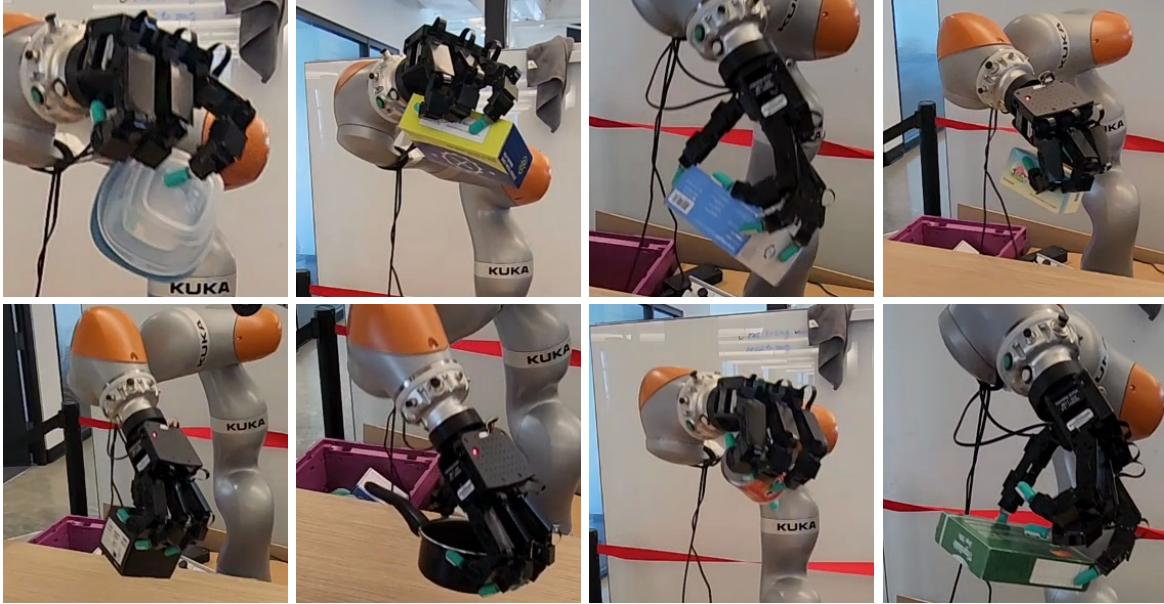


# DextrAH-RGB: Visuomotor Policies to Grasp Anything with Dexterous Hands

Ritvik Singh\*, Arthur Allshire<sup>†</sup>, Ankur Handa\*, Nathan Ratliff\*, Karl Van Wyk\*

\*NVIDIA Corporation

<sup>†</sup>University of California, Berkeley



**Fig. 1:** DextrAH-RGB (Dexterous Arm-Hand RGB) is an end-to-end RGB-based policy that can perform dexterous grasping of a wide variety of objects.

**Abstract**—One of the most important yet challenging skills for a robot is the task of dexterous grasping of a diverse range of objects. Much of the prior work is limited by the speed, dexterity, or reliance on depth maps. In this paper, we introduce DextrAH-RGB, a system that can perform dexterous arm-hand grasping end2end from stereo RGB input. We train a teacher fabric-guided policy (FGP) in simulation through reinforcement learning that acts on a geometric fabric action space to ensure reactivity and safety. We then distill this teacher FGP into a stereo RGB-based student FGP in simulation. To our knowledge, this is the first work that is able to demonstrate robust sim2real transfer of an end2end RGB-based policy for complex, dynamic, contact-rich tasks such as dexterous grasping. Our policies are able to generalize grasping to novel objects with unseen geometry, texture, or lighting conditions during training. Videos of our system grasping a diverse range of unseen objects are available at <https://dextrah-rgb.github.io/>

## I. INTRODUCTION

Accurately controlling multi-fingered robotic hands to grasp objects has remained a longstanding challenge in robotics. Any solution must be generalizable to new categories of objects, robust to environmental changes, and operate using sensors that work with the full range of objects in anthropocentric environments. Such a system would help to unlock the capabilities of multifingered hand-arm systems which remain challenging

to develop general policies for, limiting their applications on downstream tasks.

