Computing and Evaluating Adversarial Grasp Objects with Systematic Face Rotations and Applications of Dense Object Descriptors to Mechanical Search

David Tseng

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-92
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-92.html

May 21, 2019

Computing and Evaluating Adversarial Grasp Objects with Systematic Face Rotations and Applications of Dense Object Descriptors to Mechanical Search

by

David Tseng

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Trevor Darrell

Spring 2019

The thesis of David Tseng, titled Generalization of Deep Learning Models in the 3D Object Domain, is approved:

Chair _____ Date 5-16-19

_____ Date 5/15/19

_____ Date _____

University of California, Berkeley

Computing and Evaluating Adversarial Grasp Objects with Systematic Face Rotations and Applications of Dense Object Descriptors to Mechanical Search

Abstract

Computing and Evaluating Adversarial Grasp Objects with Systematic Face Rotations and Applications of Dense Object Descriptors to Mechanical Search

by

David Tseng

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

Learning-based approaches to understanding and manipulating 3D objects can handle a wide variety of objects, but may be unable to generalize all sets of objects. We will study the generalization of deep learning models in two subdomains of 3D objects: grasping and object descriptors.

In the first part of the thesis, we show how objects can be designed to be adversarial to Dex-Net, a neural network based grasping policy. We define an “adversarial grasp object" as an object that is visually similar to an original object but decreases the predicted graspability resulting from a robot grasping policy. We propose a method for synthesizing adversarial grasp objects by removing antipodal face pairs via systematic face rotations. We also explore a method that maintains local convexity and a deep-learning based method. Experiments suggest that all three algorithms consistently reduce graspability. Physical experiments demonstrate that the adversarial objects generated by the convexity preserving analytic algorithm decrease the grasp success rate by at least 87%. In simulation, the analytic rotation algorithm is able to reduce the graspability metric by 66%, 57%, and 63% on intersected cylinders, intersected prisms, and ShapeNet bottles.

In the second part of the thesis, we explore the potential application of dense object descriptors to mechanical search, where we attempt to find a target object in clutter. We find that dense object descriptors can generalize reasonably well and learn a consistent representation for unseen classes of objects. Feature matching methods can be combined with the descriptors to search for target objects in a heap.

To my parents, Bi Jiuan Tseng and Tung Fu Tseng, and my sister, Yuh-Wen Tseng

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Professor Ken Goldberg for giving me the opportunity to do research in his lab for the last two years. He has always pushed his students hard, and I have learned so many valuable skills and lessons through research that would be hard to find elsewhere. Jeff Mahler, Mike Danielczuk, and Matt Matl have provided valuable mentorship on the two projects in the thesis. I would also like to thank Roy Fox for also mentoring me in an earlier project last year. Thank you to David Wang, Yiding Jiang, Pusong Li, and Menglong Guo for their work on the Adversarial Grasp Objects project. David Wang has been an amazing research partner in two of the projects I have worked on and a great friend outside of research.

My college and graduate career would not have been complete without my friends and family. My parents, Tung Fu Tseng and Bi Jiuan Tseng, and my sister, Yuh-Wen Tseng, have supported me since the beginning and made all of this possible. I want to thank Xinge Ren for her love and support, and for the lovely time we have spent together so far. Thanks to Tim Chan, Chandler Chen, and Deborah Yang for the all laughs we have had while we all struggled and I look forward to all the trips we will take together in the future. I also want to thank Kevin Chiv for being a fun roommate and friend, and for the late night talks whenever I stayed up late. Finally, I would like to thank all the friends I have made in my classes and in the research lab, my project partners throughout the years, and all the friends back at home in San Gabriel Valley.

Chapter 1

Introduction

Recent advancements in learning based methods have facilitated progress in understanding and manipulating 3D objects, which will have a significant impact in industry. For example, robust robot grasping of a large variety of objects and mechanical search can benefit a diverse range of applications, such as the automation of industrial warehousing and home decluttering. However, this is challenging because robots must be able to rapidly and reliably handle novel objects with unique geometric and material properties such as new industrial parts, packaged consumer products, and toys. Recent research suggests that deep learning policies can work well on data distributions that the models are trained on, but can be prone to failures in examples not encountered during training. For example, while robot policies can generalize somewhat to unseen objects [23, 25, 37, 29], it can still fail on other objects that lie farther from the training distribution [30].

Adversarial images [42, 35, 22, 1] are examples of input that challenge the ability of learning-based approaches to generalize in the domain of computer vision. They are modified images that drastically alter the prediction made by a classifier while applying minimal perturbation to the original image. Similarly, we define “adversarial grasp objects," an analog of adversarial images in the domain of robust robot grasping. Adversarial grasp objects aim to reduce graspability while retaining geometric similarity to input objects. For the first project in the thesis, we present an analytical algorithm for synthesizing adversarial objects using systematic face rotations to remove antipodal faces. The chapter on adversarial grasp objects is part of a collaborative effort with David Wang*, David Tseng* Pusong Li*, Yiding Jiang*, Menglong Guo, Jeffrey Mahler, and Professor Ken Goldberg and was accepted to IEEE CASE 2018. My specific contributions to this project are listed in Sec. 3.4.

The second project in the thesis explores generalization in a different way. We evaluate how well dense object descriptors [9] generalize to objects of different classes than in the training set, and then extend them to applications in mechanical search [7].

This thesis contributes:

1. A formal definition of adversarial grasp objects.

2. An analytical algorithm to synthesize adversarial 3D objects for grasp planning from a given 3D object by performing systematic face rotations on pairs of antipodal faces.

3. Experiments studying adversarial grasp objects of several categories (bottles, intersected cylinders, and intersected prisms) generated by the algorithm 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 [28].

4. Physical experiments of a point contact gripper on the adversarial grasp objects generated by our algorithm.

5. Experiments demonstrating the performance of dense object descriptors as a general descriptor for unseen objects.

6. Exploratory experiments showing the potential application of dense object descriptors to mechanical search.

Chapter 2

Related Works

2.1 Adversarial Images

Adversarial images [42, 35, 22, 1] are inputs with a small added perturbation that can change the output of an image classifier, and the problem of finding adversarial images is typically formulated as a constrained optimization problem that can be approximately solved using gradient-based approaches [42]. 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 [46]. We build on this line of research by studying adversarial examples in the context of generating adversarial 3D objects for robotic grasping.

2.2 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 [38], empirical [2], 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 [38] or constrain the object's motion [39], possibly under perturbations to model robustness to sensor noise. Examples include GraspIt! [12], OpenGRASP [24], and the Dexterity Network (Dex-Net) 1.0 [28]. 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 [4, 6, 11, 14, 16, 20]. However, these systems do not scale well to novel objects and may be computationally expensive during execution.

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 [23, 15, 19] or using self-supervision to collect labels from successes and failures on a physical system [25, 37]. 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 [18, 36]. 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 [29, 30, 3]. In this paper, we consider synthesizing adversarial 3D objects for the analytic supervisor used to train these hybrid grasp planning methods.

2.3 Generative Models

Deep generative models map a simple distribution, such as a multivariate Gaussian distribution, to a much more complex distribution, such as natural images. Common deep generative models fall into likelihood-based models (i.e., the Variational Auto-Encoder (VAE) [21] and PixelCNN [33]) and likelihood-free models (i.e., various formulations of Generative Adversarial Networks (GANs) [13]). 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. Generative models have also been previously used in the domain of robot grasping, where Veres et al. [44] used conditional generative models to synthesize grasps from RGB-D images, and Bousmalis et al. [3] used GANs for simulation-to-reality transfer learning.

On the other hand, 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. [45], 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. [17], 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.

2.4 Object Descriptors

Object descriptors are features that can be more easily processed for semantic understanding of an object. One common use case is matching features between two views of the same object. ORB [41], LIFT [47], and SIFT [27] have all been object descriptors introduced in the past. More recently, Florence et al. proposed Dense Object Nets [9], in which networks learn to map pixels from an RGB image to a set of descriptors. In the mapping, the pixels corresponding to the same point on an object have the same descriptor, and the descriptors are consistent across different views and poses of a given object. Florence et al. demonstrate the performance of the network on single class objects as well as multiple classes of objects where the specific classes are known. In Chapter 4, we will be exploring the ability of Dense Object Nets to generalize to unseen classes and look at its potential application in mechanical search [7], in which a target object needs to be located inside clutter.

Chapter 3

Adversarial Grasp Objects

3.1 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 only consider a parallel-jaw grasping policy. We assume that the policy can be represented as:

$$\pi(\mathbf{x}) \triangleq \underset{\mathbf{u} \in \mathcal{U}(\mathbf{x})}{\text{argmax}}\ Q(\mathbf{x}, \mathbf{u}) \tag{3.1.1}$$

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 [31]:

$$g(\mathbf{x}, \pi) \triangleq \mathbb{P}_{\gamma}(Q(\mathbf{x}, \mathbf{u})) \tag{3.1.2}$$

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 $\mathbf{x}^*$:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\text{argmin}}\ g(\mathbf{x}, \pi) \text{ subject to } \sigma(\{\mathbf{x}\}, S) = 1 \tag{3.1.3}$$

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 [28]. 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 [28]. 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 [30].

