LEARNING GENERALIZABLE ROBOT POLICIES WITH TARGETED GENERATIVE AUGMENTATION

QIUYU CHEN

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington
2024

Reading Committee:
Dieter Fox, Chair
Abhishek Gupta, Chair
Ranjay Krishna

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science & Engineering

University of Washington

ABSTRACT

LEARNING GENERALIZABLE ROBOT POLICIES WITH TARGETED GENERATIVE AUGMENTATION

Qiuyu Chen

Chairs of the Supervisory Committee:
Dieter Fox
Abhishek Gupta
Paul G. Allen School of Computer Science & Engineering

Visual Imitation Learning methods have the potential to generalize across diverse tasks and environments but often encounter challenges in acquiring large and diverse datasets due to the high costs of real-world data collection. This dissertation introduces Targeted Generative Augmentation, a systematic approach designed to generate data that closely mimics real-world distributions. This method aims to effectively diversify the initial dataset that robots are more likely to encounter, thereby bridging the data scarcity gap.

In this thesis, I present realistic data generation techniques that help robots generalize to unseen scenarios. The dissertation outlines three main approaches: Static Targeted Augmentation for Visual Diversity, Dynamic Targeted Augmentation for Physical Realism, and Contextual Targeted Augmentation for Real-World Scene Generation. These methods generate data with visual realism and the complexity of the real world, which help to bootstrap the robot policy and improve generalization in unfamiliar environments.

To my husband Aaron,
my love.

ACKNOWLEDGEMENTS

I first must thank my family: my husband Aaron, my parents Wen and Hongze, my in-laws Sam, John and brother-in-law Arlo, and our dog Toast. I want to thank Aaron, for his enormous love and support every single day, his home-made chai every morning, a surprising cake with literally "65%" on it after my experiment bumped to that number, a random book "curse breaker" after the first paper in my PhD because I was joking of being cursed on publications, the best road trip to the East Coast with a car loaded with stuffed animals and board games, waving our arms running downstairs and shouting "MOVIE NIGHT!!!" to pick up fried chicken on many Friday nights, staying up late on so many nights and taking care of me through the crazy deadlines. He is there for me when the time is good and the time is tough. I want to thank my parents for their love and encouragement since I was a kid, and for their care and support for all my decisions. I want to first thank my mom for being the goofiest explorer, she is the one who always tries everything with the biggest excitement. Aaron always said I have a great imagination and I know that's from her. My dad introduced me to science when I was a kid, going to his chemistry lab on weekends was always an adventure. He even started online courses after my PhD to learn about AI. His curiosity about the world has always been an inspiration to me. I'm extremely grateful for their support and understanding during my PhD. I also want to thank my family-in-law. I have not been able to travel home due to the pandemic and visa, but they have given me a family here with their love, support, tasty food, infinite cookies and snacks. I'm grateful for all the time we've spent together: Decorating Christmas cookies, surprising new dessert Mom made, playing exit games, family karaoke, and sinking into the couch and chatting about life. I'm also grateful for our puppy Toast for bringing us happiness and joy. I am extremely lucky to have all my family in my life.

This dissertation would not have been possible without their love and support.

Next I must thank my advisors Dieter Fox and Abhishek Gupta. I want to thank Dieter for his patience and support throughout my PhD, especially when things got tough, and always cheered me on when I made progress. His sharp observation always yields creative insights, interesting discussions and ideas. Additionally, a huge thank you to Dieter who has been the key to getting our robotics space and bringing brilliant people together, making our collaborative efforts possible. I am extremely grateful to Abhishek for his constant support and mentorship in helping me navigate through the ups and downs of my research and steering me toward collaboration and internship opportunities. More than just the projects, I really appreciate his readiness to offer advice and guidance as I planned out key decisions and my career path. I came to PhD without much robotics background, I am forever grateful to them for taking a chance on me and shaping me into who I am. Their insights and advice have been really important to my personal and professional growth.

I would also like to thank my other committee members Ranjay Krishna and Ashis Banerjee for taking their time helping me in this PhD journey. Their suggestions and discussion have greatly improved this work.

I must also thank my other collaborators over various projects during my PhD and internships, Karl Van Wyk, Yu-Wei Chao, Wei Yang, Arsalan Mousavian, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Sho Kiami, Vikash Kumar, Karthikeya Vemuri, Alan Wu. It would take longer for me to finish those projects without their insightful discussions and help.

I would also like to thank all my other advisors and mentors who led me to the PhD journey. First, I'm extremely grateful to my first academic advisor Cezhou Zhao for taking me into research. I still remember being a freshman pitching an idea of building some crazy sci-fi-ish portable projector, literally door-to-door to every professor in my department (This is true). Cezhou gave me a desk in the lab and encouraged me to do research on the idea and figure out steps to

build it. My sci-fi idea did not work out, but he introduced me to the research of optics and photonics. He encouraged me to read a lot and discuss with senior students, which later led to successful research publications. I am forever grateful for his trust since the very first day. I am super grateful to my advisors and mentors Jenq-Neng Hwang, Imari Sato, Ryoma Bise and Yingqiang Zheng during my master's program, for making research encouraging and fun for me. Without them, I might have gotten lost before my PhD.

Life in PhD will be boring without the support and care from my friends. I especially want to thank Kay Ke from Personal Robotics Lab who is also my neighbor next door, for numerous food exchanges, cooking, movies, board games, walks, gossiping, and commiserating since the very first quarter of my PhD! I also want to thank all members of the Robotics and State Estimation Lab at the University of Washington for not only helping shape and refine these ideas over the years but also providing support through the ups and downs of graduate studies. Specifically Daniel Gordon, Chris Xie, Arunkumar Byravan, Junha Roh, Karthik Desingh, Xiangyun Meng, Adam Fishman, Yi Li, Mohit Shridhar, Aaron Walsman, Jens Lundell, Markus Grotz, Wentao Yuan, Marius Memmel, Jiafei Duan and Helen Wang. Similarly, the WEIRD lab at the University of Washington was an invaluable source of wisdom, inspiration, and joy, especially our weekly reading group and many social events, which were a constant source of new ideas and inspiration. Specifically Tyler Westenbroek, Chuning Zhu, Marius Memmel, Mateo Guaman Castro, Daphne Chen, Sriyash Poddar, Yunchu Zhang, Patrick Yin, Max Balsells and Marcel Torne. I must also thank other members of the CSE Robotics Lab for the casual chat about life, complaining about robots and food sharing, including Matt Schmittle, Vinitha Ranganeni, Bernie Zhu, Entong Su, Sidharth Talia, Octi Zhang, Anqi Li, Boling Yang, Carolina Higuera Arias, Rosario Scalise, Tyler Han, Rohan Baijal, Yuquan "Nil" Deng, Nick Walker, Michael Murray, Motoya Ohnishi, Sanjar Normuradov, Amal Nanavati, Ethon Gordon, Selest Nashef, Amirreza Shaban, Brian Hou and many others.

I want to also thank friends of the larger computer science community at the University of Washington and outside UW who made my journey fun and delightful. Specifically Sewon Min, Tim Dettmers, Victor Zhong, Sally Dong, Ofir Press, Keren Qin, Judit Acs for food, chats, picnics, hikings, and trips together earlier during my PhD, Ari Holtzman, Kendall Lowery, Aravind Rajeswaran for insightful conversations and catch-ups, Sehee Min, Mandi Zhao, Homanga Bharadhwaj, Jay Vakil, Unnat Jain, Priyam Parashar, Sudharshan Suresh, Wenxuan Zhou and many others for chats, gossips, picnics, boba, food and making my internships a fun experience.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Humans interact with the world through a complex integration of sensory experiences and actions, primarily relying on visual cues to navigate and manipulate their environment. This intuitive human ability to learn directly from visual information motivates the development of robotic systems that can similarly learn to react and interact with the environment through direct visual perception. Visual imitation learning aims to enable robots to observe their surroundings and predict potential interactions by processing sequences of images or video frames. This method leverages human expert demonstrations to teach robots different manipulation skills given the observation of the robot environment.

To train robots to autonomously execute a series of tasks, expert data is typically collected through demonstrations by human operators. The observation space in visual imitation learning usually consists of visual inputs such as RGB images or depth maps, representing the robot's view of the environment, while the action space is defined by specific movements a robot should perform, often represented as a series of robot joint angles or sequences of end-effector poses.

However, data collection methods that typically use kinesthetic teaching and teleoperation can be expensive and time-consuming, significantly restricting our ability to efficiently collect large-scale data in diverse environments. This is crucial because having access to diverse and comprehensive data sets is essential for enabling robot policies to generalize effectively in unseen scenarios.

Given these constraints, it is essential to explore alternative strategies for expanding the initial datasets without expensive new data collection. As shown in Figure 1, Augmentation is a widely used approach that typically involves techniques such as rotating, scaling, or cropping images to artificially increase the dataset's size and vari-

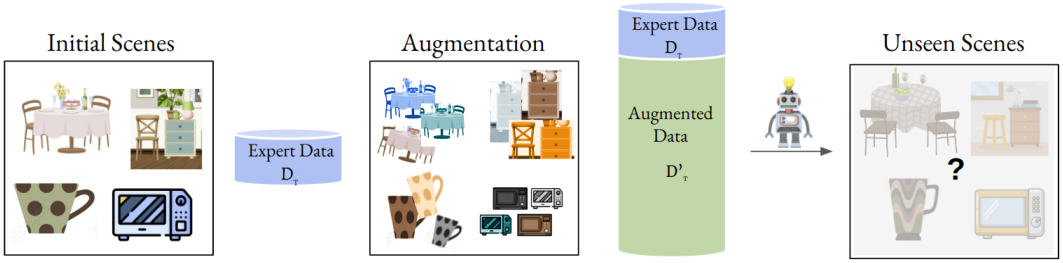


Figure 1: Starting with a small amount of expert data from a limited range of scenes, data augmentation is a technique that automatically produces a more varied dataset. Can the expanded dataset be effectively used to train a robot policy and generalize better at unseen scenarios?

ability. Additionally, augmentation sometimes may also include randomly generating layouts or inserting random background images to further diversify the scenarios. The goal of the augmentation is to expand the initial dataset and train robots to generalize better at unfamiliar situations. However, prior methods tend to focus narrowly on the initial data distribution and barely extend to more diverse scenarios. Moreover, they often fail to accurately represent the real-world distributions that a robot might encounter, resulting in limited generalizable behaviors in unseen scenarios.

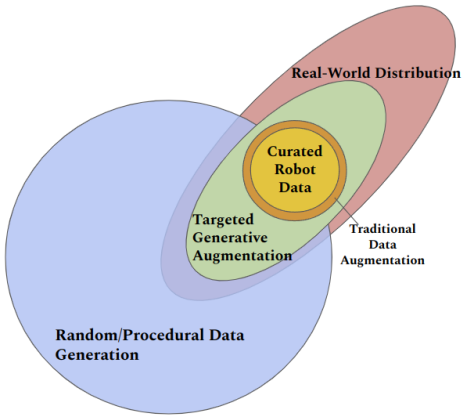


Figure 2: Illustration of Targeted Generative Augmentation: Collecting data from real-world robots is both costly and time-consuming. To address the complexities of the real world (indicated by the narrow pink region), one way to cheaply generate data is through traditional data augmentation such as random crops which does not introduce new semantic diversity (shown in the orange region). Another common strategy is to use a random or procedural generation but these approaches may not accurately reflect the real-world distribution. We propose targeted generative augmentation (shown in green region), which utilizes the initially limited set of robot data to emulate real-world conditions better.

In this dissertation, I introduce the concept of Targeted Generative Augmentation, as illustrated in Figure 2, an approach that aims to generate data that closely mimics real-world distribution. The goal of this method is to augment the initial dataset more effectively that a robot is more likely to encounter. In the following chapters, I will explore the potential benefits of targeted generative augmentation across a range of tasks in robot learning.

In Part I: Static Targeted Augmentation for Visual Diversity, I show how Targeted Generative Augmentation can be effectively used to diversify the observation space of the robot, enabling it to generalize better in previously unseen scenarios. In particular, I propose GenAug, a system that semantically augments the scenes in a controllable way by introducing variances such as new textures, distractor objects and backgrounds that are close to real-world scenarios. This controlled augmentation data is used to train the robot and improve its performance to generalize to entirely unseen environments.

In Part II: Dynamic Targeted Augmentation for Physical Realism, I discuss how to apply targeted augmentation to not only diversity static observations but also actively refine their actions in the robot grasping settings, by leveraging targeted augmentation with simulation. Particularly, I introduce ISAGrasp, a method that automatically generates diverse object shapes from an initial small dataset, as well as their corresponding successful grasping poses. This is achieved by applying a correspondence-aware generative model and leveraging simulation to effectively produce physically accurate data from a limited initial set of Mocap data.

In Part III: Contextual Targeted Augmentation for Real-World Scene Generation, I explore a challenging direction in the setting of realistic content generation. I discuss how Targeted Generative Augmentation can be conditioned on real images, ensuring that the generated dataset aligns more closely with real-world distributions. Specifically, I introduce URDFormer, a pipeline that predicts the detailed kinematic structure of articulated objects and scenes directly from RGB images. This allows large-scale generation directly from internet images, making simulation content creation significantly faster and

more scalable. By training robots in these real-world-inspired simulated environments, we can train a more robust robot policy that transfers better in real-world settings.

2

BACKGROUND AND RELATED WORK

Visual Policy Learning In the realm of robot learning, the choice of data modality is critical for achieving generalization. Vision data became particularly important because it captures intricate details necessary for complex tasks such as spatial reasoning and object manipulation. visual data can effectively form the backbone of effective robotic control policies, as shown by recent works [39, 47, 49, 88, 102, 103, 113, 136, 167]. One key step in training a generalizable visual policy for robots is to collect and train on diverse data such that the policies are robust and adaptable to understanding and interacting with their environment effectively. Recent studies have explored ways to expand the volume and variety of visual data for robot learning. A significant portion of this research is centered on gathering and analyzing data directly generated by robots [10, 173]. However, these works typically involve only a limited number of distinct environments, posing challenges for the robots to generalize effectively across a broader spectrum of unfamiliar scenes. A substantial body of research also focuses on learning image representations beyond robot demonstration data such as large-scale videos and images [83, 103, 120]. Moreover, several studies have focused on using language to learn representations from videos [98, 172].

Data Augmentation In the absence of diverse data, a promising direction is finding ways to inject structure directly into learned models for widespread generalization. The most widely used technique is various forms of data augmentation [133], such as cropping, shifting, noise injection and rotation. These methods have been used in many robot learning approaches and provide a significant improvement in data efficiency [7, 26, 71, 134]. For example, [169] investigate different augmentation modes in Meta-learning settings. In addition, several methods attempt to enforce geometric invariance through ar-

chitectural innovations such as [149] and [31]. While these methods can provide a local notion of robustness and invariance to perceptual noise, they do not provide generalization to novel object shapes or scenes. In a similar vein, domain randomization is used in robotics, where predefined parameters such as lighting, camera are randomized during training[144, 146] in order to learn a policy that is invariant to these parameters. However, this randomization does not aid with generalization to novel object shapes. It is essential to have an augmentation approach that can diversify the initial data while maintaining realism.

Leveraging Simulation Data and Scene Generation A natural solution to this problem is to leverage simulation data [30]. However, while this method is efficient at generating a large *quantity* of data, it can be challenging to create diverse content (objects meshes, physics, layouts, and visual appearances), and often the resulting simulations exhibit a significant gap from reality. To address this issue, domain randomization [144, 146] varies multiple simulation parameters in order to boost the effectiveness of trained policies in the real world. Randomization of lighting and camera parameters during training can allow a policy to be invariant to the effects of lighting and visual perturbations in the real world. Physics parameters of the scene can also be randomized to transfer policies trained in simulation to the real world [143]. While effective, challenges in creating a simulation with visual and physical realism for every task and behavior severely restrict the applicability of these methods to isolated known tasks and limited diversity. Furthermore, they require the user to define augmentation parameters and their ranges which can be very non-trivial for complex tasks [3, 50]. On the other hand, inverse graphics is a well-studied and rich field of study looking to infer the properties of a scene from images (or videos) of the scene of interest. A variety of work focuses on inferring scene properties such as geometry, lighting, and other geometric properties from single images [6]. This work has both been optimization-based [2] and learning-based[109]. In a similar vein, a rich body of work [129] focuses on mesh reconstruction and novel view synthesis using a variety of techniques such as

implicit neural fields [93, 110, 171], Gaussian splatting [65, 81], differentiable rendering [63, 76, 107] amongst many other techniques. Importantly, the focus of many of these works on inverse graphics has been on geometric reconstruction rather than our focus on scene-level simulation construction complete with kinematic and semantic structure like object relationships and articulation. There have been a number of efforts in inferring physical properties such as articulation [55, 160, 161], friction and surface properties [5, 72, 112, 157], although these typically require either interaction or video access. In contrast, our work focuses less on exact geometry reconstruction but rather on generating correct scene statistics at the articulation/kinematics/positioning level for entire scenes or complex objects from single RGB images. As opposed to these methods, the goal is not just a slow and expensive process for a single scene, but a fast generation process that can scale to generate hundreds of scenes with natural statistics. Importantly, this generation process does not require interaction or targeted data collection per domain. Generating indoor scenes is a long-standing problem in computer vision and machine learning. This has been approached by building learned generative models of indoor scenes [58, 66, 74, 124] and floorplans [56, 106, 150], while others have produced text-to-scene models [14, 15]. While generating scenes this way can be promising, these methods either fail to achieve the targeted generation of complex scenes with articulation and complex kinematic structure intact or require extremely expensive inference processes to do so. On the other hand, procedural generation techniques have been popular in generating grid-world environments [34, 37, 42, 67] and in generating home environments at scale [29]. These scenes are diverse and often rich, but are not controllable to particular target scenes or are not able to generate scenes complete with physical properties and articulation. Other techniques such as [27, 75] are able to generate large datasets of more interactive scenes but require interactive scanning with either a phone or other hardware for dataset generation specific to indoor scenes.

Asset Creation for Robot Manipulation Constructing realistic and diverse assets for robot learning and control has been an important

goal for many years. Early efforts [53] used geometric approaches to reconstruct static 3D scenes. Recently learned approaches [65, 93, 109] have in some cases improved the accuracy and visual realism of these approaches. While these techniques can offer incredible visual realism, they do not usually produce articulated models that allow for dynamic interaction.

To address this, many prior works have explored utilizing assets that are manually curated [28, 70, 141, 159], procedural-generated [29, 166], or scanned from the real world [78, 90, 158], to train robot manipulation policies. While the learned policies are supposed to apply directly to test scenes, there could be challenging scenes in which the systems fail to faithfully recognize and interact with the scene due to the scene complexity or sim-to-real gaps. In our work, we resort to a targeted system that first reconstructs a URDF given the test scene and then performs scene-specific training to learn a specialized model in the scene. There is also a rich body of literature on building digital twins of articulated objects from the real world, either based on passive visual observations such as 2D multi-view images or videos [54, 77, 117, 154, 164], 3D RGB-D, depth images or point clouds [1, 152, 155, 163], full-scene 3D scans [27, 75, 89], or through interactive perception which requires few-shot interactions with the object [55, 61, 64, 82, 108, 138]. While in some cases, these methods can successfully produce high quality models, they require either video, depth sensing or interaction with the objection which is more difficult to aquire and is rarely available in online internet data.

Part I

STATIC TARGETED AUGMENTATION FOR VISUAL DIVERSITY

"Logic will get you from A to B. Imagination will take you everywhere."

— Albert Einstein

In this chapter, we present GenAug, a semantic data augmentation framework that leverages pre-trained text-to-image generative models to train a generalizable robot policy. Visual imitation learning often requires a heavy burden on data collection by human experts. GenAug addresses this challenge by automatically generating "augmented" RGBD images that display the visual complexity and diversity that a robot might encounter in the real world. Particularly, given a dataset of image-action examples obtained from a human expert, GenAug leverages a diffusion model to modify object textures, shapes, and backgrounds, while ensuring physical consistency with the original scene. By training language-conditioned policy on this augmented dataset, GenAug enables a notable improvement in generalization capabilities across different tasks, allowing them to perform at entirely unseen real-world environments, even when provided with only a limited number of demonstrations collected in a single, simple environment. This contains material originally published as "GenAug: Retargeting behaviors to unseen situations via **Gen**erative **Aug**mentation" at RSS '23[19].

3

GENAUG: GENERATIVE AUGMENTATION FOR REAL-WORLD DATA COLLECTION

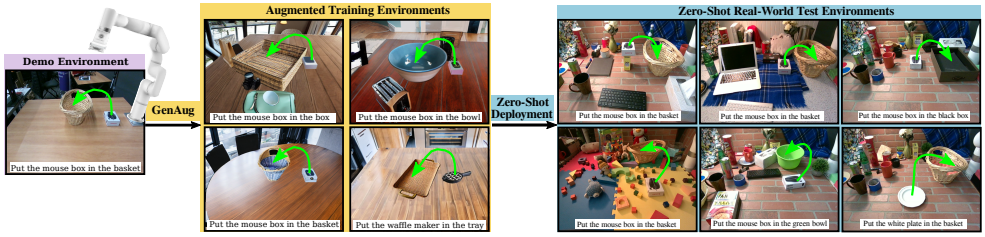


Figure 3: An illustration of the problem setting for our proposed system GenAug. GenAug takes a small set of image-action demonstration data on a robotics problem like tabletop pick-and-place and generates a diverse set of augmented image observations to supplement the real-world demonstration dataset. These augmented observations add semantically meaningful visual diversity in objects, distractors, and backgrounds while maintaining functional invariance of the actions. Training on this augmented training set leads to significant improvements in policy generalization, without requiring additional data collection.

3.1 BACKGROUND

While robot learning has often focused on the search for optimal behaviors [73, 101] or plans [119], the power of learning methods in robotics comes from the potential for *generalization*. While techniques such as motion planning or trajectory optimization are effective ways to solve the policy search problem in highly controlled situations such as warehouses or factories, they may fail to generalize to novel scenarios without significant environment modeling and replanning [40]. On the other hand, techniques such as imitation learning and reinforcement learning have the *potential* for widespread generalization without significant environment modeling and replanning, especially when combined with deep neural network function approximators [11, 62, 84].

Let us consider this question of learning from human demonstrations [114]. While imitation learning methods circumvent the challenges of exploration, these methods often impose a heavy burden on data collection by human supervisors. Human demonstrations are often collected by expensive techniques such as on-robot teleoperation or kinesthetic teaching, which limit the amount of real-world data that can be collected. Beyond the sheer quantity of data, the rigidity of most robotics setups makes it non-trivial to collect *diverse* data in a wide variety of scenarios. As a result, many robotics datasets involve a single setup with just a few hours of robot data. This is in stark contrast to the datasets that are common in vision and language problems [32, 130], both in terms of quantity *and* the diversity of the data. Given how important large-scale data has been for generalization in these domains, robot learning is likely to benefit from access to a similar scale of data.

Data augmentation can provide a way to improve model generalization, but these techniques typically perform augmentation in low-level visual space, performing operations such as color jitter, Gaussian blurring, and cropping, among others. While this may help with generalization to low-level changes in scene appearance, they are unable to handle large semantic differences in the scene such as distractors, background changes, or object appearance changes. In this work, we aim to provide *semantic* data augmentation to enable broad robot generalization, by leveraging pre-trained generative models. While on-robot data can be limited, the data that pre-trained generative models are exposed to is significantly larger and more diverse [32, 130]. Our work aims to leverage these generative models as a source of data augmentation for real-world robot learning. The key idea of our work builds on a simple intuition: demonstrations for solving one task in one environment should still largely be applicable to the same task in new environments despite the visual changes in scenes, background, and object appearances. The small amount of on-robot experience provides demonstrations of the desired behavior, while a generative model can be used to generate widely varying visual scenes, with diverse backgrounds and object appearances under which the

same behavior will still be valid. Furthermore, since such generative models are trained on realistic data, the generated scenes are visually realistic and extremely diverse. This allows us to cheaply generate a large quantity of *semantically augmented* data from a small number of demonstrations, providing a learning agent access to significantly more diverse scenes than the purely on-robot demonstration data. As we show empirically, this can lead to widely improved generalization, with minimal additional burden on human data collection.

We present GenAug, a semantic data augmentation framework that leverages pre-trained text-to-image generative models for real-world robot learning via imitation. Given a dataset of image-action examples provided on a real robot system, GenAug automatically generates "augmented" RGBD images for entirely different and realistic environments, which display the visual realism and complexity of scenes that a robot might encounter in the real world. In particular, GenAug leverages language prompts with a generative model to change object textures and shapes, adding new distractors and background scenes in a way that is physically consistent with the original scene, for table-top manipulation tasks with a real robot.

We show that training on this *semantically augmented* dataset significantly improves the generalization capabilities of imitation learning methods on entirely unseen real-world environments, with only 10 real-world demonstrations collected in a single, simple environment.

In summary, our key contributions are:

1. We present a general framework for leveraging generative models for data augmentation in robot learning.

2. We show how this framework can be instantiated in the context of tabletop manipulation tasks in the real world, building on the framework of CLIPort introduced in [135].

3. We demonstrate that GenAug policies can show widespread real-world generalization for tabletop manipulation, even when they are only provided with a few demonstrations in a simple training environment.

4. We provide a number of ablations and visualizations to understand the impact of various design decisions on learned behavior.

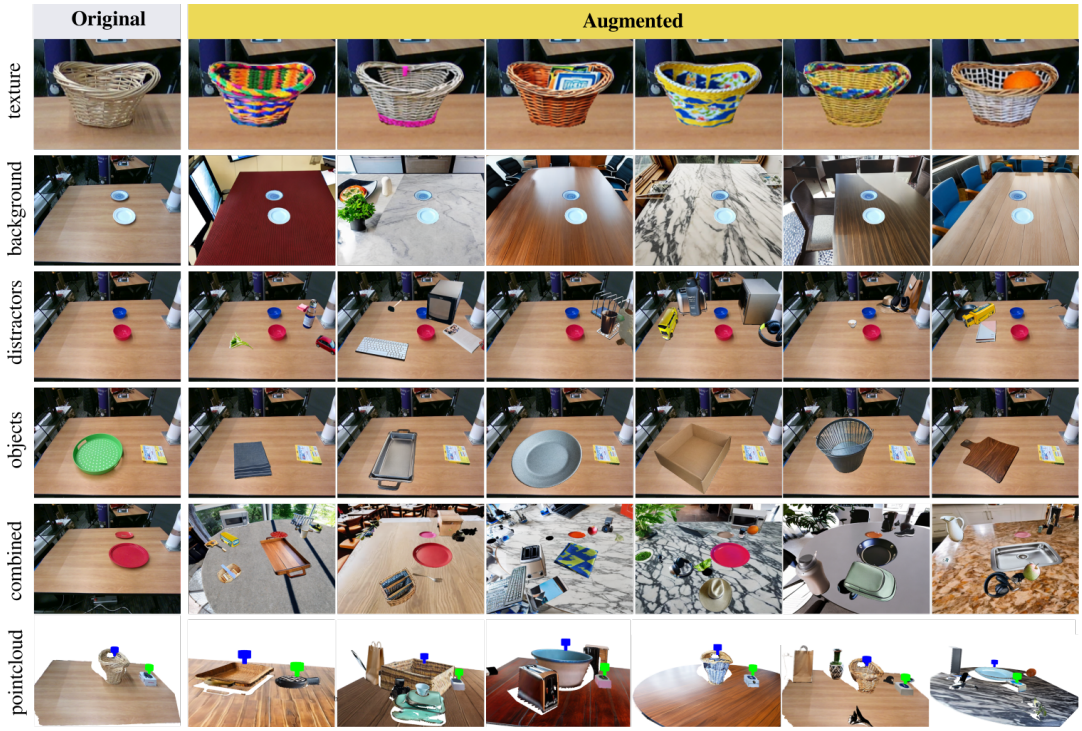


Figure 4: GenAug provides the ability to augment the scene by changing the object texture (first row), changing the background (second row), adding distractors (third row) and changing object categories (fourth row). In addition to image augmentation, we can consistently augment their depth, as shown in last row.

3.2 METHOD

In this section, we will describe the problem statement we consider in our semantic data augmentation technique - Generative Augmentation (GenAug), show how generative models can conceptually be used to inject semantic invariances into robot learning and instantiate a concrete version of this setup for learning policies for tabletop robotic manipulation tasks.

