Real-World Robotic Perception and Control Using Synthetic Data

Joshua Tobin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-104
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-104.html

June 23, 2019

Real-World Robotic Perception and Control Using Synthetic Data

by

Joshua P Tobin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Pieter Abbeel, Chair
Professor Francesco Borrelli
Professor Trevor Darrell

Spring 2019

Real-World Robotic Perception and Control Using Synthetic Data

Abstract

Real-World Robotic Perception and Control Using Synthetic Data

by

Joshua P Tobin

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Pieter Abbeel, Chair

Modern deep learning techniques are data-hungry, which presents a problem in robotics because real-world robotic data is difficult to collect. Simulated data is cheap and scalable, but jumping the "reality gap" to use simulated data for real-world tasks is challenging. In this thesis, we discuss using synthetic data to learn visual models that allow robots to perform manipulation tasks in the real world. We begin by discussing domain randomization, a technique for bridging the reality gap by massively randomizing the visual properties of the simulator. We demonstrate that, using domain randomization, synthetic data alone can be used to train a deep neural network to localize objects accurately enough for a robot to grasp them in the real world. The remainder of the thesis discusses extensions of this approach to a broader range of objects and scenes. First, we introduce a data generation pipeline inspired by the success of domain randomization for visual data that creates millions of unrealistic procedurally generated random objects, removing the assumption that 3D models of the objects are present at training time. Second, we reformulate the problem from pose prediction to grasp prediction and introduce a generative model architecture that learns a distribution over grasps, allowing our models to handle pose ambiguity and grasp a wide range of objects with a single neural network. Third, we introduce an attention mechanism for 3-dimensional data. We demonstrate that this attention mechanism can be used to perform higher fidelity neural rendering, and that models learned this way can be fine-tuned to perform accurate pose estimation when the camera intrinsics are unknown at training time. We conclude by surveying recent applications and extensions of domain randomization in the literature and suggesting several promising directions for research in sim-to-real transfer for robotics.

Contents

Acknowledgments

I am extremely fortunate to have Pieter Abbeel as my advisor. Without his willingness to take a risk to work with me and mentor me early in my graduate career (despite my lack of background in computer science) this thesis would not have been possible. I am deeply grateful for the push he gave me to learn the field fast, the suggestion to work at OpenAI before I knew I was ready, and the support and advice he provided throughout my PhD.

I would also like to thank my supervisor at OpenAI, Wojciech Zaremba. Wojciech shaped many of my ideas about machine learning and robotics, and was an invaluable thought partner and mentor in all of the work contained in this thesis. I have been lucky to have amazing collaborators for the work in this thesis, including Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, and Peter Welinder. I'd like to particularly thank Rachel, Alex, Jonas, and Peter for frequently going well beyond their duty as teammates and co-authors to help make this work possible. Thanks to everyone on the OpenAI Robotics Team - Ilge Akkaya, Nikolas Tezak, Jerry Tworek, Alex Paino, Jonas Schneider, Maciek Chociej, Peter Welinder, Lilian Weng, Glenn Powell, Lei Zhang, Matthias Plappert, Arthur Petron, Wojciech Zaremba, Qiming Yuan, and Mateusz Litwin - for helpful discussions and for creating an amazing software environment to work with.

Thanks to my parents, siblings, and friends for your love and support and for keeping me grounded.

Lastly, to Sara - for far more than is possible to capture here.

Chapter 1

Introduction

1.1 Deep learning for robotics

Deep learning

Machine learning is the science of recognizing patterns and regularities in data without explicit programming. Traditional machine learning is a two-stage process that encodes human intuition and empirical knowledge into features and then uses those features to find a solution to the pattern recognition task. Deep learning is a class of machine learning techniques that formulates the task as a prediction problem, chooses a loss function, and optimizes the parameters of a powerful function approximator (a deep neural network) using stochastic gradient descent.

Although deep learning has been studied since the 1970s [100, 164, 218], it has seen a resurgence since 2012 [93, 101] and now enables state-of-the-art results in object recognition [93], speech recognition [29], translation [223], game playing [16, 124, 181, 182], and many other fields. One of the core insights underlying the broad success of these techniques is that, with enough data and compute and an expressive enough model, features learned directly from raw data can perform better than those engineered by domain experts.

We refer the reader to [53] for a more complete overview of the field.

Applications in robotics

The early successes of deep learning in computer vision and language inspired a number of applications in robotic perception and control. The most straightforward application of deep learning-based object recognition and detection models is to learn perception primitives that provide an input to a downstream robotic control system [43].

Another application is learning deep neural networks to perform decision making and control directly. The most prevalent approach is deep reinforcement learning. Reinforcement learning (RL) formulates the robot's interactions with the world as a sequential decision making problem. At each timestep, the robotic *agent* uses *observations* of the world to choose *actions* that allow it to interact with its *environment* and receive *rewards*. An RL algorithm attempts to maximize the agent's rewards by repeated trial-and-error interaction with the environment [191]. Deep RL employs deep neural networks as function approximators within the RL context [124].

A fundamental challenge of applying deep supervised learning and deep reinforcement learning to robotics is data availability. State-of-the-art deep supervised learning techniques use between millions and tens of millions of labeled examples [30, 223], and deep RL techniques often use hundreds of millions [124] or more [16]. However, obtaining labeled data for robotics is challenging because robots are expensive, 3-dimensional robotic data can be particularly difficult to label [242], and collecting data with robotic systems can be dangerous [48].

As a result, a fundamental research direction in deep learning for robotics is developing techniques to overcome with the high data requirement.

Overcoming the data availability problem in robotics

One way to collect datasets with enough scale to perform deep learning is to create robotic systems that autonomously collect and label large amounts of data [5, 108, 145]. These systems require mechanisms that provide their own labels and automatically reset the environment, and have been used to collect hundreds or thousands of hours of interaction data.

Another approach is to reduce the data requirement of the learning algorithms. In supervised learning, using deep neural networks pre-trained on broad datasets like ImageNet [30] can significantly reduce the amount of task-specific data required [51, 112]. In deep RL there are several active research areas that reduce the need for data. Instead of learning behaviors entirely through trial-and-error, model-based RL involves building or learning a model of the environment [13, 106]. Rather than solving a single task in a single environment, meta-RL algorithms learn how to solve a distribution of tasks in many environments, with the hope of being able to rapidly solve new tasks from that distribution [33, 42]. In learning from demonstrations (LfD), the agent is provided with richer supervision in the form of demonstrations of successful task execution from a human or another agent [31, 232, 241]. Self-supervised learning algorithms use data collected from trial-and-error interaction to solve other, related learning problems like achieving different goals [6] or solving surrogate tasks like observation reconstruction [179].

Rather than building large-scale data collection systems or making learning more efficient, this thesis focuses on taking advantage of low-cost synthetic data from physics simulators, game engines, and graphics engines.

1.2 Simulated data for robotics

Performing robotic learning in a simulator instead of the real world could accelerate the impact of machine learning in robotics by making data collection cheaper, faster, and more scalable. Since the state of the world is known in the simulator, researchers can also retrieve perfectly accurate labels programatically without the need for human labelers.

However, using simulated data for real-world problems is challenging due to the *reality gap* [74]. Unmodeled physical effects and the inability of simulated sensors to accurate reproduce the richness and noise of their real-world counterparts cause models trained in simulation to generalize poorly to real data.

The rest of this section describes some approaches to overcoming the reality gap.

Improving simulators

If the underlying cause of the reality gap is that simulators are not a faithful representation of the real world, a natural approach to overcoming it is to make simulators better match reality.

Simulators like DART [103], Gazebo [88], MuJoCo [201], and Open Dynamics Engine [186] aim to provide robotics-focused physics simulation that is fast enough to run in at least real time and models robot kinematics, dynamics, and rigid-body contacts in a physically realistic way.

Most robotics simulators model only a subset of the interactions possible in the real world. Another direction in simulation development is to build simulators that can model a wider range of physical phenomenon like nonrigid bodies (e.g., cloth) [47].

A given simulator can model a wide variety of different physical systems, so closely matching the parameters of the simulator to the real system is also important. System identification techniques [2, 92, 133, 219] aim to solve this problem.

Lastly, better physics must be matched with more accurate sensor simulation in order to faithfully model the real world. In the context of machine learning for robotics, much of this work has gone into building high-quality rendered image datasets in domains like self-driving cars [157], human pose estimation [211], and indoor scene representation [170].

Better simulators may help with sim-to-real transfer, but so far better simulators alone have not been sufficient to train models that work well in the real world. Another approach combines simulated data with real world data to achieve better performance than either data source alone.

Domain adaptation

Domain adaptation methods use data from a source domain to improve performance of a learned model on a different target domain where data is less available. In the context of sim-to-real transfer, the source domain is the simulator and the target is the real-world data distribution.

If labels or rewards are available in the target domain, supervised domain adaptation can be applied. The simplest form of supervised domain adaptation is fine-tuning, where the weights of a network trained in the source domain are used as initialization for a network trained in the target domain. Fine-tuning can be more efficient than learning from scratch in the real world [8, 82]. Alternatives to naive fine-tuning in supervised domain adaptation for robotics include using adaptation-focused model architectures like progressive networks [166], learning a specific part of the control system (instead of the full model) using real data [50, 128], explicitly treating the model learned in simulation as a Bayesian prior for the real-world model [27], and using real data to search over a low-dimensional subspace of models learned in a range of simulations [89].

Instead of training the network once in each domain, iterative learning control alternates between training in simulation, deploying the model in the real world, and using the data from the real world to improve the simulation [3, 28, 63, 210].

If labels or rewards are not available in the real world, they can be estimated from the model trained in simulation (weakly-supervised domain adaptation) [184, 207]. Unsupervised domain adaptation techniques work directly with unlabeled data from the target domain to match the distribution of inputs or features between the domains [69, 15, 225, 240].

Domain randomization

Domain randomization is the idea that, instead of carefully modeling all aspects of the real world, instead the simulation should be highly randomized. The intuition is that if the model sees enough variability in the simulator, the real world may look to the model like the next simulation. Choosing to randomize an aspect of the simulator like lighting conditions forces the model not to rely on it to make its predictions.

The core idea of domain randomization in robotics has been explored since the 1990s [74]. The phenomenon that highly randomized data can be used to learn deep neural networks that generalize to real data was first discovered in [109], and the use of randomized simulations to learn deep neural networks that generalize to real robots was first explored in [168].

1.3 Overview and contributions

In this thesis, we study learning perception models using simulated data that are useful for downstream robotics tasks. Though end-to-end visuomotor control policies are a promising direction for robotics, we choose to focus on models that estimate a representation of the state of the world and can be used as input to a separate control module.

Separating perception and control provides two advantages. First, it is easier to disambiguate which errors come from perception, which makes studying sim-to-real transfer of perception models easier to study. Second, whereas task-oriented end-to-end policies are often difficult to reuse [197], perception primitives need not be.

The main contributions of this thesis are the following:

- In Chapter 2, we explore a limited form of state estimation: learning a model that estimates the position of a single object of interest. We propose a domain randomization approach that allows our models to estimate the position of objects to 1.5cm, which is accurate enough to perform grasping in a cluttered real-world environment. To our knowledge, this is the first successful transfer of a deep neural network trained only on simulated RGB images (without pre-training on real images) to the real world for the purpose of robotic manipulation. This work was previously published as [200].

- In Chapter 3, we explore training a single model to estimate the state of many objects in the context of robotic grasping. We formulate the grasping task as a generative modeling problem and propose a neural network architecture that allows us to learn a distribution from which to efficiently sample grasps. To avoid the need to create realistic meshes for every object the model must grasp, we propose applying the idea of domain randomization to object synthesis. We procedurally generate low-fidelity, random objects and show that, by training on a wide array of such objects, models are able to generalize to realistic objects and grasp them in the real world. This work was previously published as [199].

- In Chapter 4, we explore a more general form of state estimation: building an implicit 3D model of the world by learning to render a scene from arbitrary viewpoints. Implicit 3D representations can model the state of any scene, and in principle scale with the size of the input images, not the complexity of the scene. We extend the Generative Query Networks (GQN) [39] model architecture with a novel attention mechanism based on the epipolar geometry of the scene. We show our model architecture E-GQN is able to produce high-fidelity renderings of more complex, higher resolution scenes than is possible with GQN alone. We demonstrate that fine-tuning these representations allows learning pose estimation models even when the camera intrinsics are unknown at training time. A publication of this work is currently in preparation [198].

- Finally, in Chapter 5, we discuss the implications of our work, including recent applications and extensions of domain randomization since our work was released. We conclude by discussing possible future research directions in sim-to-real transfer for robotic perception and control.

Chapter 2

Domain Randomization

2.1 Introduction

Robotic learning in a physics simulator could accelerate the impact of machine learning on robotics by allowing faster, more scalable, and lower-cost data collection than is possible with physical robots. Learning in simulation is especially promising for building on recent results using deep reinforcement learning to achieve human-level performance on tasks like Atari [123] and robotic control [106, 175]. Deep reinforcement learning employs random exploration, which can be dangerous on physical hardware. It often requires hundreds of thousands or millions of samples [123], which could take thousands of hours to collect, making it impractical for many applications. Ideally, we could learn policies that encode complex behaviors entirely in simulation and successfully run those policies on physical robots with minimal additional training.

Unfortunately, discrepancies between physics simulators and the real world make transferring behaviors from simulation challenging. System identification, the process of tuning the parameters of the simulation to match the behavior of the physical system, is time-consuming and error-prone. Even with strong system identification, the real world has unmodeled physical effects like nonrigidity, gear backlash, wear-and-tear, and fluid dynamics that are not captured by current physics simulators (though learning techniques may help bridge this gap [136]). Furthermore, low-fidelity simulated sensors like image renderers are often unable to reproduce the richness and noise produced by their real-world counterparts. These differences, known collectively as the *reality gap*, form the barrier to using simulated data on real robots.

This chapter explores domain randomization, a simple but promising method for addressing the reality gap. Instead of training a model on a single simulated environment, we randomize the simulator to expose the model to a wide range of environments at training time. The purpose of this work is to test the following hypothesis: if the variability in simulation is significant enough, models trained in simulation will generalize to the real world with no additional training.

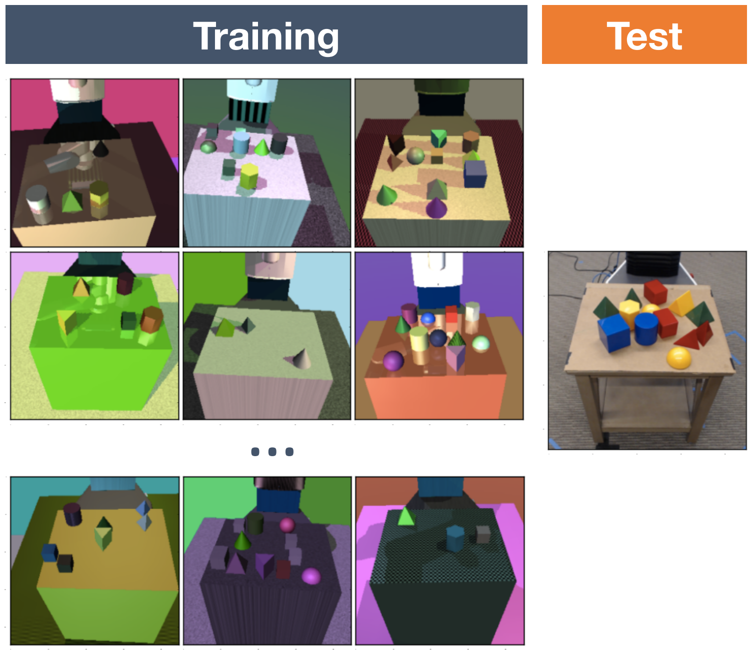

Figure 2.1: Illustration of our approach. An object detector is trained on hundreds of thousands of low-fidelity rendered images with random camera positions, lighting conditions, object positions, and non-realistic textures. At test time, the same detector is used in the real world with no additional training.

Though in principle domain randomization could be applied to any component of the reality gap, we focus on the challenge of transferring from low-fidelity simulated camera images. Robotic control from camera pixels is attractive due to the low cost of cameras and the rich data they provide, but challenging because it involves processing high-dimensional input data. Recent work has shown that supervised learning with deep neural networks is a powerful tool for learning generalizable representations from high-dimensional inputs [101], but deep learning relies on a large amount of labeled data. Labeled data is difficult to obtain in the real world for precise robotic manipulation behaviors, but it is easy to generate in a physics simulator.

We focus on the task of training a neural network to detect the location of an object. Object localization from pixels is a well-studied problem in robotics, and state-of-the-art methods employ complex, hand-engineered image processing pipelines (e.g., [24], [23], [195]). This work is a first step toward the goal of using deep learning to improve the accuracy of object detection pipelines. Moreover, we see sim-to-real transfer for object localization as a stepping stone to transferring general-purpose manipulation behaviors.

We find that for a range of geometric objects, we are able to train a detector that is accurate to around 1.5 cm in the real world using only simulated data rendered with simple, algorithmically generated textures. Although previous work demonstrated the ability to perform robotic control using a neural network pretrained on ImageNet and fine-tuned on randomized rendered pixels [168], this chapter provides the first demonstration that domain randomization can be useful for robotic tasks requiring precision. We also provide an ablation study of the impact of different choices of randomization and training method on the success of transfer. We find that with a sufficient number of textures, pre-training the object detector using real images is unnecessary. To our knowledge, this is the first successful transfer of a deep neural network trained *only* on simulated RGB images to the real world for the purpose of robotic control.

2.2 Related Work

Object detection and pose estimation for robotics

Object detection and pose estimation for robotics is a well-studied problem in the literature (see, e.g., [25], [23], [24], [38], [195], [226]). Recent approaches typically involve offline construction or learning of a 3D model of objects in the scene (e.g., a full 3D mesh model [195] or a 3D metric feature representation [23]). At test time, features from the test data (e.g., Scale-Invariant Feature Transform [SIFT] features [56] or color co-occurrence histograms [38]) are matched with the 3D models (or features from the 3D models). For example, a black-box nonlinear optimization algorithm can be used to minimize the re-projection error of the SIFT points from the object model and the 2D points in the test image [25]. Most successful approaches rely on using multiple camera frames [24] or depth information [195]. There has also been some success with only monocular camera images [25]. Neural network-based perception has also been explored [104], but collecting a large enough training set can be challenging.

Compared to our method, traditional approaches require less extensive training and take advantage of richer sensory data, allowing them to detect the full 3D pose of objects (position and orientation) without any assumptions about the location or size of the surface on which the objects are placed. However, our approach avoids the challenging problem of 3D reconstruction, and employs a simple, easy to implement deep learning-based pipeline that may scale better to more challenging problems.

Domain adaptation

The computer vision community has devoted significant study to the problem of adapting vision-based models trained in a source domain to a previously unseen target domain (see, e.g., [32], [71], [70], [94]). Approaches include re-training the model in the target domain (e.g., [231]), adapting the weights of the model based on the statistics of the source and target domains (e.g., [110]), learning invariant features between domains (e.g., [209]), and learning a mapping from the target domain to the source domain (e.g., [193]). Researchers in the reinforcement learning community have also studied the problem of domain adaptation by learning invariant feature representations [59], adapting pretrained networks [165], and other methods. See [59] for a more complete treatment of domain adaptation in the reinforcement learning literature.

In this chapter we study the possibility of transfer from simulation to the real world *without* performing domain adaptation.

Bridging the reality gap

Previous work on leveraging simulated data for physical robotic experiments explored several strategies for bridging the reality gap.

One approach is to make the simulator closely match the physical reality by performing system identification and using high-quality rendering. Though using realistic RGB rendering alone has had limited success for transferring to real robotic tasks [77], incorporating realistic simulation of depth information can allow models trained on rendered images to transfer well to the real world [147]. Combining data from high-quality simulators with other approaches like fine-tuning can also reduce the number of labeled samples required in the real world [158].

Unlike these approaches, ours allows the use of low-quality renderers optimized for speed and not carefully matched to real-world textures, lighting, and scene configurations.

Other work explores using domain adaptation to bridge the reality gap. It is often faster to fine-tune a controller learned in simulation than to learn from scratch in the real world [27, 89]. In [50], the authors use a variational autoencoder trained on simulated data to encode trajectories as a low-dimensional latent code. A policy learned on real data can overcome the reality gap by choosing latent codes via exploration that correspond to the desired physical behavior. In the evolutionary robotics literature, the work of Cully et al. [26] demonstrates that exploration in the real world can be more efficient with prior knowledge from a simulator.

Domain adaptation has also been applied to robotic vision. In [239], the authors show that modularity between the perception system and control system can aid transferability. Rusu et al. [166] find that using the progressive network architecture has better sample efficiency than fine-tuning or training in the real-world alone. In [208], the authors learn a correspondence between domains that allows the real images to be mapped into a space understood by the model. While the preceding approaches require reward functions or labeled data, Mitash and collaborators [122] pre-train an object detector using realistic rendered images to bootstrap an automated learning learning process that does not require manually labeling data and uses only around 500 real-world samples.