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

$$Q(\mathbf{x}, \mathbf{u}) \triangleq \mathbb{E}_{\mathbf{u}' \sim p(\cdot|\mathbf{u}), \mathbf{x}' \sim p(\cdot|\mathbf{x})}[R(\mathbf{x}', \mathbf{u}')] \tag{3.1.4}$$

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 [8].

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 [10]. 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:

$$\hat{Q}(\mathbf{x}, \mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} R(\mathbf{x}_i, \mathbf{u}_i) \tag{3.1.5}$$

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.

3.2 Analytical Methods

We consider approaches 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 $\mathcal{V} = \{v_1, v_2, \ldots v_n\} \subset \mathbb{R}^3$ and a set of faces $\mathcal{F} = \{f_1, f_2, \ldots f_m\}$, where each face f_i is the triangle defined by three distinct elements of $\mathcal{V}$. Also, let $F_a = \{(f_i, f_j), \ldots (f_p, f_q)\}$ be the set of pairs of antipodal faces, and let the unit normal of face f_i be denoted by $\mathbf{n_i} \in \mathbb{R}^3$. Finally, let the antipodality angle φ between two faces be defined as $\varphi(f_i, f_j) = \arccos(-\mathbf{n_i}^T \mathbf{n_j})$.

Case Study: Cube

To motivate our analytical methods, we show a technique for modifying a cube to become adversarial to grasping by the Dex-Net 1.0 policy with minimal changes. Due to the shape of the cube, every

pair of opposite facets is antipodal. To remove these pairs, we add an additional vertex to three sides of the cube and perturb that vertex outwards by a distance d. The perturbed sides are chosen so that any possible grasp would be in contact with at least one perturbed side. By perturbing the vertices in this manner, all pairs of antipodal faces can be removed. The adversarial cube is shown in Fig. 3.1.

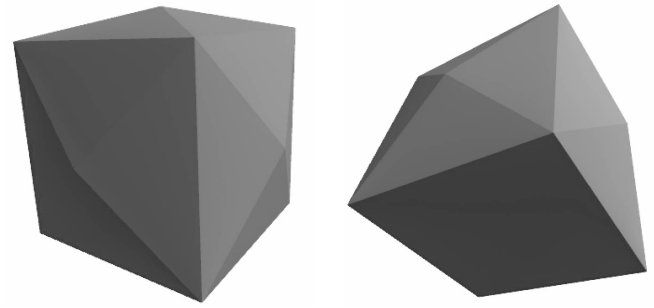

Figure 3.1: Two views of the adversarial cube described in Sec. 3.2. Here, d is chosen so that opposing faces have $\psi =$ 15 degrees.

To test this adversarial cube on a physical system, we 3D printed both the original cube and adversarial cubes with a side length of 1 inch and $d = 0.22, 0.32$, and 0.53 cm using nylon material. These values of d were chosen so that opposing faces have $\psi = 10, 15$, and 20 degrees respectively. Since a gripper with area contacts may be able to easily grasp corners, we used a gripper with a steel ball bearing to simulate point contacts. We estimated the antipodal friction angle threshold between the steel bearing and the nylon material to be around 16 degrees. For each object, we sampled 5 grasps and attempted each grasp 3 times. The results are shown in Table 3.1. The results are consistent with what we expect given the measured friction angle threshold. The object is able to grasp the original cube most of the time, the 15 degree cube almost none of the time, and is never able to grasp the 26 degree cube. This supports the idea that an adversarial object can be made by systematically perturbing faces to remove antipodal pairs.

Object	Fraction of Grasp Successes
Original Cube	14/15
Adversarial Cube (10 degrees)	9/15
Adversarial Cube (15 degrees)	2/15
Adversarial Cube (26 degrees)	0/15

Table 3.1: Results of physical trials on the 3D-printed cubes. For each object, we sampled 5 grasps and attempted each grasp 3 times for a total of 15 attempted grasps.

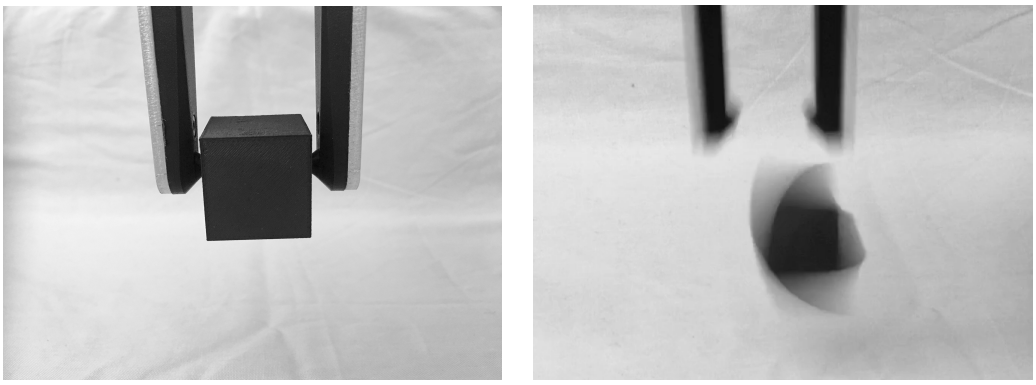

Figure 3.2: Physical experiments on the original and adversarial cubes with a gripper. Both ends of the gripper contain a steel bearing to simulate a point contact. Left: The gripper successfully grasping the original cube. Right: The gripper dropping the adversarial cube.

Antipodal Rotation Method

To generalize this idea of systematically removing pairs of antipodal faces to other objects, we present two analytical algorithms. Both of these algorithms reduce the grasp metric quality by perturbing faces or vertices in a way that increases the antipodal angle. The first analytical method increases the antipodality angle by repeatedly sampling antipodal faces and then perturbing them. At each iteration of this algorithm, we sample k points uniformly randomly on the surface of the mesh. If the face normals at those two opposing points lie within the friction cone as described in [28], then the two faces containing those two points are rotated to increase the antipodality angle, in the direction that increases φ the most. The perturbation parameter can be tuned in order to determine how strong the shape similarity constraint should be. In order to encourage the algorithm to make large structural changes instead of only local perturbations, if the mesh has more than d faces, we decimate the mesh until it has less than d faces. For this paper, we use $d = 350$. An in-depth description of the process is included in Algorithm 1.

Algorithm 1: Antipodal Rotation Algorithm

procedure SAMPLEFORANTIPODALFACES($\mathcal{F}$, k, ψ) ▷ $\mathcal{F}$ is the set of faces in the input mesh, k is the number of points to sample on the mesh to test for antipodal faces, ψ is the antipodality angle threshold. This was adapted from the grasp sampling method in [28].
    $V_1 \leftarrow$ sample k points uniformly randomly on F
    $V_2 \leftarrow$ corresponding intersections with F when shooting a ray starting from $v \in V_1$.
    $F_1 \leftarrow$ the faces $\subseteq F$ containing V_1
    $F_2 \leftarrow$ the faces $\subseteq F$ containing V_2
    $F_{ant} \leftarrow \{\}$ ▷ Stores all antipodal pairs of faces
    for all $(v_1, v_2, f_1, f_2) \in V_1, V_2, F_1, F_2$ **do**
        $n_1, n_2 \leftarrow$ face normals of f_1, f_2
        $d \leftarrow$ vector from v_1 to v_2
        if n_1, n_2 in friction cone of angle ψ from d **then**
            F_{ant}.append((f_1, f_2))
    return F_{ant}
procedure ANTIPODAL ROTATION($\mathcal{F}$, N_{iter}, k ψ, p, d) ▷ $\mathcal{F}$ is the set of faces in the input mesh, N_{iter} is the number of iterations, k is the number of points to sample on the mesh to test for antipodal faces, ψ is the antipodality angle threshold, p is the perturbation parameter, d is the decimation threshold
    if sizeof$(F) > d$ **then**
        $F \leftarrow$ decimate F until sizeof$(F) \leqslant d$
    $p \leftarrow (p) \cdot$ (mesh diameter) ▷ Scale by p by the mesh diameter
    for $i \in 1 \ldots N_{iter}$ **do**
        $F_{ant} \leftarrow$ SAMPLEFORANTIPODALFACES$(\mathcal{F}, k, \psi)$
        for all pairs of faces $f_i, f_j \in F_{ant}$ **do**
            $n_i, n_j \leftarrow$ face normals of f_i, f_j
            $\varphi =$ angle between $(-n_i, n_j)$
            if $\varphi <$ antipodality angle threshold **then**
                $u \leftarrow$ unit vector $\perp \{n_i, n_j\}$
                $\theta_i = (\psi - \varphi)p$
                $\theta_j = -\theta_i$
                $R_i \leftarrow$ rotation matrix with origin at center of f_i with angle θ_i around axis u
                $R_j \leftarrow$ rotation matrix with origin at center of f_j with angle θ_j around axis u
                Apply R_i, R_j to vertices of f_i, f_j respectively
    return $\mathcal{F}$

Random Perturbation with Convexity-Preserving Rejection Sampling

While the antipodal rotation method efficiently removes antipodal pairs, it can create concavities in parts of the mesh that were originally convex and vice versa. This may end up generating an

adversarial object that is significantly different the original. An alternative approach is to use random perturbation with rejection sampling — we randomly perturb vertices and reject changes that change the local convexity of the mesh. At each iteration, we sample points and directions to perturb, and keep the change that maintains local convexity while maximizing ψ between antipodal faces. The perturbation parameter can be tuned in order to determine how strong the shape similarity constraint should be. Similar to the antipodal rotation method, in order to encourage the algorithm to make large structural changes instead of only local perturbations, if the mesh has more than d faces, we decimate the mesh until it has less than d faces. For this paper, we use $d = 350$.