Recently, great progress has been made in leveraging reinforcement learning in simulation for manipulation and locomotion. Tools such as domain randomization have allowed roboticists to successfully transfer policies trained in simulation to the real world. It has allowed practitioners to easily scale up data collection for their policies allowing for reactive behavior that can take into account both proprioceptive and vision-based inputs.

Despite these successes, the current solutions developed using such tools remain limited. Existing systems almost always factorize the problem of creating a grasping policy, avoiding training direct end-to-end RGB-to-action visuomotor policies. One common approach is to cast the problem as a kinematic one of finding static grasp configuration. These methods are effective on the median object, but they are not continuously reactive and struggle with disturbances or to correct when dealing with unusual unseen objects. Other methods that do continuous vision-based grasping via simulation are almost always restricted to depth as most simulators are unable to render high fidelity images efficiently at scale. This causes problems with semi-transparent or translucent objects, and when dealing with noise produced by IR-based cameras.

To address these challenges, we present a method to perform Dexterous Arm-Hand grasping end2end from RGB input, or Dextrah-RGB. Our main contributions are 1) simulation-only training of an RGB-based FGP, and 2) successful deployment of our RGB policy in the real world to achieve safe, reliable, and reactive grasping behaviors.

We hope our approach to be a useful module in its own right, as well as a system which can be used to develop more complex skills in addition to serving as a source of data for larger pixels-to-action foundational policies.

## II. RELATED WORK

Dexterous Grasping has been studied extensively with a rich history of prior work. Classical methods typically involve optimizing an analytic grasp metrics [16, 5, 7]. These works are typically limited to synthesizing precision grasps. They also rely on groundtruth object models and their performance suffers when accurate models are unavailable [13]. Data driven methods to dexterous grasping involve leveraging grasp datasets to train grasp planning networks. Some examples of these datasets include MultiDex [11], DexGraspNet [27], Grasp'D-1M [25], and Get a Grip [13]. A lot of past work involves grasp synthesis based on pointcloud/depth information. Not only do they capture the geometry of the objects, but are also more feasible to simulate compared to high-fidelity RGB rendering. UniDexGrasp learns dexterous grasping policies from full pointclouds [30] and UniDexGrasp++ improved upon this method by introducing a geometry-aware curriculum [26]. Both of these works only demonstrate results in simulation. DexRepNet [12] learns a representation that combine spatial geometric features of the hand and object and demonstrate successful sim2real results; however, they require access to CAD models of their objects in order to register pointclouds for pose estimation-thereby limiting the generalizability of their method. DexPoint is able to demonstrate successful sim2real transfer of their pointcloud policies. Their key observation is that when deploying in the real world, there are typically many points on the finger that are missing due to occlusions. They use the proprioceptive states of the robot to perform forward kinematics on the hand and sample various points on the mesh of the fingers to fill in the missing points and feed this combined pointcloud into their policy [19]. Agarwal et al. [1] predict a pre-grasp pose by matching DINO-ViT features with previous instances of objects. They then have a proprioceptive-only policy execute the motion to realize the grasp. The policy predicts weights for different eigengrasps that they calculate by performing PCA on a set of grasp poses gathered from motion capture data. Singh et al. pretrain their policies on human videos [22] by reconstructing objects in simulation and retargeting human hand motions to robot hand actions. After pre-training, they are able to demonstrate impressive sim2real results by finetuning with PPO in simulation to train a policy conditioned on the depth image. The work that is most similar to our work is DextrAH-G [14]. In this work, they use a student-teacher distillation framework to train a state-based teacher policy to pick up objects from the Visual

Dexterity [3] dataset. In order to allow for more functional grasps, they transform the action space for the hand into a PCA subspace of the robot hand finger joint motions derived from retargeting human grasping data. This action space is also embedded into a geometric fabric [20] to ensure safe and reactive controllers. The teacher is then distilled into a depth-based student and exhibits remarkable sim2real transfer, being able to grasp objects unseen in the training set.