3.2.1 *Problem Formulation*

In this work, we consider robotic decision-making problems, specifically in robotic manipulation. For the sake of exposition, let us consider prediction problems where an agent is provided access to sensory observations $o \in \mathcal{O}$ (e.g. camera observations) and must predict the most appropriate action $a \in \mathcal{A}$ (e.g. where to move the robot arm for picking up an object). The goal is to learn a predictive model $f_\theta : \mathcal{O} \to \Delta\mathcal{A}$ (where $\Delta\mathcal{A}$ denotes the simplex over actions) that predicts a distribution over actions such that the action $a \sim f_\theta(.|o)$ is able to successfully accomplish a task when executed in the environment. In this work, we will restrict our consideration to supervised learning methods for learning $f_\theta(.|o)$. We will assume that a human expert provides a dataset of optimal data $\mathcal{D} = \{(o_0, a_0), (o_1, a_1), \ldots, (o_N, a_N)\}$, and learn policies with standard maximum likelihood techniques [135, 170]:

$$\max_\theta \mathbb{E}_{(o,a)\sim\mathcal{D}} \left[\log f_\theta(a|o)\right] \tag{1}$$

This training process is limited to the training dataset $\mathcal{D}$ that is actually collected by the human supervisor. Since this might be quite limited, data augmentation methods apply augmentation functions $q : \mathcal{O} \times \mathcal{A} \times \mathcal{Z} \to \mathcal{O} \times \mathcal{A}$ which generate augmented data $(o', a') = q(o, a, z); z \sim p(z)$, where different noise vectors z generate different augmentations. This could include augmentations like Gaussian noise, cropping, and color jitter amongst others [7, 26, 111, 134]. Given an augmentation function, an augmented dataset can be generated $\mathcal{D}_{\text{aug}} = \mathcal{D} \cup \{(o', a')_i\}_{i=1}^M$, where $M \gg N$, and then used for maximum likelihood training of $f_\theta(a|o)$. Typically these augmentation functions q are not learned but instead hand-specified by an algorithm designer, with no real semantic meaning. Instead, they impose invariances to the corresponding disturbances such as color variations, rotations and so on to help combat overfitting. Next, we describe how generative models can be leveraged for semantic data augmentation.

3.2.2 *Leveraging Generative Models for Data Augmentation*

While data augmentation methods typically hand-define augmentation functions $(o', a') = q(o, a, z); z \sim p(z)$, the generated data (o', a') may not be particularly relevant to the distribution of real-world data that might be encountered during evaluation. In this case, it is not clear if generating a large augmented dataset $\mathcal{D}_{\text{aug}}$ will actually help learned predictors f generalize in real-world settings. The key insight in GenAug is that pretrained generative models are trained on the distribution $p_{\text{real}}(o)$ of real images (including real scenes that a robot might find itself in). This lends them the ability to generate (or modify) the training set observations o in a way that corresponds to the distribution of real-world scenes instead of a heuristic approach such as described in [111]. We will use this ability to perform targeted data augmentation for improved generalization of the learned predictor f_θ.

Let us formalize these generative models as $g : \mathcal{T} \times \mathcal{O} \times \mathcal{Z} \to \mathcal{O}$, mapping from a text description, an image, and a noise vector to a modified image $o' = g(o, t, z); z \sim p(z)$. This includes commonly used text-to-image inpainting models such as Make-A-Video [137], DALL-E 2 [123], Stable Diffusion [126] and Imagen [128]. It is important to note that since these generative models are simply generating images, they are not able to appropriately generate novel actions a, simply novel observations o. This suggests that data generated by these generative models will be able to impose *semantic invariance* on the learned model f_θ, i.e ensure that an equivalence group $\{o, g(t_1, o, z_1), g(t_2, o, z_2), \ldots, g(t_M, o, z_M)\}$ all map to the same action a. To leverage a pretrained text-image generative model for semantic data augmentation, we can simply generate a large set of semantically equivalent observation-action pairs

$$\{(o, a), (g(t_1, o, z_1), a), (g(t_2, o, z_2), a), \ldots, (g(t_M, o, z_M), a)\}$$

for every observation $(o, a) \in \mathcal{D}$ using the generative model g. Note that the generative models cannot simply generate arbitrary observations $g(t, o, z)$, but only observations that retain semantic equivalence, i.e the ground truth actions for the generated augmentations

$\{o, g(t_1, o, z_1), g(t_2, o, z_2), \ldots, g(t_M, o, z_M)\}$ are all a. We discuss how to actually instantiate this semantic equivalence in the following section.

GenAug allows us to generate a large dataset of *realistic* data augmentations, that ensures robustness to various realistic scenes that may be encountered at test time, while still being able to perform the designated task. Unlike typical data augmentation with the hand-defined shifts described above, the generated augmented observations $\{g(t_1, o, z_1), g(t_2, o, z_2), \ldots, g(t_M, o, z_M)\}$ have high likelihood under the distribution of real images $p_{\text{real}}(o)$ that a robot may encounter on deployment. This ensures that the model generalizes to a wide variety of novel scenes, making it significantly more practical to deploy in real world scenarios, since it will be robust to changes in objects, distractors, backgrounds and other characteristics of an environment. It is important to note the limitations of doing this type of augmentation - it will not be able to generate novel actions a, but instead generate invariances to realistic observational disturbances $o' = g(t, o, z)$ that are generated by the text-image generative model. If not performed carefully, these augmentations can also possibly invalidate the original action a due to factors such as physical inconsistencies or collisions. Next, we discuss a concrete instantiation of this framework in the context of tabletop robotic manipulation.

3.2.3 *Instantiating GenAug for Tabletop Robotic Manipulation*

We scope our discussion of GenAug for this work to tabletop rearrangement tasks with a robot arm. This problem has a non-trivial degree of complexity when performed from purely visual input, especially in very cluttered and visually rich domains, and constitutes a significant body of work in robot learning [135, 170]. In particular, we consider tasks where the observation o is a top-down view of the scene, and the action is a spatial action map over the image, indicating where to pick and place objects with a suction-activated gripper. This builds on the transporter networks[135, 170] framework for visual imitation learning. In this section, we describe how to in-

stantiate GenAug for semantic data augmentation for these table-top manipulation problems.

The important question to answer is — how do we use and prompt the generative model g to generate the appropriate equivalence set of semantically equivalent augmented states for an observation o - $\{o, g(t_1, o, z_1), g(t_2, o, z_2), \ldots, g(t_M, o, z_M)\}$, such that the same action a would apply across all of them? We leverage the fact that for a tabletop manipulation task involving picking and placing objects, the same actions are applicable across a wide range of visual appearances including objects being grasped, distractor objects, target receptacles, and backgrounds, as long as the approximate position of the object of interest and the target remain unchanged.

Given a pick-and-place task on a tabletop, we can perform data augmentations on the visual appearance of 1) the object being grasped or the target receptacle, 2) distractor objects 3) the background or table. We will next describe how we can use the text-to-image depth-guided image generation for generating GenAug's augmentations and maximize visual diversity while preserving semantic invariances for each of these types of augmentations.

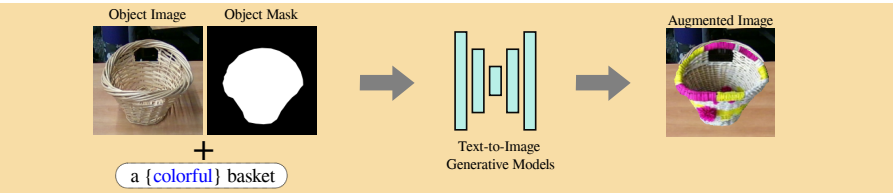


Figure 5: GenAug takes the RGB image and the object mask and uses a depth-guided diffusion model to perform in-category data augmentation.

3.2.3.1 *Generating Diverse Grasping Objects and Target Receptacles*

Simply generating new scenes with an image generation model is unlikely to retain the semantic invariance that we desire for in GenAug since the images will be generated in an uncontrolled way with no regard for functionality. To appropriately retain semantic invariance, we propose a more controlled image generation scheme. In particular, we assume access to masks $\mathcal{M}(o)$ for every observation o, labeling the object of interest and the target receptacle. Note that this is only

needed for the small number of demonstrations that are collected, not at inference time. To generate a diversity of visuals, we consider augmentation both “in-category" and “out-of-category", as described below:

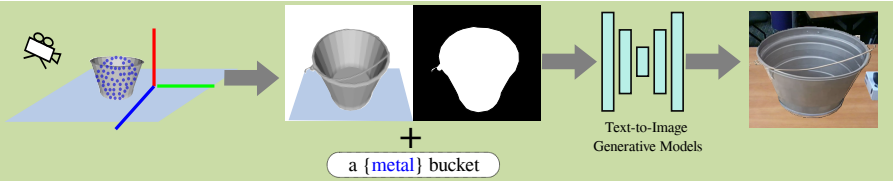


Figure 6: GenAug randomly chooses a new object and place it at the center of the original object to perform cross-category data augmentation

In-Category Generation For in-category generation, GenAug takes the provided mask $\mathcal{M}(o)$ of the object to grasp (or the target receptacle) and the original RGB image, and applies a pretrained depth-guided text-to-image inpainting model [127] to generate novel visual appearances for objects from the same category. To encourage diverse visual appearances of the object, we leverage the fact that the generative model is guided by text and randomly generate novel text prompts involving color (e.g. red, green, yellow, etc) and material (glass, marble, wood, etc) to generate visually diverse objects. Since the object masks remain the same, the resulting positions and shapes are the same, thereby retaining semantic invariance.

Cross-Category Generation While in-category generation provides a degree of visual diversity, it often falls short at generating novel objects altogether and we must consider replacing object categories altogether. When replacing the category of the original object O_i (e.g. a basket) with a new object (e.g. a bucket), one potential technique involves using inpainting models to generate images directly over the masked object (or target receptable). However, naively applying this technique does not generate physically plausible images since it does not appropriately account for geometric consistency during image generation, which causes problems for robotic manipulation.

To allow the generated images to show physical plausibility and 3-D consistency, we leverage a dataset of object meshes to assist the generative model’s generation process. In particular, we first render randomly scaled and sized object meshes from a different category

without any visual detail to get the perspective correct using the same camera pose, followed by a process of visual generation with a depth-aware generative model, as described for an in-category generation. The object meshes are able to ensure 3-D consistency and physical plausibility, while the generative model allows for significant visual diversity. We note that since we are doing top-down grasping with a suction gripper, even cross-category generation ensures semantic invariance leaving the point of interaction with the object largely unchanged while boosting visual diversity.

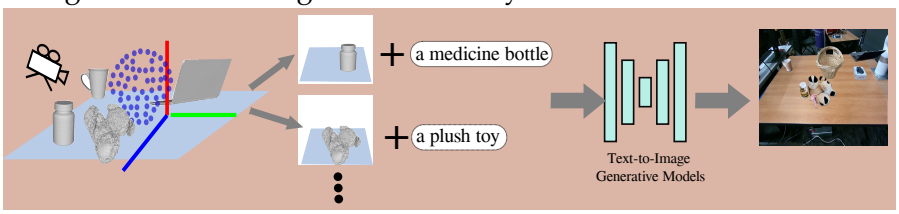


Figure 7: GenAug places a distractor with collision check on the table and uses a depth-guided model to generate realistic-looking objects that are physically plausible.

3.2.3.2 *Generating Distractors with Diverse Visual Appearances*

While Section 3.2.3.1 discusses how to augment the appearance of the object to grasp and the target receptacle, real-world scenarios are often cluttered scenes with several irrelevant distractors. GenAug leverages the same techniques described in Section 3.2.3.1 to generate scenes with a diversity of visual distractors. Similar to cross-object augmentation, to add a new distractor D_i, we randomly choose a new object mesh from a family of object assets and render it on the table, followed by visual generation with a text-to-image generative model as described in Section 3.2.3.1. Importantly since the distractors must be generated in a way that retains semantic invariance, they must not be in collision with the object to grasp or the target receptacle. To ensure this, we compute collisions by checking for overlapping bounding boxes (in image space) between the generated distractor D_i and masks $\mathcal{M}(o)$ for the object to grasp and the target receptacle and remove this distractor if it is in collision. This ensures semantic invariance while being able to generate very cluttered and diverse scenes, as shown in Figure 7.

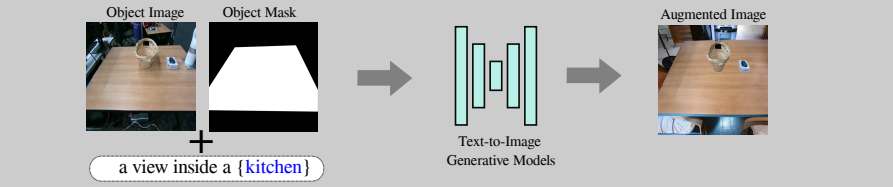


Figure 8: GenAug takes the original RGB image and the mask of the Non-background regions to generate different styles of background scenes such as kitchen, living room, or restaurant.

3.2.3.3 *Generating Diverse Backgrounds*

To augment not just the appearances of objects, but also the background of the scene while ensuring semantic invariance, we can simply invert the process of generation in Section 3.2.3.1 and 3.2.3.2. In particular, we can simply hold the object, target receptacle, and distractor masks fixed while asking a text-guided generative model to generate a diverse range of backgrounds, as shown in Figure 8. Since the object masks are all held fixed, their positions remain invariant, while ensuring that the visual appearance of the background and table varies widely.

GenAug leverages these three forms of semantic augmentation - 1) visual object generation, 2) distractor generation and 3) background generation to augment robot learning data with a large amount of semantically invariant, yet visually diverse data. This data has a significant overlap with the types of environments that might be encountered by a system in the real world, and as we show empirically in Section 5.4, is able to improve the robustness and generalization of robot learning models significantly. Once data is generated with a combination of these three forms of augmentation, we can then simply run standard maximum likelihood techniques for learning manipulation from the augmented dataset. To enable GenAug to be an effective tool for robot learning, we next describe the setup we used for real-world experiments.

3.3 SYSTEM DETAILS

3.3.1 *Hardware Setup*

Since GenAug is instantiated in this work for tabletop manipulation, we use a robot arm equipped with a suction gripper for all our hardware experiments. In particular, we use the 6 DoF xArm5 with a vacuum gripper manipulator, and control it directly in end-effector space. As shown in Figure 9, we attached the xArm to the end of a large wooden table in a brightly lit room and set up the depth camera on a tripod on one side of the table so that it has a clear view of the robot and any objects on the table.

GenAug requires RGBD observations of the demonstration scenes, masks for the object of interest and the target receptacles, as well as a calibrated camera pose. In our real-world setup, we obtain RGBD images from an Intel RealSense Camera (D435i) and manually label the object masks for the collected demonstrations. While the input for GenAug is the camera observations from the RealSense camera, the input observation and the action for the predictive model f_θ operate on a top-down view, as described in [170].

In order to guide the robot to complete the tasks in the cluttered environment, we largely build on the architecture and training scheme of CLIPort [135], which combines the benefits of language-conditioned policy learning with transporter networks [170] for data-efficient imitation learning in tabletop settings. GenAug replaces the imitation learning dataset in CLIPort with a much larger augmented one, as described in Section 3.3.3. Implementation details can be found in Appendix B.

3.3.2 *Demonstration Collection*

To collect demonstrations, we rely on humans to collect action labels for various pick-and-place tasks. We first project the front camera view to a 2D top-down image and height map of the scene. The user can manually click locations on the top-down images to indicate the

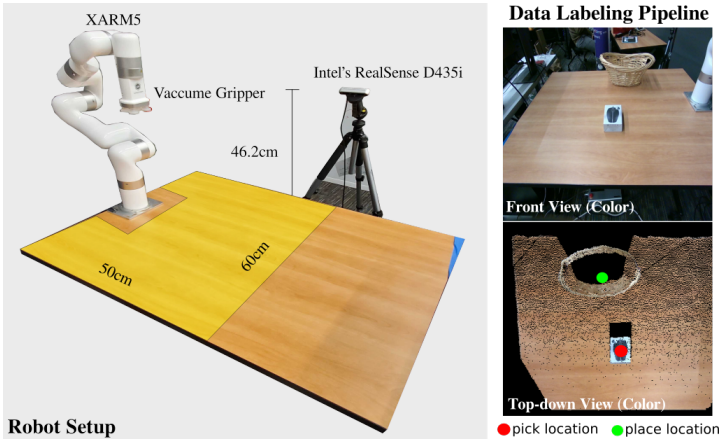


Figure 9: An illustration of our robot experiment setup and data labeling pipeline. A user clicks locations on a top-down view image, to indicate pick (red) and place (green) locations in the robot space.

pick and place locations. These locations are then converted to end-effector positions in full Cartesian space, which is then provided to a low-level controller that uses inverse kinematics and position control. We save the demonstration if the robot can successfully complete the task. We collect 10 demonstrations per task and 10 tasks in total, all in one single environment.

Table 1: Real-World Robot Experiments tested on 10 tasks. On average, GenAug achieves 85% success rate on unseen environment, 52% on unseen object to place, and 45% on unseen object to pick.

	bowl to Coaster	box to basket	bowl to bowl	plate to tray	box to tray	plate to box	plate to plate	coaster to salt	coaster to pan	box to box	**Average**
Unseen Env	0.8	0.9	1	1	1	0.9	0.9	1	0.5	0.5	0.85
Unseen Place	0.7	1	0.5	0.3	0.6	0.3	0.4	0.4	0.4	0.6	0.52
Unseen Pick	0.2	0.6	0.5	0.6	0.7	0.3	0.3	0.7	0	0.6	0.45

3.3.3 *Augmentation Infrastructure*

As described in Section 3.3.1, GenAug requires object meshes to generate cross-category augmentations and distractors. To perform this augmentation, we use 40 object meshes from the GoogleScan dataset [36] and Free3D [41]. Of these, we choose 11 objects to augment the original object of interest and 12 objects to augment the target receptacle. Any of the remaining 38 objects are then randomly chosen as distractors. During augmentation, we randomly select which compo-

Table 2: Evaluating with and without GenAug on unseen scenes collected in the real world across 10 tasks. On average, GenAug shows notable improvement in unseen environments and objects.

	box to tray			box to basket			coaster to pan			plate to tray			bowl to coaster		
	env	pick	place	env	pick	place	env	pick	place	env	pick	place	env	pick	place
No GenAug	0.8	0	0	0.2	0.2	0	0.8	0.4	0.4	0	0	0	0	0	0
GenAug	**1**	**0.6**	**1**	**0.6**	**0.6**	**0.8**	**1**	0.4	0.4	**1**	**0.4**	**0.2**	**0.6**	**0.6**	**0.6**
	plate to plate			box to box			plate to box			coaster to salt			bowl to bowl		
	env	pick	place	env	pick	place	env	pick	place	env	pick	place	env	pick	place
No GenAug	0	0	0.2	0.2	0	0	0.6	0.2	0	0.2	0	0.2	1	0.2	0
GenAug	**1**	0	**0.6**	**0.8**	**0.4**	**0.4**	**1**	**0.8**	0	**1**	**0.4**	**0.4**	1	**0.4**	**1**

nents (table, object texture, shape, distractors) to change to generate the augmented training dataset. For each demonstration, we apply GenAug 100 times resulting in 1000 augmented environments per task. The augmented data is then passed into Cliport [135] to learn a language-conditioned policy.

3.4 EXPERIMENT

We evaluate the effectiveness of GenAug in both the real world and simulation. Our goal is to: (1) demonstrate GenAug is practical and effective for real-world robot learning, (2) compare GenAug with other baselines in end-to-end vision manipulation tasks, (3) investigate different design choices in GenAug. We will first show our results in a real-world setting, followed by an in-depth baseline study in simulation.

3.4.1 *Real-world Experiment*

3.4.1.1 *Design of demo and test environment*

To show the generalization capability of a model trained with GenAug, we collect demonstrations of 10 tasks in one single environment and create different styles of test environments such as "Playground", "Study Desk", "Kitchen Island", "Garage" and "Bathroom" as shown in Figure 10. For evaluation, we randomly add and rearrange objects from each

test style and create unseen environments. Please see Appendix A for further details.

Figure 10: Demonstration environment and examples of test environments used in our robot experiments.

3.4.1.2 *Result*

We train CLIPort with augmented RGBD and text prompts for tasks collected in the real world and evaluate in various unseen environments. In particular, for each task, we randomly choose an environment style from Figure 10, randomly rearrange and add objects on the table to create 10 unseen environments, 10 scenes with unseen objects to pick, and 10 scenes with unseen objects to place. We observe that GenAug shows a significant generalization to unseen environments with an average of 85% success rate. On more challenging tasks of unseen objects to pick or place, GenAug is able to achieve 45% and 52% success rates, which are expected to improve with more demonstrations and more object meshes for augmentation.

Results for each task can be found in Table 1.

3.4.1.3 *Baselines for real-world experiments*

To further show the effectiveness of GenAug, we compare our approach with CLIPort trained without GenAug, shown in Table 2. To ensure both methods are tested with the same input observations, we evaluate the success rate by comparing the predicted pick and place

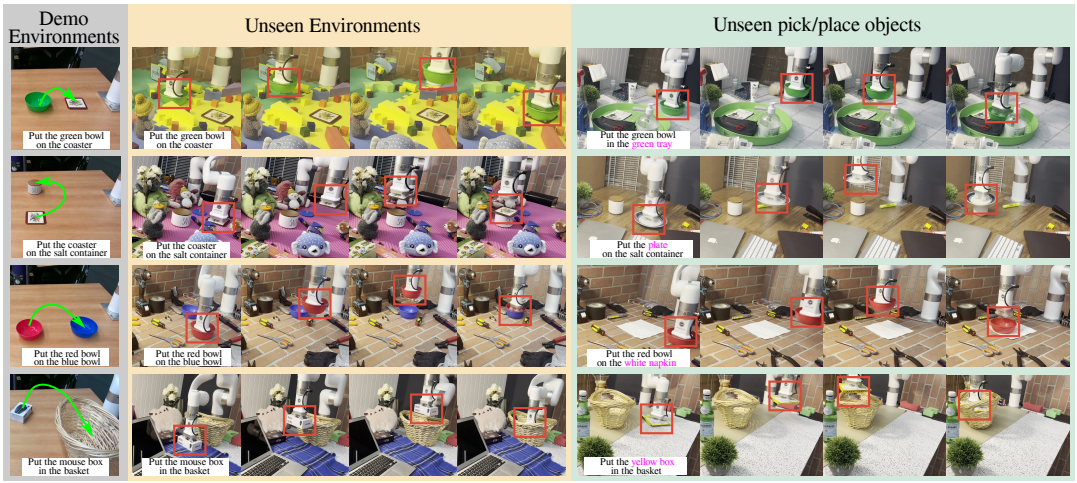


Figure 11: Examples of real-world experiments. Given demonstrations in one simple environment, GenAug enables the robot to generalize unseen environments and objects.

affordances with ground truth locations. For each task, we evaluate both methods on 5 unseen environments, 5 unseen objects to pick, and 5 unseen objects to place. By averaging the success rates from Table 2, we observe GenAug provides a notable improvement for zero-shot generalization. In particular, GenAug achieves 80% success rate on unseen environments compared to 38% without GenAug. On unseen objects to place, GenAug achieves 54% success rate compared to 8% without. Finally, GenAug achieves 46% success rate on unseen objects to pick compared to 10% without. We visualize and compare the differences in their predicted affordances in Figure 12.

3.4.2 *Simulation*

To further study in depth the effectiveness of GenAug, we conduct large-scale experiments with other baselines in simulation. In particular, we organize baseline methods as (1) in-domain augmentation methods and (2) learning from out-of-domain priors, as described below.

3.4.2.1 *In-domain augmentation methods*

(1) "No Aug" does not use any data augmentation techniques. (2) "Spatial Aug" randomly transforms the cropped object image features to learn rotation and translation equivariance, as introduced

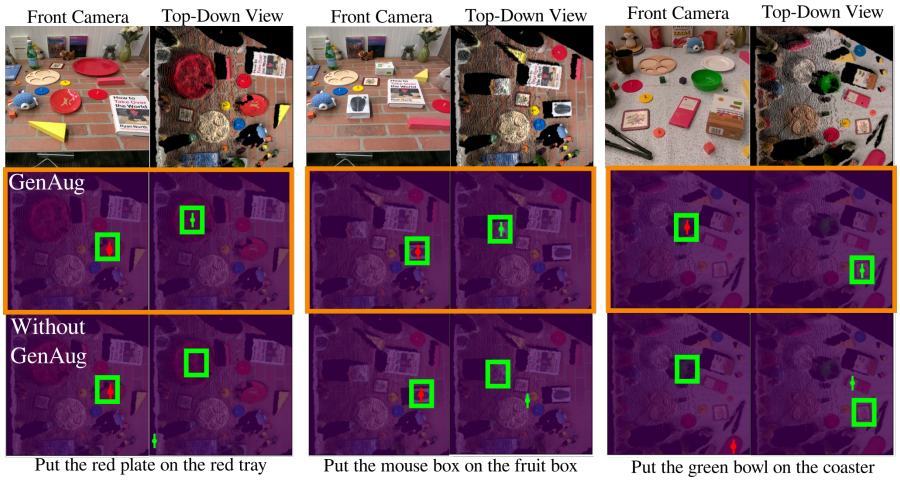


Figure 12: Comparison between training with and without GenAug by comparing pick and place affordances predicted by two models. GenAug significantly improves generalization over unseen environments and objects compared to training without GenAug. pick affordances are highlighted in red, and place affordances are highlighted in green. Ground truth locations are represented in green boxes

in TransporterNet [170]. (3)"Random Copy" randomly queries objects and their segmented images from LVIS dataset [45], and places them in the original scene. This includes adding distractors around the pick or place objects or replacing them. Further visualization of this approach can be found in Appendix A. (4)"Background" does not modify the pick or place objects but replaces the table and background with images randomly selected from LVIS dataset. (5)"Distractors" randomly selects segmented images from LVIS dataset as distractors.

3.4.2.2 *Learning from out-of-domain priors*

In addition, we investigate whether learning from a pretrained out-of-domain visual representation can improve the zero-shot capability on challenging unseen environments. In particular, we initialize the network with pre-trained R3M [104] weights and finetune it on our dataset.

We use baselines described above with two imitation learning methods: TransportNet [170] and CLIPort [135]. Since all the baselines cannot update the depth of the augmented images, we only use RGB images instead of RGBD used in the original TransporterNet and CLI-

Table 3: Baseline experiments evaluated in simulation. We compare the average performance of GenAug with other methods on 5 pick-and-place tasks and observe GenAug provides a notable improvement at unseen environments and objects.

	Unseen Environment						Unseen to place						Unseen to pick					
	TransporterNet			CLIPort			TransporterNet			CLIPort			TransporterNet			CLIPort		
	1	10	100	1	10	100	1	10	100	1	10	100	1	10	100	1	10	100
No Aug	4.8	8.1	9.8	11.7	14.3	14.4	15.1	30.4	52.6	39.4	40.8	44.6	8.5	34.6	54.9	46.0	67.0	64.1
Spatial Aug	11.0	12.2	8.3	23.3	16.1	26.7	44.3	50.5	65.3	26.1	36.9	50.7	53.6	57.2	66.4	38.2	56.9	80.3
Random Copy	**53.1**	67.0	73.5	38.2	39.8	64.3	55.1	65.4	**84.9**	39.7	55.9	73.9	48.3	67.0	76.1	52.5	65.0	81.0
Background	53.0	75.3	79.1	33.6	62.2	55.4	24.5	22.1	35.5	7.6	9.9	17.9	44.4	40.7	35.9	19.2	52.7	72.3
Distractors	10.1	9.7	13.7	15.4	36.2	35.8	28.2	60.7	66.0	27.5	51.8	54.3	42.5	47.4	62.3	31.0	64.0	69.1
R3M	4.1	6.0	4.8	22.2	16.8	20.9	43.5	40.6	41.9	30.9	43.5	57.5	45.6	45.7	41.1	46.7	50.7	72.7
GenAug	43.9	58.5	77.6	46.6	57.0	71.9	**69.1**	**76.5**	83.6	62.6	**83.9**	**86.3**	**75.3**	75.6	**87.2**	**61.5**	**77.7**	**83.1**
GenAug (Depth)	47.8	**83.8**	**91.2**	**47.2**	**60.9**	**73.4**	39.9	67.2	74.2	**64.8**	73.8	84.6	71.2	**83.4**	87.1	56.2	67.3	81.5

Port. For each baseline, we train 5 tasks in simulation and report their average success rate in Table 3. We observe GenAug notably outperforms other approaches in most of the tasks. One interesting observation is that randomly copying and pasting segmented images or replacing the background images can provide reasonable robustness in unseen environments but are not able to achieve similar performance as GenAug at unseen objects. This indicates generating physically plausible scenes that are semantically meaningful is important. Details of tasks in simulation experiments can be found in Appendix A.

3.4.3 *Ablations*

In this section, we aim to study different design choices in GenAug. In particular, our goal is to investigate (1) how the number of augmentations affects the generalization performance to unseen environments, (2) when GenAug will fail in real-world unseen environments. We further justify the choice of using depth-guided models and compare this with in-painting models in Appendix A.

