Learning Robust Real-World Dexterous Grasping Policies via Implicit Shape Augmentation

Zoey Qiuyu Chen [1*] **Karl Van Wyk**[2] **Yu-Wei Chao**[2] **Wei Yang**[2] **Arsalan Mousavian**[2]

Abhishek Gupta[1] **Dieter Fox**[1,2]

[1]University of Washington [2]NVIDIA

{qiuyuc, abhgupta, fox}@cs.washington.edu

{kvanwyk, ychao, weiy, amousavian}@nvidia.com

https://sites.google.com/view/implicitaugmentation/home

Appendix A: ISAGrasp

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.

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 [1] 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.

$$\|\pi^{\theta}_{\text{translation}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t) - (a^i)_{\text{translation}}\| \tag{1}$$

$$+G(\pi^{\theta}_{\text{rotations}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t), (a^i)_{\text{rotations}}) \tag{2}$$

$$+\|\pi^{\theta}_{\text{finger}}(\cdot \mid p, \vec{N}_p, \vec{N}_o, \vec{N}_f, \vec{N}_t) - (a^i)_{\text{finger}}\| \tag{3}$$

Appendix B: Experiment

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. 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

*Work done while the author was a part-time intern at NVIDIA.

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

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 [2]. 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.

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 1. We use 65 RescaledYCB objects, 200 ShapeNet objects, and 200 GoogleScan objects. For each object, we evaluate 5 times with object mass/friction as $(m = 0.05, \mu = 0.8)$, $(m = 0.1, \mu = 0.85)$, $(m = 0.15, \mu = 0.9)$, $(m = 0.2, \mu = 0.95)$, $(m = 0.25, \mu = 1)$.

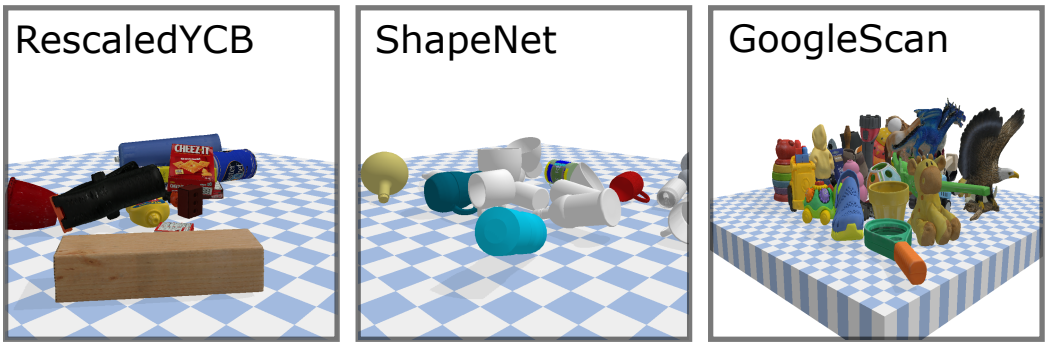

Figure 1: A subset of Examples used for evaluation.

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.

Baselines

We provide additional details for baselines described in Section 4 and table 1 as below.

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 For each object, we apply GraspIt[3] and 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. We compare qualitative examples between GraspIt and our method in Figure 2. Our policy generates more natural and stable grasps than GraspIt.

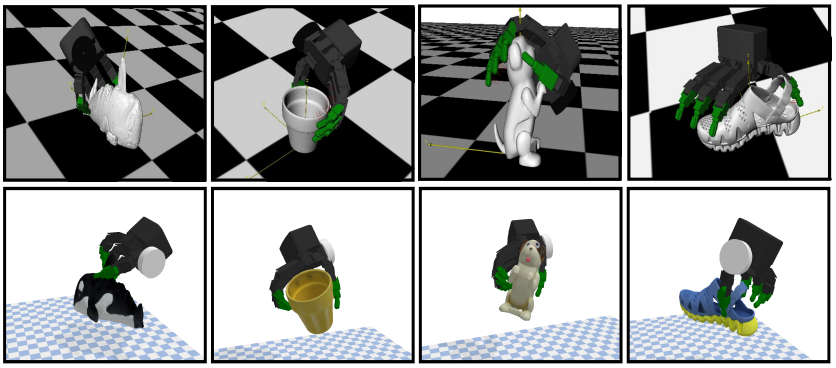

Figure 2: Qualitative comparison between GraspIt! (Top) and our method (bottom). Our policy utilizes a few human demonstrations and trained to generates more natural and stable grasps

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 found only 26% of the initial grasps can be refined, and we train a policy on these grasps and achieve 29%, 42%, 20%.