## III. DEXTRAH-RGB

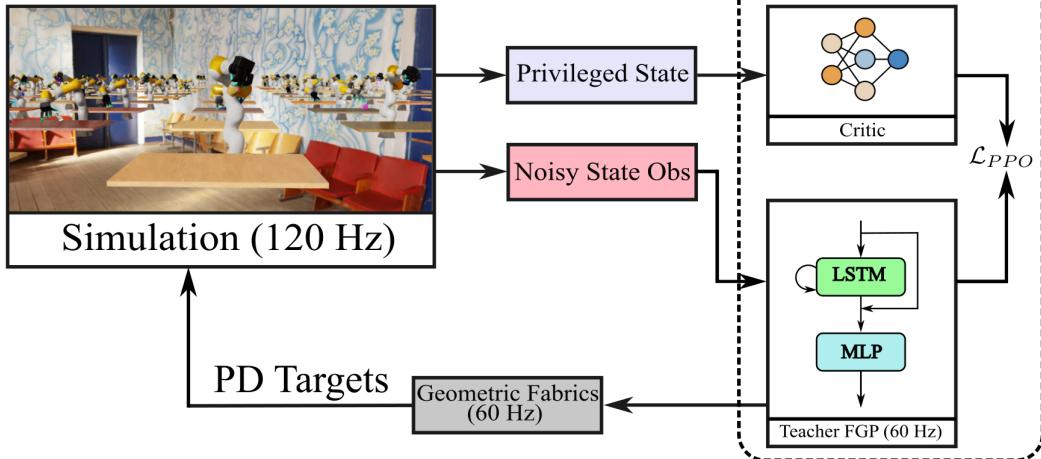
In this section we detail the training of our rgb-based policy. In order to ensure safety of the robot, we use geometric fabrics. This also has the benefit of exposing an action space that lends itself well to the task of dexterous grasping. We first train a state-based, teacher fabric guided policy (FGP) in simulation using reinforcement learning. We then use an online version of DAgger to distill the teacher FGP into a student FGP. Crucially, the student doesn't receive any state information and instead receives two RGB images from a set of cameras in a stereo configuration as input. This student is trained from purely simulated data and we use this to deploy in the real world.

### A. Geometric Fabrics and Fabric-Guided Policies (FGPs)

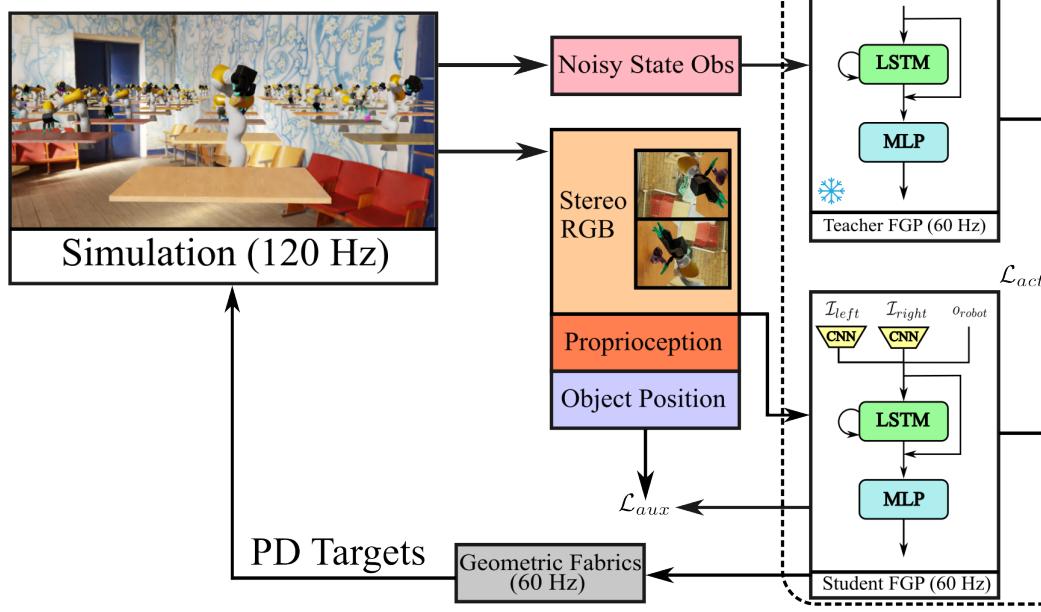
Geometric fabrics is a framework that generalizes classical mechanical systems to design safe, reactive, and robust policies. It involves specifying the desired behavior of an artificial second-order dynamical system, and then connecting this through a torque law (e.g. joint-level PD control) in order to realize the behavior on a real robot [29]. We express this behavior through a combination of *geometric* and *forcing* terms. Geometric terms are used for specifying the nominal behavior of the policy, and crucially, create speed-independent paths. This ensures that no matter how fast or slow the robot moves, it still follows the same path. Forcing terms, on the other hand, serve to perturb the robot from its nominal behavior and these are typically used to ensure safety (e.g. joint limits) or by the RL policy to guide the robot to complete a desired task. Where possible, we aim to push the desired behavior into the underlying geometry because multiple forcing terms can fight with each other resulting in a collapse of the desired behavior. For the purposes of this work, we used the exact same geometric fabric and action space as DextrAH-G, and so we briefly outline the geometric fabric and refer the reader to [14] for in-depth details.