3.4.3.1 *Impact of the number of augmentations*

Given the task "put the brown plate in the brown box" in simulation, we apply GenAug $0, 10, 50$ and 100 times and compare their success

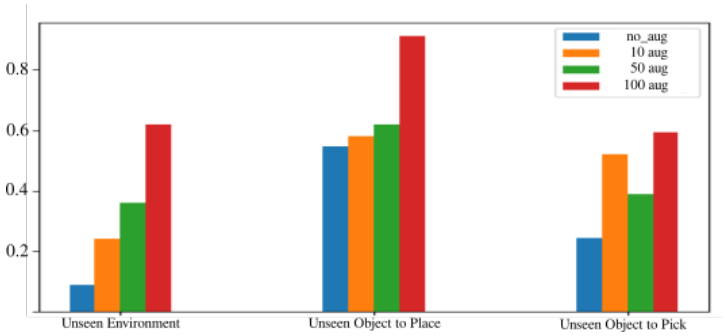


Figure 13: Analysis of the number of augmentation on unseen scenes.

rate on 100 scenes of "unseen environment", 100 scenes of "unseen object to pick" and 100 scenes of "unseen object to place". As shown in Figure 13, the performance improves as the number of augmentations increases, which indicates the importance of using augmentation as a way to robustify the generalization capability.

3.4.4 *Failure modes*

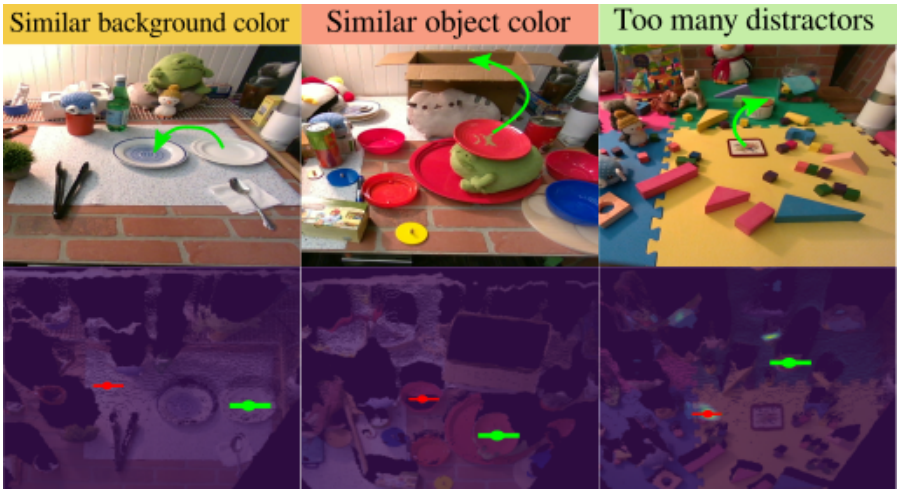


Figure 14: Failure cases observed in the real-world setting

We observe failure cases usually occur when the background color is similar to the pick or place object. Or one of a few distractors has a very bright color or similar color. We expect this can be improved by increasing the number of augmentations in the training set, such that the training data can have higher coverage of possible combinations of the scenes.

3.5 LIMITATIONS

Action Assumption: Despite showing promising visual diversity, GenAug does not augment action labels and reason about physics parameters such as material, friction, or deformation, thus it assumes the same action still works on the augmented scenes. For the augmented cluttered scenes, GenAug assumes the same action trajectory is not colliding with the augmented objects.

Augmentation and Speed: GenAug cannot guarantee visual consistency for frame augmentation in a video. GenAug usually takes about 30 seconds to complete all the augmentations for one scene, which might not be practical for some robot learning approaches such as on-policy RL.

3.6 FUTURE WORK

In the future, we are interested in extending GenAug to other grippers and tasks that are beyond simple pick-and-place tasks. In particular, it would be interesting to investigate if we can train a diffusion model that can augment robot, action and scenes together. We are also interested in applying video diffusion models such as dreamix [97] to ensure visual smoothness for tasks that requires temporal consistency. In our current setup, we only have one front camera mounted on a tripod. For future work, we hope to add a wrist camera on the robot. This will give us flexibility to control the camera and find the object. In addition, combining GenAug with the power of common sense reasoning from LLM such as chatGPT to augment actions with reasoning on object physics would be interesting.

3.7 SUMMARY

We present GenAug, a novel system for augmenting real-world robot data. GenAug leverages a data augmentation approach that bootstraps a small number of human demonstrations into a large dataset

with diverse and novel objects. We demonstrate that GenAug is able to train a robot that generalizes to entirely unseen environments and objects, with 40% improvement over training without GenAug. For future work, we hope to investigate GenAug in other domains of robot learning such as Behavior Cloning and Reinforcement learning, and extend our table-top tasks into more challenging manipulation problems. Moreover, investigating whether a combination of language models and vision-language models can yield impressive scene generations would be a promising direction in the future.

Part II

DYNAMIC TARGETED AUGMENTATION FOR PHYSICAL REALISM

> *"We have to remember that what we observe is not nature herself, but nature exposed to our method of questioning."*
>
> — Werner Heisenberg

While augmenting observation data increases the visual diversity of the initial datasets, it can unintentionally reduce physical accuracy, occasionally producing augmented data that is not physically feasible.

In this chapter, I introduce an augmentation method that generates both diverse visual input as well as physically accurate action data. Specifically, I present Implicit Shape Augmented Grasping (ISAGrasp), a novel learning system that expands a limited number of human demonstrations into a larger dataset of realistic objects and their corresponding grasps. This augmented dataset can be used to train a policy that takes pointcloud observation and predicts final endeffector poses. I will show by combining generative augmentation with a physics simulator, we can effectively train a more generalizable policy and transfer it to unseen objects in real-world settings.

This chapter contains material originally published as "Learning Robust Real-World Dexterous Grasping Policies via Implicit Shape Augmentation " at CoRL '22[18].

4

ISAGRASP: LEARNING TO GENERALIZE BY GENERATIVE SHAPE AUGMENTATION

4.1 BACKGROUND

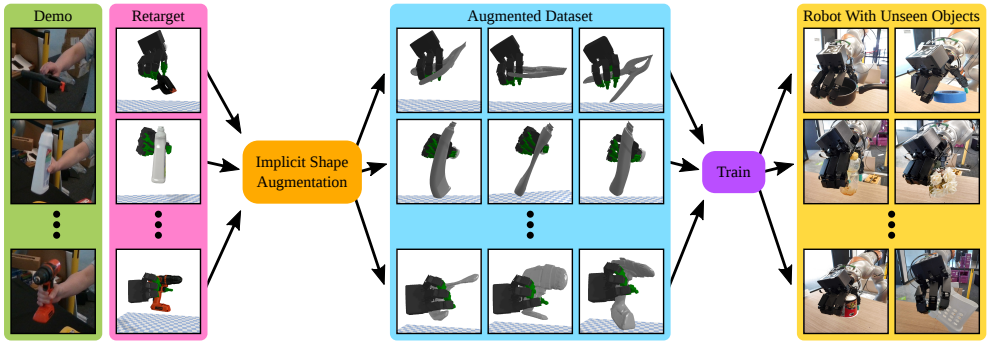


Figure 15: An illustration of the training scheme in ISAGrasp. A few human demonstrations are provided from motion capture. These demonstrations are retargeted to a four-fingered allegro robot in simulation. We use a correspondence-aware generative model to extrapolate the retargeted demonstrations to a large dataset of novel objects. We use this dataset to train a single grasping policy that is able to generalize to a variety of unseen objects in simulation and real world.

Human hands are powerful tools for manipulating a wide range of objects. Our goal is to build dexterous robotic manipulators that can robustly and adeptly interact with human-centric environments. However, robustly controlling a dexterous hand remains challenging due to its high dimensional state and action space and its multi-modal contact dynamics.

Recent work has used deep reinforcement learning algorithms to learn complex tasks with dexterous manipulators [17, 59, 86, 87, 101, 118]. These methods often require careful reward engineering and environment design, and often lack generalization, struggling with robust real world deployment.

An alternative paradigm involves collecting a large labelled dataset and using supervised learning (imitation learning). This has shown significant success with parallel jaw grippers [85, 99, 140] and suction cups [85, 170]. However, imitation learning methods have proven difficult to scale to multi-fingered robots due to the burden of human data collection. To scale robust policy learning to dexterous grasping, we require techniques that can leverage a small amount of human effort to learn robust, general grasping policies. This can be done by extrapolating a small number of human demonstrations to generate an abundance of data that interacts with *diverse* objects and is *successful* at grasping objects under real world dynamics. The key question is —how do we generate this type of diverse training data?

In this work, we propose *Implicit Shape Augmentated Grasping* (ISAGrasp), shown in Figure 15, a learning system that leverages a correspondence aware generative model [33] to extrapolate a small number of human demonstrations to a large dataset of realistic objects and their corresponding grasps. In particular, we directly perform deformations on implicit 3-D shape representations of various objects in a learned latent space. The representation of point-wise shape deformations allows us to generate transformed grasps for novel objects that can be made successful with a small amount of active interaction in simulation. In doing so, implicit shape augmentation allows us to grow from a small dataset of human demonstration data for dexterous grasping to a significantly larger and more diverse dataset with novel object shapes and dynamically successful grasps.

Given the dataset constructed by implicit shape augmentation, we can now perform large scale supervised learning. In this work, we represent a policy as predicting a pre-grasp and a final pose from pointclouds of the target object, with intermediate motion being performed with standard motion planning libraries.

We show empirically that a policy learned via supervised learning on the dataset constructed by ISAGrasp is able to generalize widely across different objects in simulation and the real world.

We demonstrate the efficacy of this pipeline on the rescaled YCB instances, ShapeNet and GoogleScans objects in simulation and achieve a 79% success rate on 22 unseen objects in the real world.

In summary, our contributions are (1) we propose a novel system, *ISAGrasp*, that is able to generate both a wide variety of objects *and* the corresponding dexterous grasps from a few human demonstrations. (2) We show that the augmented dataset can be used for learning point-cloud based control policies that are able to robustly grasp a large variety of *novel* objects in simulation (3) We demonstrate the efficacy of the proposed pipeline by deploying it in the real world. (4) We analyze the proposed pipeline through several ablation studies in simulation.

4.2 METHOD

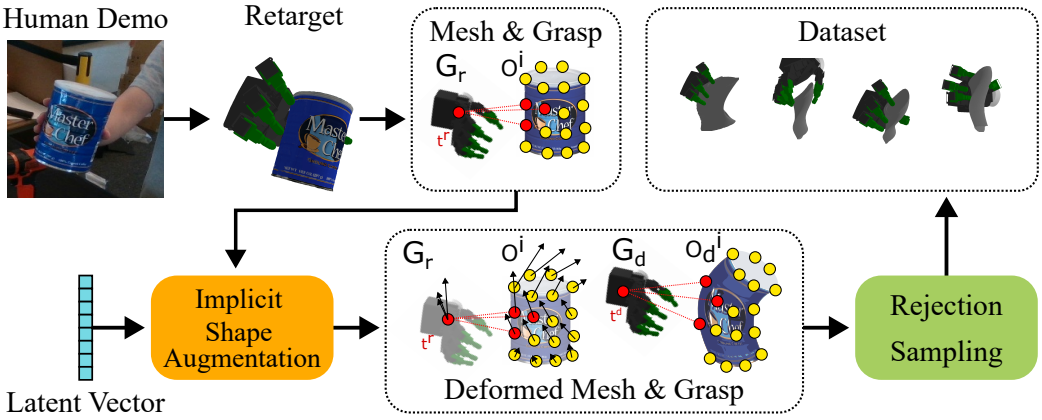


Figure 16: Implicit Shape Augmentation for generating augmented dataset from demonstrations. First a human demonstration is retargeted onto the Allegro hand to generate meshes and grasp labels. This data can then be used to generate a variety of new objects via shape augmentation with DIF-Net [33]. Grasps for these deformed objects can then be further refined with rejection sampling to generate dynamically consistent grasps.

To learn robust grasping policies that can operate in the real world for grasping novel objects of various shapes, we propose ISAGrasp —a framework for supervised learning of robust and generalizable grasping policies from successful grasps on a large variety of objects generated from a small set of human provided expert demonstrations.

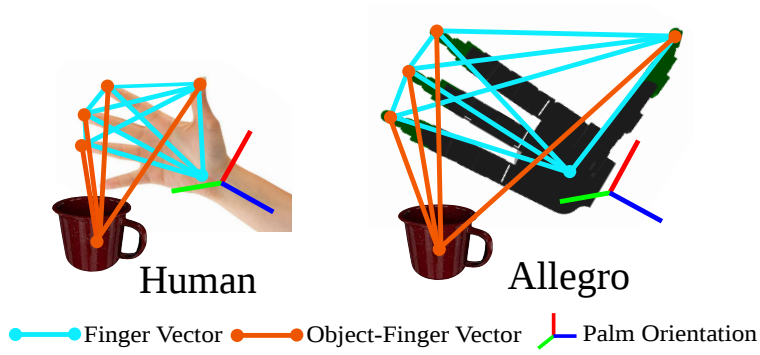


Figure 17: Illustration of retargeting a human hand demonstration to an Allegro hand

Given a pointcloud of an object, we model a grasping policy as predicting (1) a pre-grasp pose $\mathbf{T}$ that controls robot hand's translation $\mathbf{T_t}$ and rotation $\mathbf{T_q}$, and (2) a final pose $\mathbf{G_f}$ that controls the 16-DoF finger pose that can firmly grab the object.

ISAGrasp assumes access to a set of labelled human grasping demonstrations, can be used to learn grasping policies via supervised learning. However, to learn truly robust and general grasping policies, this dataset must be grown to a much more diverse set of objects and successful grasps. Generating a multi-fingered grasping dataset of diverse yet realistic objects is a non-trivial problem [9, 69, 143]. We approach this problem by leveraging a correspondence-aware, deformation-based generative model to widely augment the set of human provided demonstrations, which can then be used to learn robust and general policies via supervised learning. An overview of our ISAGrasp system is shown in Figure 16, and we detail each component below.

4.2.1 *Human-Robot Retargeting*

$$d_g = \sum_{i=0}^{N} \|\vec{\mathbf{a}_{r_i}} - s_r \vec{\mathbf{a}_{h_i}}\|^2, \tag{2}$$

$$d_c = \sum_{i=0}^{N} \|\vec{\mathbf{c}_{r_i}} - \vec{\mathbf{c}_{h_i}}\|^2, \tag{3}$$

$$d_r = G(\mathbf{M}_r, \mathbf{M}_h), \tag{4}$$

$$_{(\mathbf{q}_r, \mathbf{f}_r, \mathbf{M}_r)}(w_g d_g + w_c d_c + w_r d_r), \tag{5}$$

ISAGrasp first collects N human hand-arm demonstrations using motion capture $\mathcal{D}=\{\tau_0^h, \tau_1^h, \ldots, \tau_N^h\}$, where each demonstration $\tau_i = \{(h_0^i, o_0^i), (h_1^i, o_1^i), \ldots, (h_T^i, o_T^i)\}$ is a trajectory of hand pose h_t and object pose o_t. These demonstrations are then "retargeted" to a 22-DoF floating Allegro hand [125] in simulation. Note that this is a nontrivial problem since these demonstrations are in the morphology of the human hand and are typically not functional when directly mapped to a robot hand using standard Inverse Kinematics [43].

We formulate the retargeting objective as a non-linear optimization problem. First, to allow a robot to grasp in a similar pose to a human demonstration, we use the same cost function proposed in DexPilot [51], shown in Eq. 2, where s_r is the ratio between the sizes of the robot and human hands, and each $\vec{\mathbf{a}_{r_i}}$ and $\vec{\mathbf{a}_{h_i}}$ is a displacement vector between finger tips and the palm of the robot and human hands, shown as blue in Figure 17. While Eq. 2 encourages grasp shape similarity, we additionally minimize the differences between the relative poses to the object, as captured by the vectors between fingers and the object center for the human grasp, $\vec{\mathbf{c}_{r_i}}$, and the retargeted grasp, $\vec{\mathbf{c}_{h_i}}$ (see Eq.3 and orange lines in Fig. 17). Finally, to orient the robot palm similar to the human hand, we add Eq. 4, which optimizes the minimum geodesic distance G in SO(3) between the rotation matrix of the human palm $\mathbf{M}_h$ and the robot palm $\mathbf{M}_r$. The final retargeting goal is to find the 22 DoF configuration that minimizes the weighted sum of these three terms, which results in a candidate grasp $\mathbf{G_r}$ for each object in a human provided dataset.

As we describe in Section 4.2.3, the retargeted grasps are then refined via a rejection sampling step and used to generate the dataset with successful, dynamically consistent retargeted grasps. However, since this dataset is typically quite limited, we next outline how to augment the dataset with novel objects and grasps via implicit shape augmentation.

4.2.2 *Implicit Shape Augmentation*

Given a small set of object point clouds and successful retargeted grasps, $\mathbf{G_r}$, we build a large scale augmented dataset of novel objects and their corresponding successful grasps $\mathbf{G_d}$, using a correspondence-aware implicit generative model, DIF-Net [33].

To recap at a high level, DIF-Net aims to learn an implicit representation of 3D shapes as a scalar field. More specifically, DIF-Net represents a 3D shape via a instance agnostic template implicit field (which is common for all shapes of a particular category and represent the common features of a category), together with a latent conditional 3D deformation field (that perturbs this template field) which allows the shapes to be adapted to every particular object instance while maintaining semantic 3-D structure.

Different novel but realistic shapes can be generated by sampling different latent vectors α and generating object deformations on known object classes using the learned deformation and correction fields, while maintaining semantics. This model naturally provides dense correspondences across object instances since different object instances are pointwise deformations on the same template.

We use this generative model in two ways: (1) leverage the ability to sample a variety of object instances by sampling latent vectors α to generate various novel objects via deformations and (2) the resulting dense correspondences allow us to estimate dynamically consistent grasps from human demonstrations to novel generated objects. Specifically, we use the latent conditional deformation field from a pretrained DIF-Net to generate novel instances and dynamically consistent grasps. To generate novel objects, we sample latent vectors

α from a Gaussian that conditions a latent conditional deformation field. This deformation field generates point-wise deformations of particular object meshes chosen from the set of human demonstrations to generate novel objects.

Second, to estimate the new grasps $\mathbf{G_d}$ for the deformed objects, we find N reference points on the original mesh that are close to the root position t^r of the robot hand and compute the position t_o of the robot relative to a local coordinate system O_i, centered at each reference point i: $t_i^o = O_i^{-1} \cdot t^r$. Once the object is deformed, we compute the corresponding coordinate system O_i^d on the deformed meshes using the same deformation field, to estimate the new grasp location t_d using the average of the local offsets: $t^d = \frac{1}{N}\Sigma_{i=1}^{N}(O_i^d \cdot t_i^o)$. In this way, the dense correspondences obtained via the DIF-Net are directly useful in generating grasps for novel, deformed object instances.

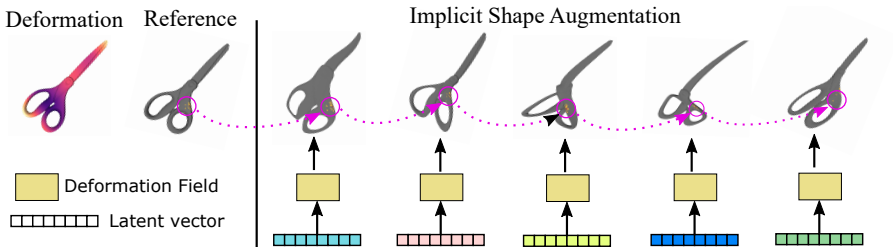


Figure 18: Deformation map and grasping correspondences for objects generated in ISAGrasp. Grasping correspondences on the original object (reference) and the deformed objects are highlighted inside the circle. As can be seen, object semantics are maintained. Different object instances are generated by sampling different latents

Figure 18 visualizes how objects and correspondences deform with different sampled latent vectors. We highlight the reference points on the original mesh, and their correspondences on deformed meshes (purple circles). The objects are deformed into a variety of realistic shapes while maintaining the original semantic structures and grasping correspondences.

Using the new grasp location t^d, together with retargeted orientation q^r and finger pose f^r, we can construct new grasps $\mathbf{G_d}(t^d, q^r, f^r)$. While $\mathbf{G_d}$ are not guaranteed to be successful grasps, they provide good starting points for the local search procedure described next.

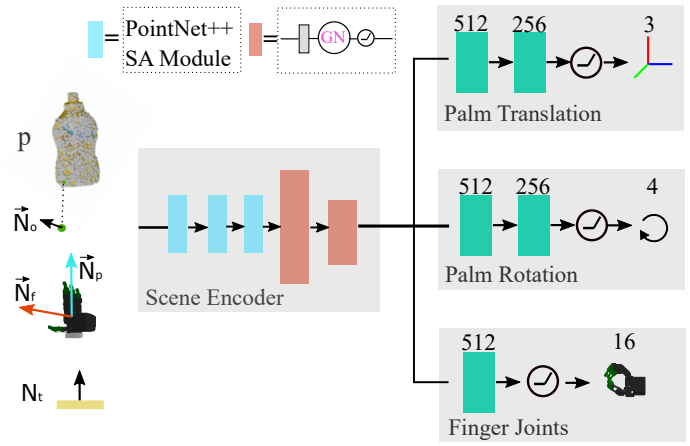


Figure 19: Policy network architecture. The point-net++ architecture inputs an object point cloud p, object surface normal $\vec{N}_o$, the table normal $\vec{N}_t$, robot facing direction $\vec{N}_f$ and the pointing direction $\vec{N}_p$ to generate palm translation, rotation and finger joints for the dexterous grasp.

4.2.3 *Grasp Refinement for Dynamics Consistency*

Given transformed grasps $\mathbf{G_d}$, we use pose perturbation with rejection sampling to generate successful and dynamically consistent grasping poses. Since our shape augmentation model transforms successful hand grasps via the 3D deformation field, we found that only a small amount of perturbation is typically needed to find successful grasps for the deformed models using rejection sampling. In particular, we sample local perturbations δ_t, δ_r and δ_f from a uniform distribution, and add these perturbations to the translation t^d, rotation q^r and finger joints f^r of the transformed poses $\mathbf{G_d}$. We evaluate success of each perturbed grasp $\mathbf{G_p}$ using a physics simulator (Pybullet [24]) and add domain randomization to save robust successful grasps and objects to the training dataset.

Using above methods, we can generate a large and successful dexterous grasping dataset with a variety of novel objects, and then perform supervised policy learning, as described in the next section.

4.2.4 *Policy Learning via Supervised Learning*

As described in the beginning of Section 5.2, we model the grasping problem as predicting a pre-grasp pose $\mathbf{T}$ and a final pose $\mathbf{G_f}$ given an object pointcloud observation. We perform a standard empirical risk minimization procedure on the above-mentioned dataset as an architecture and use a network consisting of PointNet++ SA modules [116] as a feature extractor. Instead of only feeding the raw point cloud to the network, we found significant improvements by appending object points with additional information regarding the alignment between the robot hand and the local object surface. In particular, we use the object point cloud p, the surface normal at each point $\vec{N_o}$, the normal of the table surface $\vec{N_t}$, and the hand facing direction $\vec{N_f}$ and pointing direction $\vec{N_p}$ to define the following feature vectors: $f(p)=(p_x, p_y, p_z, (\vec{N_o} \cdot \vec{N_t}), (\vec{N_o} \cdot \vec{N_f}), (\vec{N_o} \cdot \vec{N_p}), (\vec{N_f} \cdot \vec{N_t})$. We append these vectors to each point in the point cloud. We found these features provide a compact description of alignment between the robot hand and the object, and using the relative vector alignments via the pairwise dot products allows us to improve the final grasping performance.

The network outputs a 3-dim translation and 4-dim quaternion, which are used to define the pregrasp pose $\mathbf{T}$. Additionally, the network predicts a 16-dim finger poses, which is used to define final pose $\mathbf{G_f}$. Figure 19 shows our network architecture details. We provide training details in Appendix B.

4.3 SYSTEM DETAILS

Robotic System. In simulation, we control a 22-DOF Allegro robot at 12Hz using a PD controller, with the torque force that is close to a real robot: 0.6 N.m. In real world experiments, We deploy our policy to a robotic platform that has 23 actuators across a KUKA LBR iiwa 7 R800 robot arm and a Wonik Robotics Allegro robotic hand, and use two cameras to provide necessary point cloud information (Appendix B).

To control the robot in simulation, we directly reset the robot at the pregrasp pose $\mathbf{T}$ with open fingers, and linearly interpolate 10 times and move fingers to the grasping finger poses $\mathbf{G_f}$. In real world, we extend the pregrasp from the pointcloud center by 5cm, and interpolate both translation and rotation from the initial robot pose to this pose. These pose targets are passed to an underlying, manually-derived geometric fabrics policy that generates high-frequency joint position, velocity, and acceleration targets across all joints of the arm. The design and tuning of this policy is exactly the same as the one reported in [147]. Finally, the target joint positions generated by fabrics are directly fed to an underlying gravity-compensated joint PD controller at 30 Hz, which are upsampled to 1000 Hz via polynomial interpolation.

Initial Dataset Construction. We extracted human demonstrations from the DexYCB dataset [16], which contains mocap sequences $\mathcal{D}_{\text{human}}$ of humans hand poses q_T^i picking up 20 YCB objects [12] with poses o_T^i . We picked 10 demonstrations per object, resulting in 200 demonstration in total.

ISAGrasp Implementation Details. We use a pretrained DIF-Net [33] to augment objects into a variety of novel shapes. First, we sample a 128-dim latent vector from a Gaussian distribution $\sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = 0$ and $\sigma = 0.002$. Second, to estimate corresponding new grasps $\mathbf{G_d}$, we choose $N = 20$ closest points on the object surface as reference points and compute corresponding grasp location t_d (Section4.2.2). To obtain successful deformed grasps, we apply rejection sampling by uniformly sampling $\delta_t \in [-0.02\ m, 0.02\ m]$ for translation t_d, and $\delta_r \in [-0.5, 0.5]$ radians for rotation q_r, and apply the same perturbation $\delta_f \in [-0.1, 0.1]$ radians for finger joints (Appendix B).

4.4 EXPERIMENTS

Through our experimental evaluation, we aim to answer the following questions:

Table 4: Baselines evaluated in simulation

	RescaledYCB	ShapeNet	GoogleScans
Random	0.03	0.05	0.02
Train on successful random	0.15	0.42	0.18
Heuristic	0.27	0.40	0.16
Train on successful heuristic	0.09	0.15	0.02
Train on successful GraspIt	0.29	0.42	0.20
PPO with dense reward	0.12	0.10	0.06
DAPG +DR	0.46	0.51	0.53
DexYCB - DR - ISA	0.34	0.35	0.22
DexYCB + DR - ISA	0.74	0.56	0.51
DexYCB +DR +ISA (ours)	**0.74**	**0.74**	**0.70**

1. Does ISAGrasp generate realistic novel objects based on a small set of scanned objects?

2. Does ISAGrasp allow for easy generation of dynamically consistent grasps on novel objects through grasp transformation and refinement?

3. Do policies learned on the dataset produced by ISAGrasp show improved robustness and generalization on unseen objects?

4. How does the training perform with different input features.

We address these questions through a study in a PyBullet [24] simulation, followed by a real world experimental evaluation using the robotic system described in Section 4.3. In subsequent sections we describe baseline methods, evaluation protocols and present experimental results followed by an ablation study on object shape generation.Additional analysis on different elements of the ISAGrasp system are in Appendix B, and more videos in the appendix and https://sites.google.com/view/implicitaugmentation/home

Evaluation Metrics. We choose three unseen datasets with an increasing complexity: RescaledYCB, ShapeNet [13], and GoogleScans [36] (see appendix B). In particular, RescaledYCB contains 65 rescaled YCB objects, ShapeNet contains 200 unseen objects from "Can", "Bottle", "Mug" and "Bowl" categories, and GoogleScans objects contains

200 everyday objects. We place objects randomly on the table, and evaluate performance with 5 sets of object mass and friction. The success is defined as when the object is above the table by 10cm.

Domain Randomization. We use domain randomization during refinement stage with rejection sampling. We randomize object mass from $m \in [0.05\ kg, 0.25\ kg]$ and object friction from $\mu \in [0.7, 1]$. After the object is lifted, we add 1 second high-frequency perturbation sampled from a Gaussian distribution $\sim \mathcal{N}(0, 0.01)m$ on the palm translation. We repeat this process 10 times and keep grasps that are robust and successful for all 10 random physics parameters.