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:

$$r = \exp(-1 \sum \|\mathbf{f}_i - \mathbf{o}\|) + \mathbf{N}_c/\mathbf{N}_r \tag{4}$$

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 [4]. 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 -ISA 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 -ISA 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 +ISA 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.

Appendix C. Results and Analysis

Ablation Analysis.

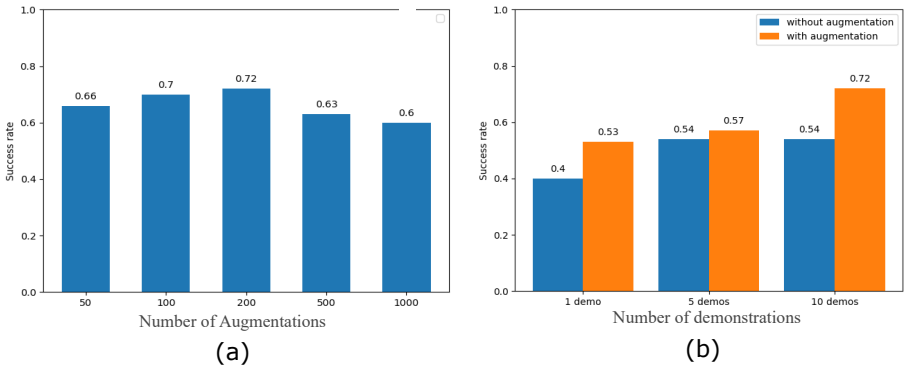

Figure 3: Ablation analysis. (a) Analysis on number of augmentations. In our experiments, we use 200 augmentations for each object. (b) analysis on number of demonstrations. More demonstrations leads to high success rate, but more object augmentation leads to better generalization.

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 3 (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 3 (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.

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 hardcoded 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” (Figure 4 top row) and requires more careful reorientation of the pregrasp pose.
2. The object is unstable to grasp from the top (Figure 4 middle row).

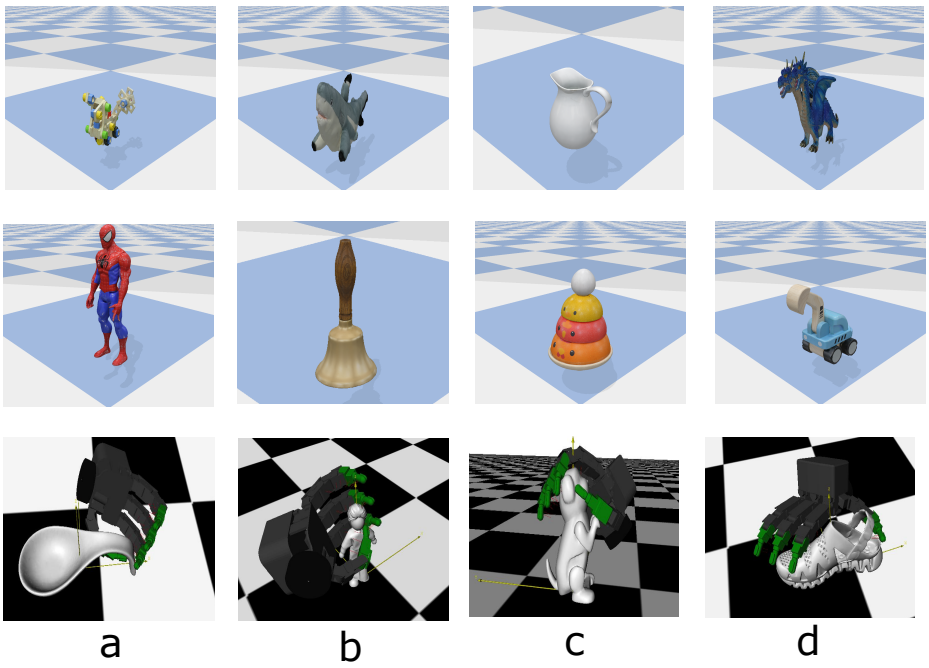

Figure 4: Baseline Failure mode. Top: Heuristic approach suffers from challenging objects which require more careful reorientation. Middle: Heuristic approach is less likely to succeed if grasping from top is less stable. Bottom: Failure mode in GraspIt. (a) Collision to table when close fingers from an open palm. (b)(c) unstable grasp when lifting the object up (d) challenging object to optimize.

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 (Figure 4 Bottom (a)).
2. Unstable grasps: GraspIt does not take dynamics into consideration, making the produced grasps potentially unstable while lifting up the object (Figure 4 Bottom (b) and (c)). In contrast, our rejection sampling with domain randomization encourages more stable grasps.
3. Challenging to optimize: When the object surface is more complex (Figure 4 Bottom (d)), 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.