A related idea, iterative learning control, uses real-world data to improve the dynamics model used to determine the optimal control behavior, rather than using real-world data to improve the controller directly. Iterative learning control starts with a dynamics model, applies the corresponding control behavior on the real system, and then closes the loop by using the resulting data to improve the dynamics model. Iterative learning control has been applied to a variety of robotic control problems, from model car control (e.g., [3] and [28]) to surgical robotics (e.g., [210]). Similar ideas have been explored in the evolutionary robotics literature, such as in the work of Koos et al. [90].

Domain adaptation and iterative learning control are important tools for addressing the reality gap, but in contrast to these approaches, ours requires no additional training on real-world data. Our method can also be combined easily with most domain adaptation techniques.

Several authors have previously explored the idea of using domain randomization to bridge the reality gap.

In the context of physics adaptation, Mordatch and collaborators [127] show that training a policy on an ensemble of dynamics models can make the controller robust to modeling error and improve transfer to a real robot. Similarly, in [7], the authors use a simulator with randomized friction and action delay and find behaviors transfer to the real world.

Rather than relying on controller robustness, Yu et al. [233] use a model trained on varied physics to perform system identification using online trajectory data, but their approach is not shown to succeed in the real world. Rajeswaran et al. [151] explore different training strategies for learning from an ensemble of models, including adversarial training and adapting the ensemble distribution using data from the target domain, but also do not demonstrate successful real-world transfer.

Earlier work in evolutionary robotics also uses domain randomization to encourage sim-to-real transfer. In [73], the authors suggested that transferability can be achieved by randomly varying the items in the *implementation set* – model parameters that are not essential to the controller achieving near-optimal performance. Our work can be interpreted in this framework by considering the rendering aspects of the simulator (lighting, texture, etc) as part of the implementation set.

Researchers in computer vision have used 3D models as a tool to improve performance on real images since the earliest days of the field (e.g., [134]). More recently, 3D models have been used to augment training data to aid transferring deep neural networks between datasets and prevent over-fitting on small datasets for tasks like viewpoint estimation [187] and object detection [189], [130]. Recent work has explored using only synthetic data for training 2D object detectors (i.e., predicting a bounding box for objects in the scene). In [142], the authors find that by pretraining a network on ImageNet and fine-tuning on synthetic data created from 3D models, better detection performance on the PASCAL dataset can be achieved than training with only a few labeled examples from the real dataset.

In contrast to our work, most object detection results in computer vision use realistic textures, but do not create coherent 3D scenes. Instead, objects are rendered against a solid background or a randomly chosen photograph. As a result, our approach allows our models to understand the 3D spatial information necessary for rich interactions with the physical world.

Sadeghi and Levine's work [168] is the most similar to our own. The authors demonstrate that a policy mapping images to controls learned in a simulator with varied 3D scenes and textures can be applied successfully to real-world quadrotor flight. However, their experiments – collision avoidance in hallways and open spaces – do not demonstrate the ability to deal with high-precision tasks. Our approach also does not rely on precise camera information or calibration, instead randomizing the position, orientation, and field of view of the camera in the simulator. Whereas their approach chooses textures from a dataset of around 200 pre-generated materials, most of which are realistic, our approach is the first to use only non-realistic textures created by a simple random generation process, which allows us to train on hundreds of thousands (or more) of unique texturizations of the scene.

2.3 Methods

Given some objects of interest $\{s_i\}_i$, our goal is to train an object detector $d(I_0)$ that maps a single monocular camera frame I_0 to the Cartesian coordinates $\{(x_i, y_i, z_i)\}_i$ of each object. In addition to the objects of interest, our scenes sometimes contain distractor objects that must be ignored. Our approach is to train a deep neural network in simulation using domain randomization. The remainder of this section describes the specific domain randomization and neural network training methodology we use.

Domain randomization

The purpose of domain randomization is to provide enough simulated variability at training time such that at test time the model is able to generalize to real-world data. We randomize the following aspects of the domain for each sample used during training:

- Number and shape of distractor objects on the table
- Position and texture of all objects on the table
- Textures of the table, floor, skybox, and robot
- Position, orientation, and field of view of the camera
- Number of lights in the scene
- Position, orientation, and specular characteristics of the lights
- Type and amount of random noise added to images

Since we use a single monocular camera image from an uncalibrated camera to estimate object positions, we fix the height of the table in simulation, effectively creating a 2D pose estimation task. Random textures are chosen among the following:

(a) A random RGB value

(b) A gradient between two random RGB values

(c) A checker pattern between two random RGB values

The textures of all objects are chosen uniformly at random – the detector does not have access to the color of the object(s) of interest at training time, only their size and shape. We render images using the MuJoCo Physics Engine's [201] built-in renderer. This renderer is not intended to be photo-realistic, and physically plausible choices of textures and lighting are not needed.

Between 0 and 10 distractor objects are added to the table in each scene. Distractor objects on the floor or in the background are unnecessary, despite some clutter (e.g., cables) on the floor in our real images.

Our method avoids calibration and precise placement of the camera in the real world by randomizing characteristics of the cameras used to render images in training. We manually place a camera in the simulated scene that approximately matches the viewpoint and field of view of the real camera. Each training sample places the camera randomly within a $(10 \times 5 \times 10)$ cm box around this initial point. The viewing angle of the camera is calculated analytically to point at a fixed point on the table, and then offset by up to 5.7 degrees (0.1 radians) in each direction. The field of view is also scaled by up to 5 % from the starting point.

Model architecture and training

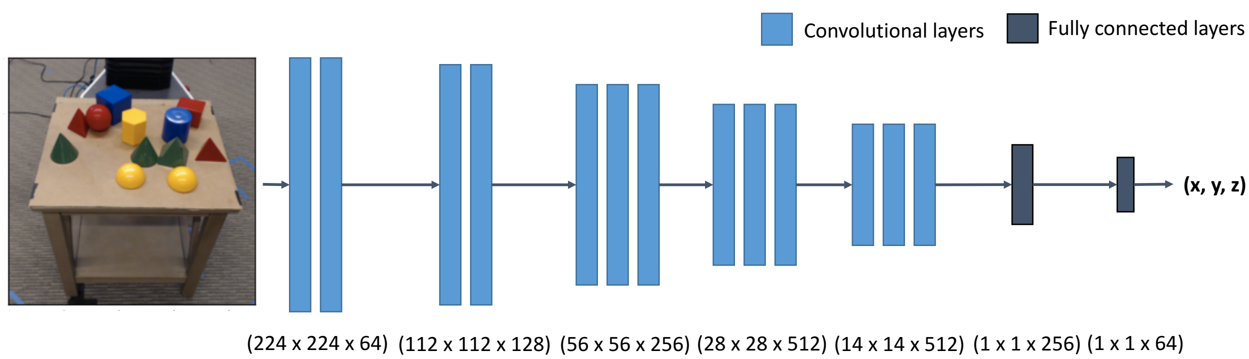

Figure 2.2: The model architecture used in our experiments. Each vertical bar corresponds to a layer of the model. ReLU nonlinearities are used throughout, and max pooling occurs between each of the groupings of convolutional layers. The input is an image from an external webcam downsized to (224×224).

We parametrize our object detector with a deep convolutional neural network. In particular, we use a modified version the VGG-16 architecture [183] shown in Figure 2.2. We chose this architecture because it performs well on a variety of computer vision tasks, and because it has a wide availability of pretrained weights. We use the standard VGG convolutional layers, but use smaller fully connected layers of sizes 256 and 64 and do not use dropout. For the majority of our experiments, we use weights obtained by pretraining on ImageNet to initialize the convolutional layers, which we hypothesized would be essential to achieving transfer. In practice, we found that using random weight initialization works as well in most cases.

We train the detector through stochastic gradient descent on the L_2 loss between the object positions estimated by the network and the true object positions using the Adam optimizer [87]. We found that using a learning rate of around 1e−4 (as opposed to the standard 1e−3 for Adam) improved convergence and helped avoid a common local optimum, mapping all objects to the center of the table.

2.4 Experiments

Experimental Setup

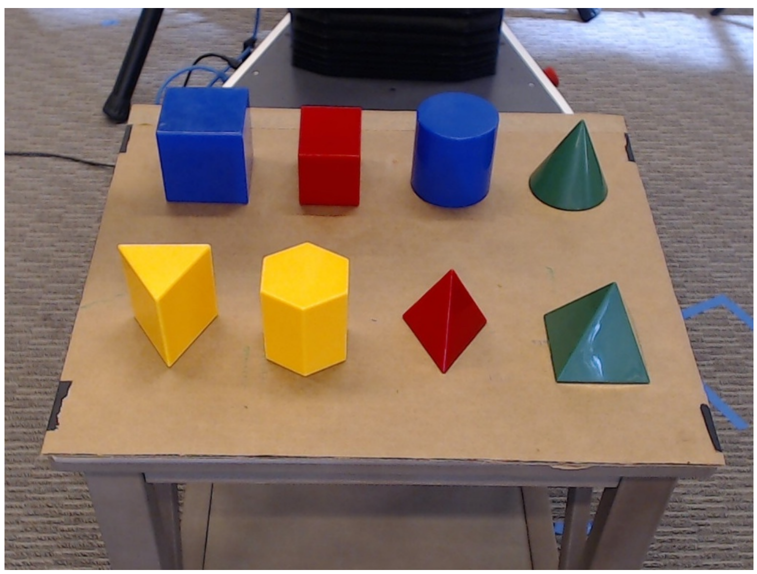

Figure 2.3: The geometric objects used in our experiments.

We evaluated our approach by training object detectors for each of eight geometric objects. We constructed mesh representations for each object to render in the simulator. Each training sample consists of (a) a rendered image of the object and one or more distractors (also from among the geometric object set) on a simulated tabletop and (b) a label corresponding to the Cartesian coordinates of the center of mass of the object in the world frame.

For each experiment, we performed a small hyperparameter search, evaluating combinations of two learning rates (1e−4 and 2e−4) and three batch sizes (25, 50, and 100). We report the performance of the best network.

The goals of our experiments are:

(a) Evaluate the localization accuracy of our trained detectors in the real world, including in the presence of distractor objects and partial occlusions

(b) Assess which elements of our approach are most critical for achieving transfer from simulation to the real world

(c) Determine whether the learned detectors are accurate enough to perform robotic manipulation tasks

Evaluation type	Object only	Distractors	Occlusions
Cone	1.1 ± 0.4	1.1 ± 0.3	0.9 ± 0.4
Cube	0.9 ± 0.5	2.0 ± 2.2	1.5 ± 1.1
Cylinder	0.9 ± 0.5	1.9 ± 3.6	2.6 ± 3.6
Hexagonal Prism	0.7 ± 0.5	0.6 ± 0.3	1.0 ± 1.0
Pyramid	0.9 ± 0.3	1.0 ± 0.5	1.1 ± 0.7
Rectangular Prism	1.1 ± 0.4	1.1 ± 0.3	0.9 ± 0.4
Tetrahedron	0.9 ± 0.5	1.3 ± 0.8	3.2 ± 5.8
Triangular Prism	0.9 ± 0.4	0.9 ± 0.3	1.6 ± 3.0

Figure 2.4: Localization error for various objects, cm.

Localization accuracy

To evaluate the accuracy of the learned detectors in the real world, we captured 480 webcam images of one or more geometric objects on a table at a distance of 70 cm to 105 cm from the camera. The camera position remains constant across all images. We did not control for lighting or the rest of the scene around the table (e.g., all images contain part of the robot and tape and wires on the floor). We measured ground truth positions by aligning the object of interest on a 1 millimeter grid on the tabletop. Note that the grid may add up to 1 mm of error, so we only report up to that resolution. Each of the eight geometric objects has 60 labeled images in the dataset: 20 with the object alone on the table, 20 in which one or more distractor objects are present on the table, and 20 in which the object is partially occluded by another object.

Figure 2.4 summarizes the performance of our models on the test set. ur object detectors are able to localize objects to within 1.5 cm (on average) in the real world and perform well in the presence of clutter and partial occlusions. Though the accuracy of our trained detectors is promising, note that domain mismatch still causes worse performance than on the simulated training data, where error is around 0.3 cm.

Variability in detector performance

The localization accuracy reported in Figure 2.4 represents the the performance of the best hyperparameter setting. Figure 2.5 summarizes how performance of trained detectors varies with different seeds and hyperparameter settings. We used 2 seeds for each of the 6 hyperparemeter settings described in the previous section.

We evaluated each on the test set and a synthetic validation set consisting of the simulated scene with more realistic non-random textures. The first six columns of Table II show that the performance varies significantly between runs on both evaluation sets.

The last column reports the cosine similarity between the performance of all models on the test set and the synthetic validation set. The high similarity scores suggest that relative performance of a model on the synthetic validation set is a good proxy for relative performance on the test set. Choosing the best-performing model on the validation set may be a reasonable strategy for finding the best-performing model in the real world.

	Synthetic validation images			Real images			Cos
	Best	Worst	Avg	Best	Worst	Avg	Sim
Cone	0.3	12.5	2.3 ± 3.9	0.9	14.3	3.4 ± 4.1	0.98
Cube	0.2	12.7	1.5 ± 3.5	1.5	11.4	2.9 ± 2.7	0.91
Cylinder	0.3	12.5	1.9 ± 3.5	1.8	14.7	3.4 ± 3.8	0.95
Hex. Prism	0.4	12.6	1.9 ± 3.5	0.8	14.8	2.5 ± 3.6	0.98
Pyramid	0.3	12.5	1.5 ± 3.1	1.3	14.2	3.0 ± 3.2	0.93
Rect. Prism	0.5	12.8	2.1 ± 3.7	1.0	12.5	2.6 ± 3.0	0.96
Tetrahedron	0.3	12.8	3.0 ± 4.7	1.8	14.6	4.7 ± 4.7	0.97
Tri. Prism	0.3	12.9	2.9 ± 4.4	1.1	14.6	4.1 ± 4.7	0.98

Figure 2.5: Variability in model performance across hyperparameters and seeds.

In addition to performance variability between seeds and hyperparameters, we looked at the performance variability for the single best detector for each object. The results are summarized in Figures 2.6 and 2.7.

Figure 2.6 summarizes the prediction error on all datapoints in the test set. The mode of each error distribution is similar, suggesting that the difference in performance between objects is largely caused by outliers like not finding the object in the image and predicting the position of the wrong object.

Figure 2.7 shows how performance varies with the position of the object. Errors are more concentrated toward the outside of the table, but are not consistent across object types.

To evaluate the performance of the models on corner cases, we tested create synthetic test images in which (a) the object of interest is not present in the scene, or (b) there are multiple copies of the object of interest on the table. In both cases, the model performed as expected – in the first case, it estimated the position of the object to be the mean of the

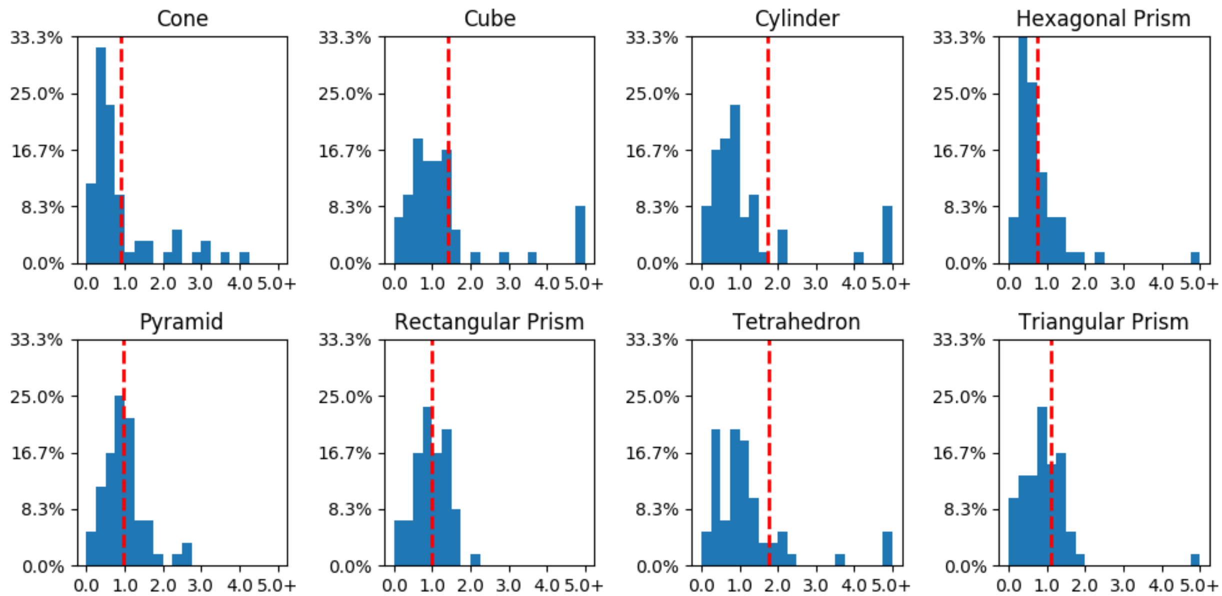

Figure 2.6: Error distributions for the best models. The x-axis corresponds to the error (in centimeters), and the y-axis corresponds to the percentage of data points within each bucket. The red line represents the mean performance.

training images (approximately the center of the table), and in the second case it predicted the object was at the mean position of all the copies of the object of interest.

Performance on non-uniform textures

To examine how our detectors would perform on objects with complex textures, we created simulated scenes in which the object of interest is given a texture from the Describable Textures Dataset (DTD) [21]. Our synthetic test set consists of 7,000 total images using 5 random textures from each of the 35 categories in the DTD. We also gave realistic textures to the table, floor, and background (wood, marble, and a warehouse background respectively). We compared the performance on this dataset to a baseline consisting of scenes with realistically textured table, floor, and background but a uniformly textured object of interest.

The results in Figure 2.4 show that the model performs comparably when tested on objects with complex, non-uniform textures, suggesting that it has learned a representation invariant to the texture of the object of interest.

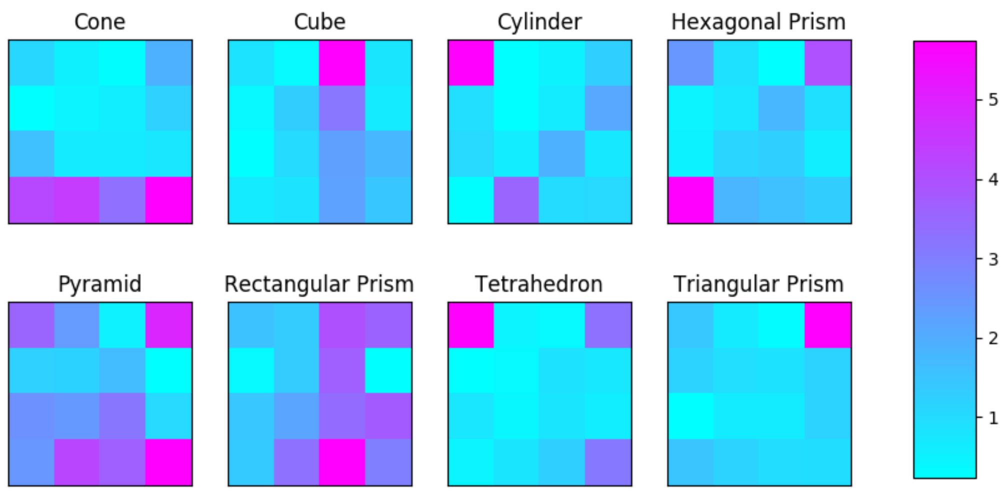

Figure 2.7: Error distribution by object position. We divided the table into 4 horizontal and 4 vertical buckets. The bottom-left of each heatmap corresponds to the front-left corner of the table in the camera frame. The scale on the right represents average error in centimeters.

Detection error for various objects, cm		
	Complex textures	Baseline
Cone	0.23 ± 0.09	0.30 ± 0.14
Cube	1.63 ± 0.37	0.64 ± 1.37
Cylinder	0.30 ± 0.06	0.34 ± 0.18
Hexagonal Prism	0.46 ± 0.26	0.37 ± 0.25
Pyramid	0.21 ± 0.11	0.30 ± 0.19
Rectangular Prism	0.5 ± 0.26	0.51 ± 0.69
Tetrahedron	0.32 ± 0.15	0.32 ± 0.19
Triangular Prism	0.27 ± 0.09	0.29 ± 0.16

Figure 2.8: Performance of trained models on simulated data with complex textures from the Describable Textures Dataset. The baseline uses uniformly textured objects.

Comparison to existing methods

The accuracy of our detectors are comparable at a similar distance to the translation error in traditional techniques for pose estimation in clutter from a single monocular camera frame [23] that use higher-resolution images.

The primary advantage of our technique over existing methods is that it uses a simple-to-implement, scalable pipeline that may be easy to extend to harder problems. However, our technique has several limitations relative to existing approaches. The primary drawback of our method is that it requires setting up simulated scenes and training separate detectors for each object, including creating 3D models for each object. Once trained, the models may not generalize well to new scenes (e.g., new positions of the table relative to the robot or large changes of the position of the camera relative to the table). Finally, since our approach is based on deep neural networks, failure cases can be extreme (e.g., greater than 15 cm error) and difficult to interpret.