The collision avoidance behavior is embedded into both the underlying geometry and a forcing fabric. Most of the collision avoidance is enforced by the geometric term and only when two objects are close to collision does the forcing term activate to push them apart. Due to the kinematic redundancy of the robot, we add a geometric attraction term that nominally brings the robot to an elbow-out, fingers-curved configuration. This is kept in the underlying geometry to ensure the perturbations from the RL policy are not canceled out by this attractor. We impose forcing terms to ensure that the robot remains within its joint position limits as this is a

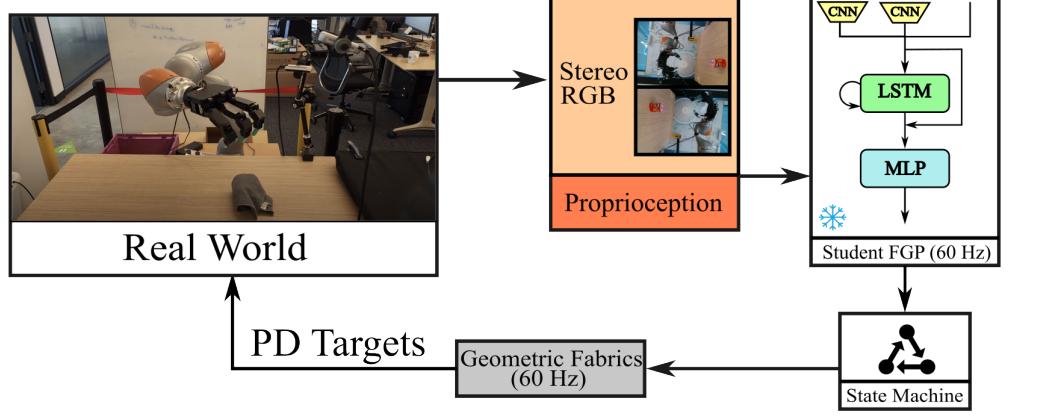
### Stage 1: Teacher Training



### Stage 2: Student Training



### Stage 3: Real World Deployment



**Fig. 3:** We train a state-based FGP teacher using PPO (top). We then distill the teacher FGP into an RGB-based student policy using DAgger (middle). We then deploy the student FGP in the real world.

Object	Pitcher	Pringles	Coffee	Container	Cup	Cheezit	Cleaner	Brick	Spam	Pot	Airplane
DextrAH-RGB (ours)	20%	<b>100%</b>	80%	<b>100%</b>	60%	40%	80%	<b>100%</b>	60%	<b>100%</b>	0%
DextrAH-G	<b>80%</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>	<b>80%</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>	<b>100%</b>	<b>60%</b>
DexDiffuser [28]	-	60%	-	-	60%	80%	<b>100%</b>	-	-	-	20%
ISAGrasp [4]	-	60%	-	40%	-	80%	-	-	-	80%	-
Matak [15]	67%	<b>100%</b>	67%	-	0%	0%	<b>100%</b>	<b>100%</b>	0%	-	-

TABLE I: Success rates of our method compared with prior work.

safety critical aspect that should not be perturbed. The action space for the RL policy is the 6-dof pose of the palm and a reduced, 5 dimensional PCA action space for the hand. We create 2 forcing terms, one for the palm pose, and another for the PCA finger space. These fabric terms, and their associated priority metrics, allow us to solve for the fabric acceleration. We then formulate a quadratic program that tries to bring the acceleration of the robot to match that of the fabric system, while respecting acceleration and jerk limits.