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 success rates significantly lower than ISAGrasp because it does not train on augmented shapes and RL optimization can be unstable with multiple different shapes.

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.

ISAGrasp Performance and Failre Mode

We find that ISAGrasp significantly outperforms training without shape augmentation on unseen ShapeNet and GoogleScans objects, and achieves the same performance on RescaledYCB objects. As seen from Table 1 in the section, the generalization performance is good, even on completely unseen object instances, although there are failure modes, which we summarize as below.

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.

Understanding ISAGrasp performance on RescaledYCB: The rescaled YCB dataset is created by scaling different dimensions of objects in the DexYCB dataset (as shown in Figure 5). 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.

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 in Figure 4 and Figure5.

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 6. Please check out more visualization on our website.

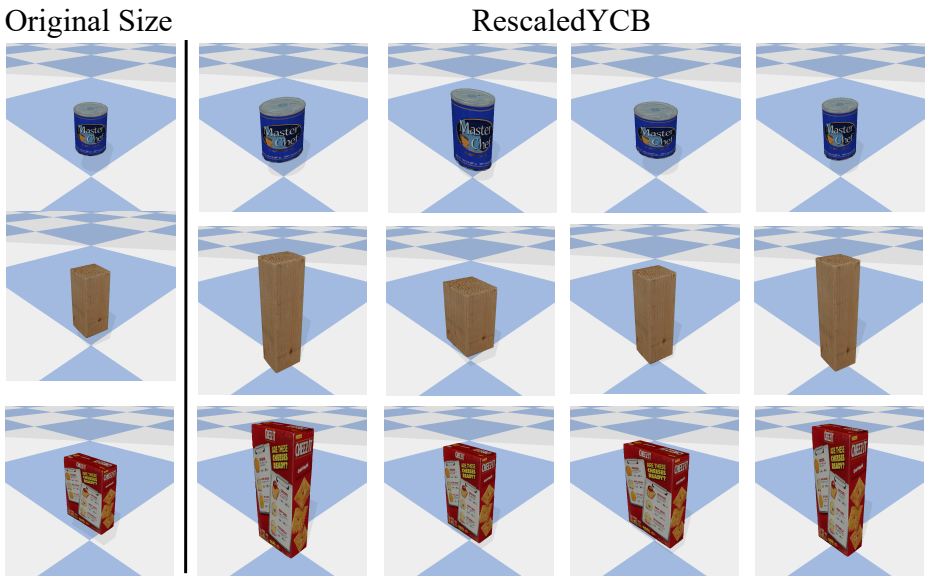

Figure 5: Comparison of training size and RescaledYCB dataset. The rescaled YCB dataset is created by scaling different dimensions of objects in the DexYCB dataset. This results in objects of widely varying sizes, rather than very different shapes.

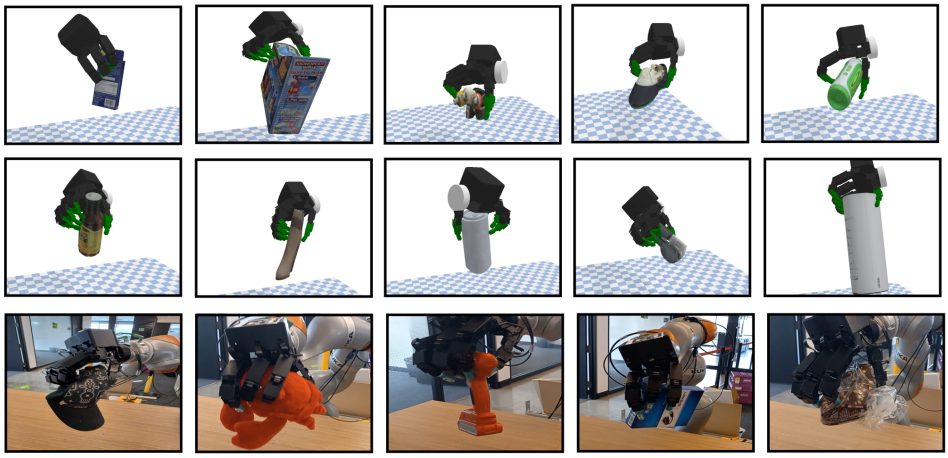

Figure 6: More qualitative results in simulation and real world. Top two rows shows policy performs on googleScans objects, bottom row shows policy on unseen daily objects in the real world.

Grasping in Real World

We evaluate our grasping policy in real world with 22 unseen objects, with 5 random poses per object. Figure7 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 fingers from an open-palm to the final predicted pose.