Baselines. Table 4 additionally compares the performance of our method to other baselines —**(1)** Randomly generate grasps around the object (Random) **(2)** use random baseline with rejection sampling to create successful dataset and train a policy **(3)** perform a predefined grasp where robot always grasps from the top (Heuristic). **(4)** use Heuristic baseline with rejection sampling **(5)** Graspit!, an optimization-based grasp planner from prior work[94] **(6)** In addition, we train two RL baselines using PPO[131], both without demonstrations (using a dense reward function) and use the demonstration as a reward function. **(7)** Training only on the human demonstrations provided on the initial 20 YCB objects using supervised learning. **(8)** We train an RL method (DAPG + DR) that combines an imitation learning objective via behavior cloning with reinforcement learning [122]. Please find further details of these baselines in the Appendix B.

We also compare with versions with domain randomization, but no shape augmentation (DexYCB + DR - ISA), and a version with no shape augmentation or domain augmentation (DexYCB - DR - ISA) in order to understand the impact of shape augmentation and the impact of domain randomization.

4.4.1 *Simulation Results.*

We first evaluate the efficacy of the ISAGrasp system on learning dexterous grasping policies in simulation. Table 4 shows the overall success rate of our method ISAGrasp on three unseen datasets in sim-

Figure 20: Qualitative results on unseen objects in simulation and in the real world. The top 3 rows shows the successful examples of our method on GoogleScans objects and the bottom 2 rows show successful examples using our policy deployed on unseen objects in real world.

ulation. ISAGrasp (bottom row) is able to achieve 74% success rate on rescaled unseen YCB objects even the policy is only trained on augmented shapes. In addition, our method achieves 74% and 70% success rate on ShapeNet and GoogleScans objects. We next describe how the various baselines perform.

4.4.1.1 *Random and Heuristic Baselines*

We find that the Random and Heuristic baselines described above do not perform well on RescaledYCB, ShapeNet and GoogleScans objects in simulation. The random grasp baseline achieves less than 5% success rate, while heuristic grasp achieves 40% success rate (only on ShapeNet objects) and performs poorly on the other two datasets. This shows the importance of actually learning the grasping behavior rather than hard coding it or sampling randomly. While the hard-coded baselines may work on certain shapes they are not adaptive enough to learn grasping behaviors on more challenging shapes.

Understanding performance of the Random Baseline: Random is referring to generating a robot pose randomly around the object. Because the action space of an allegro hand is high-dimensional, (22DoF

for floating hand, 23Dof for Kuka-Allegro), randomly generating a stable grasp is much less likely and almost fails on every object.

Understanding performance of the Heuristic baseline: We use a heuristic that we have seen being used for grasping in practical systems: grasp the object from the top, with a fixed 5cm offset from the object top surface. This method usually works when the object is small and round (can, small box), but less likely to succeed when (1)The object is less symmetric like "a mug with a handle" and requires more careful reorientation of the pregrasp pose. or (2) the object is unstable to grasp from the top.

4.4.1.2 *GraspIt Baseline*

The GraspIt baseline achieves the 48% on ShapeNet, and perform poorly on other two datasets, showing the benefit of learning from demonstrations as well as dataset augmentation that is leveraged by ISAGrasp.

Understanding performance of GraspIt Baseline: GraspIt is based on optimization using the contact energy function, but does not account for dynamics, or ensure stability. Failure modes include:

1. Collision checking: Starting from an open palm, we interpolate finger poses into the final graspIt pose. However, graspIt often fails when the robot hand is in collision with a table or starts changing the object pose while closing the fingers.

2. Unstable grasps: GraspIt does not take dynamics into consideration, making the produced grasps potentially unstable while lifting up the object. In contrast, our rejection sampling with domain randomization encourages more stable grasps.

3. Challenging to optimize: When the object surface is more complex, it is usually more challenging for GraspIt to find a good grasp solution.

The baselines of training on successful random, heuristic and GraspIt baselines augmented with rejection sampling (as seen in Table 1) can do better than the base methods, but they do not actually succeed at

solving the tasks very reliably because the refinement rates are fairly low and they do not train on a diverse range of augmented shapes.

4.4.1.3 *RL Baselines*

We also find that the reinforcement learning baselines using PPO perform very poorly, indicating the importance of using supervised learning to avoid the challenge of exploration. Even with dense reward, PPO is unable to learn very well because of the challenges of optimization with RL. The DAPG baseline achieves a success rate significantly lower than ISAGrasp because it does not actually train on augmented shapes and because RL optimization can be unstable with multiple different shapes.

4.4.1.4 *Domain Randomization Baselines*

We trained baselines using the same pipeline as ISAGrasp, but one without shape augmentation (DexYCB + DR - ISA) and one without shape augmentation or domain randomization (DexYCB - DR - ISA). We see a signficant drop from ours to DexYCB + DR - ISA on the ShapeNet and GoogleScans datasets and another drop when trained without domain randomization either (going from DexYCB + DR - ISA to DexYCB - DR - ISA). This suggests that both domain randomization and shape augmentation are important for robust learning performance.

4.4.2 *ISAGrasp Performance*

We find that ISAGrasp significantly outperforms training without shape augmentation on unseen ShapeNet and GoogleScans objects, and achieves the same performance on RescaledYCB objects (Baseline details are in Appendix B). As seen from Table 4, the generalization performance is good, even on completely unseen object instances, although there are failure modes (as described in Section 4.3).

Failure Modes of ISAGrasp: We observe the remaining performance gap stems from several factors:

1. Generalization error: We train on augmented dexYCB objects but test on rescaledYCB, ShapeNet and GoogleScan datasets. Despite the shape augmentation, there is some distribution shift between the train and test datasets, which results in some amount of the gap in performance. This is verified by seeing that the performance on the same training objects is 81%, while unseen objects on average is 73%.

2. Compounding error: The second cause of error is that we subsample the point cloud into 1024 points for GPU memory reasons. This may lead to some loss of performance since some shape properties could be lost.

3. Incomplete coverage: In order to cover unseen object poses, during training, we do domain randomization on object poses by adding orientation perturbation to the original object pose and the corresponding pregrasps. However, this does not guarantee all unseen poses are included during training.

The performance of both our method and the baselines are best appreciated from the videos on our supplementary website and through the analysis figures added into Appendix Figure 52 and Appendix Figure53 to Figure 54.

4.4.3 *Real World Experiments.*

Next, we evaluated the grasping policy learned in simulation by ISAGrasp, directly in the real world. We perform directly simulation to reality transfer of the learned policy as described in Section 4.2.4. We evaluate our policy on 22 unseen real world daily objects (see Appendix D) and evaluate each object 5 times with random poses. We report number of success on each object in Table 5. On average, the policy is able to achieve 79% success rate on real world evaluation on novel objects. We observe several failure cases: (1) if the object is more transparent (Box1: container box), the pointclouds are incomplete and the network is less robust. (2) objects which require more

Table 5: Real world test of ISAGrasp on 22 unseen objects. ISAGrasp is able to achieve 79% success rate

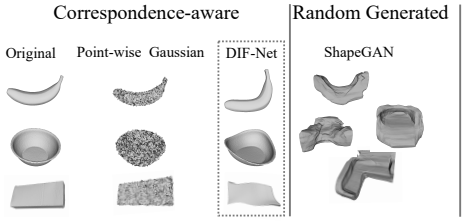
obj	Tray	Crab	Chips	Box1	Box2	Bread	Flower	Pot	Roll	Hat	Honey
#	3/5	4/5	4/5	2/5	4/5	5/5	3/5	4/5	4/5	5/5	5/5
obj	Box5	Scarf	Purse	Tape	Box3	Bottle	Cream	Cap	Drill	Bag	Pringles
#	4/5	5/5	3/5	4/5	5/5	4/5	5/5	5/5	2/5	4/5	3/5

careful grasping (Power drill) often have lower success rate. These show that we can leverage policies trained in simulation directly for real world grasping using ISAGrasp and that the robustness and generalization properties transfer from simulation to the real world.

4.4.4 *Ablations and Analysis*

We first provide some insights into the impact of various design decisions in ISAGrasp below:

Impact of using correspondence-aware generative models:



	Refinement Rate
ShapeGAN(R)	0.09
ShapeGAN(D)	0.29
DIF-Net(R)	0.13
DIF-Net(D)	0.51
DIF-Net(Corr)	**0.76**

Figure 21: Analysis on shape generation. DIF-Net generates realistic and semantically meaningful objects (Left Panel) while ShapeGAN generates novel objects but often unrealistic and without any correspondence (Middle Panel). We show the refinement rate for obtaining successful grasps on these objects (**R**: random grasps, **D**: retargeted demonstration, **Corr**: correspondence-guided grasps (right panel). The lack of correspondences makes refinement challenging, yielding only 30% success rate as compared to DIF-Net at 76%.

To understand how important using the dense correspondences provided by DIF-Net are for generating successful grasps on novel objects, we compare DIF-Net [33] with a correspondence-agnostic model, shapeGAN [69] by showing their generated meshes and the refinement rate for obtaining successful grasps. Refinement rate refers to the ratio of grasps that can be successfully refined via rejection sampling to the total number of proposed grasps. We use this met-

Table 6: Ablations on input features

	p	p+$\vec{N_o}$	p+$\vec{N_o}$+$\vec{N_f}$ +$\vec{N_p}$
Success	0.47	0.57	0.72

ric to compare the efficiency of creating stable grasp datasets generated using different approaches. Shown in Figure 21 (Left), DIF-Net can smoothly deform the original shapes and generate more realistic shapes. We conduct 50 times of perturbation with rejection sampling on 100 objects generated by both methods. Figure21 (Right) shows refinement rate (**R**: random grasps without human demonstrations, **D**: using original human demonstrations without new grasps $\mathbf{G_d}$ estimation, **Corr**: using new grasps $\mathbf{G_d}$). Initialized with correspondence-guided grasps, DIF-Net (Corr) achieves 76% refinement rate while ShapeGAN(D) achieves below 30% with the same number of refinements. This indicates the importance of transferring grasps through a deformation based transformation for novel objects.

Impact of Input Representation: To understand the choice of input representation for supervised learning, we compare feature choices used as input for training policies, evaluated on objects from the ShapeNet and GoogleScans datasets. We observe that using point-cloud p, surface normal vector $\vec{N_o}$, Robot vectors $\vec{N_f}$ and $\vec{N_p}$ performs best.

Impact of Dataset Choice: Humans are excellent at finding stable and feasible grasp based on years of experience, thus providing us a strong prior to generate good grasps much more efficiently. This is the motivation behind using the DexYCB labelled dataset rather than a method like GraspIt to generate grasps for supervised learning. In order to show the effectiveness of using human demonstrations such as the DexYCB dataset quantitatively, we conduct an experiment to compare the refinement rate of successful grasps during rejection sampling: under the same rejection sampling pipeline, if we initialize the grasp with GraspIt, only 26% grasps can be refined, while using humans as prior gives us 81% refinement rate. This suggests that learning from human demonstration is significantly more effectively than synthetically generated grasps.

Understanding ISAGrasp performance on RescaledYCB: The rescaled YCB dataset is created by scaling different dimensions of objects in the DexYCB dataset. This results in objects of widely varying sizes, rather than very different shapes. Since shape augmentation via ISAGrasp largely provides robustness to shape, and less prominently to scale, it is unable to effectively generalize to objects in rescaled YCB. We verify this quantitatively by performing experiments where we rescale all dimensions of the RescaledYCB dataset such that the object is roughly back to original size (although some dimensions may be more stretched than others). We observe 81% with shape augmentation and 75% without augmentation on this new dataset. This further shows that inability to generalize to different scales is the limiting factor here. Supplementing ISAGrasp with large dimension scales would further help.

4.5 LIMITATIONS

Some limitations of this work include:

1. **Poor performance on challenging objects:** it's more challenging for our policy to succeed when the object is large (e.g. cracker box lying on the table Appendix B) or too flat. Since our shape augmentation is built on dexYCB dataset, if the test object is too different from the training objects, or requires a more specific way of grasping. The shape augmentation also does not cover objects of widely varying scales.

2. **Real world experiments:** The method assumes access to a fairly complete point cloud. It will not succeed in scenarios with heavy amounts of occlusion or noise in the point clouds.

3. **Functional grasping:** Currently the method does not do dexterous and functional grasps, and is only designed to lift the object. The utility of a dexterous hand is perhaps best utilized with functional grasps, but the current system does not optimize for this directly.

4. **Accounting for kinematics:** The current system does not account for the kinematics of potentially hitting the table when the hand is mounted on a full arm setup. This should be accounted for as we build on this work in the future.

4.6 SUMMARY

We present ISAGrasp, a novel system for learning dexterous grasping policies in the real world. ISAGrasp leverages a data augmentation approach that bootstraps a small number of human demonstrations with a large dataset with diverse and novel objects and grasps. By using a correspondence-aware generative model, we can deform original object shapes and generate dynamically consistent new grasps. We create a large and diverse grasping dataset and train a policy via supervised learning that can then be deployed in simulation and the real world on grasping novel objects, achieving over 75% success rate on grasping novel object instances in the real world.

Part III

CONTEXTUAL TARGETED AUGMENTATION FOR REAL-WORLD SCENE GENERATION

"The Map Is Not the Territory"

— Alfred Korzybski

In the previous chapters, I have presented how static targeted augmentation can be applied to increase visual diversity and how dynamic targeted augmentation, combined with simulation, enables physically accurate data generation. Both methods, however, still rely on the availability of initial human demonstrations, which can limit scalability in diverse applications. In this chapter, I will present Contextual Targeted Augmentation for Real-World Scene Generation. I will show our effort that enables the automatic generation of robot labels by interacting with scenes that closely mimic real-world environments.

In particular, I will introduce URDFormer, a generative method designed to create realistic simulated environments conditioned on images. This system aims to replicate the kinematic properties of the scene directly from RGB images. This allows us to convert images sourced from the internet into detailed simulations that can be directly utilized for robot training. This chapter details the development of URDFormer and its several applications to scale up robot learning. This chapter contains material currently under review as " URDFormer: A Pipeline for Constructing Articulated Simulation Environments from Real-World Images" at RSS '24.

5

URDFORMER: A PIPELINE FOR CONSTRUCT- ING ARTICULATED SIMULATION ENVIRONMENTS FROM REAL-WORLD IMAGES

5.1 BACKGROUND

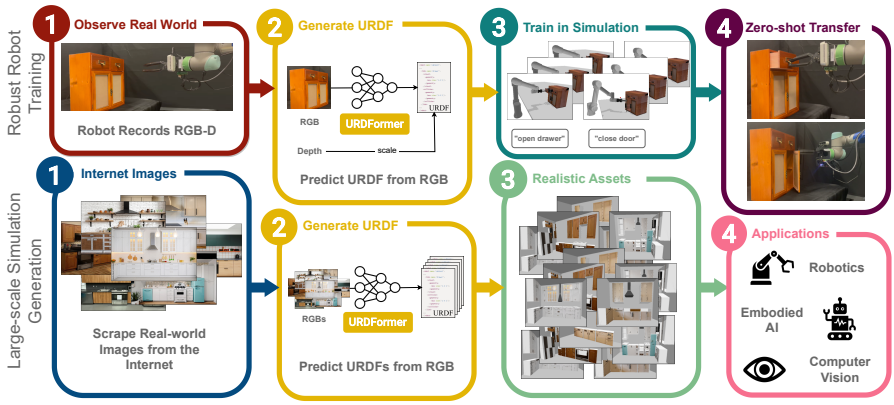


Figure 22: The URDFormer predicts realistic, kinematic scenes from images that 1) allow for zero-shot real-to-sim-to-real transfer through targeted randomization, and 2) generate internet-scale simulation assets valuable to a multitude of applications.

Simulation has become a cornerstone of a plethora of applied machine learning problems - from the natural sciences such as physics, chemistry, and biology [4, 60] to robotics [22, 105] and computer vision [91, 100]. Simulation allows for scalable and cheap data collection while providing an easy way to encode domain-specific prior knowledge into end-to-end machine learning problems. This is particularly important for data-scarce problems such as robotics, where collecting real data can lead to costly and unsafe failures or may require expensive human supervision. Critical to each of these endeavors is a rich and accurate simulation environment, complete with assets depicting complex scene layouts and kinematic structure. For instance, advances in robotic mobile manipulation in the Habitat simulator [142],

are critically dependent on the Matterport dataset for realistic scenes [162]. The creation and curation of these simulation scenes and assets is an important but often overlooked part of the process.

The most common approaches for generating simulation content are either manual [46, 70], procedural [29] or, more recently, purely generative [38, 79, 115, 151, 153]. The manual process for creating simulation scenes requires a designer to characterize, identify, and model a particular real-world scene, a painstaking and impractical process. While this approach can produce high quality results, it often leads to content that lacks diversity due to the amount of human effort required. On the other hand, rule-based procedural generation methods [29, 121] have been applied in robotics applications such as navigation, but often struggle to capture the natural complexity of the real world for problems such as manipulation. Moreover, the procedural generation process is not *controllable*, making it hard to generate simulation content corresponding to a *particular* real-world environment, which is often important in real-world robotic learning pipelines. Recently introduced generative methods for content creation [38, 79, 115, 153] can generate visually appealing 3-D geometries for particular objects, but often result in undesired physical simulation behavior and lack kinematic structure like articulation.

What are the desiderata for a content creation method for simulation? To enable a variety of downstream use cases such as robotic learning, scalable content creation in simulation must be (1) realistic enough such that machine learning models trained in the constructed simulation environments transfer back to the real world, (2) diverse in a way that captures the statistics of natural environments so as to enable learning generalizable models and policies (3) controllable in a way that allows for targeted generation of particular scenes of interest.

To generate content of this nature, we develop a pipeline to train a transformer-based network, URDFormer, that maps directly from individual real-world images to corresponding simulation content (expressed as a Unified Robot Description File (URDF)) that could plau-

sibly represent the semantics, kinematics, and structure of the scene (Fig22).

Figure 23: A summary of categories of objects and scenes that we trained URDFormer to predict. We visualize examples of URDFormer predictions (large background image) given corresponding internet images (small overlaid black box). We train a single part URDFormer for all object categories and a single global URDFormer for all scene categories. Under each object category, we also list the individual articulated parts that it contains.

We leverage controllable text-to-image generative models [126] to generate a large-scale paired dataset of structured simulation scenes and closely corresponding, realistic images. This paired dataset is then *inverted* to train URDFormer, which maps from RGB images directly to plausible simulation environments with semantic and kinematic structure.

URDFormer can then naturally be used in several use cases — (1) diverse content generation for simulation: generating a large and diverse set of realistic simulation environments that correspond directly to uncurated, real-world RGB images (e.g scraped off the web), or (2) targeted generation: generating a simulation environment (or narrow distribution of environments) corresponding to a particular set of desired images. We show that the generated simulation environments are a useful tool for robotic learning in a real-to-sim-to-real pipeline for training robotic control policies.

5.2 URDFORMER - A PIPELINE FOR SCALABLE CONTENT CREATION FOR SIMULATION FROM REAL-WORLD IMAGES

Generating simulated scenes with a high degree of visual realism that supports rich kinematic and dynamic structure, while reflecting the

natural statistics of the real world is a challenging problem. Downstream applications in robotics and computer vision typically require data that is both *realistic*, *diverse*, and *controllable*. To accomplish these requirements, we take an *inverse* approach to the problem and generate scenes by mapping RGB images of real-world scenes to scene representations in simulation complete with kinematics and semantics. This allows for scene generation that is *realistic* since it inherits natural scene and object statistics from real images. The generated scenes are *diverse* since large web-scale image datasets with diverse content can be used to seed such a generation process. Lastly, the generation is *controllable* since curated images of particular target environments can be used to generate corresponding simulation assets. We first define the inverse problem of synthetic scene generation from real-world images, then describe how to learn inverse models to solve this problem with supervised learning on a paired dataset generated using controllable text-to-image generative models [126]. Finally, we show how the learned inverse model can be used with real-world image datasets for scalable content creation, summarized in Figure 23. This tool can then be used to instantiate a real-to-simulation-real pipeline for robot learning, as described in Section 5.3.

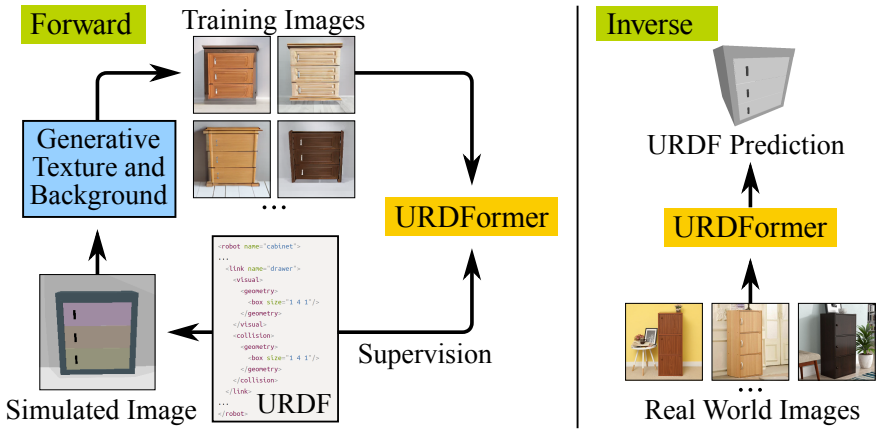


Figure 24: The URDFormer is trained on a large paired dataset of simulation assets and realistic renderings (forward). During inference, this process is inverted and it predicts the URDF from a real image (inverse).

Figure 25: **Controlled Generation:** Rendering URDF models in simulation and generating paired images with a guided diffusion model.

5.2.1 *Problem Formulation*

To formalize the problem of simulation scene generation from real-world images, let us consider a kinematic scene description z drawn from a target scene distribution $P(z)$ in the real world. For our purposes, the scene can be described as a list of objects $z = \{o_1 \dots o_n\}$, where each object o_i contains a base class label c_i, a 3D bounding box $b_i \in \mathbb{R}^6$, a 3D transform $T_i \in SE(3)$, a kinematic parent $p_i \in [1 \dots i-1]$ and a joint type j_i that specifies how that object can move relative to its parent $o_i = (c_i, b_i, T_i, p_i, j_i)$. This resembles the typical representation of scenes and robots using the unified robot description format (URDF). Let's consider a kitchen scenario that contains a row of cabinets next to a stove. The cabinets and the stove will be the kinematic children of the wall, and the kinematic parents of their respective doors. Similarly, these doors will be the kinematic parents of their handles.

As the example shows, the kinematic structure z for a particular real-world scenario is unknown without extensive human labeling effort, and instead, we only have access to the result x of an indirect "forward" function f, $x = f(z)$. For example, x could be a photograph of the real environment, or a point cloud captured with a LIDAR scanner. The goal of this work is to recover the entire kinematic and semantic structure of the scene from just having access to the forward

evaluation x, requiring complete inference of a rich scene representation z.

Unfortunately, since the scene structure z is unknown for most complex real-world scenes and difficult to generate manually, it is challenging to solve the "inverse" generation problem to infer the scene description z from the forward rendered images (or alternative sensor readings) x, $z = f^{-1}(x)$. Had there been a sizeable dataset $\mathcal{D} = \{(z_i, x_i)\}_{i=1}^{N}$ of scene descriptors z_i in simulation and their corresponding real-world counterparts x_i, the inverse problem could be solved using supervised learning (minimizing a loss $\mathcal{L}$ like the cross entropy loss or a MSE loss) to learn an f_θ^{-1} that approximates the scene structure $\hat{z}$ given an input forward-rendered image x.

However, such a paired dataset does not readily exist, making direct application of supervised learning methods challenging. In this work we take an inversion through synthesis approach —leveraging pre-trained generative models to convert procedurally generated scenes in simulation into a large paired dataset of scene content z and corresponding realistic RGB images x. This process can generate a large and diverse dataset of image and scene-description (x, z) pairs that we can use to train an approximate inverse model $f_\theta^{-1}(x)$ that generates scene descriptions $\hat{z}$ from real RGB images x. Since most scenes that we consider are object-centric, we decompose the inverse problem into two parts: (1) object-level prediction that focuses on the kinematic structure of individual objects, e. g., cabinets, and (2) global-scene prediction, e. g., kitchens, that focuses on the structure of an overall scene. We next discuss the process of generating a large paired dataset for these two components (Section 5.2.2) and then show the training process for the inverse model in detail (Section 5.2.3).

5.2.2 *Controlled Generation of Paired Datasets with Pretrained Generative Models*

Given a simulated scene z (drawn from a dataset such as PartNet [96], or procedurally generated), we use the fact that controllable genera-

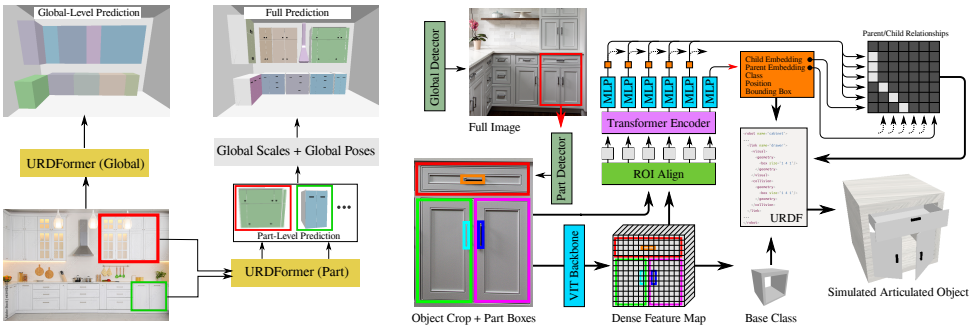


Figure 26: Depiction of the URDFormer Training Procedure and Architecture. (**Left**) Given an RGB image of the scene, i. e.a kitchen, we train two separate networks: URDFormer (Global) focuses on predicting parent and spatial info of how to place the object. URDFormer (Part) takes the cropped image containing each object and predicts detailed structure. The results of the two predictions are combined and create the full scene prediction. (**Right**) The URDFormer architecture takes as input a cropped RGB image and object part boxes and predicts a hierarchy consisting of a base class and parent-child relations that make up the final URDF file.

tive models [126] are both diverse and realistic enough to take an unrealistic simulation rendering of a scene and generate a distribution of corresponding *realistic* images. This allows the scene in simulation with unrealistic appearance and texture to be translated into a diverse set of visually realistic images that plausibly match the same underlying environment. To ensure piecewise consistency and realism of the generated images, we use two different dataset generation techniques for the global scene structure and local object structure respectively. These share the same conceptual ideas but differ to account for consistency properties in each case.

Scene-Level Dataset Generation: To generate training data for the scene model, we feed a poorly rendered image from simulation along with a templated text prompt to an image-and-text guided diffusion model [126], as shown in Fig 25. This generates a new image that attempts to simultaneously match the content described in the text prompt while retaining the global scene layout from the provided image. We found that this model is able to reliably maintain the scene layout, but it may change some individual components of the scene, e. g., replacing objects with a different but plausible category, or changing the number of subcomponents within an object such as

the drawers or handles. Despite these failures, the large-scale structural consistency still provides a useful source of training data. After running our simulated image through the generative model, we have realistic images that contain known high-level object positions and spatial relationships, but unknown category and low-level part structures (e. g.the parts for articulated objects such as cabinets), since these may have been modified by the generative model in it's forward pass. This means that the scene model dataset contains complete images, but incomplete labels. Rather than complete (x, z) pairs, we have a dataset $\mathcal{D}_{\text{scene}} = \{(x, \tilde{z})\}$ of $(x, \tilde{z})$ pairs where $\tilde{z}$ only contains the bounding boxes, transforms and parents of the high-level (non-part) objects $\tilde{z} = \{(b_1, T_1, p_1) \dots (b_n, T_n, p_n)\}$ but lacks accurate low-level information.

Object-Level Dataset Generation: The process for generating object-level training data is similar but requires more care due to the tendency of text-to-image generative models to modify low-level details. For objects with complex kinematic structures, such as cabinets, we procedurally generate a large number of examples of these objects and render them in isolation from different angles. Rather than using a generative model to construct entirely new images, we use it to produce diverse texture images, which are overlaid in the appropriate locations on the image using perspective warping. We then change the background of the image using the generative model with appropriate masking derived from the original rendering. This part-by-part texture-based rendering process ensures diversity while maintaining consistency in low-level details. Unlike the scene dataset which contains complete images but partial labels, the object dataset contains partial images (in the sense that they contain only a single object), but complete labels for the object and its kinematic parts. We can say that this dataset $\mathcal{D}_{\text{object}}$ contains $(\tilde{x}, z)$ pairs where $\tilde{x}$ is an image of a single object rather than a full scene (hence the partial x), and z is complete for the single object and its parts.

The result of these two data generation processes is a high-level scene structure dataset $\mathcal{D}_{\text{scene}}$, and a low-level object dataset $\mathcal{D}_{\text{object}}$,

that can subsequently be used to train an object-level and a scene-level inverse model.

