical algorithm.

Section 3.7. We chose these class of objects to explore in physical trials, since they were the outputs generated by the CEM + GAN algorithm with the lowest graspability in simulation.

Experimental Setup

We used a parallel jaw gripper designed for the ABB YuMi to simulate point contacts: the grippers are 3D printed, and each jaw holds a small metal bearing that makes contact with the target object to grasp. Images of the 3D printed objects and the gripper are shown in Figure 3.10.

In physical trials, for each object, we first compute the stables poses of the object and potential grasps along with their associated confidences using the Dex-Net [43] system in simulation. Then, on the physical system, we sample 5 grasps and execute each one 3 times. To sample the grasps, we first sample a stable pose, using probabilities for each stable pose proportional to the computed feasibility probability of the stable pose. We then take the grasp with the highest confidence of success that has not yet been executed. When executing the grasp, we place the object in a bin, use a depth sensor to obtain a point cloud of the object, and then align it to the known 3D model of the object using the 4-Points Congruent

Figure 3.9: Adversarial Intersected Cylinder Objects for Physical Experiments. These objects were generated by the CEM + GAN method.

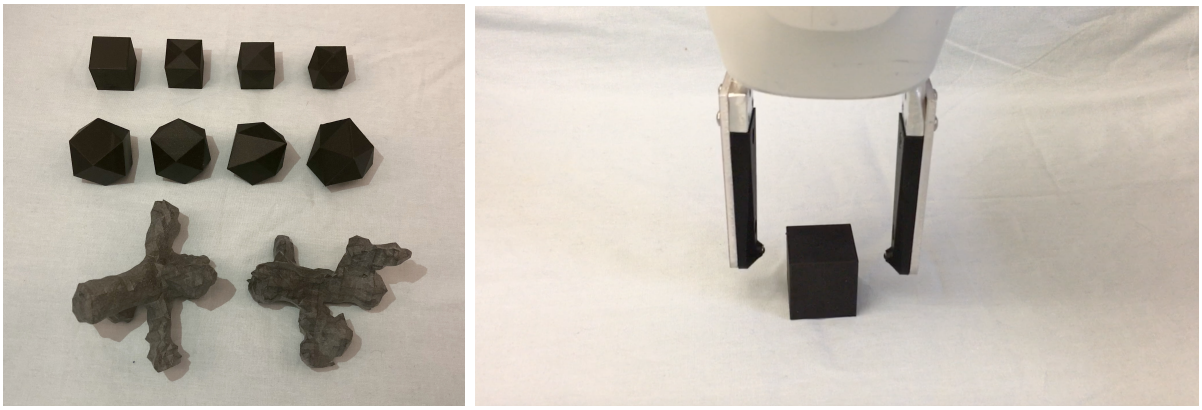

Figure 3.10: Objects and Gripper for Physical Experiments. Left: 3D printed objects used in physical experiments. Right: gripper used to simulate point contacts on a physical system.

Sets (4PCS) algorithm [1]. A grasp is defined as successful if the robot arm is able to lift the object out of the bin and successfully transport it to an adjacent bin.

Results

Using the setup described in the previous subsection, we conducted a total of 15 trials for each of the ten objects, and the success rates are shown in Table 3.4. We find that the adversarial cube of 26 degrees is never successfully grasped, and the success rate increases with lesser perturbations to the original cube, which was successfully grasped in all trials. We observe a very similar trend for the cuboctahedrons. Finally, the adversarial intersected cylinder objects generated by the CEM + GAN method are very difficult to grasp, as only we observed success in only 3 of 30 total trials for the two objects. Common failure cases on the physical system involve errors in the registration of the object as well as one jaw of the

gripper making contact with the object first and moving it such that the other jaw cannot make contact with the gripper.