Figure 7: We evaluate our method using 22 unseen daily objects in real world experiments. Here we visualize all the Objects used in our robot test

Implicit Shape Augmentation

We use a correspondence-aware generative model DIF-Net [5] to deform objects for our dataset. We show more examples of the diversity of our augmented dataset in Figure 8. DIF-Net is able to generate realistic deformation while maintaining the semantic structures of the original object meshes.

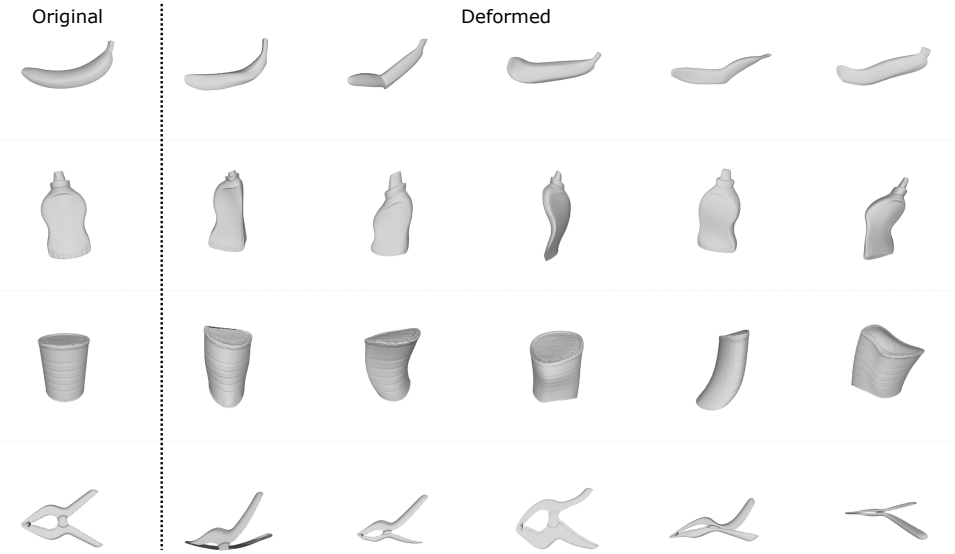

Figure 8: Examples of deformed objects in our augmented dataset

Appendix D Limitations

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 in Figure 9 a) or too flat (scissors in Figure 9 b)). 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 (e.g.

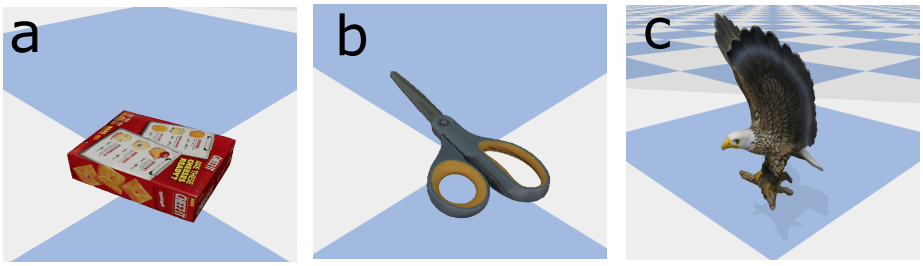

Figure 9: Failure cases of our approach. (a) large cracker box lying on the table which is too large to grasp. (b) our approach might collide with the table when trying picking up a scissor. (c) Eagle with two wings requires particular ways to grasp, e.g. grasping left wing.

Eagle with two wings: Figure 9 c). 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.

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 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.

Appendix F Further Literature

While our work uses human examples to seed the initial pre-grap positions, the key contributions of our work are complementary to works like DexMV [6] and DexVIP [7], and methods like DAPG [4] and DDPGfD [8](Learning from demonstrations with RL). DexMV and DexVIP aim to extract hand pose from video and use this as prior for training in simulation, but do not train on augmented objects for generalization. Similarly, DAPG and DDPGfD initialize from demonstrations and improve with RL on a fixed set of known objects. In comparison, the focus of our work is on learning grasping policies that generalize by explicitly generating novel, augmented shapes and then training robust policies on these shapes, which none of the other methods do. This allows policies to generalize to novel, unseen shapes. Our shape augmentation method ISAGrasp could be easily combined with pipelines for improvement with RL with demonstrations or for imposing priors from human video.

References

[1] N. Corporation. A PyTorch extension: Tools for easy mixed precision and distributed training in PyTorch. https://github.com/NVIDIA/apex, 2020.

[2] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, and N. Ratliff. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *RA-L*, 2022.

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

[4] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

[5] Y. Deng, J. Yang, and X. Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021.

[6] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. DexMV: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[7] P. Mandikal and K. Grauman. DexVIP: Learning dexterous grasping with human hand pose priors from video. In *CoRL*, 2021.

[8] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.