An in-depth description of the process is included in Algorithm 2.

Algorithm 2: Random Perturbation with Rejection Sampling Algorithm

procedure RANDOM PERTURBATION WITH REJECTION SAMPLING($\mathcal{F}$, N_{iter}, k ψ, p, d) ▷ $\mathcal{F}$ is the set of faces in the input mesh, N_{iter} is the number of iterations, k is the number of points to sample on the mesh to test for antipodal faces, ψ is the antipodality angle threshold, p is the perturbation parameter, d is the decimation threshold
  if sizeof(F) $> d$ **then**
    $F \leftarrow$ decimate F until sizeof(F) $\leqslant d$
  $p \leftarrow (p)\cdot$ (mesh diameter)
  for $i \in 1...N_{iter}$ **do**
    $F_{ant} \leftarrow$ SAMPLEFORANTIPODALFACES($\mathcal{F}, k, \psi$)
    for all pairs of faces $f_i, f_j \in F_{ant}$ **do**
      $n_i, n_j \leftarrow$ face normals of f_i, f_j
      $\varphi =$ angle between $(-n_i, n_j)$
      if $\varphi <$ antipodality angle threshold **then**
        $v \leftarrow$ random vertex in f_i
        $\varphi_{hist} \leftarrow \{\}$
        $d_{hist} \leftarrow \{\}$
        for $j \in 1...N_{directions}$ **do**
          $d \leftarrow$ random unit direction scaled by p
          $v \leftarrow v + d$
          $\varphi_{perturb} \leftarrow$ angle between $(-n_i, n_j)$
          if local convexity maintained **then**
            φ_{hist}.append($\varphi_{perturb}$)
            d_{hist}.append(d)
          Revert change
        $d_{best} \leftarrow$ direction in d_{hist} that maximizes corresponding $\varphi_{perturb}$
        $v \leftarrow v + d_{best}$
  return $\mathcal{F}$

3.3 Deep Learning Algorithm: CEM + GAN

One potential issue with the analytical methods described in Sec. 3.2 is that both assume an ideal system with point contacts that cannot grasp vertices or edges. An alternative method 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. Most significantly, the analytical methods and the CEM + GAN method all decrease the Dex-Net 1.0 metric, but the CEM + GAN method does not assume any knowledge of how the metric is designed. As a result, the CEM + GAN method can still be applied with any metric, including more accurate metrics that may be proposed in the future, such as an area-contacts based metric.

Deep Generative Models

One challenge in performing the optimization in Equation 3.1.3 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 3.1.3, but instead optimizing for a distribution of objects that we want to be similar to some prior subset $S \subset \mathcal{X}$:

$$\theta^*(\pi) = \arg\min_{\theta \in \Theta} \mathbb{E}_{\mathbf{x} \sim p_\theta(\cdot)}[g(\mathbf{x}, \pi)] \text{ subject to } \sigma(\mathcal{X}_\theta, S) = 1, \tag{3.3.1}$$

where $\mathcal{X}_\theta \subset \mathcal{X}$ is the support of the probability distribution p_θ for some parameter $\theta \in \Theta$.

We propose a deep learning method using the cross-entropy method (CEM) and generative adversarial networks (GANs) to approach this optimization problem. Let P_θ be the distribution over $\mathcal{X}$ induced by the model with parameter θ and P_S be the distribution empirically defined by S. We then define the shape similarity constraint $\sigma(\mathcal{X}_\theta, S)$ in Objective 3.1.3 as $\mathrm{D}_{\mathrm{KL}}(P_S || P_\theta) < \varepsilon$, where D_{KL} is the Kullback-Leibler divergence between two distributions, and $\varepsilon > 0$ is a hyperparameter that can be controlled through the sampling percentile γ (smaller γ means more similar distributions).

Optimization via Resampling

The cross-entropy method (CEM) [40] is an adaptive derivative-free optimization algorithm that has been widely applied. We are interested in finding the distribution of rare events that minimize a real-valued quality function $q(\mathbf{x})$ over $\mathcal{X}$. To minimize graspability, we choose $q(\mathbf{x}) = g(\mathbf{x}, \pi)$.

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 Generative Adversarial Network

To represent 3D geometry, we use the Signed Distance Function (also known as signed distance field or SDF) [34] as an alternative representation for generating 3D geometry. Binary 3D occupancy grids were considered, but produce blocky artifacts.

We draw on techniques used in Spectral-Normalization GAN (SNGAN) [32], which can generate high-fidelity images, and apply them to SDF's. 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 [26] as we empirically found that it stabilizes training. The GAN objective is then

$$\mathcal{L}_D^{data} = -\mathbb{E}_{\mathbf{x}\sim p_{data}(\cdot)}[\min(0,-1+D(\mathbf{x}))] \tag{3.3.2}$$

$$\mathcal{L}_D^{gen} = -\mathbb{E}_{\mathbf{z}\sim p_z(\cdot)}[\min(0,-1-D(G(\mathbf{z})))] \tag{3.3.3}$$

$$\mathcal{L}_D = \mathcal{L}_D^{data} + \mathcal{L}_D^{gen} \tag{3.3.4}$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim p_z(\cdot)}[D(G(\mathbf{z}))] \tag{3.3.5}$$

3.4 Experiments

We run the algorithms to minimize overall graspability on several datasets. For the first experiment, we explore and compare the results of applying the two analytical algorithms on simple convex mesh. In the subsequent sections, we run the antipodal rotation method and the CEM + GAN method on two synthetic datasets and the ShapeNet [5] bottles category. To allow a fair comparison between our two algorithms, we converted all three datasets to SDFs. 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 the synthetic datasets, we used the process presented by Bousmalis et al. [3] 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. To show that the GAN also works on non-cylindrical objects, 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 Fig. 3.7.

Unless otherwise specified, 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 $\kappa = c \cdot \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{x}_i,\pi)$, 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 Methods: Simple Convex Shapes

We run the two analytical algorithms on convex shapes. Since there are relatively few faces in these simple meshes, we iterated through all pairs of antipodal faces instead of sampling antipodal faces. We still introduce randomness by randomly shuffling the order of antipodal face pairs that we perturb. For this particular experiment, we set the antipodal angle threshold to 10 degrees, and used $p = 0.01$. We set N_{iter} to infinity and terminated the algorithm only when there were no longer any pairs of antipodal faces. The results from running the two algorithms on a dodecahedron mesh is shown in Fig. 3.3 and 3.4. Similar results can be observed when running the two algorithms on other simple meshes like a cube or cuboctahedron or meshes that are not necessarily convex. The antipodal rotation algorithm reduces the majority of the antipodal pairs in much fewer iterations than the convex rejection sampling method. However, upon inspection, the convexity-preserving rejection sampling method appears to maintain the shape of the original object more closely than the antipodal rotation method, since the antipodal rotation method produces visible concavities.