Ablation study

To evaluate the importance of different factors of our training methodology, we assessed the sensitivity of the algorithm to the number of training images, the number of unique textures seen in training, the use of random noise in pre-processing, the presence of distractors in training, the randomization of camera position in training, and the use of pre-trained weights in the detection model.

We found that the method is at least somewhat sensitive to all of the factors except the use of random noise.

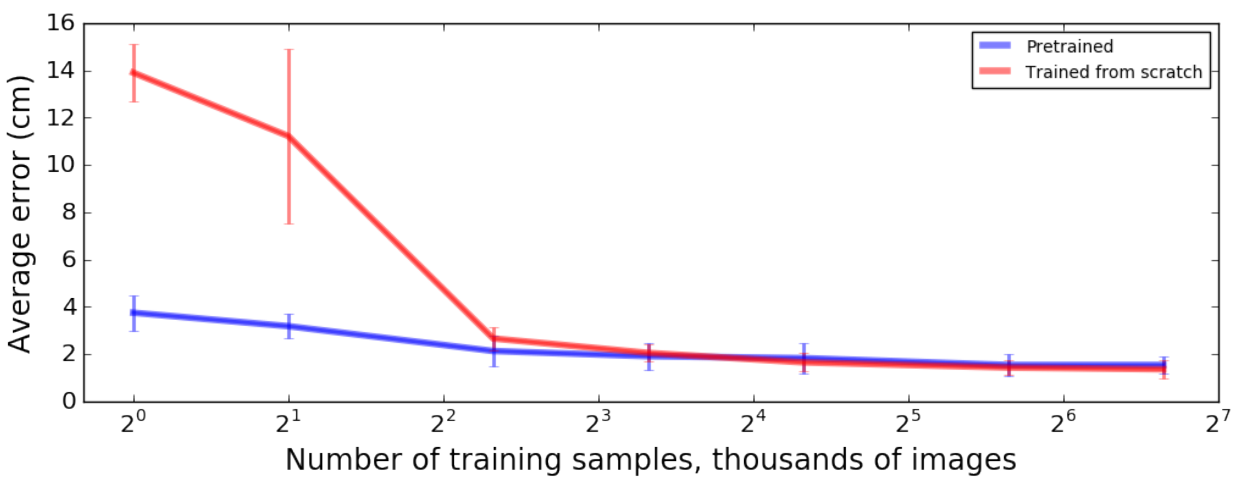

Figure 2.9: Sensitivity of test error on real images to the number of simulated training examples used. Each training example corresponds to a single labeled example of an object on the table with between 0 and 10 distractor objects. Lighting and all textures are randomized between iterations.

Figure 2.9 shows the sensitivity to the number of training samples used for pre-trained models and models trained from scratch. Using a pre-trained model, we are able to achieve relatively accurate real-world detection performance with as few as $5,000$ training samples, but performance improves up to around $50,000$ samples.

Figure 2.9 shows the performance of a model trained from scratch. Our hypothesis that pre-training would be essential to generalizing to the real world proved to be false. With a large amount of training data, random weight initialization can achieve nearly the same performance as does pre-trained weight initialization. The best detectors for a given object were often those initialized with random weights. However, using a pre-trained model can significantly improve performance when less training data is used.

Figure 2.10 shows the sensitivity to the number of unique texturizations of the scene when trained on a fixed number (10, 000) of training examples. We found that performance degrades significantly when fewer than 1, 000 textures are used, indicating that for our experiments, using a large number of random textures (in addition to random distractors and object positions) is necessary to achieving transfer. Note that when 1, 000 random textures are used in training, the performance using 10, 000 images is comparable to that of using only 1, 000 images, indicating that in the low data regime, texture randomization is more important than randomization of object positions.

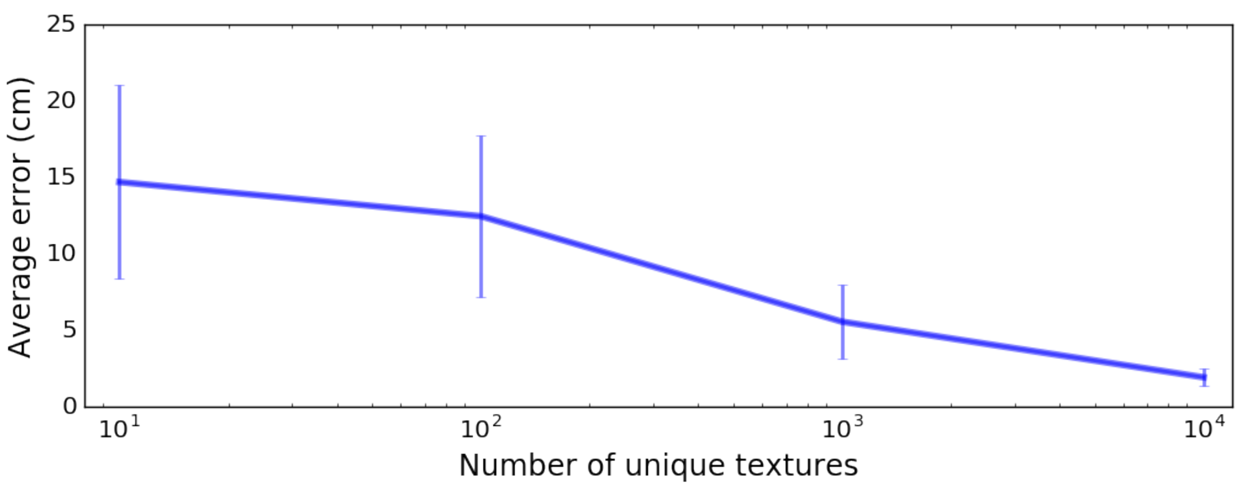

Figure 2.10: Sensitivity to amount of texture randomization. In each case, the detector was trained using 10, 000 random object positions and combinations of distractors, but only the given number of unique texturizations and lighting conditions were used.

Figure 2.11 examines the performance of the algorithm when random noise, distractors, and camera randomization are removed in training. Incorporating distractors during training appears to be critical to resilience to distractors in the real world. Randomizing the position of the camera also consistently provides a slight accuracy boost, but reasonably high accuracy is achievable without it. Adding noise during pretraining appears to have a negligible effect. In practice, we found that adding a small amount of random noise to images at training time improves convergence and makes training less susceptible to local minima.

Evaluation type	Real images		
	Object only	Distractors	Occlusions
Full method	1.3 ± 0.6	1.8 ± 1.7	2.4 ± 3.0
No noise added	1.4 ± 0.7	1.9 ± 2.0	2.4 ± 2.8
No camera randomization	2.0 ± 2.1	2.4 ± 2.3	2.9 ± 3.5
No distractors in training	1.5 ± 0.6	7.2 ± 4.5	7.4 ± 5.3

Figure 2.11: Ablation study. Localization error (in centimeters) of object detectors trained on 20,000 examples from different dataset configurations.

Robotics experiments

To demonstrate the potential of this technique for transferring robotic behaviors learned in simulation to the real world, we evaluated the use of our object detection networks for localizing an object in clutter and performing a prescribed grasp. For two of our most consistently accurate detectors, we evaluated the ability to pick up the detected object in 20 increasingly cluttered scenes using the positions estimated by the detector and off-the-shelf motion planning software [188]. To test the robustness of our method to discrepancies in object distributions between training and test time, some of our test images contain distractors placed at orientations not seen during training.

We deployed the pipeline on a Fetch robot [220], and found it was able to successfully pick up the target object in 38 out of 40 trials, including in highly cluttered scenes with significant occlusion of the target object. Note that the trained detectors have no prior information about the color of the target object, only its shape and size, and are able to detect objects placed closely to other objects of the same color.

To test the performance of our object detectors on objects with non-uniform textures, we trained a detector to localize a can of Spam from the YCB Dataset [18]. At test time, instead of using geometric object distractors like in training, we placed other food items from the YCB set on the table. The detector was able to ignore the previously unseen distractors and pick up the target in 9 of 10 trials.

Figure 2.12 shows examples of the robot grasping trials. For videos, please visit the web page associated with this paper.[1]

2.5 Conclusion

We demonstrated that an object detector trained only in simulation can achieve high enough accuracy in the real world to perform grasping in clutter. Future work will explore how to make this technique reliable and effective enough to perform tasks that require contact-rich manipulation or higher precision.

[1] `https://sites.google.com/view/domainrandomization/`

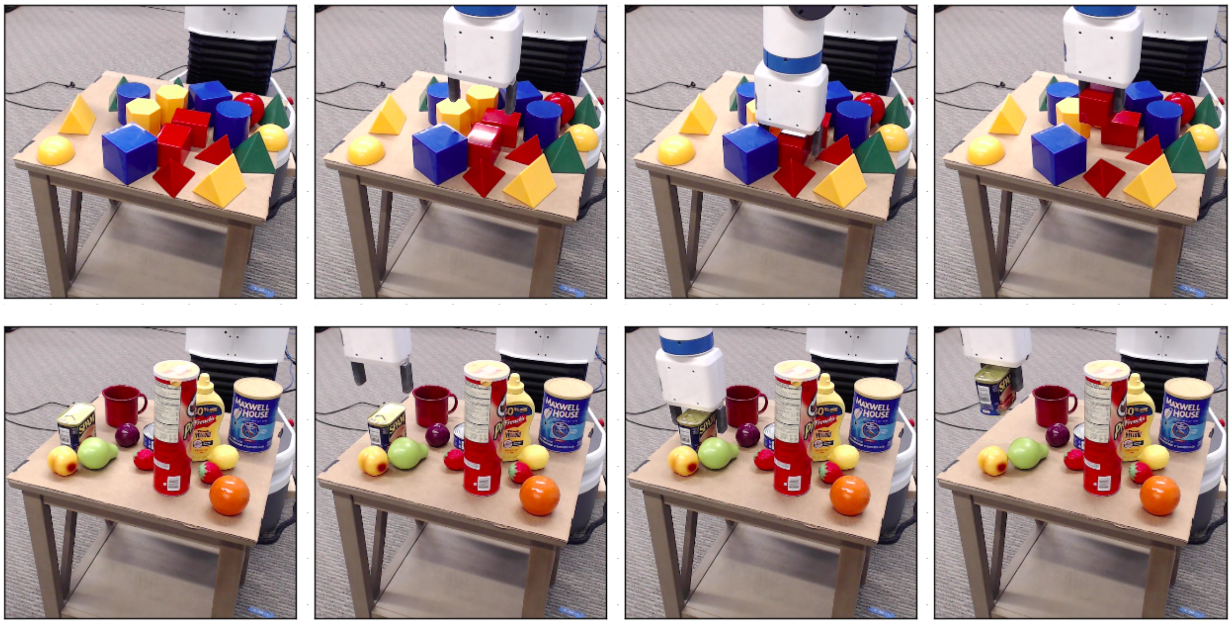

Figure 2.12: Two representative executions of grasping objects using vision learned in simulation only. The object detector network estimates the positions of the object of interest, and then a motion planner plans a simple sequence of motions to grasp the object at that location.

Future directions that could improve the accuracy of object detectors trained using domain randomization include using higher resolution camera frames, optimizing model architecture choice, introducing additional forms of texture, lighting, and rendering randomization to the simulation, training on more data, incorporating multiple camera viewpoints, stereo vision, or depth information, and combining domain randomization with domain adaptation.

Domain randomization is a promising research direction toward bridging the reality gap for robotic behaviors learned in simulation. Deep reinforcement learning may allow more complex policies to be learned in simulation through large-scale exploration and optimization, and domain randomization could be an important tool for making such policies useful on real robots.

Chapter 3

Domain Randomization for Grasping

3.1 Introduction

Robotic grasping remains one of the core unsolved problems in manipulation. The earliest robotic grasping methods used analytical knowledge of a scene to compute an optimal grasp for an object [132, 9, 135, 149, 159, 174]. Assuming a contact model and a heuristic for the likelihood of success of a grasp, analytical methods can provide guarantees about grasp quality, but they often fail in the real world due to inconsistencies in the simplified object and contact models, the need for accurate 3D models of the objects in question, and sensor inaccuracies [10].

As a result, significant research attention has been given to data-driven grasp synthesis methods [10, 120, 141, 52, 126, 125, 171]. These algorithms avoid some of the challenges of analytic methods by sampling potential grasps and ranking them according to a learned function that maps sensor inputs to an estimate of a chosen heuristic.

Recently, several works have explored using deep neural networks to approximate the grasp heuristic function [107, 144, 114, 81]. The promise of deep neural networks for learning grasp heuristics is that with diverse training data, deep models can learn features that deal with the edge cases that make real-world grasping challenging.

A core challenge for deep learning grasp quality heuristics is data availability. Due to the difficulty and expense of collecting real-world data and due to the limited availability of high-quality 3D object meshes, current approaches use as few as hundreds or thousands of unique object instances, which may limit generalization. In contrast, ImageNet [93], the standard benchmark for image classification, has about 15M unique images from 22K categories.

In order to increase the availability of training data in simulation, we explore applying the idea of *domain randomization* [168, 200] to the creation of 3D object meshes. Domain randomization is a technique for learning models that work in a test domain after only training on low-fidelity simulated data by randomizing all non-essential aspects of the simulator. One of the core hypotheses of this work is that by training on a wide enough variety of unrealistic procedurally generated object meshes, our learned models will generalize to realistic objects.

Previous work in deep learning for grasping has focused on learning a function that estimates the quality of a given grasp given observations of the scene. Choosing grasps on which to perform this estimate has received comparatively little attention. Grasps are typically chosen using random sampling or by solving a small optimization problem online. The second goal of this chapter is to propose a deep learning-based method for choosing grasps to evaluate. Our hypothesis is that a learned model for grasp sampling will be more likely to find high-quality grasps for challenging objects and will do so more efficiently.

We use an autoregressive model architecture [98, 137, 138] that maps sensor inputs to a probability distribution over grasps that corresponds to the model's weighted estimate of the likelihood of success of each grasp. After training, highest probability grasp according to the distribution succeeds on 89% of test objects and the 20 highest probability grasps contain a successful grasp for 96% of test objects. In order to determine which grasp to execute on the robot, we collect a second observation in the form of an image from the robot's hand camera and train a second model to choose the most promising grasp among those sampled from the autoregressive model, resulting in a success rate of 92%.

The contributions of this chapter can be summarized as follows:

- We explore the effect of training a model for grasping using unrealistic procedurally generated objects and show that such a model can achieve similar success to one trained on a realistic object distribution. (Another paper [14] developed concurrently to this one explored a similar idea and reached similar conclusions.)
- We propose a novel generative model architecture and training methodology for learning a sampling distribution for grasps to evaluate.
- We evaluate our object generation, training, and sampling algorithms in simulated scenes and find that we can achieve an 84% success rate on random objects and 92% success rate on previously unseen real-world objects despite training only on non-realistic randomly generated objects.
- We demonstrate that we can deploy these models in real-world grasping experiments with an 80% success rate despite having been trained entirely in simulation.

3.2 Related Work

Domain Randomization

Domain randomization involves randomizing non-essential aspects of the training distribution in order to better generalize to a difficult-to-model test distribution. This idea has been employed in robotics since at least 1997, when Jakobi proposed the "Radical Envelope of Noise Hypothesis", the idea that evolved controllers can be made more robust by completely randomizing all aspects of the simulator that do not have a basis in reality and slightly randomizing all aspects of the simulator that do have a basis in reality [73]. Recently domain randomization has shown promise in transferring deep neural networks for robotics tasks from simulation to the real world by randomizing physics [127] and appearance properties [168, 200, 238].

In another work developed concurrently with this one [14], the authors reach a similar conclusion about the utility of procedurally generated objects for the purpose of robotic grasping. In contrast to this work, theirs focuses on how to combine simulated data with real grasping data to achieve successful transfer to the real world, but does not focus on achieving a high overall success rate. Our work instead focuses on how to achieve the best possible generalization to novel objects. Ours also has a comparable real-world success rate despite not using any real-world training data.

Autoregressive models

This work uses an autoregressive architecture to model a distribution over grasps conditioned on observations of an object. Autoregressive models leverage the fact that an N-dimensional probability distribution $p(X)$ can be factored as $\prod_{n=1}^{N} p(x_n \mid x_1, \cdots, x_{n-1})$ for any choice of ordering of 1-dimensional variables x_i. The task of modeling the distribution then consists of modeling each $p(x_n \mid x_1, \cdots, x_{n-1})$ [98]. In contrast to Generative Adversarial Networks [54], another popular form for a deep generative model, autoregressive models can directly compute the likelihood of samples, which is advantageous for tasks like grasping in which finding the highest likelihood samples is important. Autoregressive models have been used for density estimation and generative modeling in image domains [98, 57, 49] and have been shown to perform favorably on challenging image datasets like ImageNet [137, 139]. Autoregressive models have also been successfully applied to other forms of data including in topic modeling [97] and audio generation [138].

Robotic grasping

Grasp planning methods fall into one of two categories: *analytical* methods and *empirical* methods [169].

Analytical methods use a contact model and knowledge of an object's 3D shape to find grasps that maximize a chosen metric like the ability of the grasp to resist external wrenches [149] or constrain the object's motion [159]. Some methods attempt to make these estimates more robust to gripper and object pose uncertainty and sensor error by instead maximizing the expected value of a metric under uncertainty [83, 216].

Most approaches use simplified Coulomb friction and rigid body modeling for computational tractability [132, 149], but some have explored more realistic object and friction models [9, 150, 161]. Typically, grasps for an object are selected based on sensor data by registering images to a known database of 3D models using a traditional computer vision pipeline [22, 67, 227] or a deep neural network [60, 236].

Empirical methods instead attempt to maximize the value of a quality metric through sampling [169]. Many approaches use simulation to evaluate classical grasp quality metrics [120, 141, 52, 119]. Others use human labeling or self-supervised learning to measure success [126, 125, 171].

Most techniques estimate the value of the quality metric in real-world trials using traditional computer vision and learning techniques [172, 99, 171, 141]. More recently, deep learning has been employed to learn a mapping directly from sensor observations to grasp quality or motor torques.

Deep learning for robotic grasping

Work in deep learning for grasping can be categorized by how training data is collected and how the model transforms noisy observations into grasp candidates.

Some approaches use hand-annotated real-world grasping trials to provide training labels [105]. However, hand-labeling is challenging to scale to large datasets. To alleviate this problem, some work explores automated large-scale data collection [107, 144]. Others have explored replacing real data with synthetic depth data at training time [114, 115, 213, 81], or combining synthetic RGB images with real images [14]. In many cases, simulated data appears to be effective in replacing or supplementing real-world data in robotic grasping. Unlike our approach, previous work using synthetic data uses small datasets of up to a few thousand of realistic object meshes.

One commonly used method for sampling grasps is to learn a visuomotor control policy for the robot that allows it to iteratively refine its grasp target as it takes steps in the environment. Levine and co-authors learn a prediction network $g(I_t, v_t)$ that takes an observation I_t and motor command v_t and outputs a predicted probability of a successful grasp if v_t is executed [107]. The cross-entropy method is used to greedily select the v_t that maximizes g. Viereck and co-authors instead learn a function $d(I_t, v_t)$ that maps the current observation and an action to an estimate of the distance to the nearest successful grasp after performing v_t [213]. Directions are sampled and a constant step size is taken in the direction with the minimum value for d. In contrast to visuomotor control strategies, planning approaches like ours avoid the local optima of greedy execution.

Another strategy to choose a grasp using a deep learning is to sample grasps and score them using a deep neural network of the form $f(I, \boldsymbol{g}) \to s$, where I are the observation(s) of the scene, $\boldsymbol{g}$ is a selected grasp, and s is the score for the selected grasp [114, 115, 81, 229]. These techniques differ in terms of how they sample grasps to evaluate at test time. Most commonly they directly optimize g using the cross-entropy method [114, 115, 229]. In contrast to these approaches, our approach jointly learns a grasp scoring function and a sampling distribution, allowing for efficient sampling and avoiding exploitation by the optimization procedure of under- or over-fit regions of the grasp score function.

Other approaches take a multi-step approach, starting with a coarse representation of the possible grasps for an object and then exhaustively searching using a learned heuristic [105] or modeling the score function jointly for all possible coarse grasps [81]. Once a coarse grasp is sampled, it is then fine-tuned using a separate network [105] or interpolation [81]. By using an autoregressive model architecture, we are able to directly learn a high- dimensional (20^4 or 20^6-dimensional) multimodal probability distribution.

3.3 Methods

Our goal is to learn a mapping that takes one or more observations $I = \{I_j\}$ of a scene and outputs a grasp $\boldsymbol{g}$ to attempt in the scene. The remainder of the section describes the data generation pipeline, model architecture, and training procedure used in our method.