5.2.3 *Learning Inverse Generative Models for Scene Synthesis*

Given the datasets $\mathcal{D}_{\text{object}} = (\tilde{x}, z)$ and $\mathcal{D}_{\text{scene}} = (x, \tilde{z})$ constructed as described above, we can use supervised learning methods to learn an *inverse model* that maps images of a complex object or scene to the corresponding simulation asset. To take advantage of these partially complete datasets, we must add some structure to our prediction model. We do this by splitting our learned inverse model in correspondence with the split in our forward data generation process: we train one network f_θ^{-1} to predict the high-level scene structure using dataset $\mathcal{D}_{\text{scene}}$ and another network g_ϕ^{-1} to predict the low-level part structure of objects using $\mathcal{D}_{\text{object}}$.

To model both the scene-level prediction model (f_θ^{-1}) and the low-level part prediction model (g_ϕ^{-1}), we propose a novel network architecture — URDFormer, that takes an RGB image and predicts URDF primitives as shown in Fig26. Note that both the scene-level prediction and the low-level part prediction use the same network architecture, the scene-level simply operates on full images with object components segmented, while the part-level operates on crops of particular objects with parts segmented. In the URDFormer architecture, the image is first fed into a vision transformer [35] (ViT) visual backbone to extract global features. We then obtain bounding boxes of the objects in the image using the masks rendered from the original procedurally generated scene in simulation (these are known at training time, and can be extracted using segmentation models at test time). We then use ROI alignment [52] to extract features for each of these bounding boxes. These feature maps are combined with an embedding of the bounding box coordinates and then fed through a Transformer [148] to produce a feature for each object in the scene. An MLP then decodes these features into an optional base class label (used only when training the object-level model), and a discretized 3D position and bounding box. In addition, it also produces a child embed-

ding and a parent embedding that are used to predict the hierarchical relationships in the scene (object to its parent and so on). To construct these relationships, the network uses a technique from scene graph generation [165] that produces an $n \times n$ relationship score matrix by computing the dot product of every possible parent with every possible child. The scene-level model also specially has a set of learned embeddings for six different root objects corresponding to the four walls, the floor, and the ceiling so that large objects like countertops and sinks can be attached to the room.

Test-time scene generation from real-world RGB images: Due to the unpredictable nature of the text-to-image generative transforms that are used to perform global dataset generation, which may change the base class identities, e. g., the diffusion model might turn a fridge into a cabinet, only the position, bounding box, and relationship information is used when computing the high-level scene structure.

To generate a full approximation of the scene structure $\hat{z}$ from a natural image at test time, the image and a list of high-level bounding boxes (from a segmentation model) are first fed into the scene prediction model f_θ^{-1}, which predicts the global structure, i.e. the location and parent for each object. The image regions corresponding to these boxes are then extracted and further segmented (using a segmentation model) to produce part-level bounding boxes. Each of these image regions that correspond to a particular object (e.g. a cabinet or a fridge) and the corresponding part-level boxes (e.g. individual drawers or doors) are then fed into the part prediction model g_ϕ^{-1} to compute the kinematic structure of the low-level objects and parts. This nested prediction structure can be used to generate entire scenes from web-scraped RGB images drawn from any image dataset to generate novel simulation content both at the scene level and at the object level. We visualize this process in Fig 26

5.3 USING URDFORMER FOR ROBOTIC CONTROL VIA A REAL-TO-SIMULATION-TO-REAL PIPELINE

As described above, URDFormer provides the ability to generate realistic, diverse and controllable scenes in simulation through inverse modeling. This suggests that on deployment, URDFormer allows a system to take a single picture of a deployment scene and then construct a fully articulated scene in simulation with *minimal* human effort. In this section, we describe how this type of controllable inverse modeling can serve as a useful tool for robotic learning through a real-to-simulation-to-real pipeline.

The most straightforward way of using URDFormer for robotic control is a model-based one - first synthesize a “digital twin" for a deployment time scene, and directly use this for planning and control in the real world with a model-based approach. This is challenging for several reasons - (1) the constructed simulations may not be perfectly accurate, (2) there is no access to Lagrangian environment state in the real world. Instead, we take a learning-based approach to the problem; we use URDFormer to generate not precisely the test time scene in simulation, but rather a narrow, representative distribution of simulation environments via a targeted randomization procedure. This distribution of environments in simulation can be used to learn generalizable robotic policies that operate from raw perceptual input, directly transferring back from simulation to the real-world. Doing so closes the real-to-simulation-to-real loop, obtaining real-world robotic policies with minimal human effort in the process. This pipeline has three major components:

Scene Generation: Given a robot’s RGB pointcloud observation of an unseen environment, we use URDFormer to generate a URDF file from RGB that captures the kinematic and dynamic structure of the real-world scene. We further resize the URDF to fit the pointcloud’s scale. Importing the URDF into simulation then provides a playground for data collection and policy training. Most importantly, this simulation is not just an arbitrary model but an approximate representation of the real world scene of interest on deployment.

Targeted Randomization: In simulation, we have access to ground truth information which we can exploit to inexpensively collect trajectory data for policy learning. We use an efficient motion planner [139] to quickly collect approximately optimal trajectories solving multiple tasks in simulation. To increase the simulation diversity and decrease the sim2real gap, we additionally randomize over minor details, e. g., texture variants, shapes, and sizes of parts. This “targeted" randomization is in contrast to procedural generation which covers many different environment variations but is not informed by the real-world environment.

Policy Synthesis: To synthesize a policy from the collected data, we can train a language-conditioned behavior cloning policy [168] operating from RGB point-clouds in simulation, applying image augmentations during training to enable policy transfer back to the real world. We stress that the proposed pipeline is not limited to data collection with motion planning and behavior cloning but can also be used to train policies with other policy search methods [48, 92, 131].

The proposed pipeline results in a robust policy that can successfully solve tasks in the real-world without the manual burden of constructing environments in simulation [29], expensive human data collection [21] or real-world reinforcement learning [44].

5.4 EXPERIMENTS

In this section, we aim to answer the following questions:

1. Does integrating URDFormer in a real2sim2real pipeline improve policies that can transfer zero-shot to the real world?
2. Can URDFormer generate plausible and accurate simulation content from internet images?
3. Can URDFormer generalize to diverse objects and scenes?
4. Can URDFormer support different robots and tasks?

5.4.1 *Does integrating URDFormer in a real2sim2real pipeline improve policies that can transfer zero-shot to the real world?*

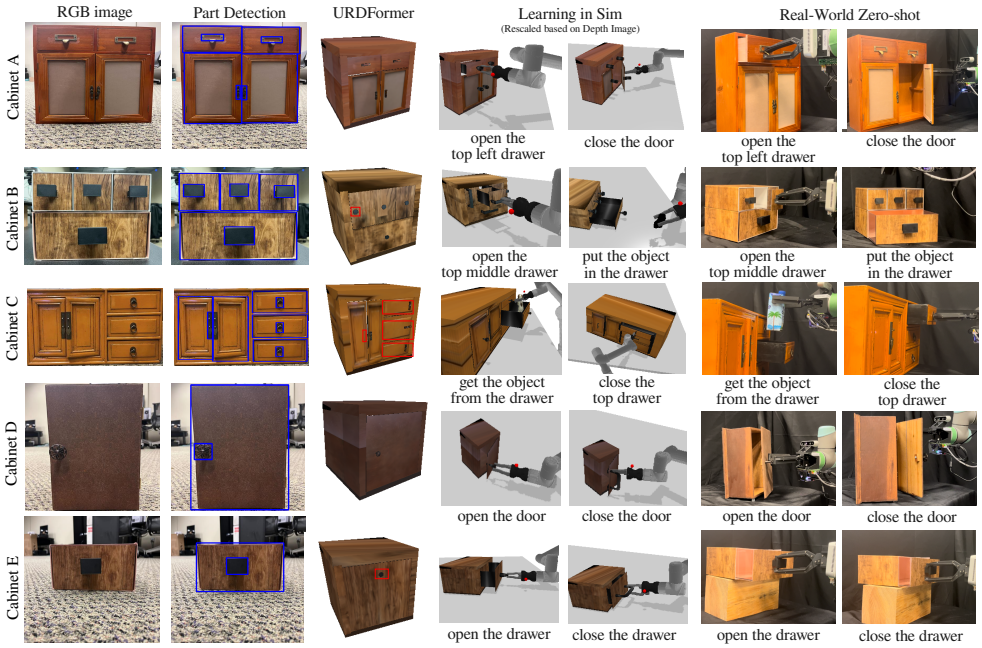


Figure 27: Qualitative Results for Real-world Robot Experiments: An RGB image is pre-processed by detecting bounding boxes of relevant parts. The URDFormer then predicts the corresponding URDF of the cabinet. When importing the cabinet into the simulation, it is re-scaled using depth measurements. Furthermore, the real-world texture is cropped using the bounding boxes and projected onto the cabinet. This realistic simulation can then be used to generate massive data with the help of motion planning, ground truth information, and targeted domain randomization. Finally, we show that training a language-conditioned multi-task policy can be zero-shot transferred to the real world to solve several opening and closing tasks.

As described in Section 5.3, URDFormer can be used to instantiate a real-to-simulation-to-real pipeline for robot learning. In this section, we describe a concrete instantiation of one such pipeline and provide a detailed evaluation of the resulting robotic behavior in the real world.

Real-to-sim-to-real pipeline: We implement our real-to-sim-to-real approach on a UR5 robot equipped with a custom-made 3D printed 2-fingered gripper and an Intel RealSense D435i mounted on the end-deffector. We evaluate our pipeline on five different cabinets with varying sizes, shapes, textures, joint types (revolute and prismatic),

and handles, with two tasks per cabinet. The details of each pipeline element is as follows:

1. Real-to-sim: First, our system takes an image of the scene and uses a finetuned Grounding DINO [80] (Appendix C)to automatically detect the parts of the cabinet, i. e., door, drawer, and handles. URDFormer then generates the corresponding URDF from the RGB image and the predicted bounding boxes using URDFormer, as described in Section 5.2.3.

2. Policy Learning in Sim: The URDF is then imported into a physics simulator (PyBullet [25]) and scaled appropriately using depth measurement. Next, a motion planner, in this case cuRobo [139], generates trajectories that solve a variety of tasks, e. g., closing/opening a drawer/door by utilizing privileged information in simulation. To account for possible prediction inaccuracies of URDFormer (red highlights in Fig 27) and to robustify the trained policy, we apply targeted randomization (TR) that randomizes the scene while maintaining its semantic configurations. In particular, doors, drawers, and handles are randomly replaced with their PartNet [96] equivalents while the size and base of the object are kept fixed. For each door or drawer that was generated in this process, the geometry was randomly replaced with alternate geometry from the same PartNet class but rescaled to be the appropriate size. Additionally, each handle or knob that was generated was similarly replaced with alternate geometry, but also randomly translated in the plane of the door or drawer that it was attached to. Textures are randomized by cropping out the real texture using the bounding boxes, generating variations by prompting Stable Diffusion [126] and fitting it back onto the shape. Finally, the RGB input is augmented by adding standard augmentations such as Gaussian noise and color jitter. After automatically collecting a dataset of successful simulation trajectories using the motion planner, we train a behavior cloning policy network that predicts end-effector poses from point clouds. The network architecture follows M2T2 [168] and predicts 6D end-effector poses from RGB pointclouds given language instructions specifying the task. Appendix C provides the full training procedure and architectural details.

Table 7: Quantitative Results for our Real-world Robot Experiments: Our Real2Sim2Real pipeline with the URDFormer and targeted (domain) randomization in simulation results in a 78% success rate across all tasks. Results are reported in success / number of trials.

	Cabinet A		**Cabinet B**		**Cabinet C**	
Task	Open left drawer	close door	place object	open middle drawer	fetch object	close top drawer
OWL-ViT [95]	0/5	0/5	0/5	0/5	0/5	0/5
DR	0/5	0/5	0/5	0/5	0/5	0/5
URDFormer-ICP	2/5	1/5	—	3/5	1/5	**3/5**
URDFormer-TR	**4/5**	**5/5**	**2/5**	**4/5**	**3/5**	**3/5**

	Cabinet D		**Cabinet E**		**Average**
Task	Open door	close door	open drawer	close drawer	
OWL-ViT [95]	0/5	0/5	0/5	0/5	0/50
DR	0/5	2/5	2/5	**5/5**	9/50
URDFormer-ICP	0/5	3/5	3/5	2/5	24/45
URDFormer-TR	**5/5**	**4/5**	**4/5**	**5/5**	**39/50**

3. Sim-to-Real: In order to transfer back to the real world, the policy takes an RGB pointcloud, current end-effector pose and a natural language instruction, and predicts the next end-effector pose, using a PD controller to execute these predictions.

We term this specific instantiation of our real2sim2real pipeline URDFormer-TR and stress that the data generation process, the policy network, and the transfer procedure are highly flexible and can be replaced depending on the exact problem setting. **Baselines:** We compare URDFormer-TR with multiple variations of our real-to-simulation-real pipeline with varying degrees of access to real-world information.

(1) **OWL-ViT [95]:** First, we evaluate against a zero-shot vision-language model baseline. Inspired by VoxPoser [57], given a language instruction (Appendix C), we used the same open-vocabulary detector (OWL-ViT [95]) to predict bounding boxes for the prompted parts and handles that are important to the task. After mapping the detection to the observed pointcloud, a motion planner can then generate plans to solve the task directly in the real world.

(2) **Domain Randomization (DR) [145]:** Another approach is Domain Randomization (DR). We follow the augmentations from URDFormer-TR but on randomly generated cabinets with different configurations.

Similarly to URDFormer-TR, since we assume depth observations during inference, we also scale the generated cabinets to the real-world size. We follow the same approach for trajectory generation, policy training, and real-world transfer procedure of URDFormer-TR.

(3) **URDFormer-ICP:** This presents a learning-free, digital twin-style approach based on the Iterative Closest Point (ICP) [8] algorithm. First, we construct a simulation with the URDF created by the URDFormer and scale it according to the depth observation. We then use the ground truth simulation to compute end effector poses which we execute directly in the real world. When the cabinet's pose changes, we use ICP to transform the computed end effector poses to the new cabinet pose.

Real-world Results of Real-to-Sim-Real Training: Running the zero-shot OWL-ViT, we find that the model fails to predict the fine-grained details required to solve the tasks, i. e., "top middle drawer", "right door", and "handles". We visualize qualitative examples of results from OWL-ViT in Appendix C. Without localizing these regions of interest, the motion planning cannot solve the tasks leading to 0% success on all tasks.

While DR works surprisingly well on simple cabinets, e. g., Cabinet E which only has a single drawer, it fails to solve any of the more complicated configurations. We observe URDFormer-TR outperforms DR by 40% on average, showing the benefits of targeting the domain randomization procedure to the real-world configuration.

URDFormer-ICP shows average performance across all opening and closing tasks. When tasking it with "put object in bottom drawer", we observe the limitations of the approach. Since the object is a lot smaller than the cabinet, ICP matches the pointcloud of the cabinet instead of the object. This results in an inability to transform the endeffector pose with respect to the graspable object and results in failure to solve the task. In general, neither baseline can reliably solve putting and getting objects in/from the drawer. While URDFormer-TR succeeds 50% of the time, it showcases the difficulty of the task and leaves space for future improvements.

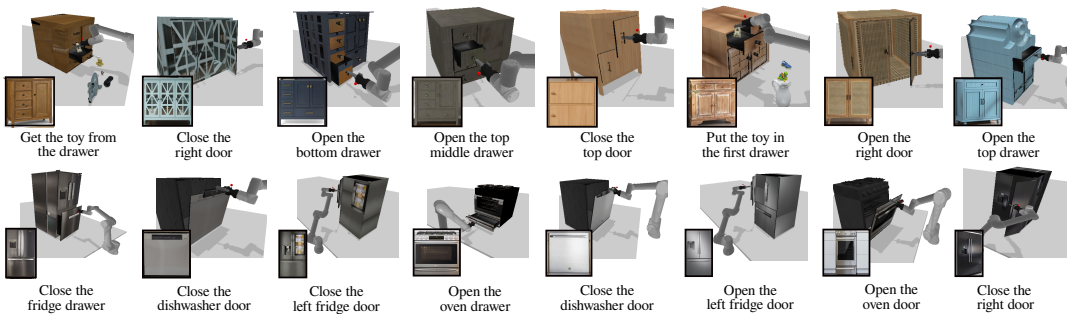


Figure 28: **Reality Gym:** A simulation environment with a variety of assets originated from internet images (black box) using URDFormer. We predict URDFs of internet images which can be loaded in any simulator. These URDFs are randomized with meshes from the Partnet dataset. We introduce 4 main tasks: (1) Open any articulated parts (2) close any articulated parts (3) fetch objects and (4) collect objects

Figure 29: **Generated Kitchen Scenes:** Examples of kitchen scenes predicted by URDFormer from internet images given labeled bounding boxes. Examples of failure parts are highlighted in red boxes.

Overall, the proposed real2sim2real pipeline using URDFormer and targeted randomization shows a **78%** success rate across all cabinet variations and tasks and an **85%** success rate on opening/closing tasks. Even though the URDFormer's predictions are not always accurate, as indicated by the red boxes in Fig27, the targeted randomization robustifies the policy network to allow for zero-shot real-world transfer more effectively than un-targeted simulation generation.

5.4.2 *Can URDFormer generate plausible and accurate simulation content from internet images?*

5.4.2.1 *Paired Training Dataset Generation*

To synthesize our paired training dataset, we first procedurally generate a set of URDF representations of scenes in simulation both for global scenes like kitchens and for single objects like ovens, cabinets, and fridges (Fig 25). We then follow the procedure in 5.2.2 to control the data generation of paired images, generating a large dataset of simulation scenes and paired realistic RGB images (Fig 25). For objects with diverse parts, we find that depth-guided Stable Diffusion [126] often ignores the semantic details of local parts, leading to inconsistencies as visualized in the Appendix C). As described in Section 5.2.2 we generated a large and diverse set of texture templates by using images of existing textures downloaded from the internet to guide the depth-guided stable diffusion. During training, we then randomly chose one template texture and warped it back to the original part region using perspective transformation. Finally, we applied a stable diffusion in-painting model [126] to smooth the boundary of the parts and generate background content (Details are described in Appendix C). In total, we generated approximately 118K image-URDF pairs across 7 categories of single articulated objects, and approximately 200K image-URDF pairs of global kitchen scenes.

5.4.2.2 *Object and Part Detection during Inverse Phase*

URDFormer takes both the RGB image and bounding boxes of the object parts and predicts URDFs. During the inverse phase, we adopt an off-the-shelf open vocabulary object detector GroundingDINO [80]. However, if we directly apply GroundingDINO on detecting parts such as drawers and handles, the detection performance is unsatisfying with an F1 score of 53.4%. Instead, if we use the same generated dataset that was used to train URDFormer (Visualized in Appendix A) and finetune groundingDINO, we observe an improvement with an F1 score of 66.2%. However, We also observe that compared to

the pretrained GroundingDINO, the finetuned GroundingDINO often fails to detect parts that have unique shapes or patterns. Inspired by recent work Model Soup[156], we simply average the pretrained weights and the finetuned weights. This leads to surprising improvement, with an F1 score of 79.7%. We additionally apply post-processing to remove duplicated boxes. Figure 30 shows Model Soup's influence on the bounding box detection.

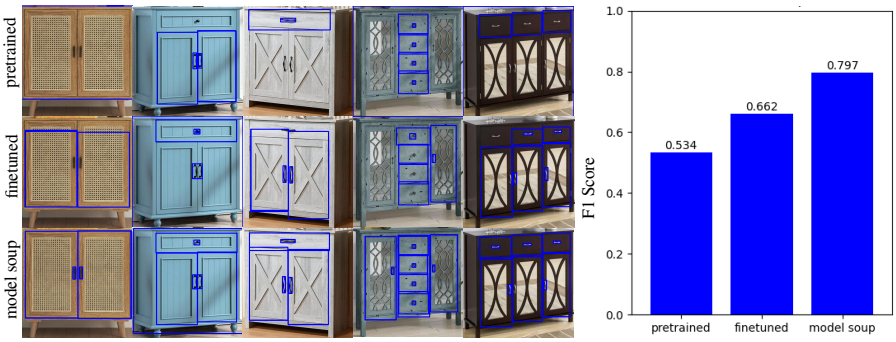


Figure 30: Comparison among pretrained, finetuned and model soup GroundingDINO on cabinet dataset

5.4.2.3 *Real World Evaluation Datasets*

We create two types of test sets for the evaluation of URDFormer: (a) *Object-Level* set includes labeled URDFs of 300 internet images of individual objects from 5 categories including 100 cabinets, 50 ovens, 50 dishwashers, 50 fridges and 50 laundry machines. (b) *Kitchen* set includes URDFs of 54 internet images of kitchens, with 5-15 articulated objects per kitchen. For each scene, we manually label the bounding box for each object and its parts, as well as the URDF primitives including mesh types, parent id, positions, and scales relative to its parent. We used the mesh types such as "left door", and "right door" to infer link axis and joint types. All the position values and scale values are discretized into 12 bins.

5.4.2.4 *Evaluation Metrics*

Evaluating entire scenes is challenging given the mixed structure and subjective nature of human labeling. We therefore measured accuracy of the predicted model using three individual sub-tasks: category ac-

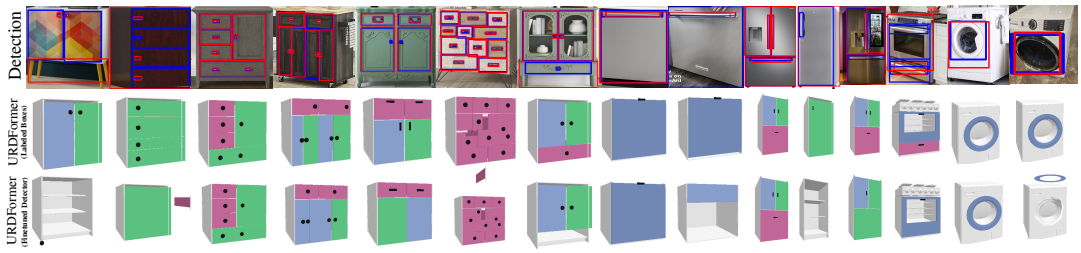


Figure 31: Successful and unsuccessful examples comparing URDFormer prediction on different articulated objects using (1) manually labeled bounding boxes and (2) bounding boxes provided by fine-tuned GroundingDINO. Image results of fine-tuned GroundingDINO (Red) compared with manually labeled boxes (Blue) are shown in the first row. Note that textures are removed for better visual comparison.

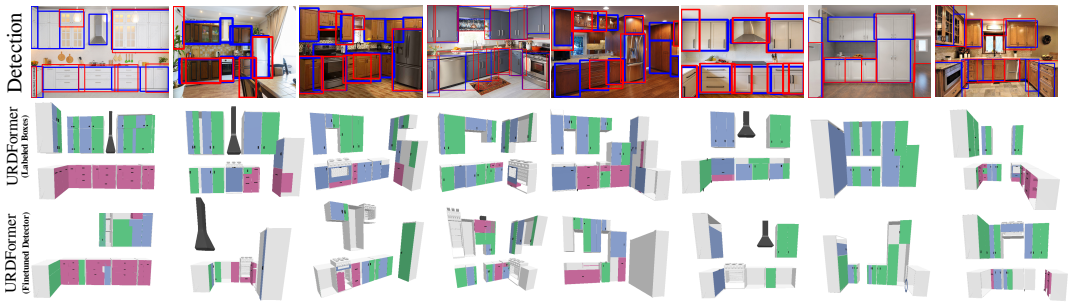


Figure 32: Successful and unsuccessful examples on kitchens comparing URDFormer prediction based on boxes generated by (1) manual labeling and (2) fine-tuned GroundingDINO. The first row also shows the boxes detected by fine-tuned GroundingDINO (Red) compared with manually labeled boxes (Blue)

curacy, parent accuracy and spatial error. In order to compute these statistics, we must first align predicted objects in the scene to their ground truth counterparts. This is accomplished by computing the intersection-over-union (IOU) of the 2D detected boxes used to instantiate the scene objects and the ground-truth bounding boxes of all the objects in the scene. Hungarian matching is then used to assign each detected box to a single ground truth box. Note that if the number of predicted boxes is not the same as the number of ground truth boxes, there will be some false positives (predicted objects that do not correspond to any ground truth objects) or false negatives (ground truth objects that do not have any associated predictions). We therefore also record the overall precision and recall of the objects in the scene. Once we have a set of aligned detected and ground-truth boxes, we say that the category of a predicted object is correct if

Figure 33: The forward phase can be applied to additional objects and create a diverse dataset for training URDFormer.

it matches the category of the assigned ground truth object. Similarly the predicted parent is correct if it matches the parent of the assigned ground truth object. Finally the spatial error is the average absolute error of the predicted discretized spatial coordinates. For larger objects, the model predicts four coordinates x_1, y_1, x_2, y_2 representing the bounding box of the object relative to its parent. For small objects such as handles and knobs, we predict only the object center x_1, y_1 and so the spatial error only considers these two values. Finally, we separate out these scores for high-level object predictions (cabinets, dishwashers, etc.) and low-level part predictions (doors, handles, etc.) so that the effects of various ablations on larger and smaller parts are more interpretable.

5.4.2.5 *Qualitative Results*

Figure 29 shows several examples of kitchens generated by URDFormer using internet images. While the model makes some mistakes, it is able to reproduce kitchen environments that largely match the structure of the original images. Figures 31 and 32 show successful and unsuccessful reconstructions of individual objects and scenes respectively. These images also show results when ground-truth boxes are provided. Again, while the model makes some mistakes, it largely captures the overall configuration of the objects and scenes.

5.4.2.6 *Ablation Study*

We compare the prediction of URDFormer trained with (1) **Ours**: realistic texture generated by part-consistency stable diffusion (2) **Random**: random texture downloaded from Describable Textures Dataset

Table 8: **Ablation Study**. We analyze which part training with generated texture benefits the most by comparing URDFormer trained with other textures, across GT boxes and boxes from the finetuned Grounding DINO detector.

		Kitchen								Object-Level							
		GT boxes				Grounding DINO				GT Boxes				Grounding DINO			
		Ours	Rand	Sim	Sel	Ours	Rand	Sim	Sel	Ours	Rand	Sim	Sel	Ours	Rand	Sim	Sel
Global	Mesh Acc (↑)	**0.578**	0.42	0.407	0.576	**0.603**	0.533	0.354	0.462	**0.740**	0.463	0.520	0.677	**0.688**	0.440	0.520	0.630
	Parent Acc (↑)	**0.833**	0.831	0.813	0.807	0.816	**0.819**	0.810	**0.819**	—	—	—	—	—	—	—	—
	Spatial Err (↓)	**0.809**	0.845	0.848	0.885	**0.987**	1.036	1.038	1.032	—	—	—	—	—	—	—	—
	Recall (↑)	1	1	1	1	0.726	0.726	0.726	0.726	—	—	—	—	—	—	—	—
	Precision (↑)	1	1	1	1	0.951	0.951	0.951	0.951	—	—	—	—	—	—	—	—
Parts	Mesh Acc (↑)	0.704	**0.719**	0.662	0.675	0.537	**0.558**	0.505	0.522	0.903	**0.927**	0.867	0.861	0.851	**0.865**	0.811	0.803
	Parent Acc (↑)	0.765	**0.773**	0.763	0.75	**0.711**	0.696	0.707	0.711	0.874	**0.878**	0.857	0.857	**0.826**	0.821	0.806	0.805
	Spatial Err (↓)	1.799	**1.699**	1.891	1.981	2.957	2.885	**2.798**	3.107	0.478	**0.420**	0.649	0.867	0.791	**0.753**	0.917	1.129
	Recall (↑)	1	1	1	1	0.495	0.495	0.495	0.495	1	1	1	1	0.853	0.832	0.832	0.832
	Pred Precision (↑)	1	1	1	1	0.927	0.927	0.927	0.927	1	1	1	1	0.986	0.987	0.987	0.987

[20] (3) **Sim**: random 3-channels RGB color and (4) **Selected**: carefully selected texture images that matches the object categories, such as wood texture (cabinet), metal texture (dishwasher) and so on. We observe that generated realistic texture is particularly helpful in global prediction, including identifying object types (cabinet or oven), parents (which wall the cabinet belongs to), and where to put the object. Surprisingly, we found the texture realism does not affect so much when predicting part structures and sometimes is slightly worse than random texture by 1% to 2%. This is likely due to using bounding box position features is sufficient for predicting simple low-level structures. For example, if a small box A is in the center of another box B, box B is likely a drawer instead of a door or a handle. On average, prediction using finetuned object detector performance is worse than using GT boxes due to detection error. However, one surprising observation is that using detected boxes helps slightly with identifying global object types. We hypothesize this is because boxes labeled by humans sometimes group multiple cabinets into one single box if they are close, making mesh prediction slightly challenging.