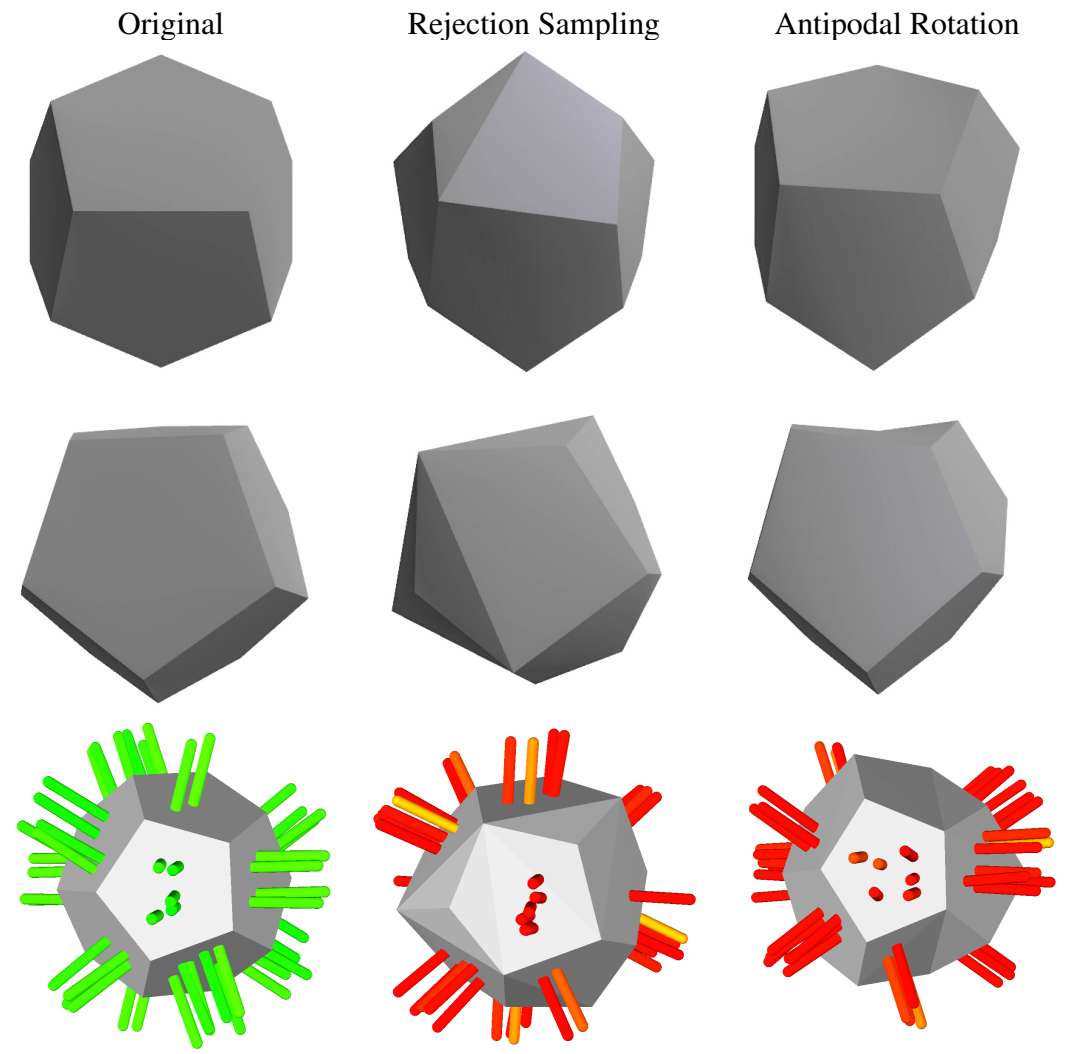

Figure 3.3: Results from applying the two analytical algorithms described in Sec. 3.2 to a dodecahedron to produce an adversarial dodecahedron. Top/middle row: Two different views of the adversarial mesh. Bottom row: The most robust 25 of 100 parallel-jaw grasps sampled on each object are displayed as grasp axes colored by relative reliability on a linear gradient from green to red, using an antipodal angle threshold of 10 degrees. From grasp quality visualization the last row, we can see that all antipodal pairs have been eliminated.

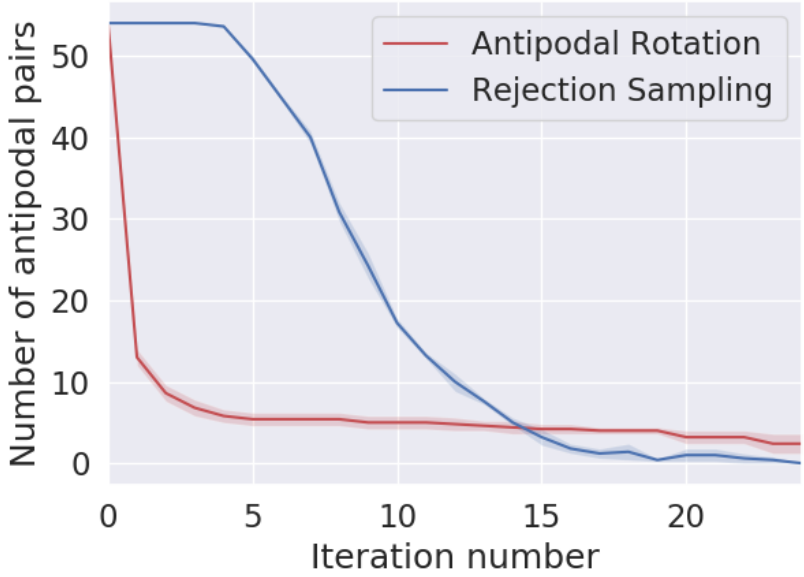

Figure 3.4: Plot of the number of antipodal pairs over the first 25 iterations of the analytical algorithms on a dodecahedron, averaged over five different iterations. The antipodal rotation algorithm described in Section 3.2 more rapidly reduces the number of antipodal pairs than the convexity-preserving rejection sampling algorithm described in Section 3.2, although the rejection sampling method reaches 0 antipodal pairs more quickly. In terms of wall time, the convexity-preserving rejection sampling method takes significantly longer due to the number of rejections — it takes 5.3 seconds compared to 0.21 seconds for the antipodal rotation method.

In addition to the dodecahedron shape, we applied the convexity-preserving rejection sampling method to cuboctahedron shape with antipodal angle thresholds $\psi = 10, 15$, and 26 degrees. We then 3D printed these objects with nylon and ran physical trials using the same experimental setup as described in Sec. 3.2. The 3D prints are shown in Fig. 3.5 and the results are shown in Table 3.2. The results suggest that the objects generated by the analytical method are indeed adversarial to real physical robotic point-contact grasps.

Figure 3.5: 3D printed adversarial cuboctahedrons generated by the convexity-preserving rejection sampling method. Antiespodal threshold from left to right: Original, 10 degrees, 15 degrees, 26 degrees.

Object	Fraction of Grasp Successes
Original Cuboctahedron	15/15
Adversarial Cuboctahedron (10 degrees)	9/15
Adversarial Cuboctahedron (15 degrees)	2/15
Adversarial Cuboctahedron (26 degrees)	0/15

Table 3.2: Results of physical trials on the 3D-printed cuboctahedrons. For each object, we sampled 5 grasps and attempted each grasp 3 times for a total of 15 attempted grasps.

Analytical Methods: Synthetic Dataset and ShapeNet Bottles

We run the antipodal rotation algorithm on 100 objects from each of the three datasets. We experimented with $p = 0.01, 0.075, 0.3$ for the shape similarity constraint described in Section 3.2. We find that the antipodal rotation algorithm decreases the graspability metric for all datasets. With a value of $p = 0.01$, the mean normalized graspability is decreased by 32% on the intersected cylinders dataset, 11% on the intersected prisms dataset, and 21% on the ShapeNet bottles datset. At each level of p, 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 perturbations. Sample object examples along with their adversarial versions, the associated graspability metrics, and the distribution of graspability metrics before and after applying the analytical algorithm are shown in Fig. 3.6. Increasing p decreases the graspability at the cost of similarity to the original object, corresponding to an increasingly relaxed shape similarity constraint.