### B. State-based Teacher FGP Training

We train DextrAH-RGB in simulation at scale across many different objects using NVIDIA Isaac Lab. Due to the sample inefficiency of RL, we do not directly train an RGB-based policy from scratch using PPO. Instead, we first train a teacher FGP which receives privileged state information. Then, as described in Section III-C, we distill the teacher policy into an RGB-based student policy. The teacher FGP is trained via PPO with the same hyperparameters as in [14] with equivalent value function and policy networks consisting of two MLP layers of 512 units each followed by a single 512 unit LSTM layer. We additionally add skip connections around the LSTM before passing through the final readout layer. The inputs to these networks consist of the measured robot joint position and velocity, the measured position and velocity of fingertip and palm points, the object pose, the object position goal, a one-hot encoding of objects, the last FGP action, and the position, velocity, and acceleration of the fabric.

We use a simplified reward function compared to [14]. We have four reward terms: a reward for driving the hand close to the object, a reward for moving the object to a position goal in freespace, a reward for lifting the object off the table, and a regularization penalty to stretch the fingers open. The first reward term is defined in terms of  $d_{hand\_obj}$  which represents the maximum distance between any point on the Allegro hand (four fingertip positions and one palm position) and the object:  $d_{hand\_obj} = \max_{i \in \{\text{palm\_pos, fingertips}\}} \|x^i - x^{obj}\|$ . We then have the reward term as  $r_{hand\_obj} = \exp(-10 d_{hand\_obj})$ . The goal reward is defined as  $r_{obj\_goal} = \exp(-\beta_{obj\_goal} \|x^{obj} - x^{goal}\|)$ . The lifting reward defined as  $r_{lift} = \exp(-\beta_{lift} (x_z^{obj} - x_z^{goal})^2)$ , where  $z$  is the vertical direction. Lastly, the regularization reward term prevents the fingers from curling too much:  $r_{curl} = -\beta_{curl} \|q_{hand} - q_{curl}\|^2$ , where  $q_{curl}$  represents a configuration. All  $\beta$  coefficients are positive scalars. The final reward is defined as a weighted sum of these reward terms:  $r = w_{hand\_obj} r_{hand\_obj} + w_{obj\_goal} r_{obj\_goal} + w_{lift} r_{lift} + w_{curl} r_{curl}$ .

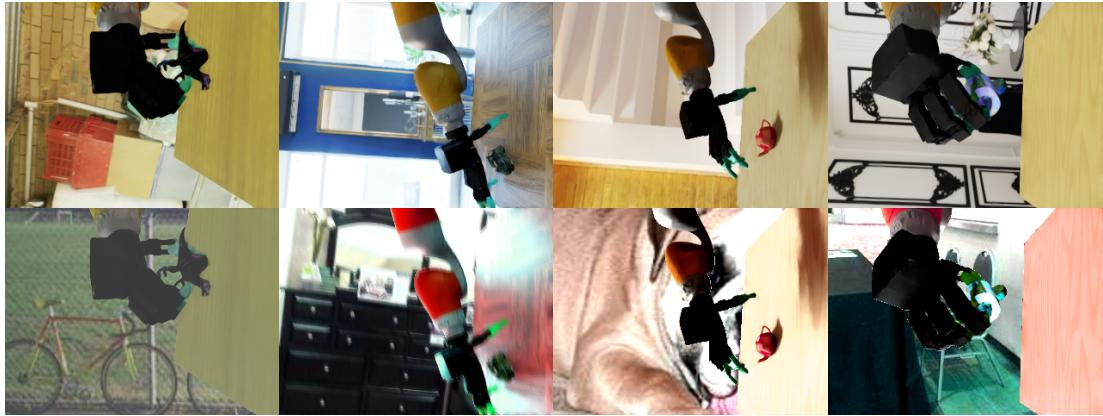
$$w_{curl} r_{curl}$$

Similar to [18, 8], we leverage Automatic Domain Randomization (ADR) when training our teacher policy. It induces a learning curriculum that progressively increases the task and environmental condition difficulty as the agent’s skill improves. In this work, we formulate ADR by setting the initial and terminal values or ranges of various parameters. When policy performance is sufficiently high, the values or ranges of the various parameters are linearly incremented towards the terminal settings. The granularity of the increments is specified beforehand. Unlike [8], all parameters under ADR control are shifted in tandem toward their maximal settings. The terminal values for the various parameters are reasonably set, and we desire policies to reach these maximal values. See Table II for more details about the parameters ADR controls and their initial and terminal ranges.