Data collection

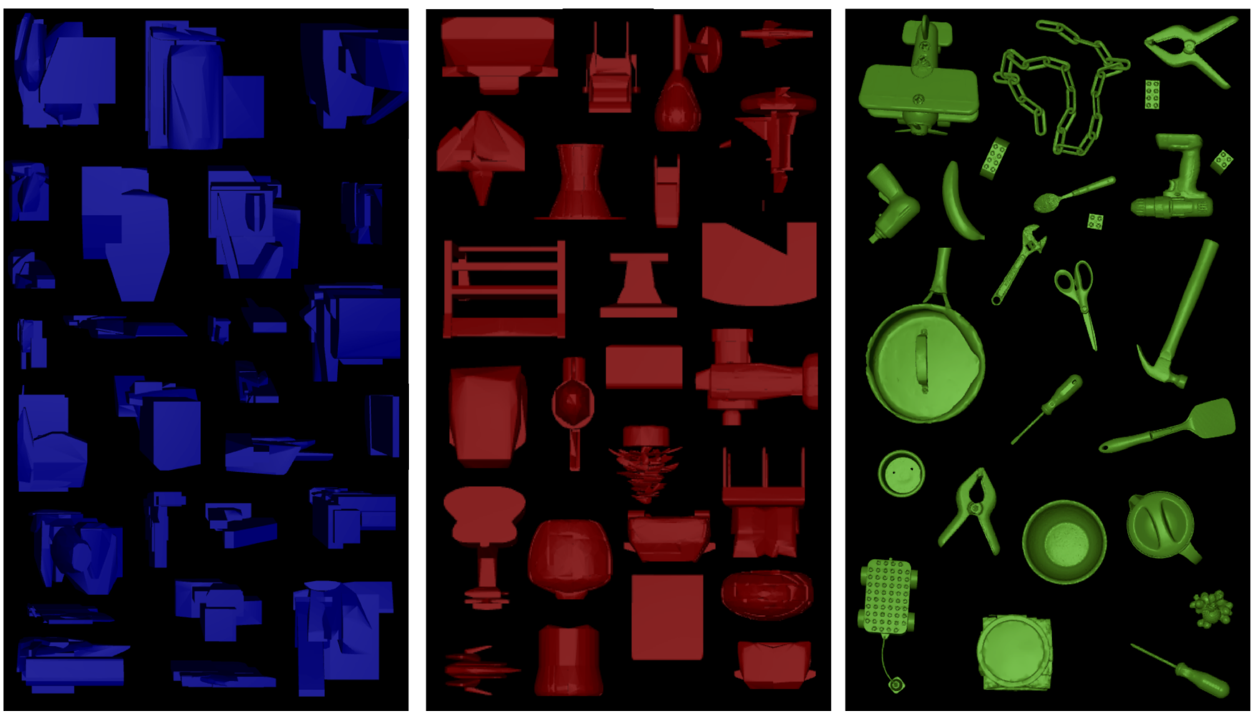

Figure 3.1: Examples of objects used in our experiments. Left: procedurally generated random objects. Middle: objects from the ShapeNet object dataset. Right: objects from the YCB object dataset.

We will first describe the process of generating training objects, and then the process of sampling grasps for those objects.

Object generation

One of our core hypotheses is that training on a diverse array of procedurally generated objects can produce comparable performance to training on realistic object meshes. Our procedurally generated objects were formed as follows:

1. Sample a random number $n_p \in \{1, \cdots 15\}$
2. Sample n_p primitive meshes from our object primitive dataset
3. Randomly scale each primitive so that all dimensions are between 1 and 15cm
4. Place the meshes sequentially so that each mesh intersects with at least one of the preceding meshes
5. Rescale the final object to approximate the size distribution observed in our real object dataset

To build a diverse object primitive dataset, we took the more than 40,000 object meshes found in the ShapeNet object dataset [19] and decomposed them into more than 400,000 convex parts using V-HACD[1]. Each primitive is one convex part.

We compared this object generation procedure against a baseline of training using rescaled ShapeNet objects.

Grasp sampling and evaluation

We sample grasps uniformly at random from a discretized 4- or 6-dimensional grasp space (4-dimensional when attention is restricted to upright grasps) corresponding to the (x, y, z) coordinates of the center of the gripper and an orientation of the gripper about that point. We discretize each dimension into 20 buckets. The buckets are the relative location of the grasp point within the bounding box of the object – e.g., an x-value of 0 corresponds to a grasp at the far left side of the object's bounding box and a coordinate of 19 corresponds to a grasp at the far right.

Grasps that penetrate the object or for which the gripper would not contact the object when closed can be instantly rejected. The remainder are evaluated in a physics simulator.

For each grasp attempt, we also collect a depth image from the robot's hand camera during the approach to be used to train the grasp evaluation function.

Model architecture

The model architecture used for our experiments is outlined in Figure 3.2. The model consists of two separate neural networks – a grasp planning module $\gamma(I) = \beta \circ \alpha(I)$ and a grasp evaluation model f. The grasp planning module is used to sample grasps that are

[1] https://github.com/kmammou/v-hacd

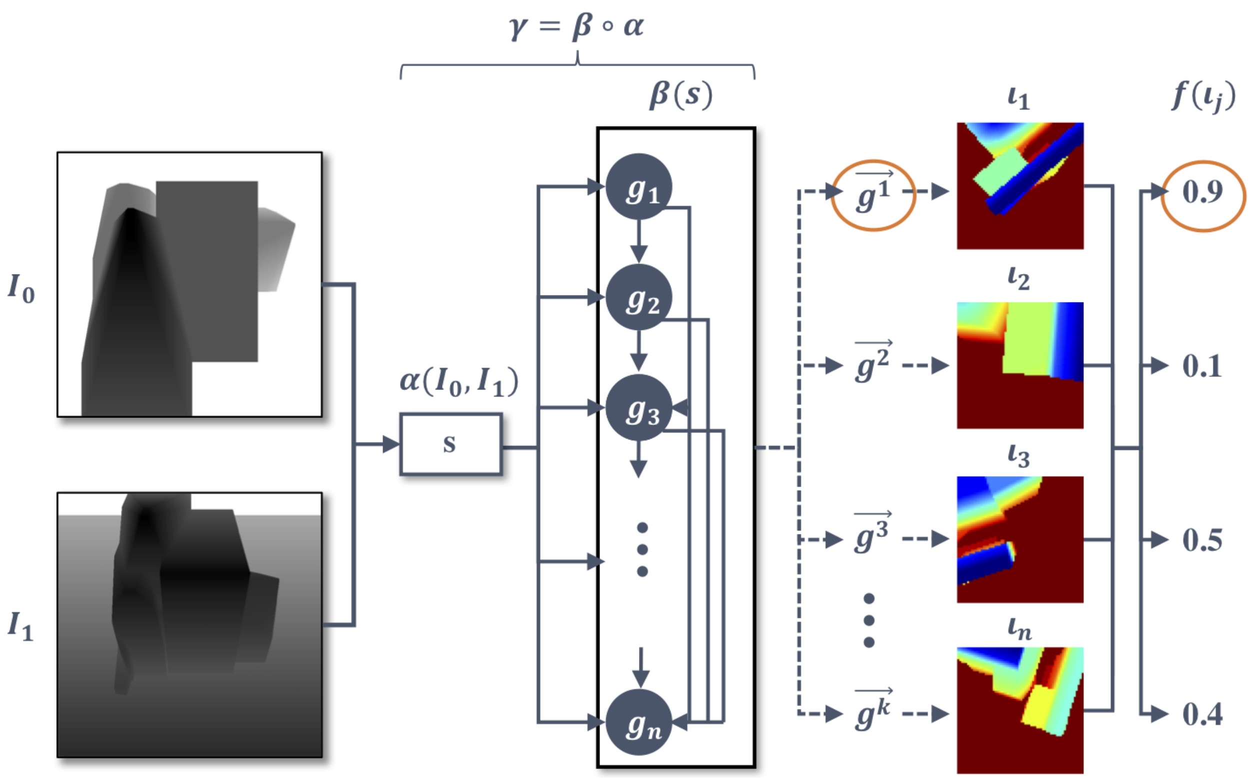

Figure 3.2: An overview of sampling from our model architecture. Solid lines represent neural networks, and dotted lines represent sampling operations. The model takes as input one or more observations of the target object in the form of depth images. The images are passed to an image representation module α, which maps the images to an embedding s. The embedding s is the input for the autoregressive module β, which outputs a distribution over possible grasps g for the object by modeling each dimension g_i of the grasp conditioned on the previous dimensions. We sample k high-likelihood grasps $g^1, \cdots, g^k$ from the model using a beam search. For each of those grasps, a second observation is captured that corresponds to an aligned image in the plane of the potential grasp. A grasp scoring model f maps each aligned image to a score. The grasp with the highest score is selected for execution on the robot.

likely to be successful. The grasp evaluation model takes advantage of more detailed data in the form of a close-up image from a camera on the gripper of the robot to form a more accurate estimate of the likelihood of each sampled grasp to be successful.

The image representation $s = \alpha(I)$ is formed by passing each image through a separate convolutional neural network. The flattened output of these convolutional layers are stacked and passed through several dense layers to produce s.

The neural network $\beta(s)$ models a probability distribution $p_\beta(\boldsymbol{g}|s)$ over possible grasps for the object that corresponds to the normalized probability of success of each grasp. The model β consists of n submodules β_i where n is the dimensionality of the grasp. For any grasp $\boldsymbol{g}$, β and $\{\beta_i\}$ are related by

$$\beta(s)(\boldsymbol{g}) = \prod_{i=1}^{n} \beta_i(g_1, \cdots, g_{i-1}, s),$$

where g_i are the dimensions of $\boldsymbol{g}$.

Each β_i is a small neural network taking s and $g_1, \cdots, g_{i-1}$ as input and outputting a softmax over the 20 possible values for g_i, the next dimension of g. We found that sharing weights between the β_i hurts performance at convergence.

The grasp evaluation model f takes as input a single observation from the hand camera of the robot and outputs a single scalar value corresponding to the likelihood of success of that grasp. The model f is parameterized by a convolutional neural network with sigmoid output.

Training methodology

Since our method involves capturing depth images from the hand of the robot corresponding to samples from $\gamma(I) = \beta \circ \alpha(I)$, the entire evaluation procedure is not differentiable and we cannot train the model end-to-end using supervised learning. As a result, our training procedure involves independently training γ and f.

Given datasets of objects $D = \{D_1, \cdots D_d\}$, observations $I = \{I^1, \cdots I^d\}$ and successful grasps $G = \{\boldsymbol{g}_1^1, \cdots \boldsymbol{g}_{m_1}^1, \cdots \boldsymbol{g}_1^2, \cdots \boldsymbol{g}_{m_d}^d\}$, γ can be optimized by minimizing the negative log-likelihood of G conditioned on the observations I with respect to the parameters θ of γ, which is given by:

$$J(\theta) = -\frac{1}{d}\sum_{i=1}^{d} \frac{1}{m_i}\sum_{j=1}^{m_i} \log p_\theta(\boldsymbol{g}_j^i \mid I^i).$$

This can be decomposed as

$$-\frac{1}{d}\sum_{i=1}^{d} \frac{1}{m_i}\sum_{j=1}^{m_i}\sum_{k=1}^{n} \log \beta_k \left(\alpha(I^i)\right) \left((\boldsymbol{g}_l^i)_{l<k}\right).$$

This function can be optimized using standard backpropogation and minibatch SGD techniques. [98]

In practice, α is usually a larger model than β and there are often tens or hundreds of successful grasp attempts $\{\boldsymbol{g}_j^i\}$ for a single observation I^i. Therefore it is computationally advantageous to perform the forward pass and gradient calculations for each $\alpha(I^i)$ once for all $\{\boldsymbol{g}_j^i\}$.

This can be achieved in standard neural network and backpropagation libraries by stacking all grasps for a given object so that SGD minibatches consist of pairs $(I_i, \boldsymbol{g}_{\{j\}}^i)$ where

$\boldsymbol{g}^i_{\{j\}}$ is the $m_i \times n$ matrix consisting of all successful grasps for object i. To deal with differing values for m_i, we choose $m = \max(\{m_i\})$ and form an $m \times n$ matrix by padding the $m_i \times n$ matrix with arbitrary values. We can then write the gradient $\nabla_\theta J(\theta)$ of our objective function as follows:

$$-\frac{1}{d}\sum_{i=1}^{d}\frac{1}{m_i}\nabla_\theta \alpha(I^i)\sum_{j=1}^{m} 1_j \sum_{k=1}^{n} \nabla_\theta \log \beta_k \left(\alpha(I^i)\right)\left((\boldsymbol{g}^i_l)_{l<k}\right)$$

where 1_j is an indicator function corresponding to whether the entry in j was one of the m_i successful grasps.

This form of the gradient allows us to compute $\nabla_\theta \alpha(I^i)$ once for each I^i, which we found to increase training speed by more than a factor of 10 in our experiments.

The grasp evaluation function f is trained using supervised learning. Inputs are the hand camera images collected during the data collection process and labels are an indicator function corresponding to whether the grasp was successful.

3.4 Experiments

The goal of our experiments is to answer the following questions:

1. Can grasping models trained on unrealistic randomly generated objects perform as well on novel realistic objects as those trained on realistic objects?

2. How efficiently can we sample grasps from our proposed autoregressive model architecture?

3. How important is using a large number of unique objects for training grasping models?

4. How well do models trained using our methodology work in the real world?

Experimental setup

We evaluated our approach by training grasping models on three datasets:

- ShapeNet-1M, a dataset of 1 Million scenes with a single object from the ShapeNet dataset with randomized orientation and object scale.

- Random-1M, a dataset of 1 Million scenes with a single object generated at random using the procedure above.

- ShapeNet-Random-1M, a dataset with 500,000 scenes from each of the previous datasets.

For each object set, we recorded 2,000 grasp attempts per object. This number of grasps was selected so that more than 95% of objects sampled had at least one successful grasp to avoid biasing the dataset to easier objects[2]. For data generation, we used a disembodied Fetch gripper to improve execution speed.

We trained the models using the Adam optimizer [87] with a learning rate of 10^{-4}. We trained each model using three random seeds, and report the average of the three seeds unless otherwise noted.

We evaluated the model on 300 training and hold-out scenes from ShapeNet-1M and Random-1M, as well as 300 scenes generated from the 75 YCB objects with meshes capable of being grasped by our robot's gripper.

All executions were done using a Fetch robot in the MuJoCo physics simulator [201]. When evaluating the model, we sampled $k = 20$ likely grasps from the model using a beam search with beam width 20. Among these grasps, only the one with the highest score according to f is attempted on the robot. An attempt is considered successful if the robot is able to use this grasp to lift the object by 30cm.

Performance using randomly generated training data

Figure 3.3 describes the overall success rate of the algorithm on previously seen and unseen data. The full version of our algorithm is able to achieve greater than 90% success rate on previously unseen YCB objects even when training entirely on randomly generated objects. Training on 1M random objects performs comparably to training on 1M instances of realistic objects.

Training set	ShapeNet Train	ShapeNet Test	Random Train	Random Test	Ycb
ShapeNet-1M	0.91	0.91	0.72	0.71	0.93
Random-1M	0.91	0.89	0.86	0.84	0.92
ShapeNet-Random-1M	0.92	0.90	0.84	0.81	0.92

Figure 3.3: Performance of the algorithm on different synthetic test sets. The full algorithm is able to achieve at least 90% success on previously unseen objects from the YCB dataset when trained on any of the three training sets.

Note that these results were achieved by limiting the robot to grasps in which the gripper is upright. The success rate is around 10% lower across experiments when using full 6-dimensional grasps. Further experimentation could look into whether significantly scaling the amount of training data or using a combination of the 4-dimensional training data and 6-dimensional training data could improve performance.

[2]This number is not closer to 100% because a small percentage of the random objects in our training set are un-graspable with the Fetch gripper.

Figure 3.4 compares the success rate of our algorithm to several baselines. In particular, our full method performs significantly better than sampling the highest likelihood grasp from the autoregressive model alone.

Training set	ShapeNet Train	ShapeNet Test	Random Train	Random Test	Ycb
Full Algorithm	0.91	0.89	0.86	0.84	0.92
Autoregressive-Only	0.89	0.86	0.80	0.76	0.89
Random	0.22	0.21	0.10	0.11	0.26
Centroid	0.30	0.25	0.10	0.12	0.54

Figure 3.4: Performance of the algorithm compared to baseline approaches. The Full Algorithm and Autoregressive-Only numbers reported are using models trained on random data. The Autoregressive-Only baseline uses the model γ to sample a single high-likelihood grasp, and executes that grasp directly without evaluating it with the model f. The Random baseline samples a random grasp. The centroid baseline deterministically attempts to grasps the center of mass of the object, with the approach angle sampled randomly.

We observed the following three main failure cases for the learned model:

1. For objects that are close to the maximum size graspable by the gripper (10cm), the grasp chosen sometimes collides with the object.
2. In some cases, the model chooses to grasp a curved object at a narrower point to avoid wider points that may cause collision, and as a result the gripper slips off.
3. The model cannot find reasonable grasp candidates for some highly irregular objects like a chain found in the YCB dataset.

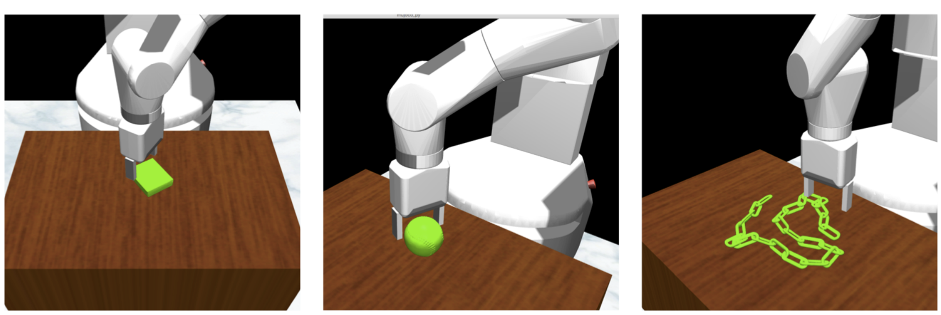

Figure 3.5: Examples of three observed failure cases for our learned models. Left: the model chooses a grasp for an object that is close to the maximum size of the gripper that collides with the object. Middle: the model chooses a grasp for a large, highly curved object that causes it to slip off. Right: the model fails to find a promising grasp for a highly irregular object.

The learned models primarily failed on objects for which all features are close in size to the maximum allowed by the gripper. Supplementing the training set with additional objects on the edge of graspability or combining planning with visual servoing could alleviate these cases.

The model also failed for a small number of highly irregular objects like a chain present in the YCB dataset. These failure cases present a larger challenge for the use of random objects in grasping, but additional diversity in the random generation pipeline may mitigate the issue.

Efficiency of the autoregressive model

To test how efficiently we are able to sample grasps from our autoregressive model, we looked at the percentage of objects for which the top k most likely grasps according to γ contain at least one successful grasp. Figure 3.6 shows that the most likely grasp according to the model succeeds close to 90% of the time on YCB objects, and the incremental likelihood of choosing a valid grasp saturates between 10 and 20 samples, motivating our choice of 20 for k. Note that more objects have successful grasps among the 20 sampled than achieve success using our method, suggesting the performance of the grasp evaluator f could be a bottleneck to the overall performance of the algorithm.

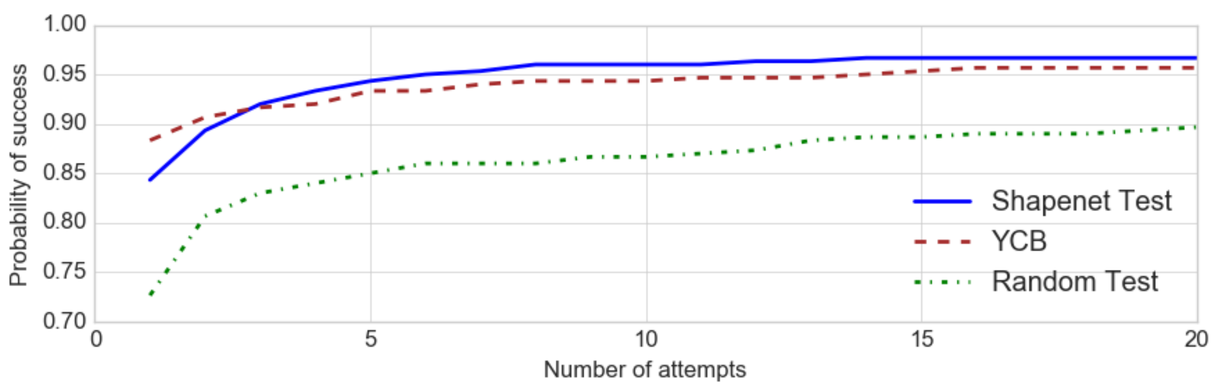

Figure 3.6: Percentage of objects that a trained model can grasp successfully as a function of number of the number of samples from the autoregressive model attempted. Here, we sample the 20 grasps from the autoregressive model that have the highest likelihood according to a beam search and count the number of times success occurred on the nth attempt for $n < 20$.

Effect of amount of training data

Figure 3.7 shows the impact of the number of unique objects in the training set on the performance of our models in validation data held out from the same distribution and out-of-sample test data from the YCB dataset. Although with enough data the model trained entirely using randomly generated data performs as well as the models trained using realistic data, with smaller training sets the more realistic object distributions perform significantly better on the test set than does the unrealistic random object distribution.