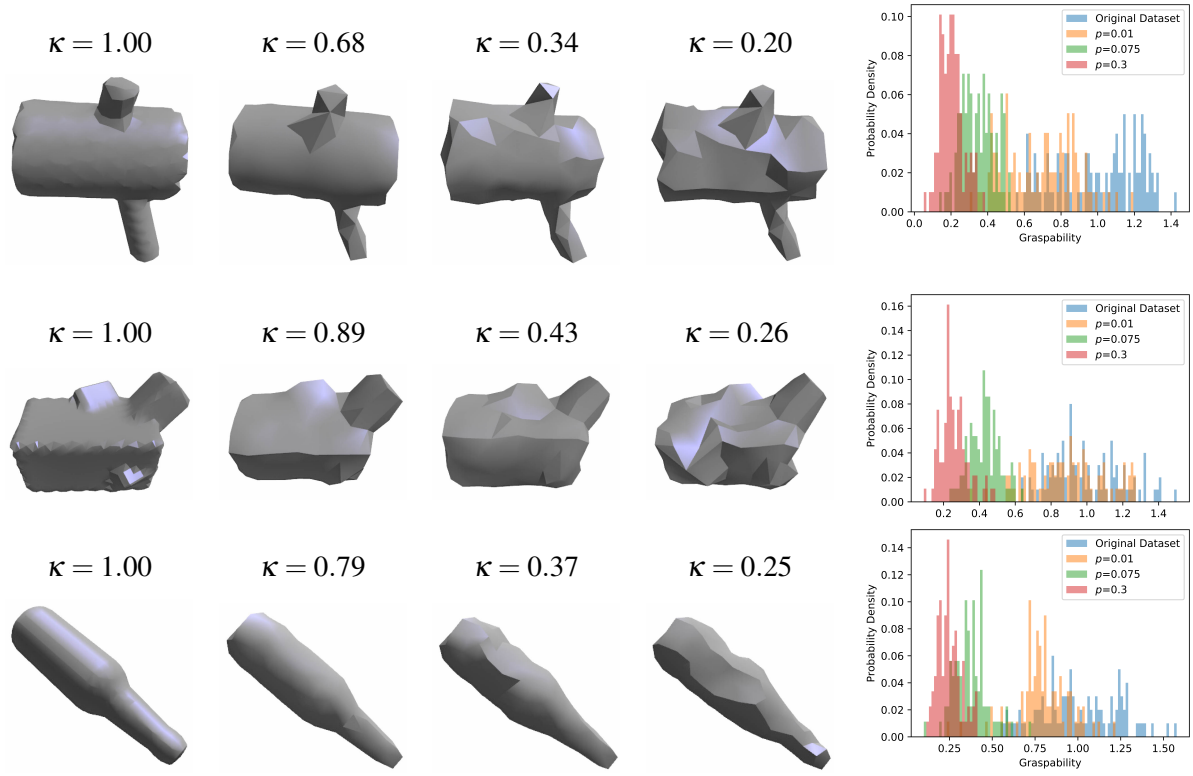

Figure 3.6: Antipodal Rotation Algorithm. We show the progression of an example from each dataset as we increase pertubation parameter p: each row (from left to right) shows the original object before decimation and then the perturbed versions using the surface normal constraint with $p = 0.01$, $p = 0.075$, and $p = 0.3$, respectively. The metric κ is the mean noramlized graspability of the generated dataset for the level of p, where the graspability is the empirical 75^{th} percentile of samples from the grasp quality function. The rightmost column shows the histograms of the graspability of all the objects. The analytical algorithm is able to decrease graspability on objects from all three datasets. The objects have been smoothed for visualization purposes with OpenGL smooth shading.

We also explored using the convexity-preserving rejection sampling algorithm on the datasets, but found that it modified the mesh shape significantly and does not reduce the grasp quality as much as the antipodal rotation algorithm. We find that while the rejection sampling method creates reasonable adversarial meshes with simple convex shapes in Sec. 3.4, it does not perform well on large complex shapes. This makes sense, since we can perturb a vertex by significant amounts and still maintain convexity. For example, a plane in the prism objects is convex, but perturbing a vertex outwards within the plane can dramatically change its appearance yet still remain convex. Fig 3.7. shows some sample results of applying the rejection sampling algorithm on the same objects from Fig. 3.6. The objects look more perturbed and still has higher grasp quality than the adversarial objects shown in Fig. 3.7.

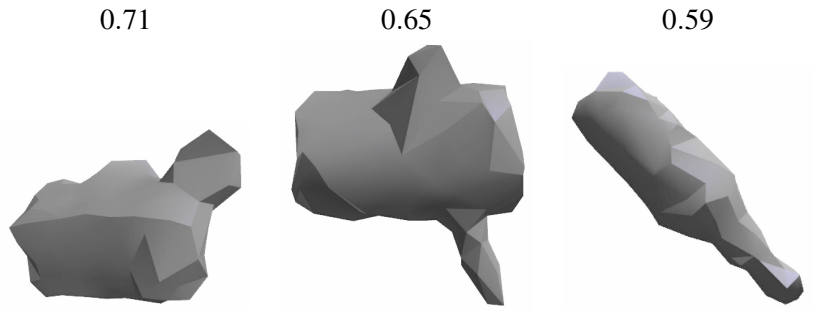

Figure 3.7: Left to right: results after applying the convexity-preserving rejection sampling algorithm to the original objects from Fig. 3.6. The number on top of each object represents the graspability metric for that object

CEM + GAN Algorithm

We train the resampling GAN on the previously described intersected prisms and cylinders datasets, as well as the ShapeNet bottles category.

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, but this rejection rate is very low: for bottles, no grids were rejected in any resampling iteration, and on the intersected sets, rejection rate remained below 10% in all episodes. Training the GAN in each iteration takes approximately 4.5 hours on a Titan X Pascal GPU and 2 hours on a Nvidia V100 GPU in the Nvidia DGX cluster.

Examples of objects from the GAN output distributions and histograms showing the overall distribution of graspability over resampling episodes are shown in Fig. 3.8. 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.

$\kappa = 1.00$, Original Dataset: $\kappa = 1.02$, Episode 0: $\kappa = 0.83$, Episode 5:

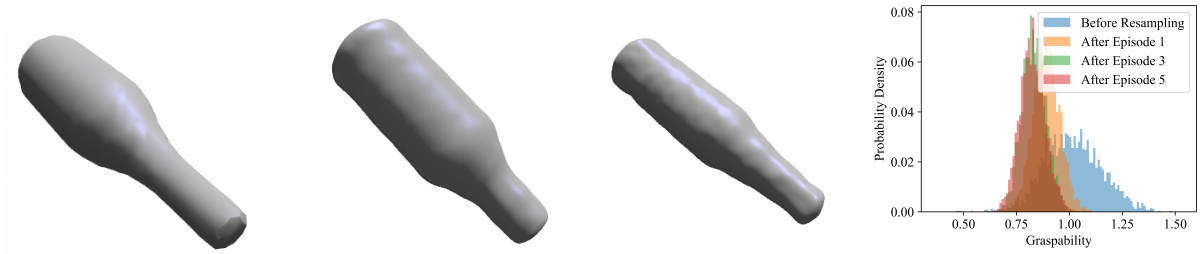

Figure 3.8: CEM + GAN Algorithm. The images on the left are example objects from the GAN output distribution as the resampling progresses. “Original" means the original SDF dataset, “Episode 0" denotes the GAN trained on the prior dataset (the first GAN trained, or Episode 0), and “Episode n" denotes the n^{th} GAN trained excluding the first. The κ values are the mean normalized graspabilities over a set of 100 objects generated during the corresponding stage in the training, where the graspability is the empirical 75^{th} percentile of samples from the grasp quality function. The right image is the histogram showing the overall distribution of the graspability metric (normalized to the mean graspability Episode 0) on the GAN output distribution as resampling progresses. As the algorithm progresses through the episodes, the probability mass shifts towards lower graspability. The objects have been smoothed for visualization purposes with OpenGL smooth shading.

Comparing Local and Global Changes

A concern of these algorithms may be that they could be reducing grasp quality by adding surface noise. To quantify how much of the grasp quality reduction is due to surface noise, we use 1 iteration of Laplacian smoothing on the generated objects from each of the two synthetic datasets and the ShapeNet bottles dataset by the antipodal rotation algorithm and the CEM + GAN algorithm to minimize surface roughness and measure the effect of smoothing on object graspability. The objects generated by the antipodal rotation algorithm use $p = 0.075$. The full results are shown in Table 3.3. After smoothing, the mean normalized grasp quality of the generated objects from the antipodal rotation algorithm is 72.8%, 107%, and 109% higher than before smoothing. After smoothing, the mean normalized grasp quality of the generated objects from the CEM + GAN algorithm is 10.0%, 34.7%, and 8.70% higher than before smoothing. This suggest that the majority of the grasp quality reduction in the analytical rotation method comes from surface perturbation, while the majority of the grasp quality reduction in the CEM + GAN method appears to come from global changes since the smoothing does not affect the mean grasp quality as significantly. However, even after smoothing, the antipodal rotation method produces adversarial objects that have a mean grasp quality that is lower than the ones generated by the CEM + GAN method in the cylinders and bottles dataset. In addition, some global changes can be seen in both algorithms — for example, both algorithms generate a tapered bottle that resembles a bottle but was not in the original bottles dataset.

Dataset	Graspability Before Smoothing		Graspability After Smoothing	
	Antipodal R.	CEM + GAN	Antipodal R.	CEM + GAN
Intersected Cylinders	0.345 ± 0.017	0.783 ± 0.011	0.596 ± 0.036	0.862 ± 0.031
Intersected Prisms	0.429 ± 0.016	0.577 ± 0.012	0.889 ± 0.039	0.777 ± 0.037
ShapeNet Bottles	0.372 ± 0.019	0.827 ± 0.012	0.779 ± 0.029	0.899 ± 0.033

Table 3.3: Comparison of the normalized mean graspability (reported with 95% confidence intervals) of objects generated by both the antipodal rotation algorithm ($p = 0.075$) and the GAN algorithm before and after Laplacian smoothing. The smoothing causes the graspability metric of the objects generated by the antipodal rotation method increase more dramatically compared to the objects generated by the CEM + GAN algorithm. However, the objects generated by the antipodal rotation algorithm still have lower graspability in the cylinders and bottles datasets before and after smoothing.