Object	Grasp Success Rate
Original Cube	15/15
Adversarial Cube (10 Degrees)	12/15
Adversarial Cube (15 Degrees)	2/15
Adversarial Cube (26 Degrees)	0/15
Original Cuboctahedron	15/15
Adversarial Cuboctahedron (10 Degrees)	9/15
Adversarial Cuboctahedron (15 Degrees)	2/15
Adversarial Cuboctahedron (26 Degrees)	0/15
Adversarial Intersected Cylinder 1	2/15
Adversarial Intersected Cylinder 2	1/15

Table 3.4: Physical Experiments on Cubes. We attempted 5 grasps with 3 trials each for each object and report the number of successful grasps on each object.

3.9 Discussion

We introduce adversarial grasp objects: objects that look visually similar to existing objects, but decrease the predicted graspability given by a robot grasping policy. We present two algorithms that generate adversarial grasp objects. The first is an analytical method that performs constrained vertex perturbations to eliminate antipodal pairs of faces in an existing triangular mesh of an object. The second method combines CEM and GANs to learn a distribution of objects parameterized by a SDF GAN that minimizes graspability. We empirically show that the two methods can decrease the grasp quality given by the DexNet 1.0 grasping algorithm.

The analytical algorithm was designed to be specifically designed to be adversarial to the Dex-Net 1.0 grasping policy by attempting to reduce pairs of antipodal faces. In contrast, the CEM + GAN algorithm can apply to other grasping policies through the generality of the resampling step: the algorithm takes the least graspable objects with respect to any arbitrary policy to use as training data for the next iteration of the generative model.

Chapter 4

Conclusion

We conclude this technical report by discussing potential future directions of work for each of the two projects: Mechanical Search and Adversarial Grasp Objects.

4.1 Mechanical Search

For the target recognition pipeline I worked on, there are several potential areas of improvement. We used a pretrained ResNet to extract features from RGB images, but it might be interesting to try featurizations from other layers in the network or completely different pretrained networks altogether. In addition, the Siamese networks we developed take in RBG images as input. There may be potential performance benefits by using depth images as well. As we were obtaining strong performance results empirically with the current version of the target recognition pipeline, we did not actively pursue these directions as much.

Instead, there are currently efforts to explore alternative versions of the entire perception pipeline. For my project, we wanted to optimize the target recognition portion, since we were getting reliable segmentations from SD Mask-RCNN [13]. However, an alternative strategy is to merge the segmentation and target recognition portions to extract the target mask in one shot, which could cut down on the computation time for the perception system. Another potential direction being explored is using dense object descriptors [16] to find correspondences between the ground truth target object and the visible portions of the target object in the bin to locate it.

On a broader level, other directions of work currently being pursued on this project include exploring how to use reinforcement learning to bridge the performance gap between the heuristic action selection policies and the human supervisor. Improving the individual primitive actions may also result in better overall performance to extract the target object from the heap of clutter in fewer total actions. Mechanical Search is a broad class of tasks, and it would be very interesting to study other instances of the problem in more complex settings as well, possibly those that involve navigation of a robot.

4.2 Adversarial Grasp Objects

In this project, the metric we explore in this project models point contacts instead of area contacts, which can be disproportionately affected by surface roughness, as we saw with the analytical algorithm. Replacing the point contact model with one that considers the full contact area could counteract the outsized effect of local surface roughness. In this case, different types of analytical algorithms would need to be explore to design objects that are adversarial to the area contact model, but the CEM + GAN method can still work by finding objects that are most adversarial to the new metric through iterative resampling.

Another extension is to consider different types of grasps. In this work, we focus on parallel jaw grasps, but it would also be interesting to explore suction grasps. Similar to above, we would need newly designed analytical algorithms to generate adversarial objects in this case, but the CEM + GAN method can still apply.

We also demonstrated the grasping performance of a robot on both adversarial and non-adversarial objects. It would be interesting use this physical system to conduct more extensive physical trials to evaluate the graspability of other objects.