RealityGym: With the ability to cheaply generate an arbitrary number of realistic simulation assets directly from the internet, we introduce *RealityGym*, a robot learning suite with a collection of realistic

Figure 34: Qualitative study of URDFormer generalization to other scenes and objects with successful and unsuccessful (highlighted in red) examples. In particular, We train URDFormer on the new dataset with added categories shown in Fig33 and evaluate on internet images for both object-level and global prediction. Interestingly, we did not train URDFormer on the Laundry Room and Study Room but found URDFormer can generalize to these unseen categories, which is likely due to the two-stage training of URDFormer Global and URDFormer Part. Please note that URDFormer does not reconstruct accurate meshes or predict properties such as friction or mass, which is detailed in the limitation and future work sections.

simulation assets and scenes created from real world RGB images using URDFormer. We provide an initial set of 300 objects (Cabinets, Ovens, Fridges, wahsers and Dishwashers) and 50 kitchen scenes from internet images, with future work looking to expand this into a bigger dataset. In addition, we also provide 84 meshes of cabinet frames, 20 door meshes, 59 drawers, 440 handles and 116 knobs from PartNet[96]. We can randomly incorporate these meshes into the initial URDFs to generate diverse scenes complete with articulated objects. We define 4 main tasks: (1) Open any articulated parts i. e.top middle drawer (2) Close any articulated parts (3) Fetch objects (4) Collect objects. We automatically generate tasks and their language descriptions, and use a motion planner (Curobo [139]) to complete the tasks. Fig 28 and Fig 29 show examples of robots performing in RealityGym on a variety of generated simulation environments and assets. Details about RealityGym can be found in Appendix C.

5.4.3 *Can URDFormer generalize to diverse objects and scenes?*

In order to demonstrate the generalization capability of URDFormer, we used the same techniques discussed on Section 5.2 to create five additional object categories and four additional scene categories for qualitative evaluation. The new object categories are toilet, microwave, desk, laptop and chair, while the four scene categories are bedroom, bathroom, laundry room and study room. We trained a single new part model that incorporates these additional categories by adding approximately 6k training examples per object category to the original training dataset. We took the same approach to train a single global scene model and added approximately 10k training examples for the bathroom and bedroom. We found that the laundry room and study could be adequately captured with no new global scene examples, as these categories only contain objects that are present in the other scene categories. Figure 33 shows example training data generated by the forward pipeline, while Figure 34 shows qualitative examples of objects and scenes inferred by a URDFormer trained on this data. More examples are available in Appendix C.

5.4.4 *Can URDFormer support different robots and tasks?*

Figure 35: We trained a Stretch robot on a multi-step task "Clean Up the Table Surface" using URDFormer prediction

To demonstrate that URDFormer supports multi-step tasks and different robots, we train an additional policy for a Stretch robot to place an

Figure 36: URDFormer can be applied to a different robot such as a Stretch Robot to perform a multi-step task such as "clean up the table surface". We apply URDFormer to predict the URDF of a study desk, and render a dataset to train a vision-based policy to predict an affordance map at each step.

object in a desk drawer as shown in Figure 35. Figure 36 shows the data generation process for training the mobile robot's policy. First, a single image of the robot's environment is passed to a pretrained URDFormer model to produce a scene description of the desk. Then, this scene description is used to generate training data in simulation that can instruct the robot how to accomplish it's multi-stage objective. When generating this data, we use an inpainting model [126] to reduce the sim2real gap by inpainting the pixels covered by the simulated object overlaid onto the original image. Finally the robot policy, which predicts a per-step affordance map from an initial image, is trained on this new generated data. This policy is implemented as a UNet, which takes the initial observation, as well as the task embedding and predicts multiple object affordance maps. These affordance maps are used with a motion planner to guide the robot at each step. Details on training the policy can be found in Appendix C.

5.5 LIMITATIONS AND FUTURE WORK

Please see the full list of limitations and future work in Appendix C.

Part Detection URDFormer relies on the performance of bounding box detection. Although the finetuned Grounding DINO improves performance than the pretrained model, there is still a gap for improvement, especially on global scene detection.

Texture and Meshes URDFormer focuses on predicting kinematic URDF structures and uses predefined meshes that might not match the real-world scenes. To apply textures, we simply assume all parts

are rectangular shapes, and use the bounding box of each object part to crop the image and import it into a uv map template. However, this does not work for irregular meshes such as a donut-shape door, or when the object in the image is tilted.

Limited URDF Primitives URDFormer currently only supports articulated objects that have limited joint types such as prismatic and revolute, and cannot predict complex objects such as cars and lamps.

Link Collisions URDFormer only predicts URDF primitives for each bounding box, which sometimes leads to a collision between two links. Further post-processing is required to resolve this issue.

Multiple Trained Components Our pipeline is not trained end-to-end and consists of multiple learning components. While this increases the complexity of the system, it is necessary to ensure consistency when using the generative model, and make the most use of existing pretrained components.

Inferred Physical Properties Our system does not presently attempt to infer physical properties such as mass, inertial moments or friction directly from observations in the scene. In theory, the visual information present in the images may allow for a rough approximation of these quantities. We expect this to be a productive direction for future work.

5.6 SUMMARY

We present a scalable pipeline for creating articulated simulation assets from real-world images. In particular, we introduce a forward-inverse framework that generates realistic and consistent images of articulated objects and trains a transformer-based network URDFormer to predict their corresponding URDFs. We present RealityGym, a robot learning suite with realistic simulation assets generated from real-world images. Additionally, we show that the predicted URDFs with targeted domain randomization enable better zero-shot performance in the real-world. Our pipeline provides the first step towards cheaper and scalable realistic scene generation for robot learning.

6

CONCLUSION

Visual imitation learning enables robots to interact with their environment using image-based observations. However, training such visual policies can be expensive and time-consuming due to the need for extensive data collection. Traditional augmentation methods often fail to accurately represent real-world distributions, which limits robot generalization in unseen situations.

To address this challenge, I have introduced targeted generative augmentation. This approach aims to generate data that closely mimics real-world scenarios, thereby effectively expanding the initial dataset in a semantically meaningful way. In this dissertation, I explored the use of targeted generative augmentation through three lines of work:

Static Targeted Augmentation for Visual Diversity: I introduced GenAug, a controllable augmentation system that increases scene diversity by leveraging a text-to-image diffusion model. I have shown that this augmented dataset notably improves robot policies in completely unseen scenes and objects, with only a minimal amount of initial expert demonstrations.

Dynamic Targeted Augmentation for Physical Realism : I developed ISAGrasp, a method that combines targeted augmentation with simulation to synthesize diverse object shapes and their successful grasping poses. This approach demonstrates that integrating targeted augmentation with a physics simulator can produce physically accurate augmented data, thereby enhancing robot performance with unseen objects and scenarios.

Contextual Targeted Augmentation for Real-World Scene Generation: I presented URDFormer, a pipeline that constructs detailed kinematic structures of articulated objects and scenes directly from RGB images. This method utilizes image data from the internet to create more realistic and potentially large-scale simulated environments.

The generated simulations are applicable for training downstream robotic tasks and can be effectively transferred to real-world scenes.

This thesis highlights the role of targeted generative augmentation in overcoming the challenges of expensive data collection and in emulating real-world distributions for visual imitation learning. In the future, it would be interesting to explore the integration of data generation in a closed-loop system. This system would observe real-world failures of robots and generate more challenging data to improve the robustness of robot policies in scenarios where robots are more likely to fail. Additionally, combining contextual targeted augmentation with real-world 3D scans to create accurate and interactive simulation assets is another direction worth pursuing. Another interesting direction is the development of more accurate digital twins by incorporating common sense physics, such as object mass and friction based on materials, into simulation content generation. As training a generalizable robot policy for unstructured environments remains a challenging problem, I hope this work inspires fruitful steps in this direction.

BIBLIOGRAPHY

[1] Hameed Abdul-Rashid, Miles Freeman, Ben Abbatematteo, George Konidaris, and Daniel Ritchie. "Learning to infer kinematic hierarchies for novel object instances." In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 8461–8467.

[2] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. "Building rome in a day." In: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[3] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. "Solving rubik's cube with a robot hand." In: *arXiv preprint arXiv:1910.07113* (2019).

[4] Mark S. Alber et al. "Integrating machine learning and multiscale modeling - perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences." In: *npj Digit. Medicine* 2 (2019). DOI: 10.1038/s41746-019-0193-y. URL: https://doi.org/10.1038/s41746-019-0193-y.

[5] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. "End-to-end differentiable physics for learning and control." In: *Advances in neural information processing systems* 31 (2018).

[6] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. "Simulation as an engine of physical scene understanding." In: *Proceedings of the National Academy of Sciences* 110.45 (2013), pp. 18327–18332.

[7] Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew G Wilson. "Learning invariances in neural networks from train-

ing data." In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17605–17616.

[8] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes." In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606. URL: https://www.researchgate.net/publication/3191994_A_method_for_registration_of_3-D_shapes_IEEE_Trans_Pattern_Anal_Mach_Intell.

[9] Samarth Brahmbhatt, Ankur Handa, James Hays, and Dieter Fox. "ContactGrasp: Functional Multi-finger Grasp Synthesis from Contact." In: *IROS*. 2019.

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. "Rt-1: Robotics transformer for real-world control at scale." In: *arXiv preprint arXiv:2212.06817* (2022).

[11] Anthony Brohan et al. "RT-1: Robotics Transformer for Real-World Control at Scale." In: *CoRR* abs/2212.06817 (2022). DOI: 10.48550/arXiv.2212.06817. arXiv: 2212.06817. URL: https://doi.org/10.48550/arXiv.2212.06817.

[12] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. "The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research." In: *2015 International Conference on Advanced Robotics (ICAR)*. 2015.

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

[14] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. "Text to 3d scene generation with rich lexical grounding." In: *arXiv preprint arXiv:1505.06289* (2015).

[15] Angel Chang, Manolis Savva, and Christopher D Manning. "Learning spatial knowledge for text to 3D scene generation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 2028–2038.

[16] Yu-Wei Chao et al. "DexYCB: A Benchmark for Capturing Hand Grasping of Objects." In: *CVPR*. 2021.

[17] Tao Chen, Jie Xu, and Pulkit Agrawal. "A System for General In-Hand Object Re-Orientation." In: *CoRL*. 2021.

[18] Zoey Qiuyu Chen, Karl Van Wyk, Yu-Wei Chao, Wei Yang, Arsalan Mousavian, Abhishek Gupta, and Dieter Fox. "Learning robust real-world dexterous grasping policies via implicit shape augmentation." In: *arXiv preprint arXiv:2210.13638* (2022).

[19] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. "Genaug: Retargeting behaviors to unseen situations via generative augmentation." In: *arXiv preprint arXiv:2302.06671* (2023).

[20] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. "Describing Textures in the Wild." In: *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2014. URL: https://arxiv.org/pdf/1311.3618.pdf.

[21] Open X-Embodiment Collaboration et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. https://arxiv.org/abs/2310.08864. 2023.

[22] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. "A Review of Physics Simulators for Robotic Applications." In: *IEEE Access* 9 (2021), pp. 51416–51431. DOI: 10.1109/ACCESS.2021.3068769. URL: https://doi.org/10.1109/ACCESS.2021.3068769.

[23] NVIDIA Corporation. *A PyTorch Extension: Tools for Easy Mixed Precision and Distributed Training in PyTorch*. https://github.com/NVIDIA/apex. 2020.

[24] Erwin Coumans and Yunfei Bai. *PyBullet: a Python module for physics simulation for games, robotics and machine learning*. https://pybullet.org. 2016–2021.

[25] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021.

[26] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. "Autoaugment: Learning augmentation policies from data." In: *arXiv preprint arXiv:1805.09501* (2018).

[27] Matt Deitke, Rose Hendrix, Ali Farhadi, Kiana Ehsani, and Aniruddha Kembhavi. "Phone2Proc: Bringing Robust Robots into Our Chaotic World." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 9665–9675. DOI: 10.1109/CVPR52729.2023.00932. URL: https://doi.org/10.1109/CVPR52729.2023.00932.

[28] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. "Objaverse: A universe of annotated 3d objects." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13142–13153.

[29] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. "ProcTHOR: Large-Scale Embodied AI Using Procedural Generation." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 5982–5994. URL: https://arxiv.org/pdf/2206.06994.pdf.

[30] Matt Deitke et al. "ProcTHOR: Large-Scale Embodied AI Using Procedural Generation." In: *CoRR* abs/2206.06994 (2022). DOI: 10.48550/arXiv.2206.06994. arXiv: 2206.06994. URL: https://doi.org/10.48550/arXiv.2206.06994.

[31] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. "Vector neurons: A general framework for so (3)-equivariant networks." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12200–12209.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. URL: https://doi.org/10.1109/CVPR.2009.5206848.

[33] Yu Deng, Jiaolong Yang, and Xin Tong. "Deformed implicit field: Modeling 3d shapes with learned dense correspondence." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10286–10296.

[34] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. "Emergent complexity and zero-shot transfer via unsupervised environment design." In: *Advances in neural information processing systems* 33 (2020), pp. 13049–13061.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." In: *arXiv preprint arXiv:2010.11929* (2020). URL: https://arxiv.org/pdf/2010.11929.pdf.

[36] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. "Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items." In: *arXiv preprint arXiv:2204.11918* (2022).

[37] Sam Earle, Maria Edwards, Ahmed Khalifa, Philip Bontrager, and Julian Togelius. "Learning controllable content generators." In: *2021 IEEE Conference on Games (CoG)*. IEEE. 2021, pp. 1–9.

[38] Chuan Fang, Xiaotao Hu, Kunming Luo, and Ping Tan. "Ctrl-Room: Controllable Text-to-3D Room Meshes Generation with Layout Constraints." In: *arXiv preprint arXiv:2310.03602* (2023). URL: https://arxiv.org/pdf/2310.03602.pdf.

[39] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. "One-shot visual imitation learning via meta-learning." In: *Conference on robot learning*. PMLR. 2017, pp. 357–368.

[40] Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. "Motion Policy Networks." In: *CoRR* abs/2210.12209 (2022). DOI: 10.48550/arXiv.2210.12209. arXiv: 2210.12209. URL: https://doi.org/10.48550/arXiv.2210.12209.

[41] Free3D. *Free3D*. https://free3d.com/.

[42] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. "Adversarial Reinforcement Learning for Procedural Content Generation." In: *2021 IEEE Conference on Games (CoG), Copenhagen, Denmark, August 17-20, 2021*. IEEE, 2021, pp. 1–8. DOI: 10.1109/CoG52621.2021.9619053. URL: https://doi.org/10.1109/CoG52621.2021.9619053.

[43] Michael Gleicher. "Retargetting Motion to New Characters." In: *SIGGRAPH*. 1998.

[44] Abhishek Gupta, Justin Yu, Tony Z. Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. "Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention." In: *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 6664–6671. DOI: 10.1109/ICRA48506.2021.9561384. URL: https://doi.org/10.1109/ICRA48506.2021.9561384.

[45] Agrim Gupta, Piotr Dollar, and Ross Girshick. "LVIS: A Dataset for Large Vocabulary Instance Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

[46] Arjun Gupta, Max Shepherd, and Saurabh Gupta. "Predicting Motion Plans for Articulating Everyday Objects." In: *International Conference on Robotics and Automation (ICRA)*. 2023. URL: https://ieeexplore.ieee.org/document/10160752.

[47] David Ha and Jürgen Schmidhuber. "World models." In: *arXiv preprint arXiv:1803.10122* (2018).

[48] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1856–1865. URL: http://proceedings.mlr.press/v80/haarnoja18b.html.

[49] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination." In: *arXiv preprint arXiv:1912.01603* (2019).

[50] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality." In: *arXiv preprint arXiv:2210.13702* (2022).

[51] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. "DexPilot: Vision-Based Teleoperation of Dexterous Robotic Hand-Arm System." In: *ICRA*. 2020.

[52] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969. URL: https://arxiv.org/pdf/1703.06870.pdf.

[53] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments." In: *The international journal of Robotics Research* 31.5 (2012), pp. 647–663.

[54] Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Rares Andrei Ambrus, Jeannette Bohg, Abhinav Valada, and Thomas Kollar. "Carto: Category and joint

agnostic reconstruction of articulated objects." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21201–21210.

[55] Cheng-Chun Hsu, Zhenyu Jiang, and Yuke Zhu. "Ditto in the house: Building articulation models of indoor scenes through interactive perception." In: *arXiv preprint arXiv:2302.01295* (2023). URL: https://arxiv.org/abs/2302.01295.

[56] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. "Graph2plan: Learning floorplan generation from layout graphs." In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 118–1.

[57] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. "VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models." In: *arXiv preprint arXiv:2307.05973* (2023).

[58] Drew A. Hudson and C. Lawrence Zitnick. "Generative Adversarial Transformers." In: *CoRR* abs/2103.01209 (2021). arXiv: 2103.01209. URL: https://arxiv.org/abs/2103.01209.

[59] Rae Jeong, Jost Tobias Springenberg, Jackie Kay, Dan Zheng, Alexandre Galashov, Nicolas Heess, and Francesco Nori. "Learning Dexterous Manipulation from Suboptimal Experts." In: *CoRL*. 2020.

[60] Xiaowei Jia, Yiqun Xie, Sheng Li, Shengyu Chen, Jacob Zwart, Jeffrey M. Sadler, Alison P. Appling, Samantha Oliver, and Jordan S. Read. "Physics-Guided Machine Learning from Simulation Data: An Application in Modeling Lake and River Systems." In: *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*. Ed. by James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu. IEEE, 2021, pp. 270–279. DOI: 10.1109/ICDM51629.2021.00037. URL: https://doi.org/10.1109/ICDM51629.2021.00037.

[61] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. "Ditto: Building Digital Twins of Articulated Objects from Interaction." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[62] Dmitry Kalashnikov et al. "QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation." In: *CoRR* abs/1806.10293 (2018). arXiv: 1806.10293. URL: http://arxiv.org/abs/1806.10293.

[63] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. "Differentiable rendering: A survey." In: *arXiv preprint arXiv:2006.12057* (2020).

[64] Dov Katz, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. "Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects." In: *2013 IEEE International Conference on Robotics and Automation*. IEEE. 2013, pp. 5003–5010.

[65] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. "3d gaussian splatting for real-time radiance field rendering." In: *ACM Transactions on Graphics (ToG)* 42.4 (2023), pp. 1–14.

[66] Mohammad Keshavarzi, Aakash Parikh, Xiyu Zhai, Melody Mao, Luisa Caldas, and Allen Y Yang. "Scenegen: Generative contextual scene augmentation using scene graph priors." In: *arXiv preprint arXiv:2009.12395* (2020).

[67] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. "Pcgrl: Procedural content generation via reinforcement learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 16. 1. 2020, pp. 95–101.

[68] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. "Segment Anything." In: *arXiv preprint arXiv:2304.02643* (2023).

[69] Marian Kleineberg, Matthias Fey, and Frank Weichert. "Adversarial generation of continuous implicit shape representations." In: *arXiv preprint arXiv:2002.00349* (2020).

[70] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, et al. "Ai2-thor: An interactive 3d environment for visual ai." In: *arXiv preprint arXiv:1712.05474* (2017). URL: https://arxiv.org/pdf/1712.05474.pdf.

[71] Ilya Kostrikov, Denis Yarats, and Rob Fergus. "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels." In: *arXiv preprint arXiv:2004.13649* (2020).

[72] James R Kubricht, Keith J Holyoak, and Hongjing Lu. "Intuitive physics: Current research and controversies." In: *Trends in cognitive sciences* 21.10 (2017), pp. 749–759.

[73] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-End Training of Deep Visuomotor Policies." In: *CoRR* abs/1504.00702 (2015). arXiv: 1504.00702. URL: http://arxiv.org/abs/1504.00702.

[74] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. "Grains: Generative recursive autoencoders for indoor scenes." In: *ACM Transactions on Graphics (TOG)* 38.2 (2019), pp. 1–16.

[75] Zhengqin Li et al. "OpenRooms: An Open Framework for Photorealistic Indoor Scene Datasets." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 7190–7199. DOI: 10.1109/CVPR46437.2021.00711. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Li_OpenRooms_An_Open_Framework_for_Photorealistic_Indoor_Scene_Datasets_CVPR_2021_paper.html.

[76] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. "Paparazzi: surface editing by way of multi-view image processing." In: *ACM Trans. Graph.* 37.6 (2018), pp. 221–1.

[77] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. "PARIS: Part-level Reconstruction and Motion Analysis for Articulated Objects." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 352–363.

[78] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. "Akb-48: A real-world articulated object knowledge base." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14809–14818.

[79] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. "One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization." In: *arXiv preprint arXiv:2306.16928* (2023). URL: https://arxiv.org/pdf/2306.16928.pdf.

[80] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. "Grounding dino: Marrying dino with grounded pre-training for open-set object detection." In: *arXiv preprint arXiv:2303.05499* (2023). URL: https://arxiv.org/pdf/2303.05499.pdf.

[81] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. "Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis." In: *arXiv preprint arXiv:2308.09713* (2023).

[82] Liqian Ma, Jiaojiao Meng, Shuntao Liu, Weihang Chen, Jing Xu, and Rui Chen. "Sim2Real2: Actively Building Explicit Physics Model for Precise Articulated Object Manipulation." In: *International Conference on Robotics and Automation (ICRA)*. 2023.

[83] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training." In: *arXiv preprint arXiv:2210.00030* (2022).

[84] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics." In: *Robotics:*

Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017. Ed. by Nancy M. Amato, Siddhartha S. Srinivasa, Nora Ayanian, and Scott Kuindersma. 2017. DOI: 10.15607/RSS.2017.XIII.058. URL: http://www.roboticsproceedings.org/rss13/p58.html.

[85] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics." In: *arXiv preprint arXiv:1703.09312* (2017).

[86] Priyanka Mandikal and Kristen Grauman. "DexVIP: Learning Dexterous Grasping with Human Hand Pose Priors from Video." In: *CoRL*. 2021.

[87] Priyanka Mandikal and Kristen Grauman. "Learning Dexterous Grasping with Object-Centric Visual Affordances." In: *ICRA*. 2021.

[88] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. "Roboturk: A crowdsourcing platform for robotic skill learning through imitation." In: *Conference on Robot Learning*. PMLR. 2018, pp. 879–893.

[89] Yongsen Mao, Yiming Zhang, Hanxiao Jiang, Angel Chang, and Manolis Savva. "MultiScan: Scalable RGBD scanning for 3D environments with articulated objects." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9058–9071.

[90] Roberto Martín-Martín, Clemens Eppner, and Oliver Brock. "The RBO dataset of articulated objects and interactions." In: *The International Journal of Robotics Research* 38.9 (2019), pp. 1013–1019.

[91] Marius Memmel, Roman Bachmann, and Amir Zamir. "Modality-invariant Visual Odometry for Embodied Vision." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21549–21559. URL: https://openaccess.thecvf.com/content/CVPR2023/papers/Memmel_Modality-

Invariant_Visual_Odometry_for_Embodied_Vision_CVPR_2023_paper.pdf.

[92] Marius Memmel, Puze Liu, Davide Tateo, and Jan Peters. "Dimensionality Reduction and Prioritized Exploration for Policy Search." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 2134–2157. URL: https://proceedings.mlr.press/v151/memmel22a.html.

[93] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. "Nerf: Representing scenes as neural radiance fields for view synthesis." In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[94] 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.

[95] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. "Simple open-vocabulary object detection." In: *European Conference on Computer Vision*. 2022. URL: https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136700714.pdf.

[96] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 909–918. DOI: 10.1109/CVPR.2019.00100. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Mo_PartNet_A_Large-Scale_Benchmark_for_Fine-Grained_and_Hierarchical_Part-Level_3D_CVPR_2019_paper.html.

[97] Eyal Molad, Eliahu Horwitz, Dani Valevski, Alex Rav Acha, Yossi Matias, Yael Pritch, Yaniv Leviathan, and Yedid Hoshen.

"Dreamix: Video diffusion models are general video editors." In: *arXiv preprint arXiv:2302.01329* (2023).

[98] Liliane Momeni, Mathilde Caron, Arsha Nagrani, Andrew Zisserman, and Cordelia Schmid. "Verbs in action: Improving verb understanding in video-language models." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 15579–15591.

[99] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation." In: *ICCV*. 2019.

[100] Matthias Müller, Vincent Casser, Jean Lahoud, Neil Smith, and Bernard Ghanem. "Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications." In: *Int. J. Comput. Vis.* 126.9 (2018), pp. 902–919. DOI: 10.1007/s11263-018-1073-7. URL: https://doi.org/10.1007/s11263-018-1073-7.

[101] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. "Deep Dynamics Models for Learning Dexterous Manipulation." In: *CoRL*. 2019.

[102] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. "Visual reinforcement learning with imagined goals." In: *Advances in neural information processing systems* 31 (2018).

[103] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. "R3m: A universal visual representation for robot manipulation." In: *arXiv preprint arXiv:2203.12601* (2022).

[104] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. "R3m: A universal visual representation for robot manipulation." In: *arXiv preprint arXiv:2203.12601* (2022).

[105] Yashraj S. Narang et al. "Factory: Fast Contact for Robotic Assembly." In: *Robotics: Science and Systems XVIII, New York City, NY, USA, June 27 - July 1, 2022*. Ed. by Kris Hauser, Dylan A.

Shell, and Shoudong Huang. 2022. DOI: 10.15607/RSS.2022.XVIII.035. URL: https://doi.org/10.15607/RSS.2022.XVIII.035.

[106] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. "House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13632–13641.

[107] Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. "Rendernet: A deep convolutional network for differentiable rendering from 3d shapes." In: *Advances in neural information processing systems* 31 (2018).

[108] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. "Structure from Action: Learning Interactions for Articulated Object 3D Structure Discovery." In: *arXiv preprint arXiv:2207.08997* (2022).

[109] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "Deepsdf: Learning continuous signed distance functions for shape representation." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[110] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan Goldman, Steven Seitz, and Ricardo Martin-Brualla. "Deformable Neural Radiance Fields." In: *https://arxiv.org/abs/2011.12948* (2020).

[111] Luis Perez and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." In: *arXiv preprint arXiv:1712.04621* (2017).

[112] Luis S Piloto, Ari Weinstein, Peter Battaglia, and Matthew Botvinick. "Intuitive physics learning in a deep-learning model inspired by developmental psychology." In: *Nature human behaviour* 6.9 (2022), pp. 1257–1267.

[113] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. "Asymmetric actor critic for image-based robot learning." In: *arXiv preprint arXiv:1710.06542* (2017).

[114] Dean Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network." In: *Advances in Neural Information Processing Systems 1, [NIPS Conference, Denver, Colorado, USA, 1988]*. Ed. by David S. Touretzky. Morgan Kaufmann, 1988, pp. 305–313. URL: http://papers.nips.cc/paper/95-alvinn-an-autonomous-land-vehicle-in-a-neural-network.

[115] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. "DreamFusion: Text-to-3D using 2D Diffusion." In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: https://openreview.net/pdf?id=FjNys5c7VyY.

[116] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space." In: *NIPS*. 2017.

[117] Shengyi Qian, Linyi Jin, Chris Rockwell, Siyi Chen, and David F. Fouhey. "Understanding 3D Object Articulation in Internet Videos." In: *CVPR*. 2022.

[118] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. "DexMV: Imitation Learning for Dexterous Manipulation from Human Videos." In: *arXiv preprint arXiv:2108.05877* (2021).

[119] Ahmed Hussain Qureshi, Mayur J. Bency, and Michael C. Yip. "Motion Planning Networks." In: *CoRR* abs/1806.05767 (2018). arXiv: 1806.05767. URL: http://arxiv.org/abs/1806.05767.

[120] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. "Real-world robot learning with masked visual pre-training." In: *Conference on Robot Learning*. PMLR. 2023, pp. 416–426.

[121] Alexander Raistrick et al. "Infinite Photorealistic Worlds Using Procedural Generation." In: *IEEE/CVF Conference on Com-*

puter Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023. IEEE, 2023, pp. 12630–12641. DOI: 10.1109/CVPR52729.2023.01215. URL: https://doi.org/10.1109/CVPR52729.2023.01215.

[122] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations." In: *RSS*. 2018.

[123] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. "Hierarchical text-conditional image generation with clip latents." In: *arXiv preprint arXiv:2204.06125* (2022).

[124] Daniel Ritchie, Kai Wang, and Yu-an Lin. "Fast and flexible indoor scene synthesis via deep convolutional generative models." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6182–6190.

[125] Wonik Robotics. *Allegro Hand*. https://www.wonikrobotics.com/research-robot-hand.

[126] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695. URL: https://openaccess.thecvf.com/content/CVPR2022/papers/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.pdf.

[127] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[128] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. "Photorealistic text-to-image diffusion models with deep language understanding." In: *arXiv preprint arXiv:2205.11487* (2022).