Failure Modes

GANs are prone to mode collapse [43], 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 observed mode collapse by the 9^{th} iteration on all three datasets.

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. Some examples are shown in Fig. 3.9.

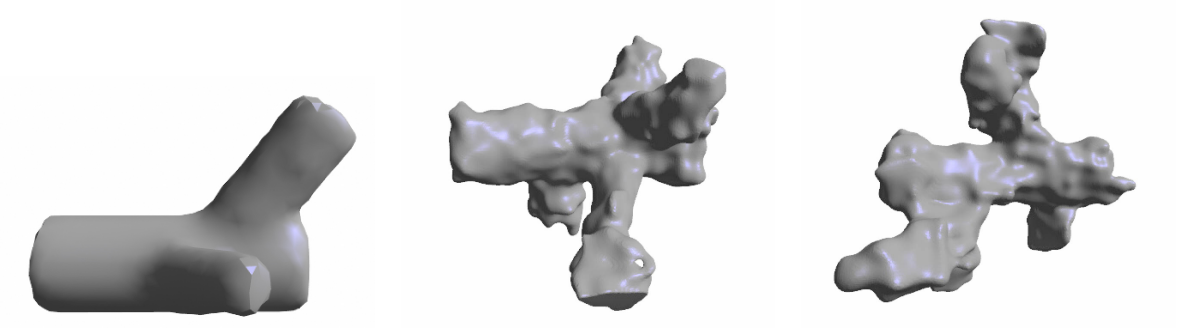

Figure 3.9: The left object is from the initial input prior distribution of intersected cylinders, and the others are objects sampled from the GAN's output distribution when it is trained on this prior without spectral normalization. The objects shown have been smoothed via Laplacian smoothing to emphasize that the GAN produces significant structural changes rather than simply adding surface roughness. However, this modified GAN also generates objects that deviate more from the original dataset.

CEM + GAN Generated Objects: Physical Trials

While the CEM + GAN generated shapes shown in Fig. 3.9 deviate significantly from the prior dataset, the shapes are relatively interesting. We collaborated with Hewlett-Packard to obtain high resolution prints of these objets and also ran physical trials on these objects.

Figure 3.10: High resolution prints from Hewlett-Packard of two objects generated by the CEM + GAN method without spectral normalization.

Dex-Net 4.0 was successful on 10/12 attempts, 83.3% with soft area contact grippers. However, with the point contact grippers, the grasps were successful on 1/30 attempts.

Individual Contributions

This work on Adversarial Grasp Objects was made in collaboration with multiple authors: David Wang*, David Tseng* Pusong Li*, Yiding Jiang*, Menglong Guo, Michael Danielczuk, Jeffrey Mahler, and Professor Ken Goldberg. Our paper was accepted to IEEE CASE 2019. I was responsible for the design and evaluation of the two analytical methods described in this chapter. David Wang worked on a variant of the two analytical methods using constrained perturbations that is explored in detail in that paper. The CEM + GAN algorithm was designed by Pusong Li and Yiding Jiang, while I helped train different GAN architectures. For the physical experiments, I worked with David Wang on the design of the adversarial cube and on running all of the physical experiments. The design of the cuboctahedron was generated using the convexity-preserving rejection sampling method described in this chapter. Michael Danielczuk provided help with registering objects and integrating grasp sampling from Dex-Net in the physical experiments. Menglong Guo designed the point-contact gripper and 3D printed all of the objects. Jeff Mahler provided extensive advice and feedback on our project and helped with editing our submitted paper, and his work on Dex-Net

provided the foundation for our project. Professor Goldberg provided valuable input and ideas on the direction of the project.

Chapter 4

Characterization of Dense Object Descriptors on Unseen Objects

4.1 Overview and Objective

Dense Object Nets is a work by Florence et al. [9] in which networks learn to map pixels from an RGB image to a set of descriptors. In the mapping, the pixels corresponding to the same point on an object have the same descriptor, and the descriptors are consistent across different views and poses of a given object. Florence et al. demonstrate the performance of the network on single class objects as well as multiple classes of objects where the specific classes are known. In this chapter, we explore the generalization of dense object descriptors to any unseen object as well as its potential applications in the mechanical search domain as described in [7]. Since this chapter is focused on the characterization of dense object descriptors, the main focus will be on experiments and applications. The training process and loss functions are the same as in [9].

4.2 Dataset Generation

For our experiments, we use a simulated dataset for rapid experimentation. To do this, we use objects from the Dex-Net dataset [28], sample random poses of the objects, and save the RGB and depth images. For our RGB datasets, we use textureless blue material on the objects with shading. Each dataset consists of 3,700 images of one object. Example objects in our datasets are shown in Fig. 4.4.

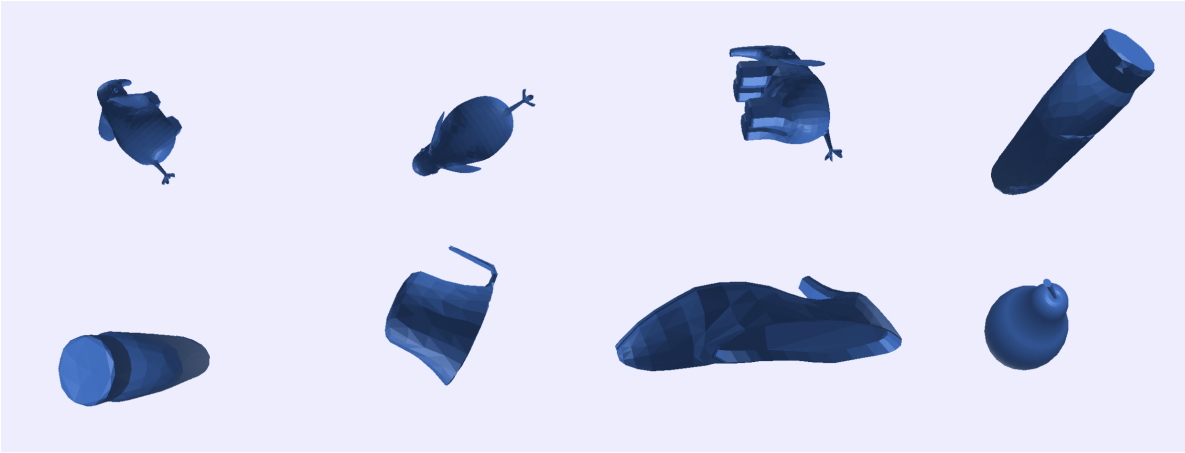

Figure 4.1: Sample RGB images from the generated datasets. The rendered objects are textureless and consist of a single color.

4.3 Generalization to Depth Data

We trained the dense object nets on depth-only data to test the generalization of dense object descriptors to depth. The input to the network is the depth data stacked together to create three channels, replacing the usual RGB channels. Examples of the network trained on a dataset consisting of images of a simulated shark object is shown in Fig. 4.4. We plot the cumulative distribution plot of the L2 distance between the best matches and the ground truth matches, and find that the network trained on depth images suffers some performance loss compared to the network trained on RGB images. However, sample images show that the network trained on depth images still performs reasonably well and is able to learn semantic regions of the shark, such as the fin and head.

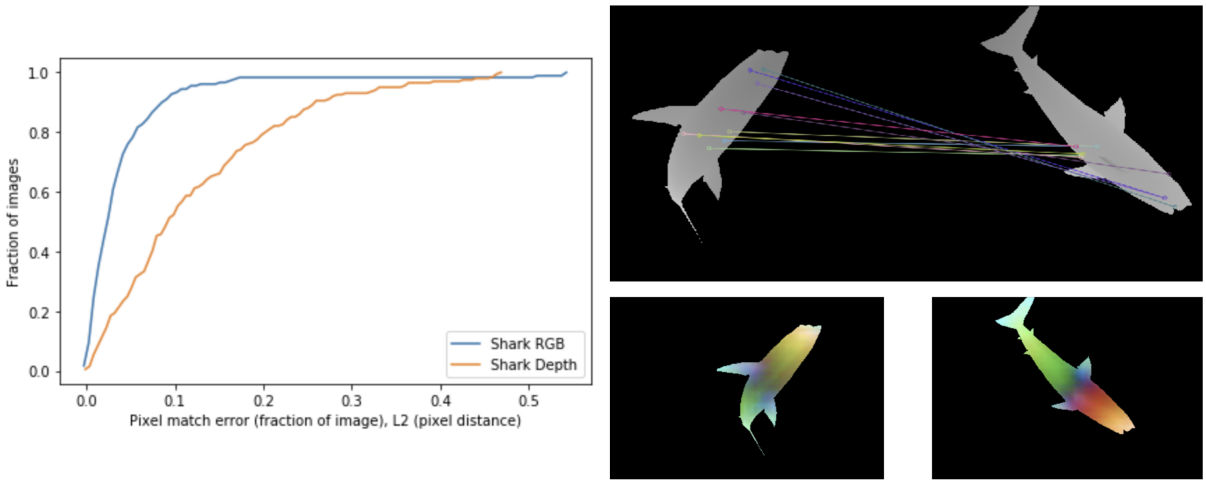

Figure 4.2: Results from applying dense object descriptors to a simulated shark dataset with only depth images. The cumulative distribution plot is shown on the left, and sample images are shown on the right. The top right image shows the best matches between the two images, where points are randomly sampled on the left image and matched with the pixel with the closest corresponding descriptor on the right image. The bottom left images show the mapping of the shark image to descriptor space.