Finally, in this project, we developed algorithms that can efficiently generate many adversarial grasp objects. However, the long-term vision of this work is to close the loop and use these synthesized objects to train better grasp planners based on neural network policies. This is similar to the idea of using adversarial images to train more robust image classifiers. This is a difficult direction to pursue, but can be extremely valuable in the ultimate goal improving robot grasping performance.

Bibliography

[1] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. "4-points congruent sets for robust pairwise surface registration". In: *ACM transactions on graphics (TOG)*. Vol. 27. 3. Acm. 2008, p. 85.

[2] Anish Athalye et al. "Synthesizing Robust Adversarial Examples". In: *CoRR* abs/1707.07397 (2017). arXiv: 1707.07397. URL: http://arxiv.org/abs/1707.07397.

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

[4] R. Bajcsy. "Active perception". In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005. ISSN: 0018-9219. DOI: 10.1109/5.5968.

[5] Jeannette Bohg et al. "Data-driven grasp synthesisfffdfffdfffda survey". In: *IEEE Trans. Robotics* 30.2 (2014), pp. 289–309.

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

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

[8] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. "Collaborative grasp planning with multiple object representations". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2011, pp. 2851–2858.

[9] Angel X Chang et al. "Shapenet: An information-rich 3d model repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[10] Xiaozhi Chen et al. "3d object proposals using stereo imagery for accurate object class detection". In: *IEEE transactions on pattern analysis and machine intelligence* 40.5 (2018), pp. 1259–1272.

[11] Matei Ciocarlie et al. "Towards reliable grasping and manipulation in household environments". In: *Experimental Robotics*. Springer. 2014, pp. 241–252.

[12] Michael Danielczuk et al. "Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter". In: *2019 IEEE International Conference on Robotics and Automation (ICRA) [To Appear]*. IEEE. 2019.

[13] Michael Danielczuk et al. "Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic data". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2019.

[14] Pieter-Tjerk De Boer et al. "A tutorial on the cross-entropy method". In: *Annals of operations research* 134.1 (2005), pp. 19–67.

[15] C. Ferrari and J. Canny. "Planning optimal grasps". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 1992, pp. 2290–2295.

[16] Peter R Florence, Lucas Manuelli, and Russ Tedrake. "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation". In: *arXiv preprint arXiv:1806.08756* (2018).

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[18] Corey Goldfeder and Peter K Allen. "Data-driven grasping". In: *Autonomous Robots* 31.1 (2011), pp. 1–20.

[19] Corey Goldfeder et al. "The Columbia grasp database". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2009, pp. 1710–1716.

[20] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: 2014.

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

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

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

[24] K He et al. "Mask r-cnn. arXiv preprint arXiv: 170306870". In: (2017).

[25] Carlos Hernandez et al. "Team Delft's Robot Winner of the Amazon Picking Challenge 2016". In: *arXiv preprint arXiv:1610.05514* (2016).

[26] Alexander Herzog et al. "Learning of grasp selection based on shape-templates". In: *Autonomous Robots* 36.1-2 (2014), pp. 51–65.

[27] Stefan Hinterstoisser et al. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. IEEE. 2011, pp. 858–865.

[28] Chiyu Jiang, Philip Marcus, et al. "Hierarchical Detail Enhancing Mesh-Based Shape Generation with 3D Generative Adversarial Network". In: *arXiv preprint arXiv:1709.07581* (2017).

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

[30] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. "Leveraging big data for grasp planning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4304–4311.

[31] Ben Kehoe et al. "Cloud-based robot grasping with the google object recognition engine". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2013, pp. 4263–4270.

[32] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition". In: *ICML deep learning workshop*. Vol. 2. 2015.

[33] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *CoRR* abs/1607.02533 (2016). arXiv: 1607.02533. URL: http://arxiv.org/abs/1607.02533.

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

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

[36] Beatriz León et al. "Opengrasp: a toolkit for robot grasping simulation". In: *Proc. IEEE Int. Conf. on Simulation, Modeling, and Programming of Autonomous Robots (SIMPAR)*. Springer. 2010, pp. 109–120.