[129] Taha Samavati and Mohsen Soryani. "Deep learning-based 3D reconstruction: a survey." In: *Artif. Intell. Rev.* 56.9 (2023), pp. 9175–9219. DOI: 10.1007/s10462-023-10399-2. URL: https://doi.org/10.1007/s10462-023-10399-2.

[130] Christoph Schuhmann et al. "LAION-5B: An open large-scale dataset for training next generation image-text models." In: *CoRR* abs/2210.08402 (2022). DOI: 10.48550/arXiv.2210.08402. arXiv: 2210.08402. URL: https://doi.org/10.48550/arXiv.2210.08402.

[131] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347* (2017). URL: https://arxiv.org/pdf/1707.06347.pdf.

[132] Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. "iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 7520–7527.

[133] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." In: *J. Big Data* 6 (2019), p. 60. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[134] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning." In: *Journal of big data* 6.1 (2019), pp. 1–48.

[135] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "CLIPort: What and Where Pathways for Robotic Manipulation." In: *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2021.

[136] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation." In: *Conference on Robot Learning*. PMLR. 2022, pp. 894–906.

[137] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. "Make-a-video: Text-to-video generation without text-video data." In: *arXiv preprint arXiv:2209.14792* (2022).

[138] J ürgen Sturm, Vijay Pradeep, Cyrill Stachniss, Christian Plagemann, Kurt Konolige, and Wolfram Burgard. "Learning Kinematic Models for Articulated Objects." In: ().

[139] Balakumar Sundaralingam et al. *cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation*. 2023. arXiv: 2310.17274 [cs.RO]. URL: https://arxiv.org/pdf/2310.17274.pdf.

[140] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13438–13444.

[141] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. "Habitat 2.0: Training home assistants to rearrange their habitat." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 251–266.

[142] Andrew Szot et al. "Habitat 2.0: Training Home Assistants to Rearrange their Habitat." In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan. 2021, pp. 251–266. URL: https://proceedings.neurips.cc/paper/2021/hash/021bbc7ee20b71134d53e20206bd6feb-Abstract.html.

[143] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, 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.

[144] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world." In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[145] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*. IEEE, 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133. URL: https://doi.org/10.1109/IROS.2017.8202133.

[146] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "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.

[147] Karl Van Wyk et al. "Geometric Fabrics: Generalizing Classical Mechanics to Capture the Physics of Behavior." In: *RA-L* (2022).

[148] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems* 30 (2017). URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[149] Dian Wang, Mingxi Jia, Xupeng Zhu, Robin Walters, and Robert Platt. "On-Robot Policy Learning with O(2)-Equivariant SAC." In: *CoRR* abs/2203.04923 (2022). DOI: 10.48550/arXiv.2203.04923. arXiv: 2203.04923. URL: https://doi.org/10.48550/arXiv.2203.04923.

[150] Kai Wang, Xianghao Xu, Leon Lei, Selena Ling, Natalie Lindsay, Angel X. Chang, Manolis Savva, and Daniel Ritchie. "Roominoes: Generating Novel 3D Floor Plans From Existing 3D Rooms." In: *Comput. Graph. Forum* 40.5 (2021), pp. 57–69. DOI: 10.1111/cgf.14357. URL: https://doi.org/10.1111/cgf.14357.

[151] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. "GenSim: Generating Robotic Simulation Tasks via Large Language Models." In: *Arxiv*. 2023. URL: https://arxiv.org/pdf/2310.01361.pdf.

[152] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. "Shape2motion: Joint analysis of motion parts and attributes from 3d shapes." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8876–8884.

[153] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. "Robogen: Towards unleashing infinite data for automated robot learning via generative simulation." In: *arXiv preprint arXiv:2311.01455* (2023). URL: https://arxiv.org/pdf/2311.01455.pdf.

[154] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. "Self-supervised neural articulated shape and appearance models." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15816–15826.

[155] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J Guibas. "Captra: Category-level pose tracking for rigid and articulated objects from point clouds." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13209–13218.

[156] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time." In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998.

[157] Jiajun Wu, Erika Lu, Pushmeet Kohli, Bill Freeman, and Josh Tenenbaum. "Learning to see physics via visual de-animation." In: *Advances in Neural Information Processing Systems* 30 (2017).

[158] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments." In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 713–720.

[159] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. "Sapien: A simulated part-based interactive environment." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11097–11107. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Xiang_SAPIEN_A_SimulAted_Part-Based_Interactive_ENvironment_CVPR_2020_paper.pdf.

[160] Zhenjia Xu, Zhanpeng He, and Shuran Song. "Universal manipulation policy network for articulated objects." In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2447–2454.

[161] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions." In: *arXiv preprint arXiv:1906.03853* (2019).

[162] Karmesh Yadav et al. "Habitat-Matterport 3D Semantics Dataset." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 4927–4936. DOI: 10.1109/CVPR52729.2023.00477. URL: https://doi.org/10.1109/CVPR52729.2023.00477.

[163] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. "RPM-Net: recurrent prediction of motion and parts from point cloud." In: *arXiv preprint arXiv:2006.14865* (2020).

[164] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. "Lasr: Learning articulated shape reconstruction from a monocular video." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15980–15989.

[165] Jingkang Yang et al. "Panoptic Video Scene Graph Generation." In: *CVPR*. 2023. URL: https://openaccess.thecvf.com/content/CVPR2023/papers/Yang_Panoptic_Video_Scene_Graph_Generation_CVPR_2023_paper.pdf.

[166] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, et al. "Holodeck: Language Guided Generation of 3D Embodied AI Environments." In: *arXiv preprint arXiv:2312.09067* (2023).

[167] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. "Visual imitation made easy." In: *Conference on Robot Learning (CoRL)*. 2020.

[168] Wentao Yuan, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. "M2T2: Multi-Task Masked Transformer for Object-centric Pick and Place." In: *7th Annual Conference on Robot Learning*. 2023. URL: https://arxiv.org/pdf/2311.00926.pdf.

[169] Afia Zafar, Muhammad Aamir, Nazri Mohd Nawi, Ali Arshad, Saman Riaz, Abdulrahman Alruban, Ashit Kumar Dutta, and Sultan Almotairi. "A Comparison of Pooling Methods for Convolutional Neural Networks." In: *Applied Sciences* 12.17 (2022), p. 8643.

[170] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. "Transporter networks: Rearranging the visual world for robotic manipulation." In: *arXiv preprint arXiv:2010.14406* (2020).

[171] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. "NERF++: Analyzing and Improving Neural Radiance Fields." In: *https://arxiv.org/abs/2010.07492* (2020).

[172] Yue Zhao, Ishan Misra, Philipp Krähenbühl, and Rohit Girdhar. "Learning Video Representations from Large Language Models." In: *arXiv preprint arXiv:2212.04501*. 2022.

[173] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control." In: *Conference on Robot Learning*. PMLR. 2023, pp. 2165–2183.

APPENDIX

A

GENAUG

A.1 REAL-WORLD EXPERIMENTS

A.1.1 *Real-World Tasks*

We collect 10 pick-and-place tasks in the real world in one single environment, as shown in Figure 38. All tasks are collected with only 10 demonstrations in one single environment. For each demonstration, we randomly place objects within the work zone of the table. To augment the demonstration, we apply GenAug 100 times per demonstration, resulting in 1000 augmented demonstrations for each task.

A.1.2 *depth-guided diffusion model vs inpainting*

We further justify the choice of using a depth-guided diffusion model, as shown in Figure 37. Directly using the inpainting model often does not result in reasonable visual augmentation. Instead, GenAug uses a depth-guided diffusion model together with predefined 3D meshes, resulting in realistic new objects and scenes.

A.1.3 *Real-World Unseen Environments*

In this section, we visualize examples of unseen test scenes that are used for evaluation for all 10 tasks, as shown in Figure 46. For each task, we evaluate GenAug performance on 10 unseen environments, 10 unseen objects to place and 10 unseen objects to pick.

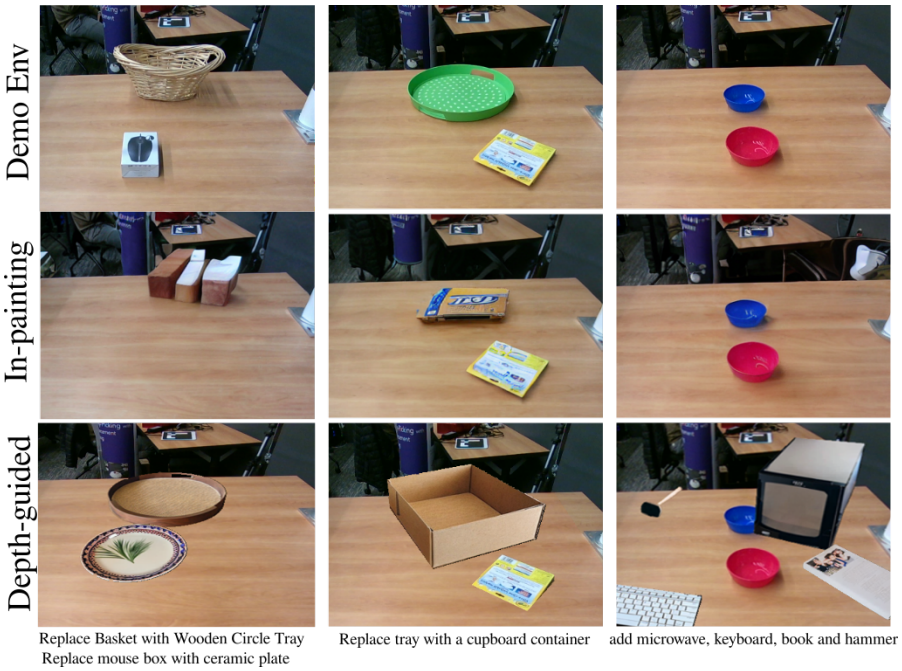


Figure 37: Comparison between depth-guided diffusion model with access to predefined 3D meshes and inpainting models.

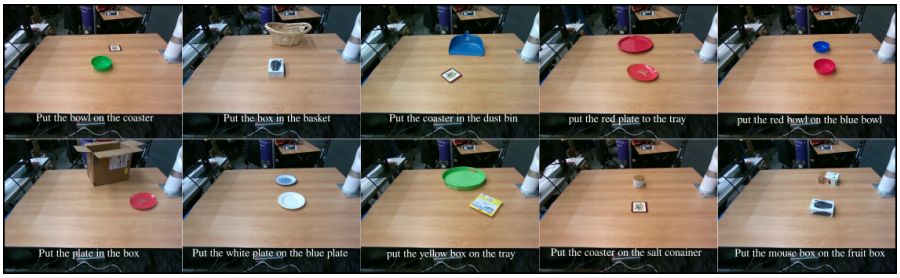


Figure 38: Tasks used in the real-world experiments.

A.1.4 *Real-World Evaluation*

We visualize the demonstration environments collected in the real world as well as their corresponding unseen test environments and objects. Please see our website for better visualization https://genaug.github.io

We visualize pick and place affordances predicted by CLIPort trained with GenAug in Figure 45

A.2 SIMULATION EXPERIMENTS

A.2.1 *Table-top Pick and Place Tasks*

We perform a large-scale evaluation in simulation. In particular, we collect 1, 10, 100 demonstrations for 5 tasks: "Pack the brown round plate", "Pack the straw hat", "Pack the green and white striped towel", "Pack the grey soccer shoe" and "Pack the porcelain cup", and report the average success rate across all tasks. Following evaluation metrics defined in CLIPort [135] and TransporterNet [170], the success rate is defined as the total volume of the pick object inside the place object, divided by the total volume of the pick object in the scene.

A.2.2 *Behavior Cloning*

In addition, we perform another experiment to show GenAug can apply to a different task: "close the top drawer" with a fetch robot. In particular, we collected 100 demonstrations and trained a CNN-MLP behavior cloning policy finetuned with R3M [104] embeddings. The input for the network is the RGB observation and the output is a 8-dim action vector. We tested on 100 unseen backgrounds using iGibson [132] rooms and observed GenAug is able to achieve 60% success rate while policy without Genaug is only 1%, leading to almost 60% improvement. Please see the visualizations on the website https://genaug.github.io

A.2.3 *Augmented Dataset in Simulation*

Given demonstrations from a task collected in simulation, we apply GenAug 100 times for each demonstration. We visualize examples of the augmented dataset in Figure 39.

We also observe diverse visual augmentation on the same object template, as shown in Figure 40. Given different text prompts, GenAug is able to generate different and realistic textures.

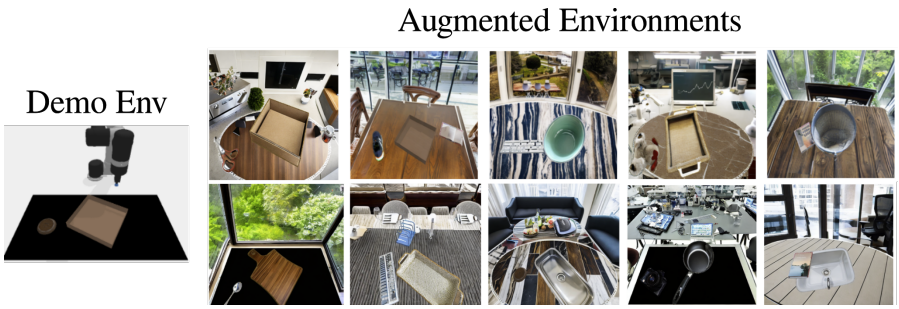


Figure 39: Augmented dataset for demonstrations collected in simulation.

Figure 40: Diversity of the appearance of the generated objects

A.3 VISUALIZATION OF BASELINE DATA AUGMENTATION

We visualize some examples of randomly copying and pasting segmented images from LVIS dataset [45], as shown in Figure 41.

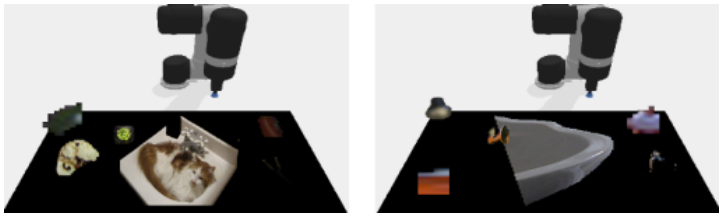


Figure 41: Examples of random copy and paste baseline. We extracted queried segmented images from LVIS dataset and paste them directly on the original demonstration image. This usually leads to low-quality and incomplete image generation.

We observe this baseline often results in unrealistic, low-quality image generation, which is not usually matching observations during test time in both real-world and simulation.

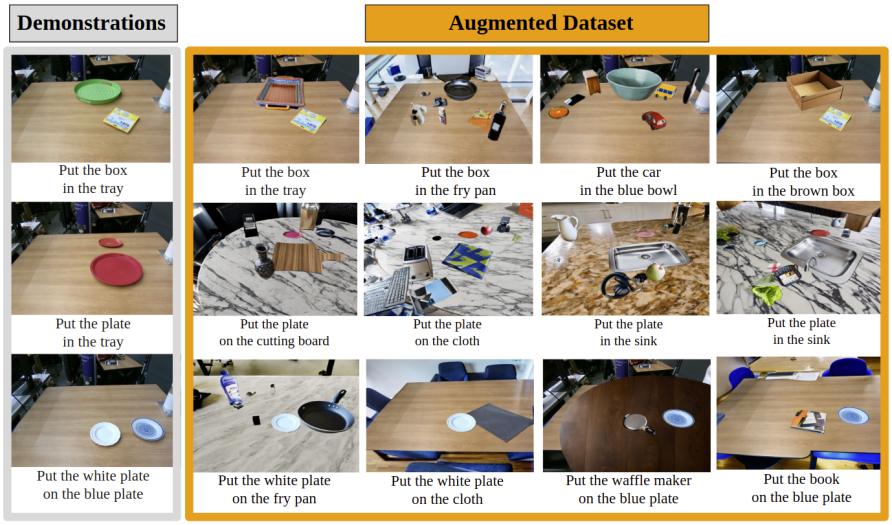


Figure 42: Examples of augmented dataset given observations of demonstrations collected in a simple environment.

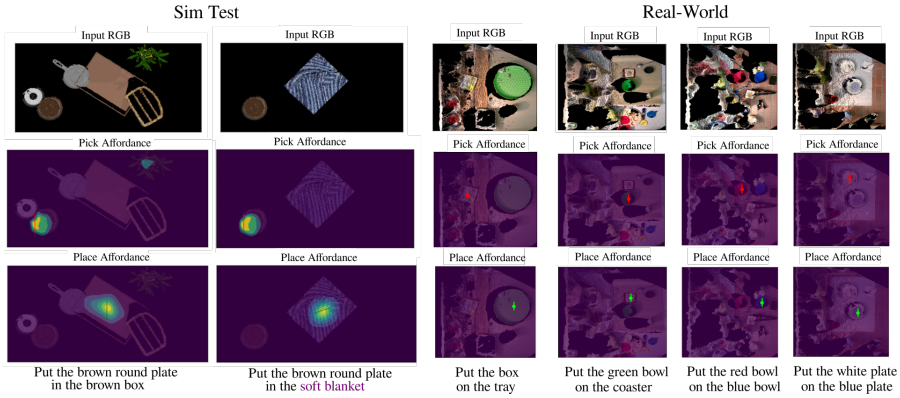


Figure 43: Pick and Place affordance predicted by CLIPort that trained on GenAug on unseen environments and objects in simulation and the real world.

A.4 VISUALIZATIONS FOR REAL-WORLD EXPERIMENTS

In this section, we visualize more examples of applying GenAug on demonstrations collected in the real world, as shown in Figure 42. We train CLIPort [135] with such a dataset and evaluate unseen environ-

ments and objects for 10 tasks. We further show affordance predictions in Figure 45 and Figure 43.

A.5 ASSUMPTIONS ON OBJECT MASKS

We found applying diffusion models on zoom-in image crops usually results in better results, especially for small objects. Note that with the advances in open-VLM, object masks can be automatically obtained from such as SAM [68], shown in Fig.44. In addition, GenAug uses object masks to update correct depth and provide more controllable augmentations such as only changing the object texture. However, GenAug is still able to do global augmentation without masks if depth update is not required.

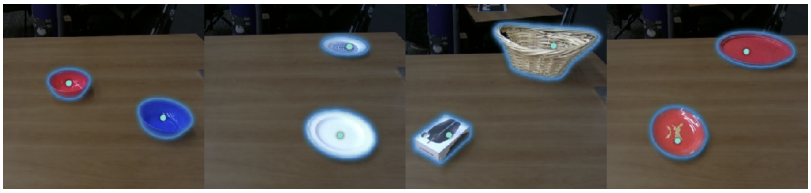


Figure 44: Prediction masks by the Segment Anything Model[68]

A.6 FUTURE WORK

In the future, we are interested in extending GenAug to other grippers and tasks that are beyond simple pick-and-place tasks. In particular, it would be interesting to investigate if we can train a diffusion model that can augment robot, action and scenes together. We are also interested in applying video diffusion models such as dreamix [97] to ensure visual smoothness for tasks that requires temporal consistency. In our current setup, we only have one front camera mounted on a tripod. For future work, we hope to add a wrist camera on the robot. This will give us flexibility to control the camera and find the object. In addition, combining GenAug with the power of common sense reasoning from LLM such as chatGPT to augment actions with reasoning on object physics would be interesting.

Figure 45: Prediction of pick and place locations on various tasks with GenAug

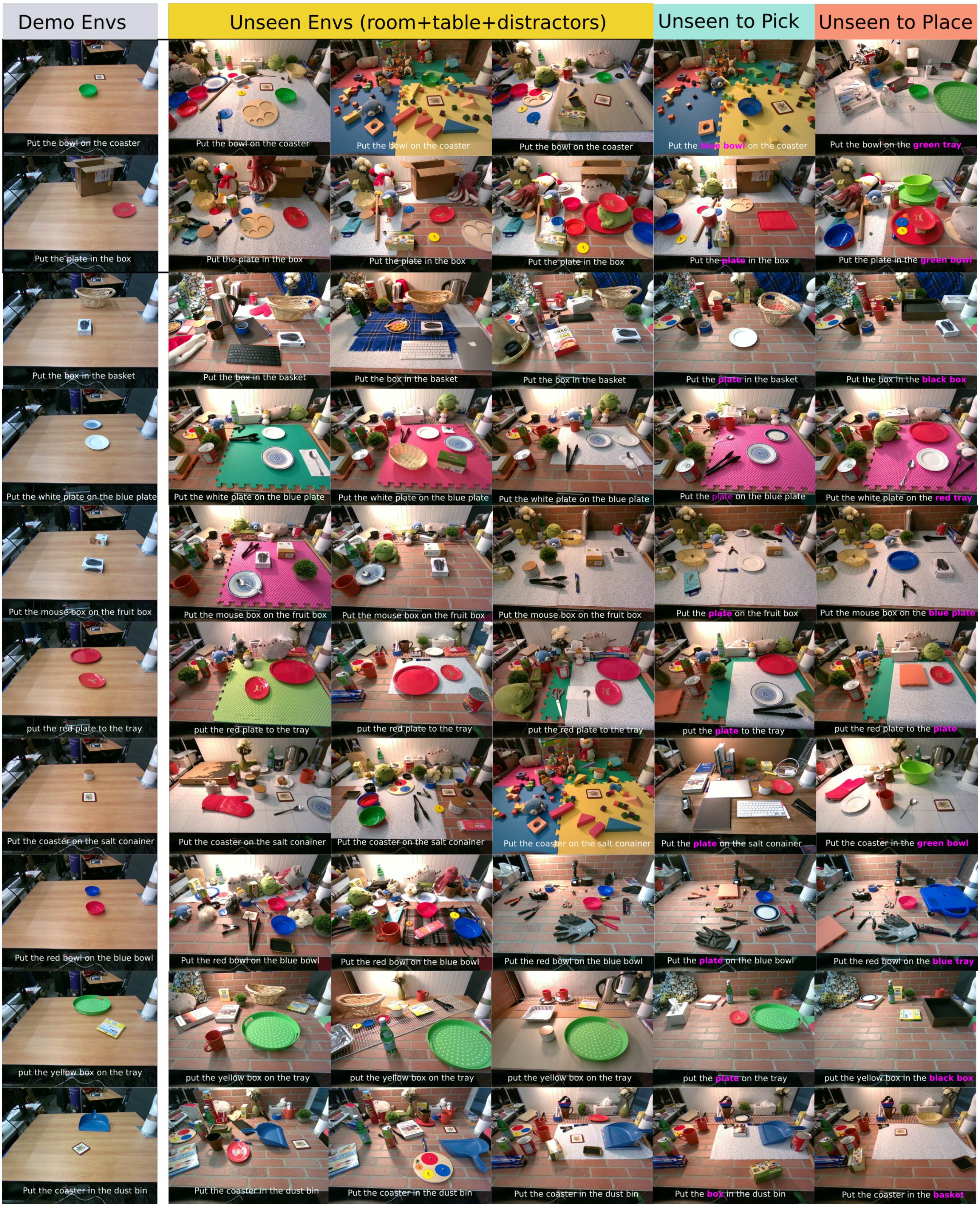


Figure 46: Unseen test set in the real-world experiments.

B

ISAGRASP

B.1 METHOD

B.1.1 *Domain Randomization*

We use domain randomization during refinement stage with rejection sampling. We randomize object mass from $m \in [0.05\ kg, 0.25\ kg]$ and object friction from $\mu \in [0.7, 1]$. After the object is lifted, we add 1 second high-frequency perturbation sampled from a Gaussian distribution $\sim \mathcal{N}(0,\ 0.01)m$ on the palm translation. We repeat this process 10 times and keep grasps that are robust and successful for all 10 random physics parameters.

B.1.2 *Network*

We describe our training details in this section. We sample $N = 1024$ points from the target object, and predict pregrasp translation, rotation and final finger pose. We use batch size $bs = 256$, learning rate $lr = 0.0002$ and use a Pytorch Adam optimizer to train our network with two GPUs. During training, we use Nvidia Apex [23] to achieve mixed-precision training through all our experiments.

We shift the input pointcloud to its center and predict translation relative to this center. To achieve rotation invariance, we online augment rotation of the robot rotation, represent centered input observation relative to the current robot rotation, and predict pregrasp rotation relative to the current robot rotation. The training loss is defined

as L1 loss on translation, geodesic loss on rotation and L1 loss on finger joints.

$$\|\pi^{\theta}_{\text{translation}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t) - (a^i)_{\text{translation}}\| \tag{6}$$

$$+G(\pi^{\theta}_{\text{rotations}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t), (a^i)_{\text{rotations}}) \tag{7}$$

$$+\|\pi^{\theta}_{\text{finger}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t) - (a^i)_{\text{finger}}\| \tag{8}$$

B.2 APPENDIX B: EXPERIMENT

B.2.1 *Dataset and Evaluation*

In simulation, we evaluate our policies on three dataset: RescaledYCB, ShapeNet and GoogleScans. We visualize examples of the three datasets in Figure 47. We use 65 RescaledYCB objects, 200 ShapeNet objects, and 200 GoogleScan objects. For each object, we evaluate 5 times with object mass/friction as $(m = 0.05, \mu = 0.8)$, $(m = 0.1, \mu = 0.85)$, $(m = 0.15, \mu = 0.9)$, $(m = 0.2, \mu = 0.95)$, $(m = 0.25, \mu = 1)$.

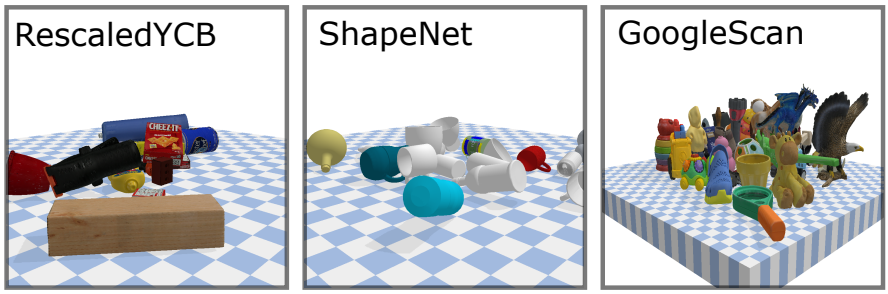


Figure 47: A subset of Examples used for evaluation.

B.2.2 *Baselines*

Random: We randomly generate robot pose (22DOF) around the object. In particular, we uniformly sample translation from $t \in [0.1m, 0.1m]$, rotation $r \in [-0.5rad, 0.5rad]$ and fingers $f \in [0.5rad, 1rad]$. This method performs poorly and only only achieves 3%, 5% and 2% success rate on RescaledYCB, ShapeNet and GoogleScans.

Train on successful random: Using the random approach above, we first randomly generate 10 grasp candidates, then apply rejection sampling with domain randomization to further remove unstable grasps. At this stage, 26% random grasps can be successfully refined and form a dataset. We train a policy using this dataset and achieve 15%, 42% and 18% on RescaledYCB, ShapeNet and GoogleScans.

Heuristic: We use a heuristic that we have seen being used for grasping in practical systems: grasp the object from the top, with a fixed 5cm offset from the object top surface. We observe 27%, 40%, and 16% success rate achieved on RescaledYCB, ShapeNet and GoogleScans.

Train on successful heuristic: Followed by Heuristic method we described above, we perturb the initial heuristic grasp and generate 10 grasp candidate for each object, then apply rejection sampling with domain randomization. We observe 30% grasps can be successfully refined. We train a policy on these refined grasps and achieve 9%, 15%, 2% success rate on RescaledYCB, ShapeNet and GoogleScans.

GraspIt! We use GraspIt![94] as one of baselines. For each object, we run 70000 steps to optimize the grasps based on contact energy. Each optimization takes about 50-70seconds. This can achieve 14%, 48%, 24% success rate on RescaledYCB, ShapeNet and GoogleScans.

During our experiments, we found GraspIt! often generates grasps that are impossible to reach from a open-fingered pregrasp pose due to the collision of the table surface. We further reduce this "penalty" by disabling table-robot collision when closing fingers. Despite this additional step, GraspIt! still has low success rate on all three dataset (See Section 4). We compare qualitative examples between GraspIt! and our method in Figure 48. Our policy generates more natural and stable grasps than GraspIt!.

Train on successful GraspIt Using GraspIt method described above, we generate 10 grasp candidates for each object and apply the same rejection sampling with domain randomization. We train a policy on these grasps and achieve 29%, 42%, 20%.