Note that performance does not appear to have saturated yet in these examples. We conjecture that more training data and more training data diversity could help reduce the effects of the first two failure cases above, but may not allow the model to overcome the third failure case.

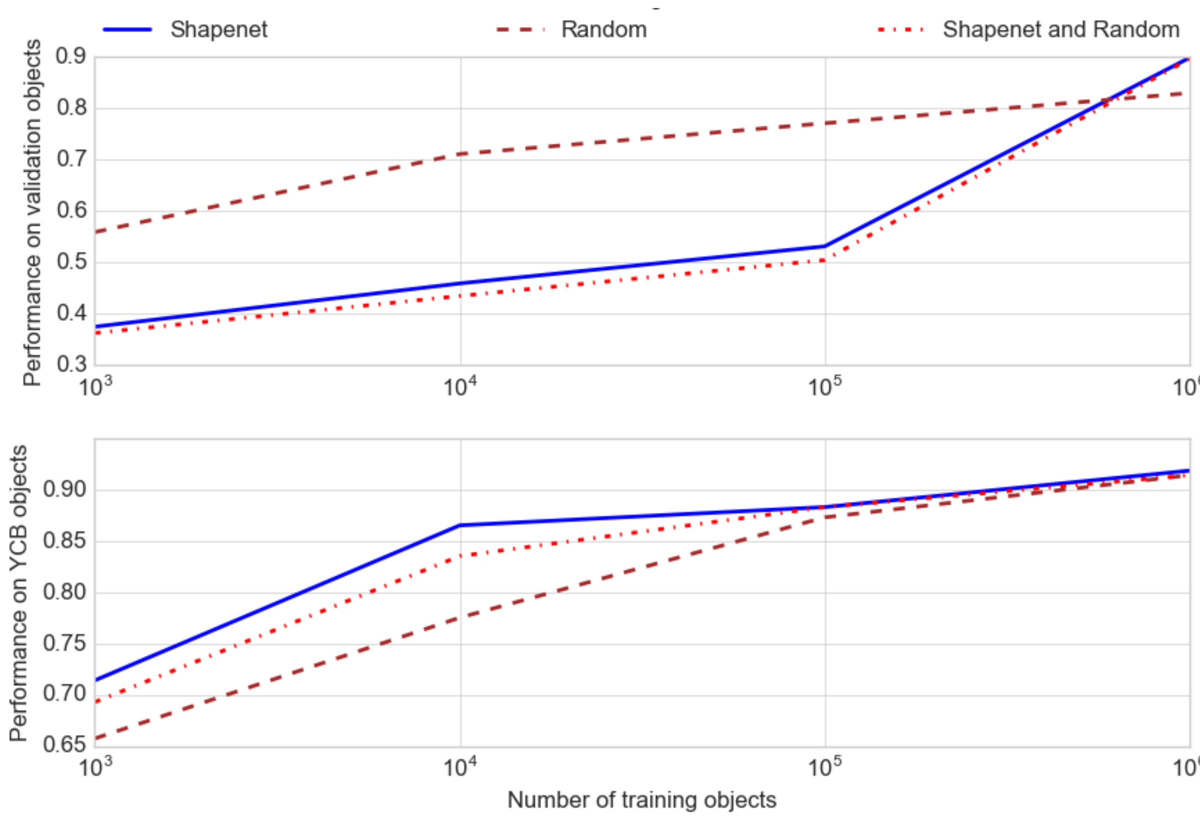

Figure 3.7: Impact of number of unique training objects used on the performance of the learned model on held out data at test time. Each line represents a different training set. The top chart indicates the performance of the models on held out data from the same distribution (i.e., held out ShapeNet or Random objects depending on the training set), and the bottom chart shows performance on out-of-sample objects from the ShapeNet dataset.

Physical robot experiments

We evaluated the ability of models learned on synthetic data using our method to transfer to the real world by attempting to grasp objects from the YCB dataset with a Fetch robot. Figure 3.8 shows the objects used in our experiments. Models executed in the real-world were the Autoregressive-Only variant of our method trained on the Random-1M dataset with a single depth image from directly above the scene as input. Figure 3.9 depicts typical grasp executions on the Fetch robot during our experiments.

At test time, the depth input was produced by an Intel RealSense D435 [84]. To model the noise in real-world depth images, we followed the method of Mahler et al. [114], who propose the observation model $I = \alpha\hat{I} + \epsilon$, where $\hat{I}$ is the rendered depth image, α is a Gamma random variable and ϵ is a zero-mean Gaussian random variable.

We tested the learned model on 30 previously unseen objects chosen to capture the diversity of the YCB object dataset. We observed an overall success rate of 80% (24 / 30), which is comparable to the success rate reported on novel objects in other recent papers applying deep learning to parallel jaw robotic grasping [114, 107].

In addition to the failure modes described above, we saw two additional failure modes in real-world experiments. The first was an inability to deal with highly translucent objects like the clear lid in our object dataset. This likely results from the inability of the RealSense to provide accurate readings for such objects. The second is objects with highly nonuniform densities like the hammer in our dataset. One reason for this failure is likely that our training data pipeline does not generate objects with components of different densities.

A video of our real-world experiments is available on the website for this paper.[3]

Figure 3.8: The objects used in our real-world experiments.

[3]https://sites.google.com/openai.com/domainrandomization-grasping/home

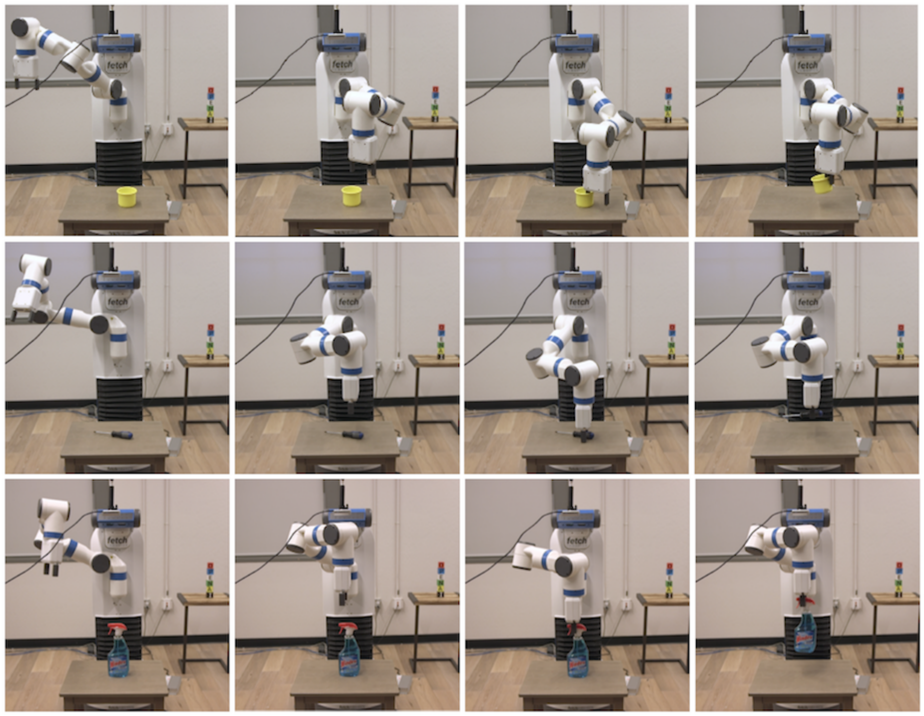

Figure 3.9: Typical grasp executions on the physical robot.

3.5 Conclusion

We demonstrated that a grasping model trained entirely using non-realistic procedurally generated objects can achieve a high success rate on realistic objects despite no training on a realistic objects. Our grasping model architecture allows for efficient sampling of high-likelihood grasps at evaluation time, with a successful grasp being found for 96% of objects in the first 20 samples. By scoring those samples, we can achieve an overall success rate of 92% on realistic objects on the first attempt. We also demonstrated that models learned using our method can be transferred successfully to the real world.

Future directions that could improve the success rate of the trained models include scaling up to larger training sets, providing the model with feedback from failed grasps to influence further grasp selection, combining our grasp planning module with work on visual servoing for grasping, and incorporating additional sensor modalities like haptic feedback.

Another exciting direction is to explore using domain randomization for generalization in other robotic tasks. If realistic object models are not needed, tasks like pick-and-place, grasping in clutter, and tool use may benefit from the ability to randomly generate hundreds of thousands or millions of 3D scenes.

Chapter 4

Geometry-Aware Neural Rendering

4.1 Introduction

The ability to understand 3-dimensional structure has long been a fundamental topic of research in computer vision [46, 96, 129, 176]. Advances in 3D understanding, driven by geometric methods [64] and deep neural networks [39, 162, 206, 221, 224] have improved technologies like 3D reconstruction, augmented reality, and computer graphics. 3D understanding is also important in robotics. To interact with their environments, robots must reason about the spatial structure of the world around them.

Robotic agents can learn 3D structure implicitly (e.g., using end-to-end reinforcement learning [121]), but these techniques can be data-inefficient and the representations often have limited reuse. An explicit 3D representation can be created using keypoints and geometry [64] or neural networks [224, 221, 156], but these can lead to inflexible, high-dimensional representations. Some systems forgo full scene representations by choosing a lower-dimensional state representation. However, not all scenes admit a compact state representation and learning state estimators often requires expensive labeling.

Previous work demonstrated that Generative Query Networks (GQN) [39] can perform neural rendering for scenes with simple geometric objects. However, robotic manipulation applications require precise representations of high degree-of-freedom (DOF) systems with complex objects. The goal of this paper is to explore the use of neural rendering in such environments.

To this end, we introduce an attention mechanism that leverages the geometric relationship between camera viewpoints called Epipolar Cross-Attention (ECA). When rendering an image, ECA computes a response at a given spatial position as a weighted sum at all *relevant positions* of feature maps from the context images. Relevant features are those on the epipolar line in the context viewpoint.

Unlike GQN, GQN with ECA (E-GQN) can model relationships between pixels that are spatially distant in the context images, and can use a different representation at each layer in the decoder. And unlike more generic approaches to non-local attention [214], E-GQN

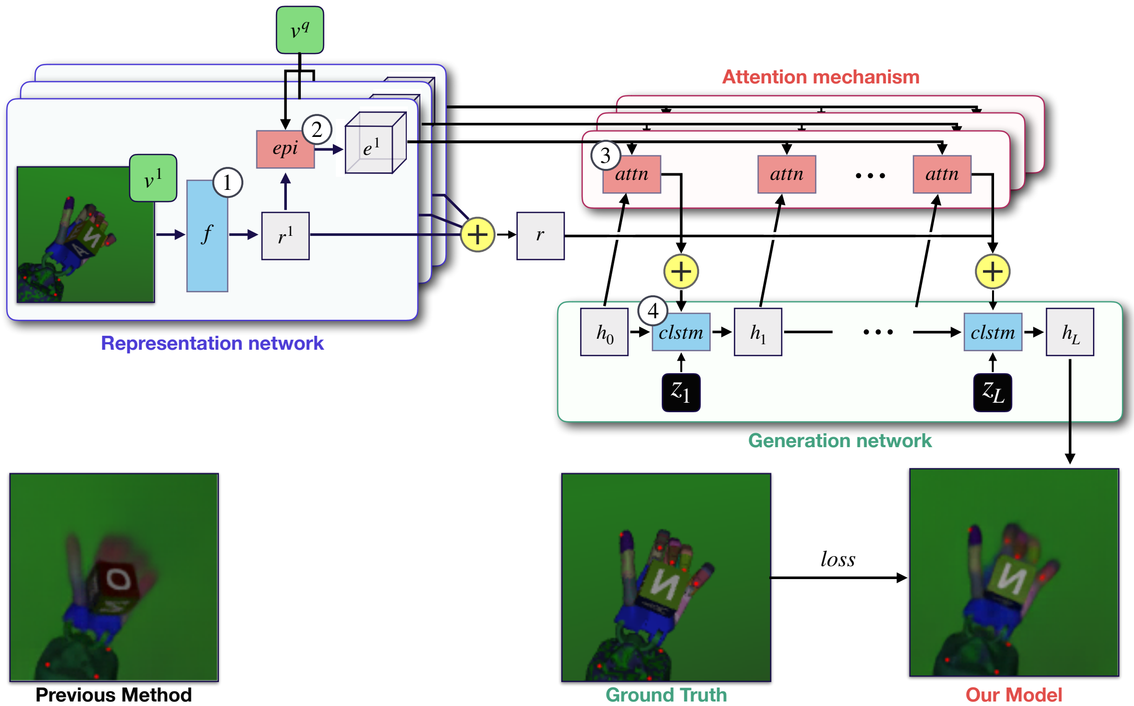

Figure 4.1: Overview of the model architecture used in our experiments. Grey boxes denote intermediate representations, z_i latent variables, $+$ element-wise addition, and blue and red boxes subcomponents of the neural network architecture. Green components are model inputs, blue are as in GQN [39], and red are contributions of our model. (1) Context images and corresponding viewpoints are passed through a convolutional neural network f to produce a context representation. We use the Tower architecture from [39]. (2) We use the epipolar geometry between the query viewpoint and the context viewpoint to extract the features in the context representation that are relevant to rendering each spatial point in the query viewpoint. These extracted features are stored in a 3-dimensional tensor called the epipolar representation. See Figure 4.3 for more details. (3) At each generation step, we compute the decoder input by attending over the epipolar representation. The attention map captures the weighted contribution to each spatial position in the decoder hidden state of all relevant positions in the context representations. See Figure 4.4 for more details. (4) The decoder, or generation network, is the skip-connection convolutional LSTM cell from [39].

only requires each feature to be compared to n other features instead of n^2 other features.

We evaluate our approach on datasets from the original GQN paper and three new datasets designed to test the ability to render systems with many degrees of freedom and a wide variety of objects. We find significant improvements in a lower bound on the negative log likelihood (the ELBO), per-pixel mean absolute error, and qualitative performance on most of these datasets. Finally, to asses the utility of our approach in problems in robotics, we show that using E-GQN pre-training for a pose estimation task allows us to achieve similar error with randomized camera positions than a baseline approach with fixed camera positions.

To summarize, our key contributes are as follows:

1. We introduce a novel attention mechanism, Epipolar Cross-Attention (ECA), that leverages the geometry of the camera poses to perform efficient non-local attention.

2. We introduce three datasets: *Disco Humanoid*, *OpenAI Block*, and *Room-Random-Objects* as a testbed for neural rendering with complex objects and high-dimensional state.

3. We demonstrate the ECA with GQN (E-GQN) improves neural rendering performance on those datasets.

4. We demonstrate the utility of E-GQN in robotics by using as pretraining to improve pose estimation performance with free-moving cameras.

4.2 Related work

Multi-view 3D reconstruction

Constructing models of 3D scenes from multiple camera views is a widely explored subfield of computer vision. If the camera poses are unknown, Structure-from-Motion (SfM) techniques [173, 4] (for unordered images) or Simultaneous Localization and Mapping (SLAM) techniques [34] (for ordered images from a real-time system) are typically used. If camera poses are known, multi-view stereo or multi-view reconstruction (MVR) can be applied.

MVR techniques differ in how they represent the scene. Voxels [37], level-sets [40], depth maps [192], and combinations thereof are common [177]. They also differ in how they construct the scene representation. Popular approaches include adding parts that meet a cost threshold [176], iteratively removing parts that do not [96, 46], or fitting a surface to extracted feature points [129].

Most MVR techniques do not rely on ground truth scene representations and instead depend on some notion of consistency between the generated scene representation and the input images like scene space or image space photo consistency measures [80, 96, 177].

Deep learning for 3D reconstruction

Recently, researchers have used deep learning to learn the mapping from images to a scene representation consisting of voxels [206, 224, 221, 156, 230] or meshes [156], with supervisory signal coming from verifying the 3D volume against known depth images [206, 230] or coming from a large-scale 3D model database [224, 156]. Loss functions include supervised losses

[230], generative modeling objectives [156], a 3D analog of deep belief networks [102, 224], and a generative adversarial loss [221, 55].

Some neural network approaches to 3D understanding instead create implicit 3D models of the world. By training an agent end-to-end using deep reinforcement learning [121] or path planning and imitation learning [61], agents can learn good enough models of their environments to perform tasks in them successfully. Like our work, Gupta and coauthors also incorporate geometric primitives into their model architecture, transforming viewpoint representations into world coordinates using spatial transformer layers [72]. Instead of attempting to learn 3D representations that help solve a downstream task, other approaches learn generic 3D representations by performing multi-task learning on a variety of supervised learning tasks like pose estimation [235].

View Synthesis and Neural rendering

Neural rendering or view synthesis approaches learn an implicit representation of the 3D structure of the scene by training a neural network end-to-end to render the scene from an unknown viewpoint. In [196], the authors map an images of a scene to an RGB-D image from an unknown viewpoint with an encoder-decoder architecture, and train their model using supervised learning. Others have proposed incorporating the geometry of the scene into the neural rendering task. In [45], plane-sweep volumes are used to estimate depth of points in the scene, which are colored by a separate network to perform view interpolation (i.e., the input and output images are close together). Instead of synthesizing pixels from scratch, other work explores using CNNs to predict appearance flow [243].

In [39], the authors propose the generative query network model (GQN) model architecture for neural rendering. Previous extensions to GQN include augmenting it with a patch-attention mechanism [162] and extending it to temporal data [95].

4.3 Background

Problem description

Given one or more images x^k and corresponding camera viewpoints v^k of a scene S, the goal of neural rendering is to learn a model that can accurately predict what the camera would see from a query viewpoint v^q. More formally, for distributions of scenes $p(S)$ and images with corresponding viewpoints $p(v^k, x^k \mid S)$, the goal of neural rendering is to learn a model that maximizes

$$\mathbb{E}_{S\sim p(S)}\mathbb{E}_{v^q,x^q\sim p(v^q,x^q|S)}\mathbb{E}_{v^k,x^k\sim p(v^k,x^k|S)} \log p\left(x^q|(x^k)_{k=\{1,\cdots,K\}},(v^k)_{k=\{1,\cdots,K\}},v^q\right)$$

This can be viewed as an instance of few-shot density estimation [154].

Generative Query Networks

Generative Query Networks model the likelihood above with an encoder-decoder neural network architecture. The encoder, or *representation network* is a convolutional neural network that takes v^k and x^k as input and produces a representation r.

The decoder, or *generation network*, takes r and v^q as input and predicts the image rendered from that viewpoint. Uncertainty in the output is modeled using stochastic latent variables z, producing a density $g(x^q \mid v^q, r) = \int g(x^q, z \mid v^q, r)dz$ that can be approximated tractably with a variational lower bound [86, 39]. The generation network architecture is based on the skip-connection convolutional LSTM decoder from DRAW [58].

See [39] for more details on the GQN model architecture.

Epipolar Geometry

The epipolar geometry between camera viewpoints v^1 and v^2 describes the geometric relationship between 3D points in the scene and their projections in images x^1 and x^2 rendered from pinhole cameras at v^1 and v^2. Figure 4.2 describes the relationship.

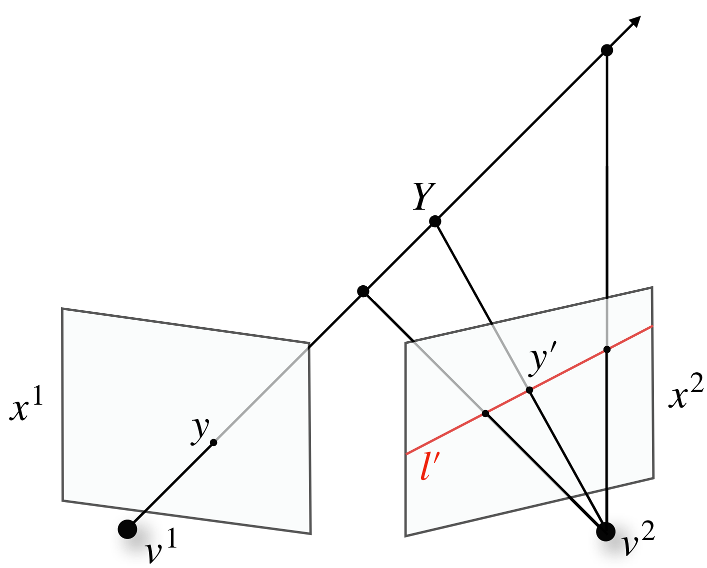

Figure 4.2: Illustration of the epipolar geometry. For any image point y in x^1 corresponding to a 3D point Y, the image point y' in x^2 that corresponds to Y is constrained to lie on a line l' in x^2. This line corresponds to the projection onto x^2 of the ray passing through the camera center of v^1 and y. This line depends only on the intrinsic geometry between v^1 and v^2, not the content of the scene.

There is a linear mapping called the *fundamental matrix* that captures this relationship. The fundamental matrix is a mapping F from an image point y in x^1 to the epipolar line

l'. F is a 3×3 matrix that depends on v^1 and v^2. The image point $y' = [w', h', 1]^T$ lies on the line l' corresponding to $y = [w, h, 1]^T$ if (w', h') lies on the line $ax_0 + bx_1 + c = 0$, where $Fy = [a, b, c]^T$.