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

[38] Jae Hyun Lim and Jong Chul Ye. "Geometric Gan". In: *arXiv preprint arXiv:1705.02894* (2017).

[39] Jeffrey Mahler et al. "Dex-Net 1.0: A Cloud-Based Network of 3D Objects for Robust Grasp Planning Using a Multi-Armed Bandit Model with Correlated Rewards". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2016.

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

[41] Jeffrey Mahler et al. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *CoRR* abs/1703.09312 (2017). arXiv: 1703.09312. URL: http://arxiv.org/abs/1703.09312.

[42] Jeffrey Mahler et al. "Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[43] Jeffrey Mahler et al. "Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[44] Jeffrey Mahler et al. "Privacy-preserving Grasp Planning in the Cloud". In: *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*. IEEE. 2016, pp. 468–475.

[45] Claudio Michaelis, Matthias Bethge, and Alexander Ecker. "One-Shot Segmentation in Clutter". In: *Proc. Int. Conf. on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 3549–3558.

[46] Claudio Michaelis, Matthias Bethge, and Alexander S Ecker. "One-Shot Segmentation in Clutter". In: *arXiv preprint arXiv:1803.09597* (2018).

[47] Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *CoRR* abs/1802.05957 (2018). arXiv: `1802.05957`. URL: `http://arxiv.org/abs/1802.05957`.

[48] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*. Vol. 153. Springer Science & Business Media, 2006.

[49] Nicolas Papernot et al. "Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples". In: *CoRR* abs/1602.02697 (2016). arXiv: `1602.02697`. URL: `http://arxiv.org/abs/1602.02697`.

[50] Andreas ten Pas et al. "Grasp pose detection in point clouds". In: *Int. Journal of Robotics Research (IJRR)* 36.13-14 (2017), pp. 1455–1473.

[51] Pedro O Pinheiro et al. "Learning to refine object segments". In: *European Conference on Computer Vision*. Springer. 2016, pp. 75–91.

[52] Lerrel Pinto and Abhinav Gupta. "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2016.

[53] Domenico Prattichizzo and Jeffrey C Trinkle. "Grasping". In: *Springer handbook of robotics*. Springer, 2008, pp. 671–700.

[54] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[55] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Proc. Advances in Neural Information Processing Systems*. 2015, pp. 91–99.

[56] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. "From caging to grasping". In: *Int. Journal of Robotics Research (IJRR)* (2012), p. 0278364912442972.

[57] Reuven Y. Rubinstein. "Optimization of computer simulation models with rare events". In: *European Journal of Operational Research* 99.1 (1997), pp. 89–112. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(96)00385-2. URL: http://www.sciencedirect.com/science/article/pii/S0377221796003852.

[58] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

[59] Amirreza Shaban et al. "One-shot learning for semantic segmentation". In: *arXiv preprint arXiv:1709.03410* (2017).

[60] Christian Szegedy et al. "Intriguing Properties of Neural Networks". In: *CoRR* abs/1312.6199 (2013). arXiv: 1312.6199. URL: http://arxiv.org/abs/1312.6199.

[61] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. "On catastrophic forgetting and mode collapse in Generative Adversarial Networks". In: *arXiv preprint arXiv:1807.04015* (2018).

[62] Koen EA Van de Sande et al. "Segmentation as selective search for object recognition". In: *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*. IEEE. 2011, pp. 1879–1886.

[63] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Proc. Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[64] David Wang et al. "Adversarial Grasp Objects". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE) [To Appear]*. IEEE. 2019.

[65] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920.

[66] Dawei Yang et al. "Realistic adversarial examples in 3d meshes". In: *arXiv preprint arXiv:1810.05206* (2018).

[67] Sergey Zagoruyko and Nikos Komodakis. "Learning to compare image patches via convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4353–4361.