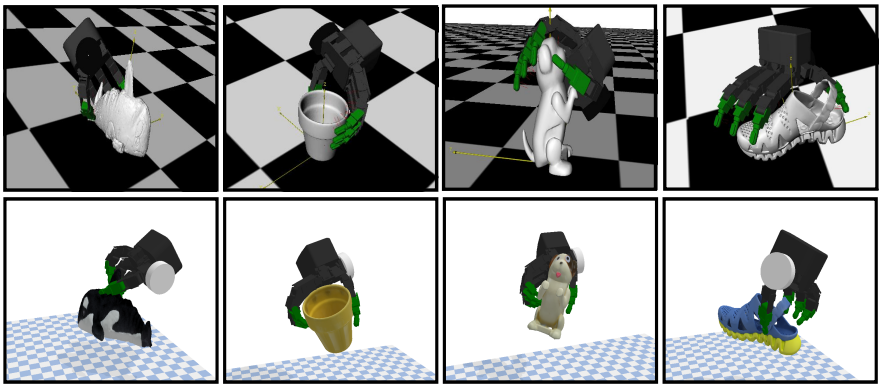


Figure 48: Qualitative comparison between GraspIt! (Top) and our method (bottom). Our policy utilizes a few human demonstrations and trained to generates more natural and stable grasps

PPO w/o Demonstration: We train two RL baselines: (1) PPO without demonstrations (2) PPO with demonstrations. We detail the implementations of these two methods as below.

Starting from an initial robot pose, we first move the robot along a linear path towards the object center and stop when its 5cm away from the object. Starting from this pose, we train RL policies using PPO and compare when demonstration is or not used as a reward function. After 5 timesteps, we define a lifting action from the last pose. In particular, the demonstration reward at each time step is defined as:

$$r^t = \exp(-20\,\|\mathbf{d}_t - \mathbf{g}_t\|^2) \tag{9}$$

$$r^q = \exp(-10\,\|\mathbf{d}_q - \mathbf{g}_q\|^2) \tag{10}$$

$$r^f = \exp(-10\,\|\mathbf{d}_f - \mathbf{g}_f\|^2) \tag{11}$$

$$r = r^t + r^q + r^f \tag{12}$$

Here $\mathbf{d_t}$ represents final pose translation from the demonstration, $\mathbf{d_q}$ represents final pose quaternion from the demonstration, and $\mathbf{d_f}$ represents the final finger pose from the demonstration. $\mathbf{g}$ represent predicted poses. In addition, we add final success as the sparse reward. We train each method with three seeds and visualize the performance of the two policies during training on 20 seen YCB objects in Figure 49. RL policies suffer at generalizing to a wide range of objects even with training objects, with below 5% success rate.

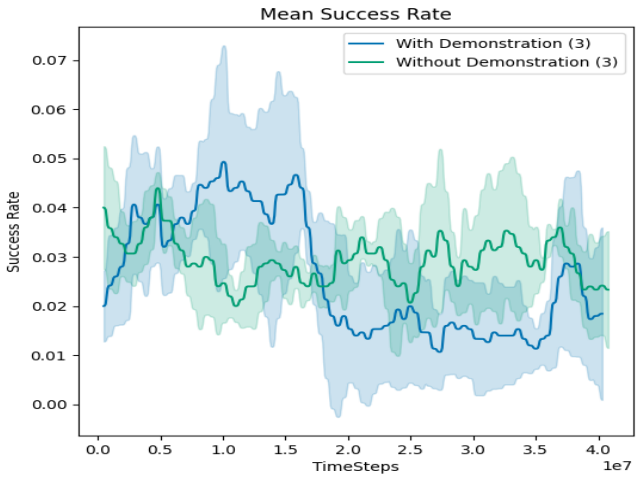


Figure 49: RL baselines evaluated on training set. Blue: training with demonstration as reward, Green: training without demonstration as reward

PPO with dense distance reward We refined the above PPO by adding a dense distance reward together with a contact reward. More formally, at each timestep, we define reward function as:

$$r = \exp(-1 \sum \|\mathbf{f}_i - \mathbf{o}\|) + \mathbf{N}_c/\mathbf{N}_r \tag{13}$$

Here f_i represents fingertip i on allegro hand, o represents the object center, N_c represents number of fingers contacting the object, and N_r represents number of robot fingers, which is 4 for allegro hand.

DAPG + DR To combine RL with imitation learning, we implement the DAPG algorithm described in [122]. We use the original DAPG code and replace the original MLP policy network with our PointNet++ style architecture in order to take pointclouds as input. Directly learning pointnet++ from scratch using DAPG results in significant long time when computing natural policy gradient (about 5hr per iteration). Therefore, we load a pretrained pointnet++ weights, and freeze this part during training. During training, we randomize the object weight and friction in order to predict more stable grasps. We train DAPG for about 3days, and found this approach achieves 46%, 51%, 53% on RescaledYCB, ShapeNet and GoogleScans.

DexYCB -DR -SA In order to compare the importance of domain randomization, we generate a new dataset without domain randomization. We found that policies trained on this dataset can only achieve 34%, 35% and 22% success rate on RescaledYCB, ShapeNet and GoogleScans.

DexYCB +DR -SA We generate stable grasps on DexYCB using rejection sampling with domain randomization. We found policies trained on this dataset achieve 74%, 56% and 51% success rate on RescaledYCB, ShapeNet and GoogleScans.

DexYCB +DR +SA We run our approach that is trained on a dataset combines implicit shape augmentation and domain randomization and observe 74%, 74% and 70% success rate on RescaledYCB, ShapeNet and GoogleScans.

B.2.3 *Ablation Analysis.*

We provide additional analysis to investigate two questions: (1) how many demonstrations is necessary (2) How much augmentation is needed.

Impact of Number of Human Demonstrations. We compare policies trained when 1, 5, and 10 demonstrations per object are provided. Figure 50 (b) compares the average success rate evaluated on 200 ShapeNet objects and 200 GoogleScans objects. In particular, we found the number of demonstration does not necessarily improve the generalization for training explicitly with initial 20 YCB objects, the more demonstrations helps the overall generalization on training with augmented objects.

Impact of amount of augmented training data. Figure 50 (a) compares the impact of number of augmentations with 10 demonstrations per object. We create different datasets with different numbers of augmentation to train policies and compare on ShapeNet objects and GoogleScans objects. Interestingly, we found as number of augmentations goes up, the performance does not consistently go up. Overall, generating 200 augmented objects leads to the highest success rate.

B.3 APPENDIX C. QUALITATIVE RESULTS

B.3.1 *Grasping in Simulation*

To evaluate performance in simulation, we directly reset the robot from the starting pose to the predicted pregrasp pose. and do linear interpolation between an open palm to the final finger pose over 10 timesteps. We define a lifting trajectory that to move the robot translation up by 20cm. We evaluate the policy on 465 unseen objects and show qualitative examples in Section 4, and Figure 51. Please check out more visualization in our supplementary video.

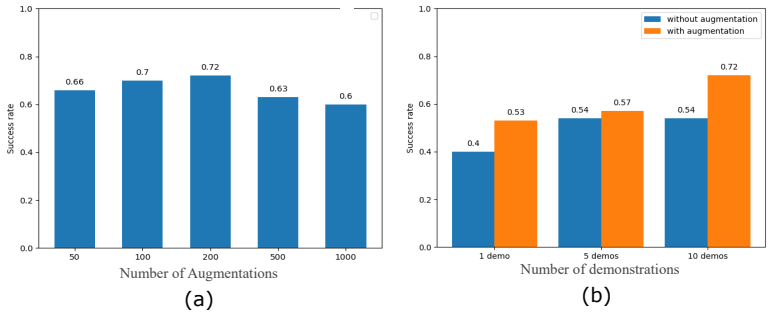


Figure 50: Ablation analysis. (a) Analysis on number of augmentations. In our experiments, we use 200 augmentations for each object. (b) analysis on number of demonstrations. More demonstrations leads to high success rate, but more object augmentation leads to better generalization.

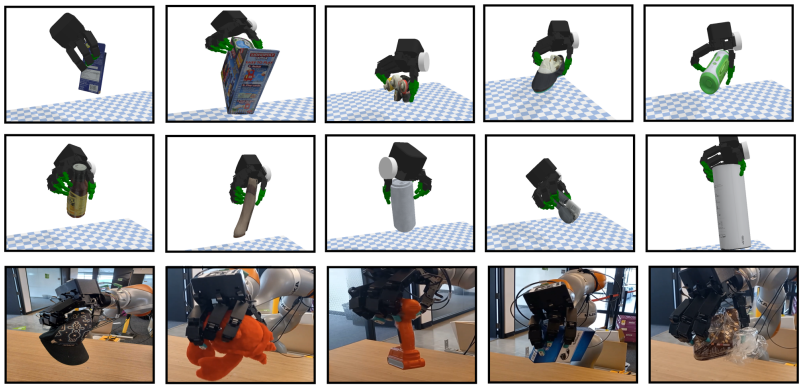


Figure 51: More qualitative results in simulation and real world. Top two rows shows policy performs on googleScans objects, bottom row shows policy on unseen daily objects in the real world.

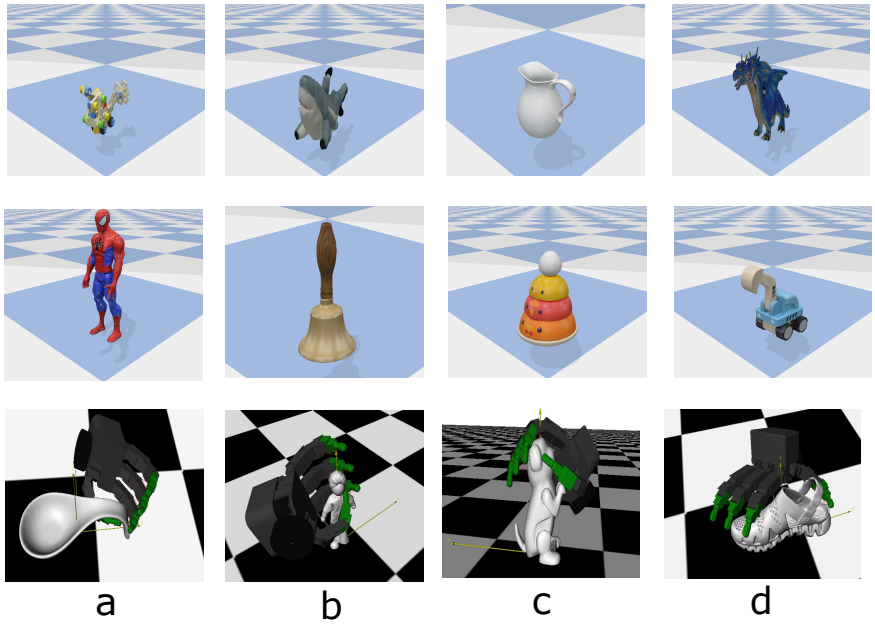


Figure 52: Baseline Failure mode. Top: Heuristic approach suffers from challenging objects which require more careful reorientation. Middle: Heuristic approach is less likely to succeed if grasping from top is less stable. Bottom: Failure mode in GraspIt. (a) Collision to table when close fingers from an open palm. (b)(c) unstable grasp when lifting the object up (d) challenging object to optimize.

B.3.2 *Grasping in Real World*

We evaluate our grasping policy in real world with 22 unseen objects, with 5 random poses per object. Figure55 shows all the objects used in the test. We place two cameras to capture enough pointclouds of the object. Starting from the default pose, the robot first retracts to a pose such that two cameras can capture the scene without occlusions. To get object pointclouds, we use RGB images to subtract the table background, which is used to extract pointclouds of the object. We feed the object pointcloud to the network and predict a pregrasp pose and a final pose. We extend the location of the pregrasp from the object center by 10cm and interpolate from the robot starting pose to the new pose. Once the new pose is reached, we follow the linear path that leads to the pregrasp pose. When the pregrasp is reached, we perform linear interpolation to control finger from an open-palm to the final predicted pose.

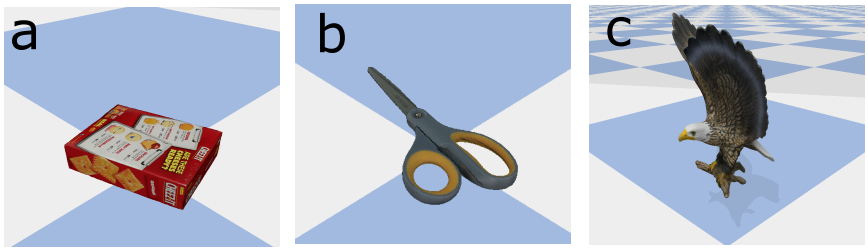


Figure 53: Failure cases of our approach. (a) large cracker box lying on the table which is too large to grasp. (b) our approach might collide with the table when trying picking up a scissor. (c) Eagle with two wings requires particular ways to grasp, e.g. grasping left wing.

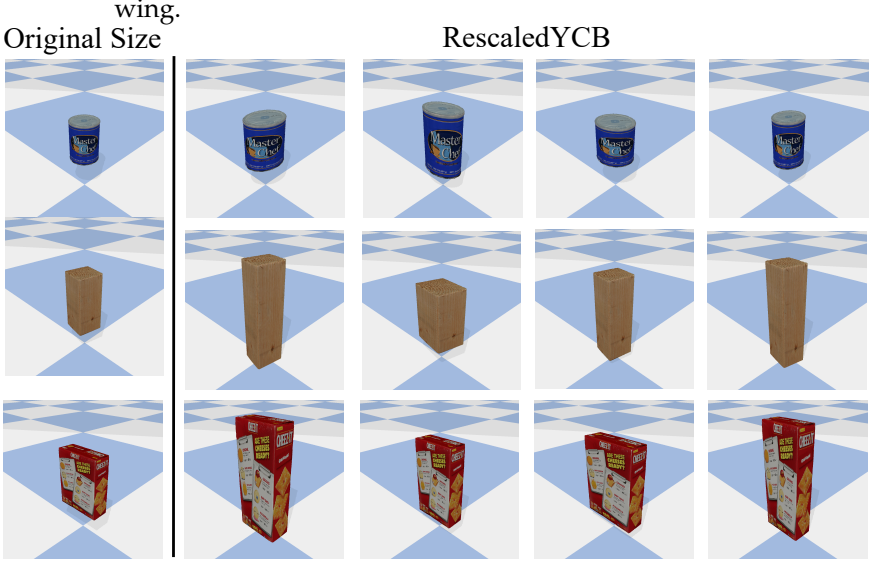


Figure 54: Comparison of training size and RescaledYCB dataset. The rescaled YCB dataset is created by scaling different dimensions of objects in the DexYCB dataset. This results in objects of widely varying sizes, rather than very different shapes.

B.3.3 *Implicit Shape Augmentation*

We use a correspondence-aware generative model DIF-Net [33] to deform objects for our dataset. We show more examples of the diversity of our augmented dataset in Figure 56. DIF-Net is able to generate realistic deformation while maintaining the semantic structures of the original object meshes.

B.4 APPENDIX E FUTURE WORK

There are several interesting future directions to explore. For example, it would be very interesting to see if we can adapt implicit shape

Figure 55: We evaluate our method using 22 unseen daily objects in real world experiments. Here we visualize all the Objects used in our robot test

augmentation and create diverse multi-object environments. It would be worth exploring augmentation in the task space, where demonstrations can be divided and recombined for a different task. We are also interested in shape augmentation with an adversarial setting, where a separate network is learning to deform shapes that policy are more likely to fail.

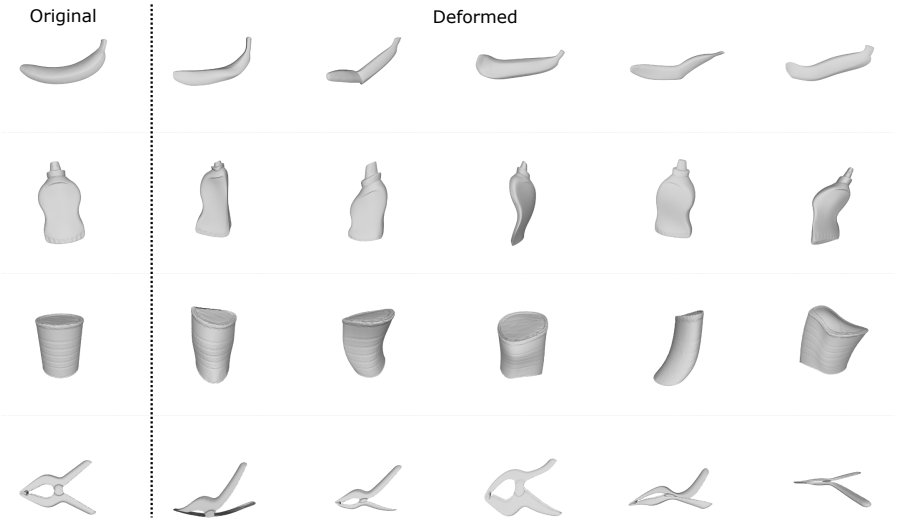


Figure 56: Examples of deformed objects in our augmented dataset

B.5 CONCLUSION

We present ISAGrasp, a novel system for learning dexterous grasping policies in the real world. ISAGrasp leverages a data augmentation approach that bootstraps a small number of human demonstrations with a large dataset with diverse and novel objects and grasps. By using a correspondence-aware generative model, we can deform original object shapes and generate dynamically consistent new grasps. We create a large and diverse grasping dataset and train a policy via supervised learning that can then be deployed in simulation and the real world on grasping novel objects, achieving over 75% success rate on grasping novel object instances in the real world.

C

URDFORMER

C.1 PAIRED DATASET GENERATION FOR TRAINING URDFORMER

C.1.1 *Consistency-aware Image Generation*

We observe that depth-guided or in-painting stable diffusion models [126] often ignore local consistency, making it difficult to render high-quality images that are paired with the simulation content. To overcome this issue, we propose texture-guided image generation. Instead of asking stable diffusion to texture the entire object which might change the low-level details, we utilize the advantage of the diffusion models that help to diversify an initial small set of texture images. In particular, given an initial 100 cabinet texture images we downloaded from the internet, we apply stable diffusion to change the style, pattern, or color, which significantly increases the texture dataset. Then we randomly choose a texture and simply apply it to regions of interest (i. e.drawer, door) obtained by the simulator using perspective warping. One important note is that these texture images are not UV texture maps, instead they are simply 2D images, and the texturing step is only at 2D image space because we only care about the photo realism for that particular image. As shown in Fig 11, this simple approach is surprisingly effective and creates realistic images while maintaining low-level part consistency.

C.1.2 *Part-Consistency*

We compare our part-wise generation method with other approaches qualitatively in Fig 58. off-the-shelf stable diffusion models often ignore low-level details, making it challenging to create high-quality accurately paired dataset. Instead, our approach helps to preserve se-

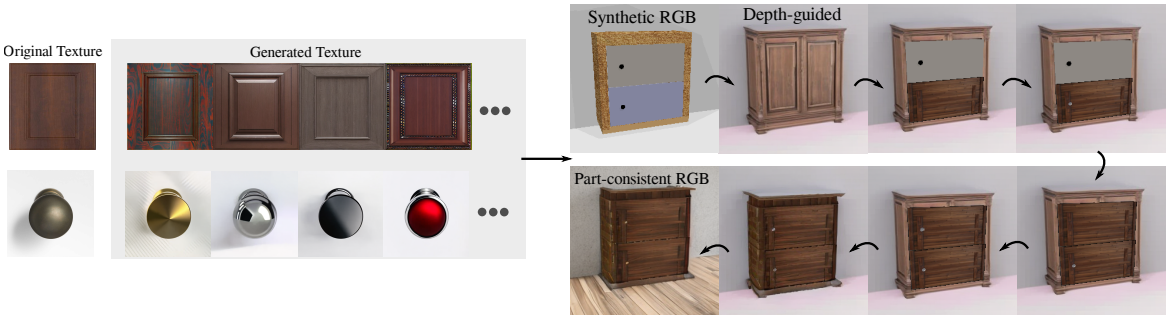


Figure 57: Paired dataset generation using texture and prompt templates to guide Stable Diffusion [126] and create a diverse texture dataset, which can be then warped on the targeted individual part of the object.

Figure 58: Qualitative comparison among different rendering methods: depth-guided diffusion models, inpainting stable diffusion and part-wise generation

mantic low-level details while creating photorealistic paired images that match the original textureless simulation data.

C.1.3 *Visualization on Ablations*

We visualize the different training data shown in Table II: URDFormer with random colors, selected textures, random textures, and generated textures. All training inputs are captured from the same camera angles. As shown in Fig 59, the generated texture shows high pixel realism that is closer to the distribution of the real world. As shown in Table II, training on such data improves performance in predicting mesh classes such as identifying ovens or cabinets.

Figure 59: Visual comparison for ablation study with different training input: Random Colors, selected textures, random textures and generated textures. Generated textures show photo-realism that closer to the real-world distribution.

C.2 URDFORMER

C.2.1 *Dataset*

In total, we generated approximately 118K image-URDF pairs across 5 common categories of articulated objects in the kitchen including cabinets, ovens, dishwashers, washer, and fridges, as well as 2 types of rigid objects oven fans and shelves. These articulated objects include part meshes in 8 types: drawer, left door, right door, oven door, down door (dishwasher), circle door, handle and knob.

C.2.2 *Training Details*

URDFormer is trained on one A40 GPU with batch-size of 256. For the ablation study shown in Table II, all the methods are trained with an equal number of epochs and evaluated using the last checkpoint. In particular, we train 130 epochs for global scenes and 80 epochs for objects.

Figure 60: Reality Gym Assets: Converting real-world images into fully interactive simulation assets. Given internet images, we apply finetuned groundingDINO to obtain 2D part boxes. These boxes and RGB images are fed into URDFormer to generate articulated simulation assets that match the real-world structure.

C.3 REALITY GYM

C.3.1 *Realistic Assets from Images*

Fig 60 shows examples of assets predicted from internet images. Given an RGB image, we first apply finetuned GroundingDINO (model soup applied) to obtain 2D bounding boxes for each part. These boxes and the original RGB image are fed into URDFormer to predict the corresponding URDF which matches the kinematic structure of the image. We introduce 4 main tasks (1) Open any part (2) Close any part (3) Fetch the object (4) Collect the object, and automatically generate successful demonstrations using Curobo [139] and their corresponding language descriptions.

C.3.2 *Targeted Domain Randomization*

To covert the real-world distribution, we further apply targeted domain randomization. This includes (1) Mesh Randomization: we randomly replace the original meshes with meshes from the PartNet [96] dataset for parts such as doors, handles as well as cabinet frames, shown as Fig 61(2) Texture Randomization: we use stable diffusion to

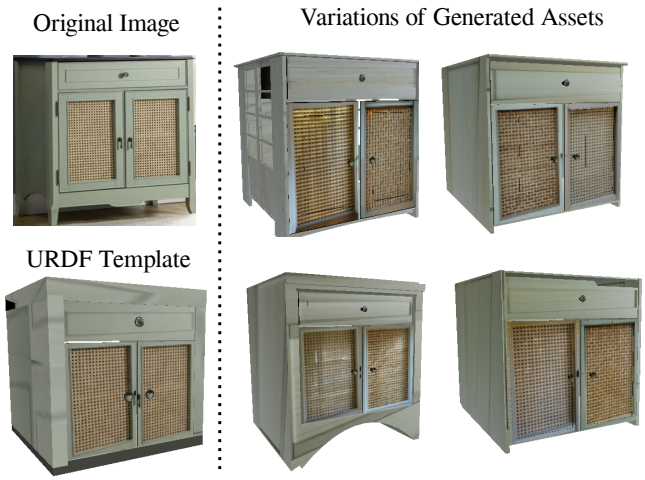


Figure 61: URDFormer does not reconstruct meshes. To cover the distribution of the real world, we randomly choose meshes from the PartNet dataset to replace the original part meshes for handles, drawers and so on.

Figure 62: We randomize the texture of the object using stable diffusion to cover the distribution of the real world

generate different style of cropped texture images to form UV texture maps, shown as Fig 62.

C.4 REAL-WORLD EXPERIMENTS

C.4.1 *Data Collection in Sim*

We introduce 4 different tasks and distribute 2 tasks for each object. (1) Open any articulated part (2) Close any articulated part (3) Fetch the object from a particular drawer (4) collect the object into a particular drawer. For each task, we collect 500 demonstrations in simulation with objects in different poses. During training, we additionally apply online pose augmentation, as well as color augmentation for RGB pointcloud to reduce the sim2real gap.

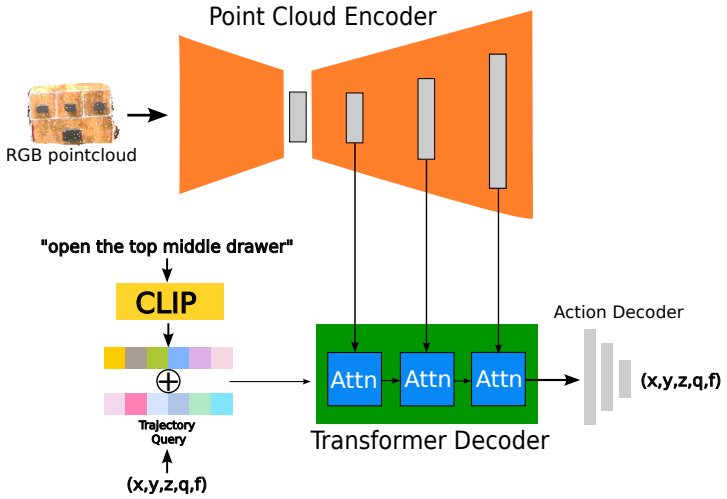


Figure 63: We train language-conditioned policy based on M2T2 network structure. The network takes RGB pointcloud and predicts endeffector poses to complete the task.

C.4.2 *Visualization of OWL-VIT*

Figure 64: Examples on using OWL-VIT [95] in our robot experiments.

We also visualize examples of OWL-VIT [95] on our test observation. We tried multiple prompts and shows the best predicted results in Fig. 64. Unfortunately, this approach fails to detect useful parts given the language instruction.

We also provide additional visualization on more test results. We observe OWL-VIT [95] fails on spatial reasoning and cannot provide accurate detection on specific regions or articulated parts such as "top drawer" or "opened door".

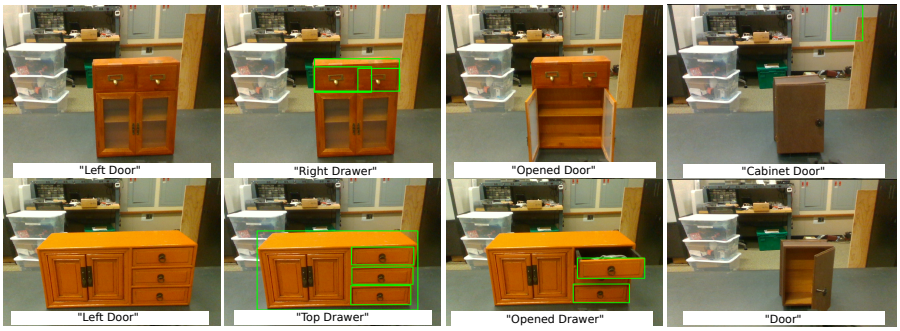


Figure 65: Examples of OWL-VIT on more tasks. We observe that this method fails to detect parts that require spatial reasoning such as "top drawer" or "opened door".

C.4.3 *Policy Training*

We train a language-conditioned policy for two tasks per object. We adopt network architecture from M2T2 [168], that takes RGB point-cloud and CLIP embeddings, to predict two end-effector poses, i.e where to place the gripper to grab the handle and where to release the handle to open the door. In particular, the transformer decoder concatenates the current endeffector pose and the text features, and combines the point features from the point cloud encoder, and predicts the endeffector pose in the next step.

C.5 LIMITATIONS AND FUTURE WORK

Part Detection URDFormer relies on the performance of bounding box detection. Although the finetuned Grounding DINO improves performance than the pretrained model, there is still a gap for improvement, especially on global scene detection. It would also be interesting for future work to remove the bounding box constraints and directly predict URDFs in language format from images.

Texture and Meshes URDFormer focuses on predicting kinematic URDF structures and uses predefined meshes that might not match the real-world scenes. To apply textures, we simply assume all parts are rectangular shapes, and use the bounding box of each object part to crop the image and import it into a UV map template. However, this does not work for irregular meshes such as a donut-shape door,

or when the object in the image is tilted, which can be an interesting future work.

Limited URDF Primitives URDFormer currently only supports articulated objects that have limited joint types such as prismatic and revolute, and cannot predict complex objects such as cars and lamps.

Link Collisions URDFormer only predicts URDF primitives for each bounding box, which sometimes leads to a collision between two links. Further post-processing is required to resolve this issue.

Global Scene We observe it's challenging for object detectors such as GroundingDINO to produce high-quality part detection results. For future work, we are interested in training a better URDFormer for global scenes and expanding kitchens into other scenes such as living rooms or bathrooms.

Object Placement URDFormer only predicts the kinematic structure of the scene for a few common articulated objects, and it cannot predict poses for objects such as water bottles on the counter. In our future work, we are interested in expanding URDFormer to cover more diverse assets and object placement.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of June 7, 2024 (`classicthesis` version 4.2).