4.4 Methods

In GQN, the scene representation is an element-wise sum of context representations from each context viewpoint. The context representations are created from the raw camera images through convolutional layers. Since convolutions are local operations, long-range dependencies are difficult to model [65, 68, 214]. As a result, information from distant image points in the context representation may not propagate to the hidden state of the generation network.

The core idea of Epipolar Cross-Attention is to allow the features at a given spatial position y in the generation network hidden state to depend directly on all of the relevant spatial positions in the context representation. Relevant spatial positions are those that lie on the epipolar line corresponding to y in the context viewpoint.

Figure 4.1 describes our model architecture. Our model is a variant of GQN [39]. Instead of using $r = \sum_k r^k$ as input to compute the next generation network hidden state h_l, we use an attention map computed using our epipolar cross-attention mechanism. The next two subsections describe the attention mechanism.

Computing the Epipolar Representation

For a given spatial position $y = (h, w)$ in the decoder hidden state h_l, the epipolar representation e^k stores at (h, w) all of the features from r^k that are relevant to rendering the image at that position.[1]

Figure 4.3 shows how we construct the epipolar representation. To compute the epipolar line l'_y in r^k, we first compute the fundamental matrix F_q^k arising from camera viewpoints v^q and v^k, and then find $l'_y = F_q^k[h, w, 1]^T$.

If h_l has shape (H, W), then for each $0 \le w' < W$,

$$e^k_{h,w,w'} = r^k_{h',w'}$$

where the subscripts denote array indexing and h' is the point on l'_y corresponding to w'.[2]

All of these operations can be performed efficiently and differentiably in automatic differentiation libraries like Tensorflow [1] as they can be formulated as matrix multiplication or gather operations.

[1]Note that care must be taken that the representation network does not change the effective field of view of the camera.

[2]To make sure h' are valid array indices, we round down to the nearest integer. For indices that are too large or too small, we instead use features of all zeros.

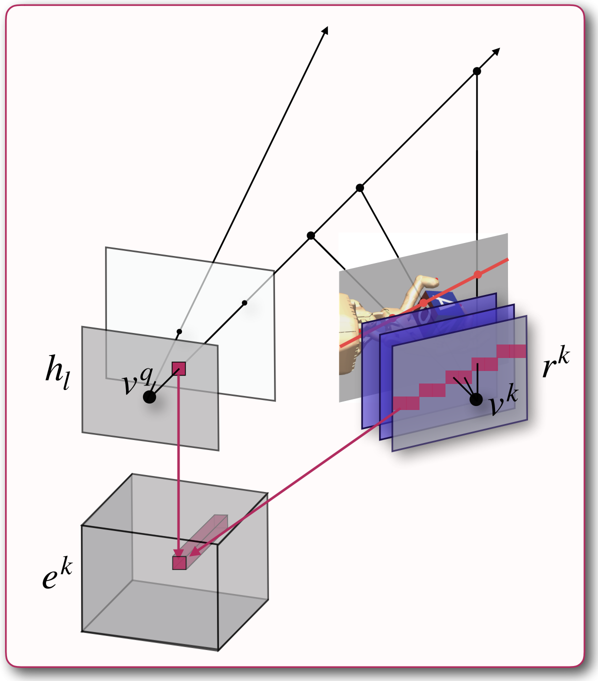

Figure 4.3: Constructing the epipolar representation e^k for a given camera viewpoint v^k. For a given spatial position in the decoder state h_l, there is a 1-dimensional subset of feature maps l' in r^k arising from the epipolar geometry. This can be viewed as the projection of the line passing from the camera center at v^q through the image point onto r^k. The epipolar representation e^k is constructed by stacking these lines along a third spatial dimension.

Attention mechanism

Figure 4.4 describes our attention mechanism in more detail. We map the previous decoder hidden state h_{l-1} and the epipolar representations e^k to an attention score a_l^k. a_l^k represents the weighted contribution to each spatial position of all of the geometrically relevant features in the context representation r^k.

Typically the weights for the projections are shared between context images and decoder steps. To facilitate passing gradients to the generation network, the attention maps a_l^k are provided a skip connection to r^k, producing

$$a_l = \lambda \sum_k a_l^k + \sum_k r^k$$

.

where λ is a learnable parameter. a_l is provided as input to the decoder to produce the next hidden state h_l.

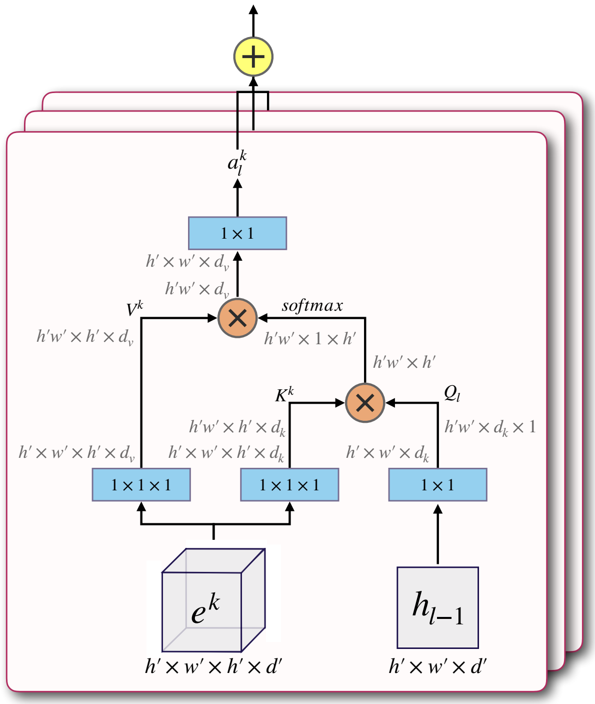

Figure 4.4: Our attention mechanism. Blue rectangles denote convolutional layers with given kernel size. "×" denotes batch-wise matrix multiplication, and "+" element-wise summation. The previous decoder hidden state h_{l-1} is used to compute a query tensor Q_l by linear projection. The epipolar representation e^k is also linearly projected to compute a key tensor K^k and value tensor V^k. K^k and Q_l are matrix-multiplied to form unnormalized attention weights, which are scaled by $1/\sqrt{d_k}$. A softmax is computed along the final dimension, and the result is multiplied by V^k to get an attention score as in [212]. All of the attention scores are linearly projected into the correct output dimension and summed element-wise.

4.5 Experiments

Datasets

To evaluate our proposed attention mechanism, we trained GQN + Epipolar Cross-Attention (E-GQN) on four datasets from the GQN paper: Rooms-Ring-Camera (RRC), Rooms-Free-Camera (RFC), Jaco, and Shepard-Metzler 7 parts (SM7) [39, 180].

The GQN datasets are missing several important features for robotic representation learning. First, they contain only simple geometric objects. Second, they have relatively few degrees of freedom: objects are chosen from a fixed set and placed with two positional and 1 rotational degrees of freedom. Third, they do not require generalizing to a wide range of objects. Finally, with the exception of the Rooms-Free-Camera dataset, all images are size 64×64 or smaller.

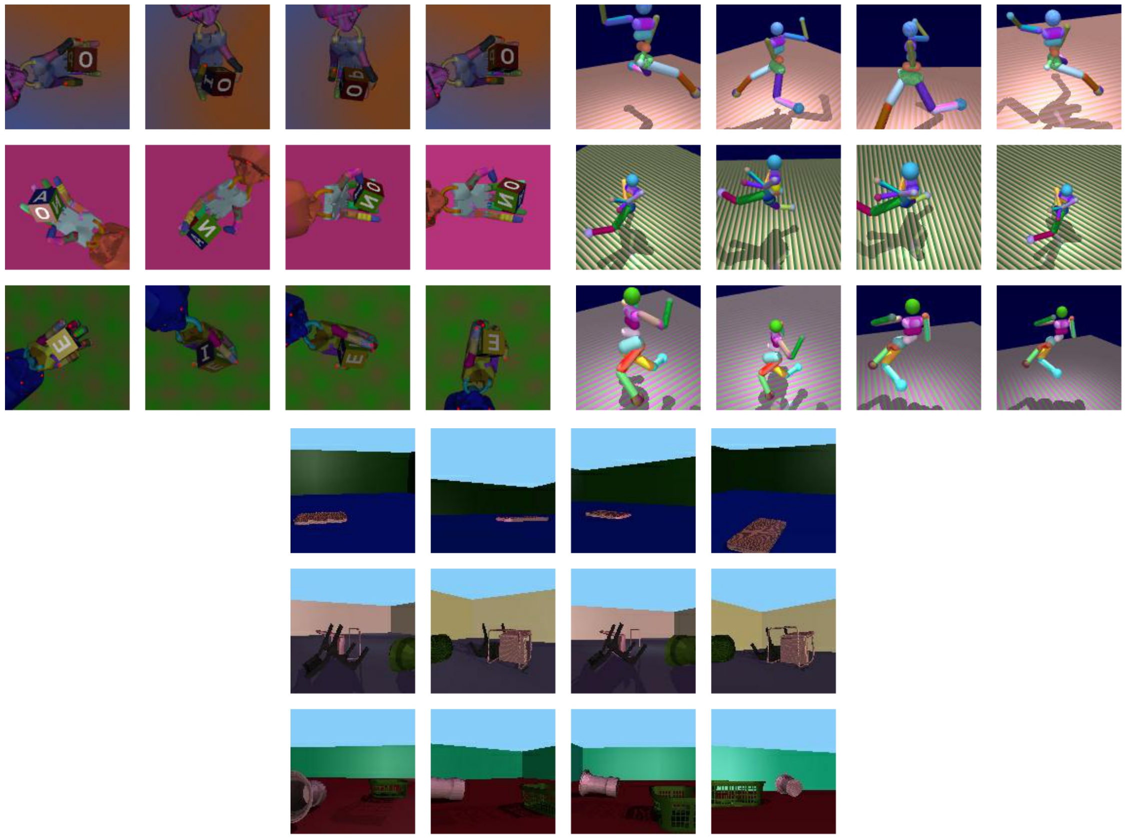

Figure 4.5: In addition to the GQN datasets, we designed the following additional datasets to test the ability of models to capture detail and object variability. (a) OpenAI Block (OAB). (b) Disco-Humanoid (Disco), (c) Rooms-Random-Objects (RRO)

To address these limitations, we created three new datasets: OpenAI Block (OAB), Disco Humanoid (Disco), and Rooms-Random-Objects (RRO). All of our datasets are rendered at size 128×128. Examples from these datasets are shown in Figure 4.5.

The OAB dataset is a modified version of the domain randomized [200] in-hand block manipulation dataset from [140] where cameras poses are additionally randomized. Since this dataset is used for sim-to-real transfer for real-world robotic tasks, it captures much of the complexity needed to use neural rendering in real-world robotics, including a 24-DOF robotic actuator and a block with letters that must be rendered in the correct 6-DOF pose.

The Disco dataset is designed to test the model's ability to accurately capture many degrees of freedom. It consists of the 27-DOF MuJoCo [201] model from OpenAI Gym [17] rendered with each of its joints in a random position in $[-\pi, \pi)$. Each of the geometric shape components of the Humanoid's body are rendered with a random simple texture.

The RRO dataset captures the ability of models to render a broad range of complex objects. Scenes are created by sampling 1-3 objects randomly from the ShapeNet [19] object

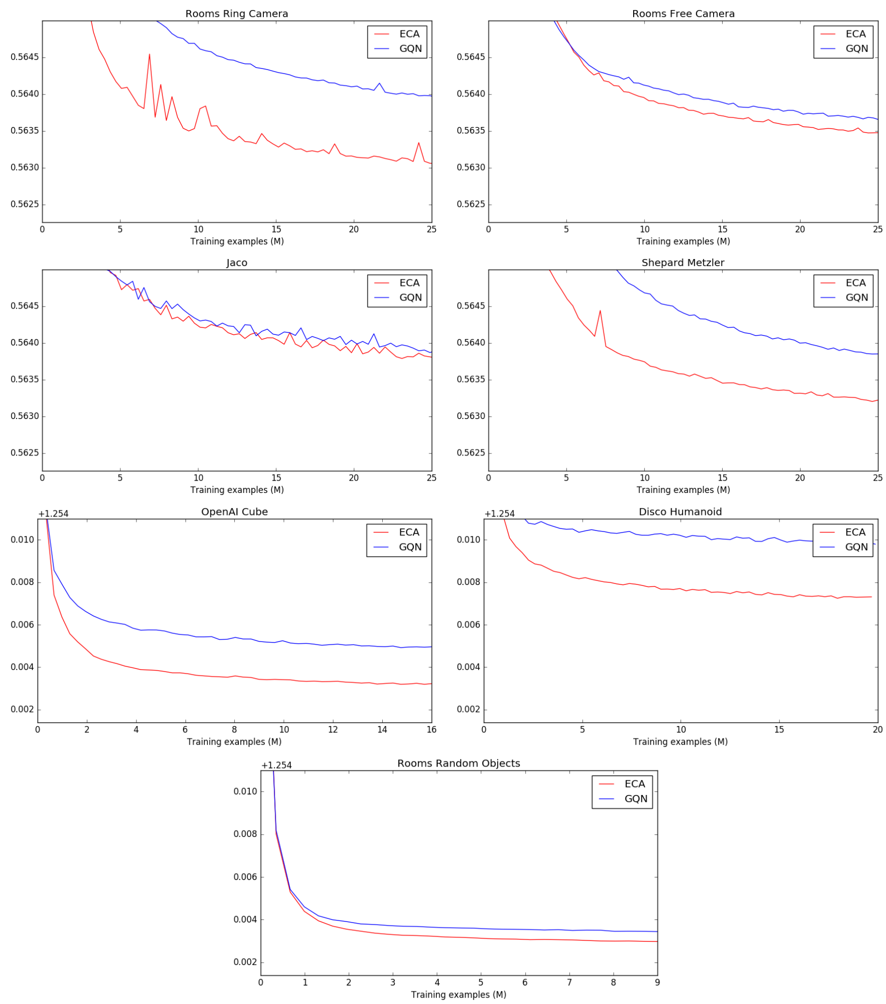

Figure 4.6: ELBO (nats/dim) on the test set. The minimum y-value denotes the theoretical minimum error (computed by assuming perfect predictions from the model). Note: the scale is different on ours vs GQN because we use a different output variance.

database. The floor and walls of the room as well as each of the objects are rendered using random simple textures.

Experimental setup

We use the the "Tower" representation network from [39]. Our generation network is from Figure S2 of [39] with the exception of our attention mechanism. The convolutional LSTM hidden state and skip connection state have 192 channels. The generation network has 12 layers and weights are shared between generation steps. We always use 3 context images. Key dimension $d_k = 64$ for all experiments, and value dimension $d_v = 128$ on the GQN datasets with $d_v = 192$ on our datasets.

We train our models using the Adam optimizer [87]. We ran a small hyperparameter sweep to choose the learning rate schedule and found that a learning rate of 1e-4 or 2e-4 linearly ramped up from 2e-5 over 25,000 optimizer steps and then linearly decayed by a factor of 10 over 1.6M optimizer steps performs best in our experiments.

We use a batch size of 36 in experiments on the GQN datasets and 32 on our datasets. We train our models on 25M examples on 4 Tesla V-100s (GQN datasets) or 8 Tesla V-100s (our datasets).

As in [39], we evaluate samples from the model with random latent variables, but taking the mean of the output distribution. Input and output images are scaled to $[-0.5, 0.5]$ on the GQN datasets and $[-1, 1]$ on ours. Output variance is scaled as in [39] on the GQN datasets but fixed at 1.4 on ours.

Quantitative results

Model	rrc	rfc	jaco	sm7	oab	disco	rro
GQN	7.40	12.44	4.30	3.13	10.99	18.86	10.12
E-GQN	3.59	12.05	4.00	2.14	5.47	12.46	6.59

Figure 4.7: Per-pixel mean average error (measured in pixels) for GQN and E-GQN on the test set.

Figure 4.6 shows the learning performance of our method. Figure 4.7 shows the mean average error (in pixels). Both show that our method significantly outperforms the baseline on most datasets, with the exception of Jaco and RFC, where both methods perform about the same.

Jaco has relatively few degrees of freedom, and both methods perform well. In RFC, since the camera moves freely, the objects are not always contained in context and target images. We hypothesize that our method provides comparatively little value in scenes with little content overlap between images.

Qualitative results

Figures 4.9-4.15 show randomly chosen samples rendered by our model. Differences in rendering quality are difficult to determine in the RRC, Jaco, and SM7 datasets as both models perform near perfectly. Our model performs better on examples from RFC containing objects.

Larger differences can be seen in the OAB, Disco, and RRO datasets. On OAB, our model near-perfectly captures the pose of the block and hand and faithfully reproduces their textures, whereas the baseline model often misrepresents the pose and textures. On Disco, ours more accurately renders the limbs and shadow of the humanoid. On RRO, ours faithfully (though not always accurately) renders the shape of objects, whereas the baseline often renders the wrong object in the wrong location.

Pose estimation results

Dataset	Position Error (cm)	Angular Error (Degrees)
Fixed Cameras (oracle)	0.56	3.80
Random Cameras	2.67	8.92
Fine-tuned GQN	2.71	5.63
Fine-tuned E-GQN	2.66	3.98

Figure 4.8: Test set pose estimation error on the OpenAI Block dataset. Fixed Cameras: same camera poses throughout training as in [140]. Random Cameras: cameras are randomized, but no pre-training.

To evaluate whether the representations learned by our model are useful for downstream robotics tasks, we explore using them for pose estimation with randomized cameras. Figure 4.8 shows our results. A model trained from scratch to estimate the pose of the block in the OpenAI Block dataset using the architecture from [140] performs significantly worse when camera poses are randomized. If instead we fine-tune a pre-trained E-GQN (using the ELBO as an auxilliary objective), we learn an object pose model that is agnostic to camera pose.

4.6 Conclusion

Robots require an understanding of the 3D world to interact with their environment. In this work, we present a geometrically motivated attention mechanism that allows neural rendering models to learn more accurate 3D representations and scale to more complex datasets with higher dimensional images.