4.4 Generalization to Unseen Classes of Objects

We trained the network on 21 object classes and tested on unseen classes. Each of the 21 object classes consisted of 3700 training images, resulting in a total of approximately 77,700 training images. We used 16 dimensional descriptors, RGB images, and normalization for this experiment. Unit normalization is a technique that was suggested by the authors where the output descriptors are normalized to the unit sphere and we confirmed to have a significant positive impact on the network performance. We find that the network is able to generalize reasonably well to unseen classes of objects. Sample results can be seen in Fig. 4.3. For meshes with more features, the network is better able to match descriptors between the pairs of views of the same object. Ranking the matches by L2 distance shows a more accurate representation of where the target object is. However, for objects with fewer features, like the bottle, the network fails to distinguish one part of the object from another. In the case of the bottle, the network incorrectly matches the top and bottom of the bottle. The overall results indicate that dense object descriptors can potentially be used as a general object descriptor. Overall, the dense object descriptors does has a potential to be used in mechanical search, since it is able to narrow down the location of the target object inside the heap.

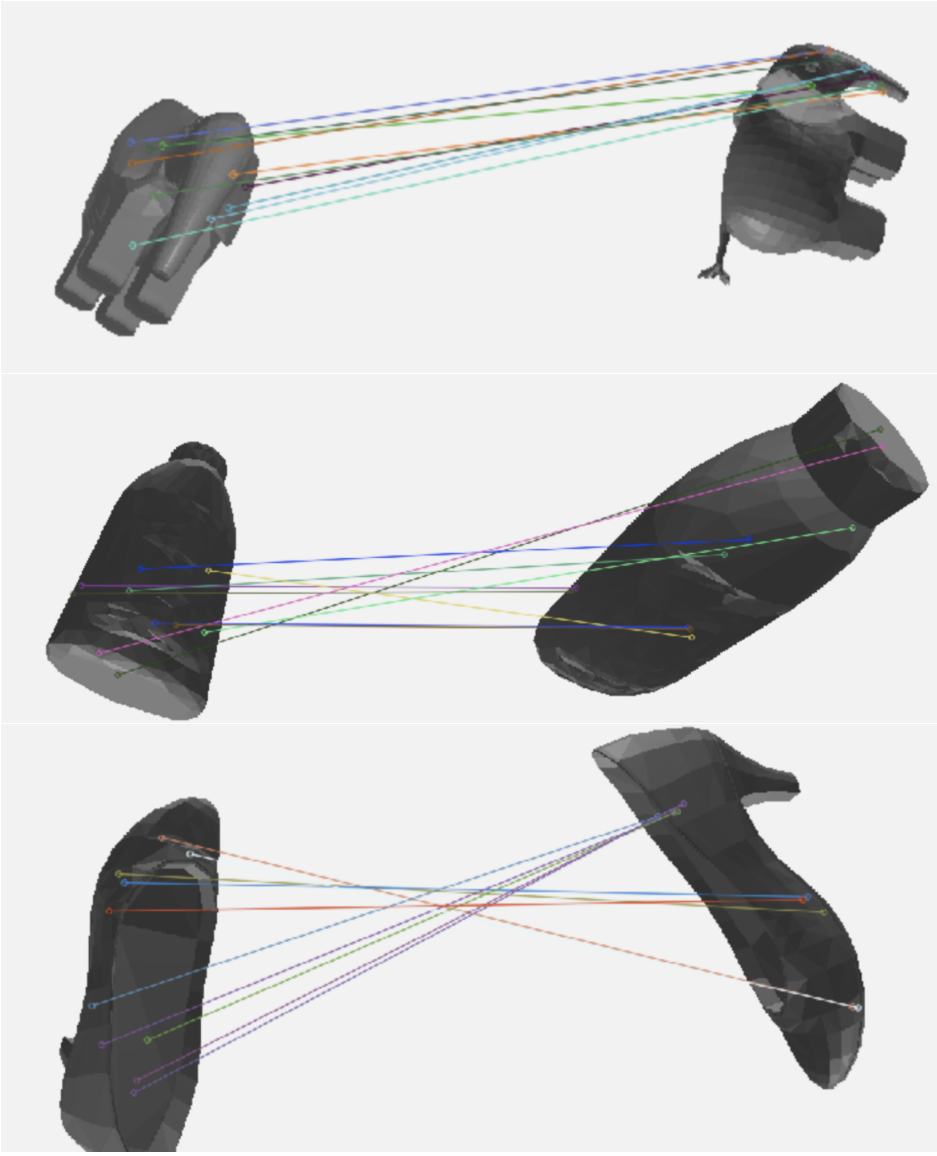

Figure 4.3: Results from applying a network trained on 21 classes on unseen classes. The input to the network is an RGB image, but the images shown are grayscale in order to show the matches between the objects, where points are randomly sampled on the left image and matched with the pixel with the closest corresponding descriptor on the right image.

4.5 Heap Experiments

Since the network is able to generalize decently well to unseen objects, we explore a possible application of dense object descriptors to mechanical search [7]. In this setting, it is not necessary for the descriptors of the same object to match exactly. Instead, we only need descriptors of an object to match to anywhere on the same object inside a heap of other objects. Since we are aiming to use

the dense object descriptors as a general descriptor, we cannot attempt to separate different classes of objects in descriptor space. As a result, it is possible for different objects to have significant overlap in descriptors. This is different from the work done in [9], where the authors have a fixed set of classes and attempt to separate them in descriptor space. For all experiments below, we use the network from Sec. 4.4 trained on 21 classes of objects with 16 dimensional descriptors, RGB images, and unit normalization.

Classification Experiment

To explore how much overlap different objects have in descriptor space, we run a simple classification experiment. Given 49 single object “heaps" and 49 target objects with 6 views, we would like to find the corresponding target object for each heap. We find the corresponding target object by sampling pairs of descriptors and ranking targets by the overall distance between the pairs. We define the distance between the pairs $(d_{1,1}, d_{1,2})$ and $(d_{2,1}, d_{2,2})$ to be $\min(\|d_{1,1} - d_{2,1}\|^2 + \|d_{1,2} - d_{2,2}\|^2, \|d_{1,2} - d_{2,1}\|^2 + \|d_{1,1} - d_{2,2}\|^2)$. This was found to have the highest success rate. However, using 16 dimensional descriptors with unit normalization and RGB images, we observe that only 36.7% of the heaps have the correct target object in the top 4 ranking. This suggests that there is indeed significant overlap in descriptor space between different object classes.

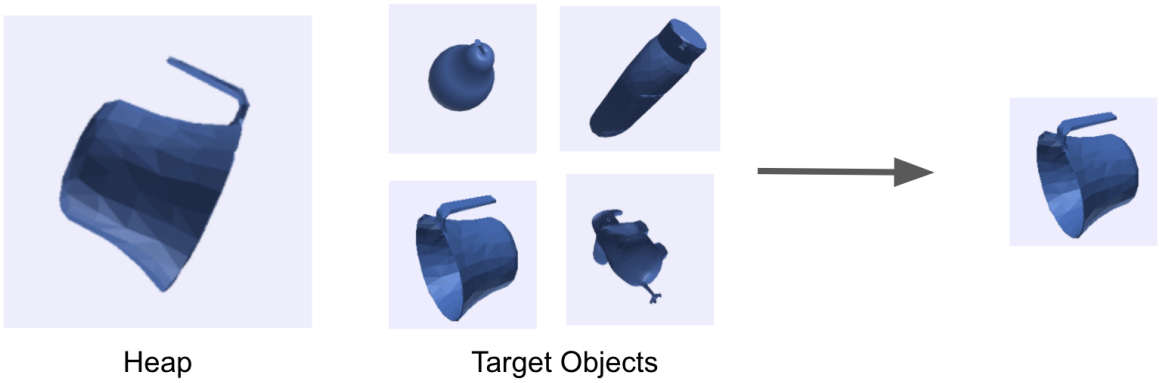

Figure 4.4: Classification experiment

Heatmap Experiment

Although there is significant overlap in descriptor space, we can still use the object descriptors as a heatmap for locating a target object within a heap. Since sampling pairs of points and ranking distance is a slow process, for this experiment we use a different filtering process. Here, we only consider pixels in images A and B to be matching only if the best match from A to B is the best match the other way around. Here, we only consider matches (i, j) such that the i-th descriptor in image A has j-th descriptor in image B as the best match and vice-versa. We rank matches by the L2 distance between the matching descriptors, so that the match with the lowest L2 distance has the highest probability of being correctly matched. We observe the following results shown in Fig. 4.5.