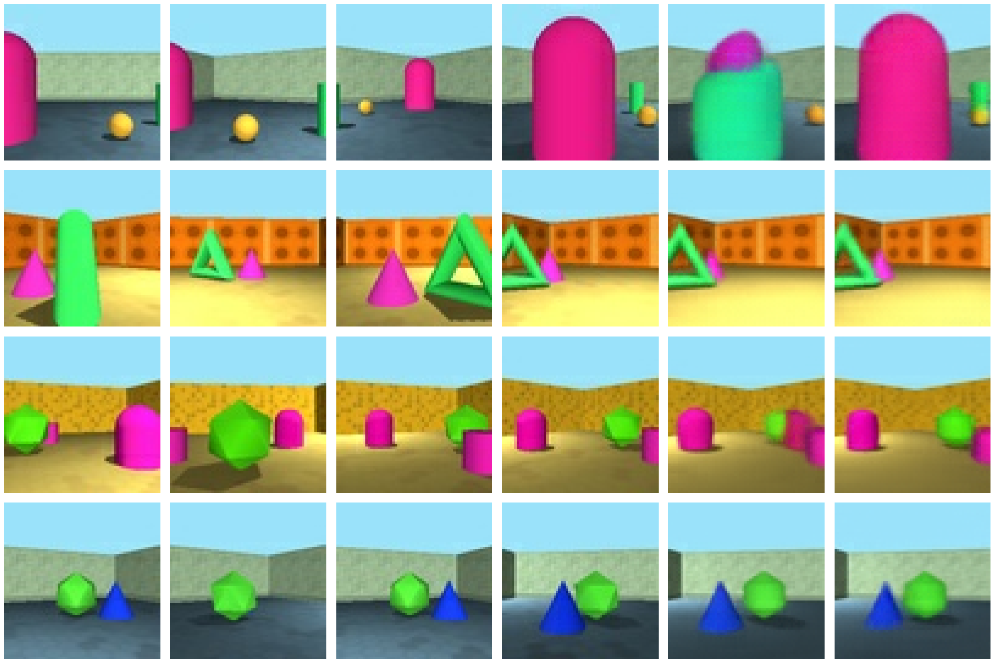

Figure 4.9: Rooms Ring Camera examples. From left to right: context image 1, context image 2, context image 3, ground truth, GQN prediction, E-GQN prediction

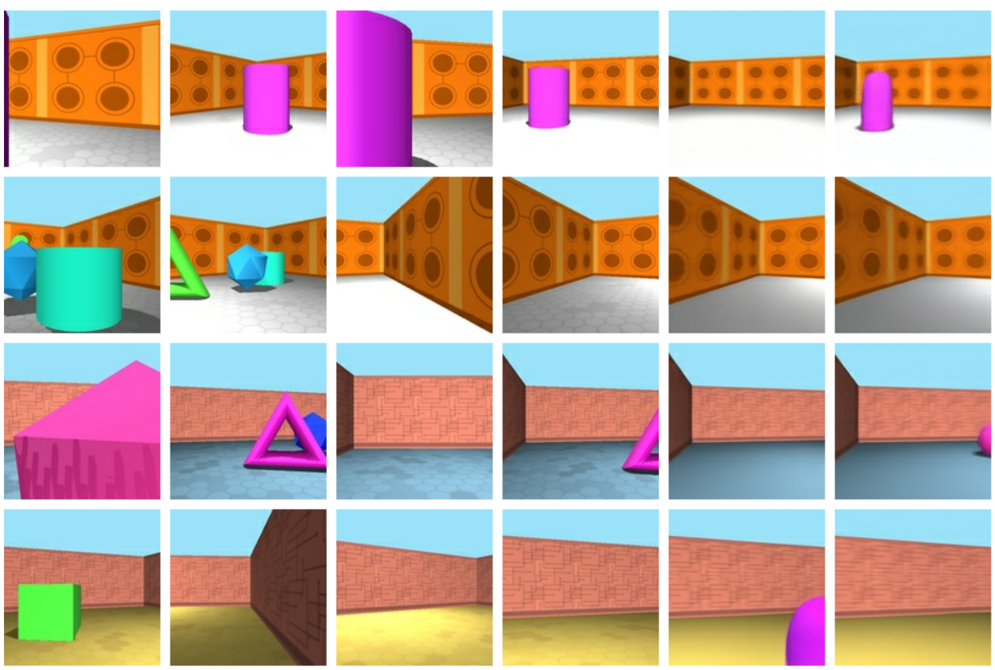

Figure 4.10: Rooms Free Camera Examples

Figure 4.11: Jaco Examples

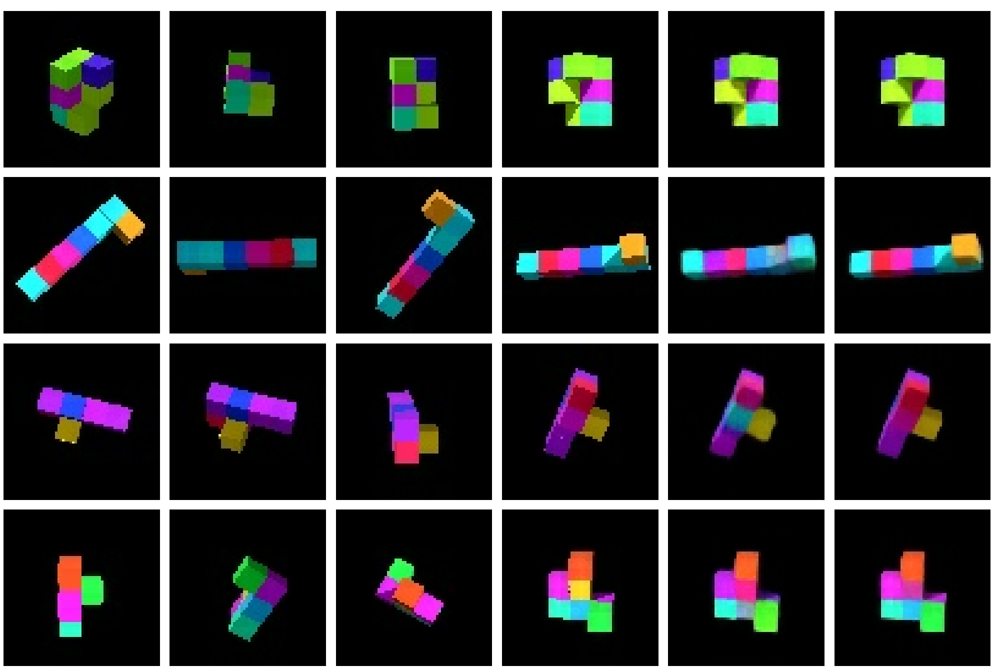

Figure 4.12: Shepard-Metzler 7 Examples

Figure 4.13: OpenAI Block Examples

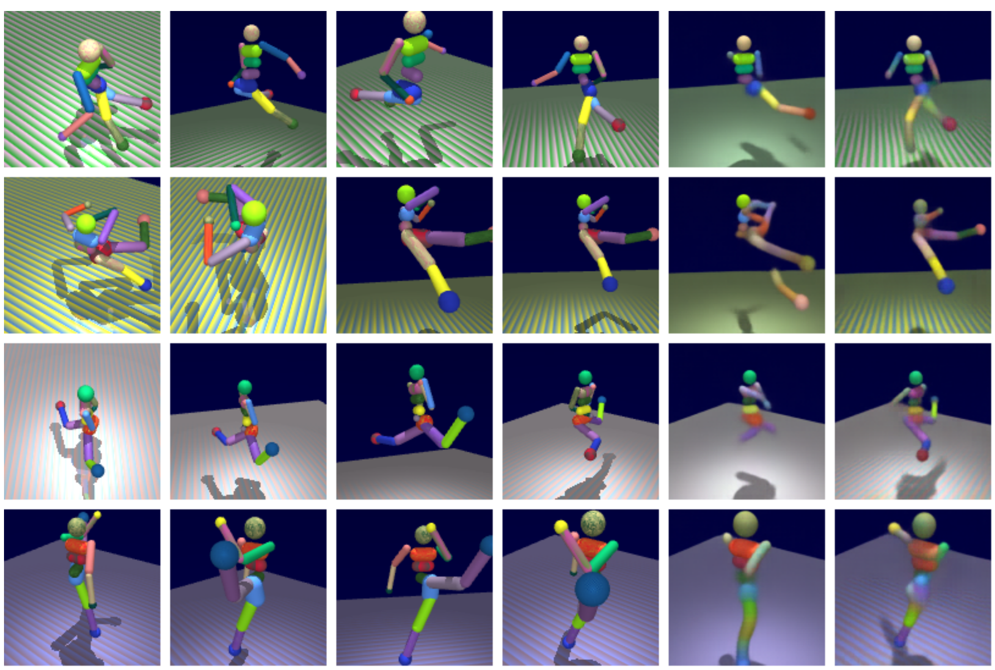

Figure 4.14: Disco Humanoid Examples

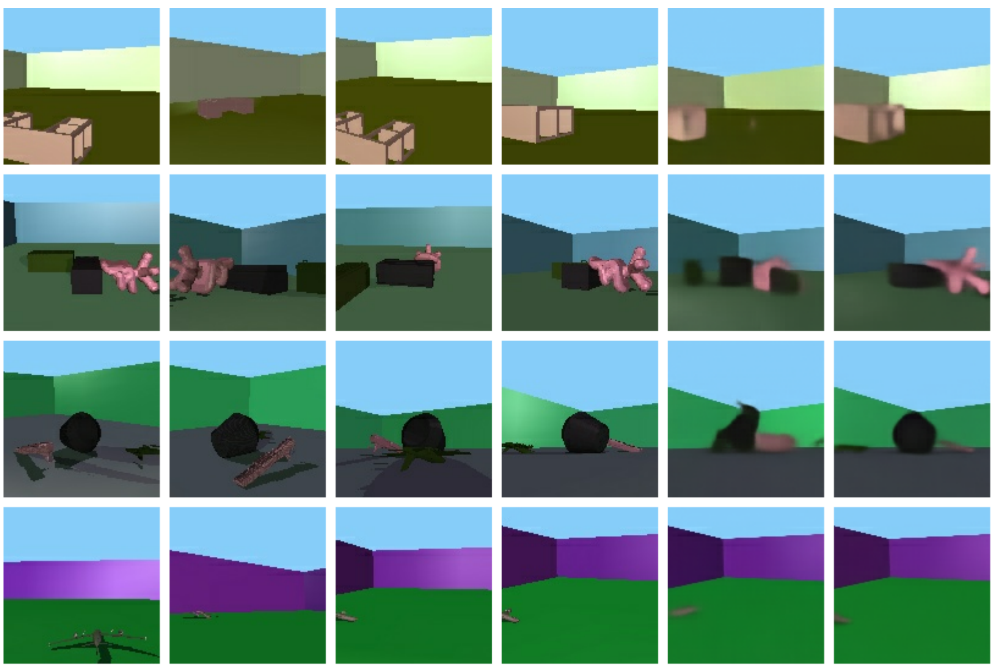

Figure 4.15: Rooms Random Objects Examples

Chapter 5

Discussion

5.1 Conclusion

In this thesis, we investigated how to learn perception models using simulated data that are useful for downstream robotic control tasks. The main idea we deployed throughout is domain randomization, the concept of massively randomizing aspects of the simulator instead of carefully modeling them. We explore a sequence of progressively more general state estimation techniques, beginning from learning an estimate of the position of a single object, then learning a model that can estimate a distribution over grasp points for any object, and finally developing a model that builds an implicit 3D representation of the entire scene.

Since we began investigating domain randomization, this idea has been applied to a range of other problems. We outline some of them here.

5.2 Other applications of domain randomization

Domain randomization in the visual domain has been applied to a number of other robotics tasks. It has been explored in the context of 6-DOF pose estimation of objects relative to a fixed reference point [190], relative to a robot gripper [111], and using active perception [155]. Visual state estimation models trained with domain randomization have been used as input to control systems that perform block stacking [217, 203], peg-insertion [205], in-hand dextrous manipulation [140], grasping [202, 78, 116, 228, 62], manipulating non-rigid bodies like cloth [117], bin picking of reflective parts [36], picking fish from a bucket [35], closed-loop control of low-cost, low-precision robot arms [245, 244], and locomotion [185].

Domain randomization has also been used to train end-to-end visuomotor policies in simulation that are transfered to the real world for pick-and-place [76, 75, 146], pushing [146], visual servoing [167], reaching in clutter [238, 237], real-time active object tracking [113], and autonomous quadrotor landing [148] tasks.

Beyond robotics, other work has explored the use of visual domain randomization for learning pixel-wise dense multi-object descriptors [44], bounding box detection of indoor objects [152, 12] and cars [204, 85], detecting packaged food objects in refrigerators [153], tracking underwater vehicles [91], performing segmentation of leaves in outdoor vegetation scenes [215], human face tracking [160], estimating the joint angles of robotic arms [66], and localizing a camera inside a human lung [178].

The idea of domain randomization has also been applied outside of vision. Randomizing simulator dynamics has been shown to be a useful technique for transferring control policies to the real world for pushing tasks [143], quadruped robots [194], flapping wing robots [41], and dexterous in-hand manipulation [140]. It has been applied outside robotics to fields like seismic phase association [163] and traffic control [79].

Designing simulated environments and randomization distributions is one the most challenging parts of using this technique. A promising direction toward making this easier is simulations and environments that are designed with randomization in mind. Examples include the mujoco-py[1] Python bindings for MuJoCo [201], the Random Worlds Gazebo plugin [11], and the House3D dataset [222].

Another way to mitigate this difficulty is to find a metric that quantifies how much learned models are overfitting to simulation [131].

A promising direction toward making domain randomization easier to use is learning a randomization distribution directly by optimizing the distribution with respect to a chosen objective. One such objective is to better match the simulation distribution to the real-world data distribution, e.g., by iteratively learning a policy in simulation and using real rollouts to minimize a distance function between the real data distribution and the simulated data distribution [20]. A second objective is to choose randomizations that maximally confuse the downstream model. To avoid constructing simulations that are too difficult, randomizations can be constrained to perturbations from specific categories like background color, lighting, etc [234]. A third possible objective is choosing randomizations that are informative in the sense that they aid in generalizing to a reference simulated environment [118].

5.3 Future work in sim-to-real transfer

Though domain randomization shows promise in many domains, there are many opportunities to improve its usefulness. Today, we only have an informal understanding of why domain randomization works. More comprehensive empirical studies examining new randomizations to add, the effect of image size and simulator quality on transferability, the effect of model architecture on transferability, failure cases of domain randomization, and domain randomization in uncontrolled environments could aid our understanding. Theoretical knowledge about why domain randomization works and when it fails could be relevant to the study of generalization in machine learning and practically useful in designing real world robotic and computer vision systems.

[1]https://github.com/openai/mujoco-py

An under-explored approach is combining domain randomization with domain adaptation. These classes of techniques are complimentary, but it is not yet known which domain adaptation algorithms work best with randomized data. The intuition behind domain randomization could also aid in creating better domain adaptation techniques - for example, by encouraging adapted features to be more random.

A principal limitation of domain randomization is the difficulty of designing simulations and randomization distributions. The ability to automatically select the randomization distribution explored in [20, 234, 118] is a promising approach, but more work here is needed to produce dramatically better results in both vision and dynamics. A significant step forward in the field would be the ability to automatically design simulations from unstructured real-world data (real-to-sim). Our work on E-GQN represents a step in this direction, but more work is needed to make these models work in the real world and to recognize not only 3D structure, but also semantic information and physical interactions.

Instead of using real data to design better simulations, another approach is to build on our work on procedurally generated objects to design simulators that are even more random. Beyond procedurally generated objects, a generative model could be used to generated objects that are similar to real objects, dissimilar from existing objects, or difficult for the current model to handle.

Simulated data could be the key to translating recent progress in deep supervised learning and deep reinforcement learning into real-world robotic results. We look forward to low-cost, perfectly labeled data being used to build cheaper, safer, and more capable autonomous systems.

Bibliography

[1] Martín Abadi et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Pieter Abbeel, Varun Ganapathi, and Andrew Y Ng. "Learning vehicular dynamics, with application to modeling helicopters". In: *Advances in Neural Information Processing Systems*. 2006, pp. 1–8.

[3] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. "Using inaccurate models in reinforcement learning". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 1–8.

[4] Sameer Agarwal et al. "Building rome in a day". In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 72–79.

[5] Pulkit Agrawal et al. "Learning to poke by poking: Experiential learning of intuitive physics". In: *Advances in Neural Information Processing Systems*. 2016, pp. 5074–5082.

[6] Marcin Andrychowicz et al. "Hindsight experience replay". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5048–5058.

[7] Rika Antonova et al. "Reinforcement Learning for Pivoting Task". In: *arXiv preprint arXiv:1703.00472* (2017).

[8] Jeroen van Baar et al. "Sim-to-Real Transfer Learning using Robustified Controllers in Robotic Tasks involving Complex Dynamics". In: *arXiv preprint arXiv:1809.04720* (2018).

[9] Antonio Bicchi and Vijay Kumar. "Robotic grasping and contact: A review". In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 1. IEEE. 2000, pp. 348–353.

[10] Jeannette Bohg et al. "Data-driven grasp synthesis - a survey". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 289–309.

[11] Joao Borrego et al. "A generic visual perception domain randomisation framework for Gazebo". In: *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE. 2018, pp. 237–242.

[12] João Borrego et al. "Applying domain randomization to synthetic data for object category detection". In: *arXiv preprint arXiv:1807.09834* (2018).

[13] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[14] Konstantinos Bousmalis et al. "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping". In: *arXiv preprint arXiv:1709.07857* (2017).

[15] 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.

[16] Greg Brockman et al. "OpenAI Five". In: (2018). URL: `https://openai.com/blog/openai-five/`.

[17] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[18] Berk Calli et al. "The YCB object and model set: Towards common benchmarks for manipulation research". In: *Advanced Robotics (ICAR), 2015 International Conference on*. IEEE. 2015, pp. 510–517.

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

[20] Yevgen Chebotar et al. "Closing the sim-to-real loop: Adapting simulation randomization with real world experience". In: *arXiv preprint arXiv:1810.05687* (2018).

[21] Mircea Cimpoi et al. "Describing textures in the wild". In: (2014), pp. 3606–3613.

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

[23] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. "The MOPED framework: Object recognition and pose estimation for manipulation". In: *The International Journal of Robotics Research* 30.10 (2011), pp. 1284–1306.

[24] Alvaro Collet and Siddhartha S Srinivasa. "Efficient multi-view object recognition and full pose estimation". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 2050–2055.

[25] Alvaro Collet et al. "Object recognition and full pose registration from a single image for robotic manipulation". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 48–55.

[26] Antoine Cully et al. "Robots that can adapt like animals". In: *arXiv preprint arXiv:1407.3501* (2014).

[27] Mark Cutler and Jonathan P How. "Efficient reinforcement learning for robots using informative simulated priors". In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2605–2612.

[28] Mark Cutler, Thomas J Walsh, and Jonathan P How. "Reinforcement learning with multi-fidelity simulators". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 3888–3895.

[29] George E Dahl et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *IEEE Transactions on audio, speech, and language processing* 20.1 (2012), pp. 30–42.

[30] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[31] Coline Devin et al. "Deep object-centric representations for generalizable robot learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7111–7118.

[32] Lixin Duan, Dong Xu, and Ivor Tsang. "Learning with augmented features for heterogeneous domain adaptation". In: *arXiv preprint arXiv:1206.4660* (2012).

[33] Yan Duan et al. "RL ̃2: Fast Reinforcement Learning via Slow Reinforcement Learning". In: *arXiv preprint arXiv:1611.02779* (2016).

[34] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[35] Jonatan S Dyrstad and John Reidar Mathiassen. "Grasping virtual fish: A step towards robotic deep learning from demonstration in virtual reality". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2017, pp. 1181–1187.

[36] Jonatan S Dyrstad et al. "Bin Picking of Reflective Steel Parts Using a Dual-Resolution convolutional Neural Network Trained in a Simulated Environment". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE. 2018, pp. 530–537.

[37] Peter Eisert, Eckehard Steinbach, and Bernd Girod. "Multi-hypothesis, volumetric reconstruction of 3-D objects from multiple calibrated camera views". In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*. Vol. 6. IEEE. 1999, pp. 3509–3512.

[38] Staffan Ekvall, Danica Kragic, and Frank Hoffmann. "Object recognition and pose estimation using color cooccurrence histograms and geometric modeling". In: *Image and Vision Computing* 23.11 (2005), pp. 943–955.

[39] SM Ali Eslami et al. "Neural scene representation and rendering". In: *Science* 360.6394 (2018), pp. 1204–1210.

[40] Olivier Faugeras and Renaud Keriven. *Variational principles, surface evolution, PDE's, level set methods and the stereo problem*. IEEE, 2002.

[41] Fan Fei et al. "Learning extreme hummingbird maneuvers on flapping wing robots". In: *arXiv preprint arXiv:1902.09626* (2019).

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1126–1135.

[43] Chelsea Finn et al. "Deep spatial autoencoders for visuomotor learning". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 512–519.

[44] 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).

[45] John Flynn et al. "Deepstereo: Learning to predict new views from the world's imagery". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5515–5524.

[46] Pascal Fua and Yvan G Leclerc. "Object-centered surface reconstruction: Combining multi-image stereo and shading". In: *International Journal of Computer Vision* 16.1 (1995), pp. 35–56.

[47] NVIDIA Gameworks. *Nvidia FleX*. 2018.

[48] Javier Garcıa and Fernando Fernández. "A comprehensive survey on safe reinforcement learning". In: *Journal of Machine Learning Research* 16.1 (2015), pp. 1437–1480.

[49] Mathieu Germain et al. "MADE: masked autoencoder for distribution estimation". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015, pp. 881–889.

[50] Ali Ghadirzadeh et al. "Deep Predictive Policy Training using Reinforcement Learning". In: *arXiv preprint arXiv:1703.00727* (2017).

[51] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[52] Corey Goldfeder et al. "Grasp planning via decomposition trees". In: *Robotics and Automation, 2007 IEEE International Conference on*. IEEE. 2007, pp. 4679–4684.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[54] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

[55] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[56] Iryna Gordon and David G Lowe. “What and where: 3D object recognition with accurate pose”. In: *Toward category-level object recognition*. Springer, 2006, pp. 67–82.

[57] Karol Gregor et al. “Deep autoregressive networks”. In: *arXiv preprint arXiv:1310.8499* (2013).

[58] Karol Gregor et al. “DRAW: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623* (2015).

[59] Abhishek Gupta et al. “Learning Invariant Feature Spaces to Transfer Skills With Reinforcement Learning”. In: *ICLR 2017, to appear* (2017). URL: `https://openreview.net/pdf?id=Hyq4yhile`.

[60] Saurabh Gupta et al. “Aligning 3D models to RGB-D images of cluttered scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4731–4740.

[61] Saurabh Gupta et al. “Unifying map and landmark based representations for visual navigation”. In: *arXiv preprint arXiv:1712.08125* (2017).

[62] Aleksi Hämäläinen et al. “Affordance Learning for End-to-End Visuomotor Robot Control”. In: *arXiv preprint arXiv:1903.04053* (2019).

[63] Josiah P Hanna and Peter Stone. “Grounded action transformation for robot learning in simulation”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[64] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[65] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[66] Christoph Heindl et al. “3D Robot Pose Estimation from 2D Images”. In: *arXiv preprint arXiv:1902.04987* (2019).

[67] Stefan Hinterstoisser et al. “Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 858–865.

[68] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[69] Judy Hoffman et al. “Cycada: Cycle-consistent adversarial domain adaptation”. In: *arXiv preprint arXiv:1711.03213* (2017).

[70] Judy Hoffman et al. “Efficient learning of domain-invariant image representations”. In: *arXiv preprint arXiv:1301.3224* (2013).

[71] Judy Hoffman et al. “LSDA: Large Scale Detection through Adaptation”. In: *Neural Information Processing Symposium (NIPS)*. 2014.

[72] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

[73] Nick Jakobi. "Evolutionary robotics and the radical envelope-of-noise hypothesis". In: *Adaptive behavior* 6.2 (1997), pp. 325–368.

[74] Nick Jakobi, Phil Husbands, and Inman Harvey. "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *European Conference on Artificial Life*. Springer. 1995, pp. 704–720.

[75] Stephen James, Michael Bloesch, and Andrew J Davison. "Task-embedded control networks for few-shot imitation learning". In: *arXiv preprint arXiv:1810.03237* (2018).

[76] Stephen James, Andrew J Davison, and Edward Johns. "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task". In: *arXiv preprint arXiv:1707.02267* (2017).

[77] Stephen James and Edward Johns. "3D Simulation for Robot Arm Control with Deep Q-Learning". In: *arXiv preprint arXiv:1609.03759* (2016).

[78] Stephen James et al. "Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks". In: *arXiv preprint arXiv:1812.07252* (2018).

[79] Kathy Jang et al. "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles". In: *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM. 2019, pp. 291–300.

[80] Hailin Jin et al. "Tales of shape and radiance in multiview stereo". In: *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE. 2003, pp. 974–981.

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

[82] Janne Karttunen et al. "From Video Game to Real Robot: The Transfer between Action Spaces". In: *arXiv preprint arXiv:1905.00741* (2019).

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

[84] Leonid Keselman et al. "Intel RealSense Stereoscopic Depth Cameras". In: *arXiv preprint arXiv:1705.05548* (2017).

[85] Rawal Khirodkar, Donghyun Yoo, and Kris Kitani. "Domain Randomization for Scene-Specific Car Detection and Pose Estimation". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1932–1940.

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

[87] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

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

[89] J Zico Kolter and Andrew Y Ng. "Learning omnidirectional path following using dimensionality reduction." In: *Robotics: Science and Systems*. 2007.

[90] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. "The transferability approach: Crossing the reality gap in evolutionary robotics". In: *IEEE Transactions on Evolutionary Computation* 17.1 (2013), pp. 122–145.

[91] Karim Koreitem et al. "Synthetically trained 3d visual tracker of underwater vehicles". In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE. 2018, pp. 1–7.

[92] Kristinn Kristinsson and Guy Albert Dumont. "System identification and control using genetic algorithms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 22.5 (1992), pp. 1033–1046.

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[94] Brian Kulis, Kate Saenko, and Trevor Darrell. "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 1785–1792.

[95] Ananya Kumar et al. "Consistent generative query networks". In: *arXiv preprint arXiv:1807.02033* (2018).

[96] Kiriakos N Kutulakos and Steven M Seitz. "A theory of shape by space carving". In: *International journal of computer vision* 38.3 (2000), pp. 199–218.

[97] Hugo Larochelle and Stanislas Lauly. "A neural autoregressive topic model". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2708–2716.

[98] Hugo Larochelle and Iain Murray. "The neural autoregressive distribution estimator". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 29–37.

[99] Quoc V Le et al. "Learning to grasp objects with multiple contact points". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 5062–5069.

[100] Yann LeCun. "Une procedure d'apprentissage ponr reseau a seuil asymetrique". In: *Proceedings of Cognitiva 85* (1985), pp. 599–604.

[101] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[102] Honglak Lee et al. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 609–616.

[103] Jeongseok Lee et al. “DART: Dynamic animation and robotics toolkit”. In: *The Journal of Open Source Software* 3.22 (2018), p. 500.

[104] Jurgen Leitner et al. “Artificial neural networks for spatial perception: Towards visual object localisation in humanoid robots”. In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–7.

[105] Ian Lenz, Honglak Lee, and Ashutosh Saxena. “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* 34.4-5 (2015), pp. 705–724.

[106] Sergey Levine et al. “End-to-end training of deep visuomotor policies”. In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[107] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* (2016), p. 0278364917710318.

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

[109] Ofir Levy and Lior Wolf. “Live repetition counting”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3020–3028.

[110] Yanghao Li et al. “Revisiting Batch Normalization For Practical Domain Adaptation”. In: *arXiv preprint arXiv:1603.04779* (2016).

[111] Vianney Loing, Renaud Marlet, and Mathieu Aubry. “Virtual training for a real application: Accurate object-robot relative localization without calibration”. In: *International Journal of Computer Vision* 126.9 (2018), pp. 1045–1060.

[112] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[113] Wenhan Luo et al. “End-to-end Active Object Tracking and Its Real-world Deployment via Reinforcement Learning”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[114] Jeffrey Mahler et al. “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”. In: *arXiv preprint arXiv:1703.09312* (2017).

[115] Jeffrey Mahler et al. “Dex-Net 3.0: Computing Robust Robot Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning”. In: *arXiv preprint arXiv:1709.06670* (2017).

[116] Jeffrey Mahler et al. "Learning ambidextrous robot grasping policies". In: *Science Robotics* 4.26 (2019), eaau4984.

[117] Jan Matas, Stephen James, and Andrew J Davison. "Sim-to-real reinforcement learning for deformable object manipulation". In: *arXiv preprint arXiv:1806.07851* (2018).

[118] Bhairav Mehta et al. "Active Domain Randomization". In: *arXiv preprint arXiv:1904.04762* (2019).

[119] Andrew T Miller and Peter K Allen. "Graspit! a versatile simulator for robotic grasping". In: *IEEE Robotics & Automation Magazine* 11.4 (2004), pp. 110–122.

[120] Andrew T Miller et al. "Automatic grasp planning using shape primitives". In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on.* Vol. 2. IEEE. 2003, pp. 1824–1829.

[121] Piotr Mirowski et al. "Learning to navigate in complex environments". In: *arXiv preprint arXiv:1611.03673* (2016).

[122] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. "A Self-supervised Learning System for Object Detection using Physics Simulation and Multi-view Pose Estimation". In: *arXiv preprint arXiv:1703.03347* (2017).

[123] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533.

[124] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[125] Luis Montesano et al. "Learning object affordances: from sensory–motor coordination to imitation". In: *IEEE Transactions on Robotics* 24.1 (2008), pp. 15–26.

[126] Antonio Morales et al. "Using experience for assessing grasp reliability". In: *International Journal of Humanoid Robotics* 1.04 (2004), pp. 671–691.

[127] Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids". In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE. 2015, pp. 5307–5314.

[128] Igor Mordatch et al. "Combining model-based policy search with online model learning for control of physical humanoids". In: *2016 IEEE International Conference on Robotics and Automation (ICRA).* IEEE. 2016, pp. 242–248.

[129] Daniel D Morris and Takeo Kanade. "Image-consistent surface triangulation". In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662).* Vol. 1. IEEE. 2000, pp. 332–338.

[130] Yair Movshovitz-Attias, Takeo Kanade, and Yaser Sheikh. "How useful is photo-realistic rendering for visual learning?" In: *Computer Vision–ECCV 2016 Workshops.* Springer. 2016, pp. 202–217.

[131] Fabio Muratore et al. “Domain Randomization for Simulation-Based Policy Optimization with Transferability Assessment”. In: *Conference on Robot Learning*. 2018, pp. 700–713.

[132] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[133] Oliver Nelles. *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[134] Ramakant Nevatia and Thomas O Binford. “Description and recognition of curved objects”. In: *Artificial Intelligence* 8.1 (1977), pp. 77–98.

[135] Van-Duc Nguyen. “Constructing force-closure grasps”. In: *The International Journal of Robotics Research* 7.3 (1988), pp. 3–16.

[136] Duy Nguyen-Tuong and Jan Peters. “Model learning for robot control: a survey”. In: *Cognitive processing* 12.4 (2011), pp. 319–340.

[137] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks”. In: *arXiv preprint arXiv:1601.06759* (2016).

[138] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[139] Aaron van den Oord et al. “Conditional Image Generation with PixelCNN Decoders”. In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee et al. Curran Associates, Inc., 2016, pp. 4790–4798. URL: `http://papers.nips.cc/paper/6527-conditional-image-generation-with-pixelcnn-decoders.pdf`.

[140] OpenAI. “Learning dexterous in-hand manipulation”. In: *arXiv preprint arXiv:1808.00177* (2018).

[141] Raphael Pelossof et al. “An SVM learning approach to robotic grasping”. In: *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*. Vol. 4. IEEE. 2004, pp. 3512–3518.

[142] Xingchao Peng et al. “Learning deep object detectors from 3D models”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1278–1286.

[143] Xue Bin Peng et al. “Sim-to-real transfer of robotic control with dynamics randomization”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[144] Lerrel Pinto, James Davidson, and Abhinav Gupta. “Supervision via competition: Robot adversaries for learning tasks”. In: *arXiv preprint arXiv:1610.01685* (2016).

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

[146] Lerrel Pinto et al. "Asymmetric actor critic for image-based robot learning". In: *arXiv preprint arXiv:1710.06542* (2017).

[147] Benjamin Planche et al. "DepthSynth: Real-Time Realistic Synthetic Data Generation from CAD Models for 2.5 D Recognition". In: *arXiv preprint arXiv:1702.08558* (2017).

[148] Riccardo Polvara et al. "Autonomous quadrotor landing using deep reinforcement learning". In: *arXiv preprint arXiv:1709.03339* (2017).

[149] Domenico Prattichizzo and Jeffrey C Trinkle. "Grasping". In: *Springer handbook of robotics*. Springer, 2016, pp. 955–988.

[150] Domenico Prattichizzo et al. "On the manipulability ellipsoids of underactuated robotic hands with compliance". In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 337–346.

[151] Aravind Rajeswaran et al. "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles". In: *arXiv preprint arXiv:1610.01283* (2016).

[152] Param S Rajpura, Hristo Bojinov, and Ravi S Hegde. "Object detection using deep cnns trained on synthetic images". In: *arXiv preprint arXiv:1706.06782* (2017).

[153] Param Rajpura et al. "Transfer learning by finetuning pretrained CNNs entirely with synthetic images". In: *National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics*. Springer. 2017, pp. 517–528.

[154] Scott Reed et al. "Few-shot autoregressive density estimation: Towards learning to learn distributions". In: *arXiv preprint arXiv:1710.10304* (2017).

[155] Xinyi Ren et al. "Domain Randomization for Active Pose Estimation". In: *arXiv preprint arXiv:1903.03953* (2019).

[156] Danilo Jimenez Rezende et al. "Unsupervised learning of 3d structure from images". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4996–5004.

[157] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. "Playing for benchmarks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2213–2222.

[158] Stephan R Richter et al. "Playing for data: Ground truth from computer games". In: *European Conference on Computer Vision*. Springer. 2016, pp. 102–118.

[159] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. "From caging to grasping". In: *The International Journal of Robotics Research* 31.7 (2012), pp. 886–900.

[160] Mikko Ronkainen et al. "Dense tracking of human facial geometry-aware". In: (2017).

[161] Carlos Rosales et al. "On the synthesis of feasible and prehensile robotic grasps". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 550–556.

[162] Dan Rosenbaum et al. "Learning models for visual 3D localization with implicit mapping". In: *arXiv preprint arXiv:1807.03149* (2018).

[163] Zachary E Ross et al. "PhaseLink: A deep learning approach to seismic phase association". In: *Journal of Geophysical Research: Solid Earth* 124.1 (2019), pp. 856–869.

[164] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[165] Andrei A Rusu et al. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[166] Andrei A Rusu et al. "Sim-to-real robot learning from pixels with progressive nets". In: *arXiv preprint arXiv:1610.04286* (2016).

[167] Fereshteh Sadeghi. "DIViS: Domain Invariant Visual Servoing for Collision-Free Goal Reaching". In: *arXiv preprint arXiv:1902.05947* (2019).

[168] Fereshteh Sadeghi and Sergey Levine. "(CAD)ˆ2 RL: Real Single-Image Flight without a Single Real Image". In: *arXiv preprint arXiv:1611.04201* (2016).

[169] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. "An overview of 3D object grasp synthesis algorithms". In: *Robotics and Autonomous Systems* 60.3 (2012), pp. 326–336.

[170] Manolis Savva et al. "MINOS: Multimodal indoor simulator for navigation in complex environments". In: *arXiv preprint arXiv:1712.03931* (2017).

[171] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. "Robotic grasping of novel objects using vision". In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173.

[172] Ashutosh Saxena, Lawson LS Wong, and Andrew Y Ng. "Learning Grasp Strategies with Partial Shape Information." In: *AAAI*. Vol. 3. 2. 2008, pp. 1491–1494.

[173] Johannes L Schonberger and Jan-Michael Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113.

[174] John D Schulman, Ken Goldberg, and Pieter Abbeel. "Grasping and fixturing as submodular coverage problems". In: *Robotics Research*. Springer, 2017, pp. 571–583.

[175] John Schulman et al. "Trust Region Policy Optimization." In: *ICML*. 2015, pp. 1889–1897.

[176] Steven M Seitz and Charles R Dyer. "Photorealistic scene reconstruction by voxel coloring". In: *International Journal of Computer Vision* 35.2 (1999), pp. 151–173.

[177] Steven M Seitz et al. "A comparison and evaluation of multi-view stereo reconstruction algorithms". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. IEEE. 2006, pp. 519–528.

[178] Jake Sganga et al. "OffsetNet: Deep Learning for Localization in the Lung using Rendered Images". In: *arXiv preprint arXiv:1809.05645* (2018).

[179] Evan Shelhamer et al. "Loss is its own reward: Self-supervision for reinforcement learning". In: *arXiv preprint arXiv:1612.07307* (2016).

[180] Roger N Shepard and Jacqueline Metzler. "Mental rotation of three-dimensional objects". In: *Science* 171.3972 (1971), pp. 701–703.

[181] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), p. 484.

[182] David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

[183] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[184] Avi Singh, Larry Yang, and Sergey Levine. "Gplac: Generalizing vision-based robotic skills using weakly labeled images". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5851–5860.

[185] Avinash Siravuru et al. "Deep visual perception for dynamic walking on discrete terrain". In: *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE. 2017, pp. 418–424.

[186] Russell Smith et al. "Open dynamics engine". In: (2005).

[187] Hao Su et al. "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2686–2694.

[188] Ioan A Sucan and Sachin Chitta. "Moveit!" In: *http://moveit.ros.org* ().

[189] Baochen Sun and Kate Saenko. "From Virtual to Reality: Fast Adaptation of Virtual Object Detectors to Real Domains." In: *BMVC*. Vol. 1. 2. 2014, p. 3.

[190] Martin Sundermeyer et al. "Implicit 3d orientation learning for 6d object detection from rgb images". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 699–715.

[191] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[192] Richard Szeliski. "A multi-view approach to motion and stereo". In: *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. Vol. 1. IEEE. 1999, pp. 157–163.

[193] Yaniv Taigman, Adam Polyak, and Lior Wolf. "Unsupervised Cross-Domain Image Generation". In: *arXiv preprint arXiv:1611.02200* (2016).

[194] Jie Tan et al. "Sim-to-real: Learning agile locomotion for quadruped robots". In: *arXiv preprint arXiv:1804.10332* (2018).

[195] Jie Tang et al. "A textured object recognition pipeline for color and depth image data". In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3467–3474.

[196] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. "Multi-view 3d models from single images with a convolutional network". In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337.

[197] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey". In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[198] Josh Tobin, OpenAI Robotics, and Pieter Abbeel. "Geometry-Aware Neural Rendering". In: *in preparation* (2019).

[199] Josh Tobin et al. "Domain randomization and generative models for robotic grasping". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3482–3489.

[200] Josh Tobin et al. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: *arXiv preprint arXiv:1703.06907* (2017).

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

[202] Jonathan Tremblay et al. "Deep object pose estimation for semantic robotic grasping of household objects". In: *arXiv preprint arXiv:1809.10790* (2018).

[203] Jonathan Tremblay et al. "Synthetically trained neural networks for learning human-readable plans from real-world demonstrations". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–5.

[204] Jonathan Tremblay et al. "Training deep networks with synthetic data: Bridging the reality gap by domain randomization". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 969–977.

[205] Joshua C Triyonoputro, Weiwei Wan, and Kensuke Harada. "Quickly Inserting Pegs into Uncertain Holes using Multi-view Images and Deep Network Trained on Synthetic Data". In: *arXiv preprint arXiv:1902.09157* (2019).

[206] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. "Multi-view consistency as supervisory signal for learning shape and pose prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2897–2905.

[207] Eric Tzeng et al. "Adapting deep visuomotor representations with weak pairwise constraints". In: *arXiv preprint arXiv:1511.07111* (2015).

[208] Eric Tzeng et al. "Adapting deep visuomotor representations with weak pairwise constraints". In: *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. 2016.

[209] Eric Tzeng et al. "Deep domain confusion: Maximizing for domain invariance". In: *arXiv preprint arXiv:1412.3474* (2014).

[210] Jur Van Den Berg et al. "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 2074–2081.

[211] Gul Varol et al. "Learning from synthetic humans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 109–117.

[212] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[213] Ulrich Viereck et al. "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images". In: *arXiv preprint arXiv:1706.04652* (2017).

[214] Xiaolong Wang et al. "Non-local neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7794–7803.

[215] Daniel Ward, Peyman Moghadam, and Nicolas Hudson. "Deep leaf segmentation using synthetic data". In: *arXiv preprint arXiv:1807.10931* (2018).

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

[217] Peter Welinder et al. *Robots that Learn*. May 2017. URL: `https://openai.com/blog/robots-that-learn/`.

[218] Paul Werbos. "Beyond Regression:" New Tools for Prediction and Analysis in the Behavioral Sciences". In: *Ph. D. dissertation, Harvard University* (1974).

[219] Paul J Werbos. "Neural networks for control and system identification". In: *Proceedings of the 28th IEEE Conference on Decision and Control,* IEEE. 1989, pp. 260–265.

[220] Melonee Wise et al. "Fetch and freight: Standard platforms for service robot applications". In: *Workshop on Autonomous Mobile Service Robots*. 2016.

[221] Jiajun Wu et al. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems*. 2016, pp. 82–90.

[222] Yi Wu et al. "Building generalizable agents with a realistic and rich 3d environment". In: *arXiv preprint arXiv:1801.02209* (2018).

[223] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). URL: `http://arxiv.org/abs/1609.08144`.

[224] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[225] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. "Addressing appearance change in outdoor robotics with adversarial domain adaptation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1551–1558.

[226] Patrick Wunsch and Gerd Hirzinger. "Real-time visual tracking of 3D objects with dynamic handling of occlusion". In: *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. Vol. 4. IEEE. 1997, pp. 2868–2873.

[227] Ziang Xie et al. "Multimodal blending for high-accuracy instance recognition". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE. 2013, pp. 2214–2221.

[228] Mengyuan Yan et al. "Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control". In: *arXiv preprint arXiv:1712.03303* (2017).

[229] Xinchen Yan et al. "Learning grasping interaction with geometry-aware 3D representations". In: *arXiv preprint arXiv:1708.07303* (2017).

[230] Xinchen Yan et al. "Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1696–1704.

[231] Jason Yosinski et al. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems*. 2014, pp. 3320–3328.

[232] Tianhe Yu et al. "One-shot imitation from observing humans via domain-adaptive meta-learning". In: *arXiv preprint arXiv:1802.01557* (2018).

[233] Wenhao Yu, C Karen Liu, and Greg Turk. "Preparing for the Unknown: Learning a Universal Policy with Online System Identification". In: *arXiv preprint arXiv:1702.02453* (2017).

[234] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. "DeceptionNet: Network-Driven Domain Randomization". In: *arXiv preprint arXiv:1904.02750* (2019).

[235] Amir R Zamir et al. "Generic 3d representation via pose estimation and matching". In: *European Conference on Computer Vision*. Springer. 2016, pp. 535–553.

[236] Andy Zeng et al. "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 1386–1383.

[237] Fangyi Zhang et al. "Adversarial Discriminative Sim-to-real Transfer of Visuo-motor Policies". In: *arXiv preprint arXiv:1709.05746* (2017).

[238] Fangyi Zhang et al. "Sim-to-real Transfer of Visuo-motor Policies for Reaching in Clutter: Domain Randomization and Adaptation with Modular Networks". In: *arXiv preprint arXiv:1709.05746* (2017).

[239] Fangyi Zhang et al. "Vision-Based Reaching Using Modular Deep Networks: from Simulation to the Real World". In: *CoRR* abs/1610.06781 (2016). URL: `http://arxiv.org/abs/1610.06781`.

[240] Jingwei Zhang et al. "Vr-goggles for robots: Real-to-sim domain adaptation for visual control". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1148–1155.

[241] Tianhao Zhang et al. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[242] Yiran Zhong, Yuchao Dai, and Hongdong Li. "Self-supervised learning for stereo matching with self-improving ability". In: *arXiv preprint arXiv:1709.00930* (2017).

[243] Tinghui Zhou et al. "View synthesis by appearance flow". In: *European conference on computer vision*. Springer. 2016, pp. 286–301.

[244] Yiming Zuo et al. "CRAVES: Controlling Robotic Arm with a Vision-based Economic System". In: ().

[245] Yiming Zuo et al. "Towards Accurate Task Accomplishment with Low-Cost Robotic Arms". In: *arXiv preprint arXiv:1812.00725* (2018).