For many cases, we observe that we are able to predict reasonably the location of the target object in the heap by using this scheme to match the object descriptors. One failure case is when the object is nestled in the middle of heap. One possible explanation for this is that receptive field does not encounter heaps in the training process, which could potentially be mitigated by introducing heaps to the training process.

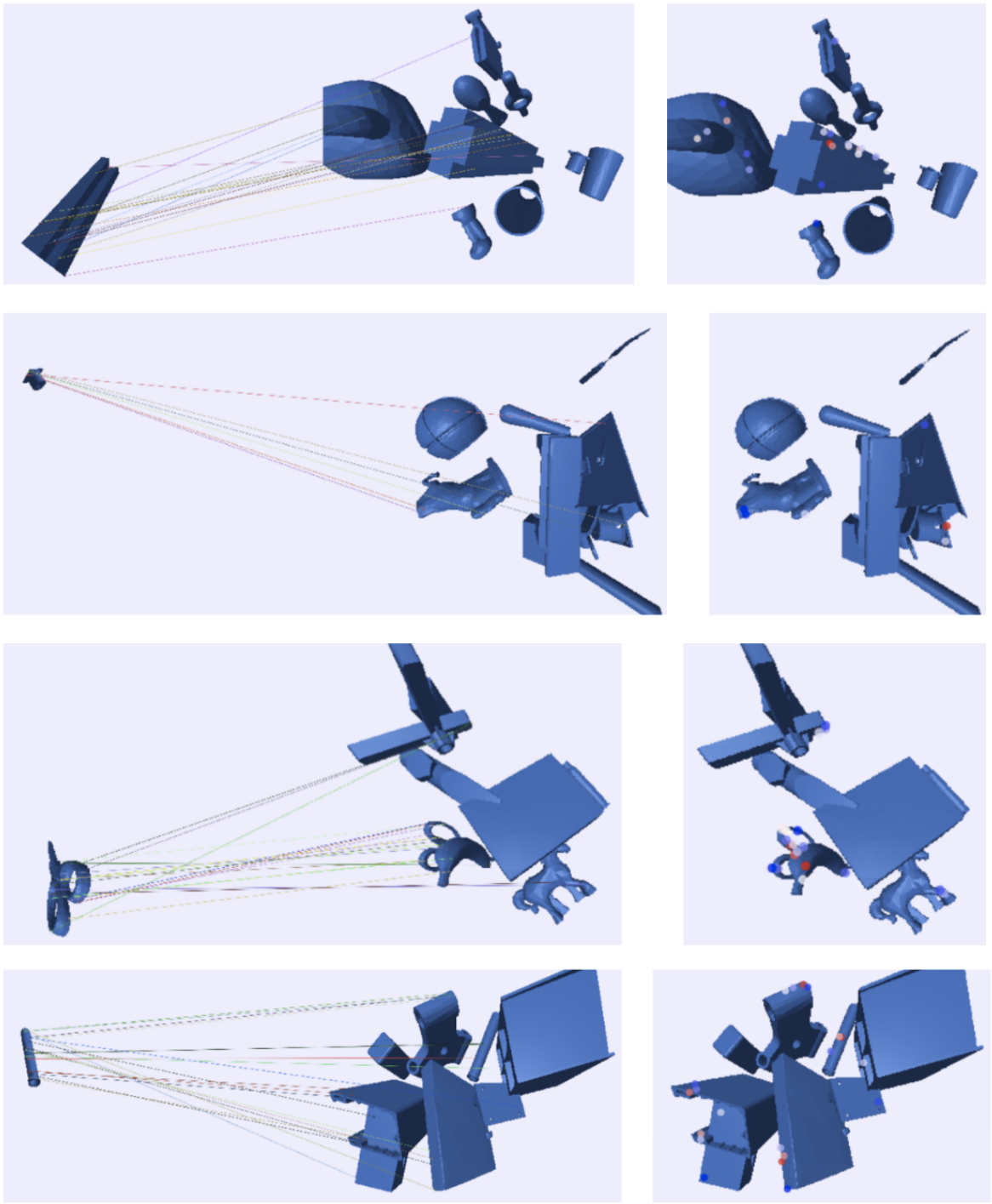

Figure 4.5: Left column: Matches between a target object and a heap. The target object is shown isolated on the left side of the image while the heap is shown on the right side of the image. Right column: Example images of the matches colored by the overall likelihood of the matches on a linear gradient from blue to red, with red being the highest probability of matching. Adding the color map increases the accuracy in which the network matches the target object. A failure case is shown in the last row, where the object is nestled deeply inside the heap.

Chapter 5

Discussion and Future Work

We explore the generalization of deep learning models on 3D objects in two subdomains: adversarial grasp objects and object descriptors. 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 three algorithms that generate adversarial grasp objects. The first is an analytic method (“convexity-preserving rejection sampling method") that perturbs vertices on antipodal faces while maintaining local convexity. The second (“antipodal rotation method") is a more efficient analytic method that rotates faces directly to remove antpodal pairs. The third is a deep-learning based method using a variation of the Cross-Entropy Method (CEM) augmented with a generative adversarial network (GAN) to synthesize adversarial grasp objects represented by a Signed Distance Field through more global geometric changes.

In each of the algorithms, we find that overall, the generated objects have significantly reduced grasp quality compared to the original set of objects. We observe that the generated distribution tend toward objects with fewer parallel surfaces. For example, the resulting distribution of the GAN trained on the bottle prior has primarily thin bottles with both a conical upper portion instead of a cap or stem structure, as well as a tapered main body.

We recognize that the current experiments focus on the DexNet 1.0 parallel-jaw point contact grasping model, and in future work can explore the generation of adversarial objects for area contacts. Replacing the point contact model with one that considers the full contact area could counteract the outsized effect of local surface roughness. In addition, objects that are designed to be adversarial to area contacts are more likely to be adversarial to an actual physical system found in industry, since most grippers are able to grasp corners and edges. We can also explore methods of incorporating adversarial grasp objects into the training process of a robot grasping policy to improve grasping performance. We believe that training augmentation with adversarial grasp objects could result in a grasping policy that is more robust to objects with subtle geometry.

We also further explored the generalization of dense object descriptors and its applications to mechanical search. We found that the dense object descriptors performed well even with depth-only data. They could also be used as general object descriptors if the networks are trained on a large number of classes. Finally, we found that dense object descriptors are able to find a target object within a heap by matching the descriptors between the target object and the heap, suggesting a

potential application in mechanical search. For future work, we can experiment with training the dense object descriptors on simulated depth images and testing on real depth images. To utilize this in mechanical search, more architecture and training modifications can be made in the future to encourage the network to match the target object in the heap more accurately. For example, including clutter in the training process can potentially mitigate the the failure case mentioned in Sec. 4.5.

Bibliography

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

[2] Jeannette Bohg et al. “Data-driven grasp synthesis—a survey”. In: *IEEE Trans. Robotics* 30.2 (2014), pp. 289–309.

[3] 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.

[4] 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.

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

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

[7] Michael Danielczuk et al. “Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter”. In: *arXiv preprint arXiv:1903.01588* (2019).

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

[9] 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).

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

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

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

[13] Ian J. Goodfellow et al. “Generative Adversarial Networks”. In: *Proc. Advances in Neural Information Processing Systems*. 2014.

[14] Carlos Hernandez et al. “Team Delft’s Robot Winner of the Amazon Picking Challenge 2016”. In: *arXiv preprint arXiv:1610.05514* (2016).

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

[16] 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.

[17] 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).

[18] 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.

[19] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. “Leveraging big data for grasp planning”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2015.

[20] 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.

[21] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel Recurrent Neural Networks”. In: *CoRR* abs/1601.06759 (2016). arXiv: 1601.06759. URL: http://arxiv.org/abs/1601.06759.

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

[35] 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.

[36] 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.

[37] 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.

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

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

[40] 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.

[41] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF.” In: *ICCV*. Vol. 11. 1. Citeseer. 2011, p. 2.

[42] 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.

[43] 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).

[44] Matthew Veres, Medhat Moussa, and Graham W Taylor. “Modeling grasp motor imagery through deep conditional generative models”. In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 757–764.

[45] 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.

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

[47] Kwang Moo Yi et al. “Lift: Learned invariant feature transform”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 467–483.