### C. RGB Student FGP Training

In order to train the RGB-based student policy, we leverage student-teacher distillation and use an online version of DAgger [21], similar to DextrAH-G [14]. The student receives proprioceptive data in the form of robot joint states and velocities, as well as two RGB images corresponding to the left and right camera. We opted to use this stereo camera setup to allow the student to infer depth from the images. We use the Isaac Lab [17] simulation framework which offers ray-traced tiled rendering functionality to allow for fast and realistic rendering in each environment.

To create realistic looking scenes, we follow a similar method as Synthetica [23]. We randomize dome-light HDRI backgrounds with a probability of 30%. At the beginning of every episode, the material properties such as albedo tint, roughness, metallic constant, and specularity of the robot, table, and object are randomized. Furthermore, the texture of the table and objects are also randomized. The objects we used initially came textureless, and so to make them look more realistic we bind textures from random, everyday objects found in the Omniverse Asset Library. While the textures may not semantically match the geometry because the UV mapping will be completely off, the objects will still look somewhat realistic. Along with these randomizations, we also add data augmentations such as random background, colour jitter, and motion blur. Tables III and IV contains more details about the randomization ranges and probabilities. We set the ADR increment to the maximum when training the student. Figure 4 shows various examples of rendered images from the left camera. The top row shows the camera renderings from



**Fig. 4:** The top row shows the left camera renderings for different envs in simulation. The bottom row shows various data augmentations applied to these sim renderings that are passed to the student policy.

simulation with various randomizations applied to the lighting and materials. The bottom row shows how the images look after going through the aforementioned data augmentations.

The architecture of the student is shown in Figure 3. We picked a stereo-based setup because we found stereo policies to perform better than monocular ones in simulation. The policy takes in two images that are of shape  $320 \times 240$ . An example of the two stereo images from sim is shown in Figure 5. The left and right image both are fed individually through an image encoder to output image embeddings. The image encoder consists of a CNN with [16, 32, 64, 128] filters and ReLU activations. The output of the CNN goes through an Average Pool layer before being flattened and projected to a 32-dimensional space which represents the size of our embedding vector. These embeddings are concatenated with the proprioceptive input and are fed into an LSTM with 512 units. The output of the LSTM is concatenated with the input to it and fed into an MLP. This architecture is similar to the connections in DenseNet [10] and previous work [24] have shown that dense architectures are better for policy learning. The MLP has three layers with [512, 512, 256] units. The output from the LSTM and input from this MLP are concatenated and fed into an auxiliary head that predicts the object position which is just a single MLP with [512, 256] units. All activations are elu. The inputs to the student FGP are the same as the teacher FGP except that the object pose and one-hot encoding are replaced with the stereo-RGB pair. We opted to use a small CNN encoder rather than a pre-trained backbone such as ResNet[9] or ViT[6]. The reason for this is because back-propagating gradients through our image encoder allows the network to learn task-specific features which we could not otherwise learn with a frozen backbone. We could not finetune these backbones because they are very large models and having to track their gradients would significantly reduce the amount of environments we could run in parallel, thus slowing down training.

The student outputs the same actions as the teacher. It is jointly supervised on the imitation loss and auxiliary loss with  $\mathcal{L} = \mathcal{L}_{action} + \mathcal{L}_{aux}$ , where  $\mathcal{L}_{action} =$

$D_{KL}(\pi_{student} \| \pi_{teacher})$ , and  $\mathcal{L}_{aux} = \|\hat{x}_{obj} - x_{obj}\|$ .  $x_{obj}$  refers to the groundtruth object position and  $\hat{x}_{obj}$  is the network's prediction of the object position. For the imitation loss, we chose to use a KL loss instead of an  $l2$  loss on the mean and variance because we noticed across all 4 seeds that were tested, policies trained on the KL loss always outperformed their  $l2$  counterparts. Since the variance for both the teacher and student policies are fixed, the error in the variance is driven to zero. Thus, the KL loss effectively reduces to  $(\mu_{student} - \mu_{teacher})^\top \Sigma_{teacher}^{-1} (\mu_{student} - \mu_{teacher})$ . Since we use diagonal Gaussians, this further reduces to  $\sum_i \frac{1}{\sigma_i^2} (\mu_{student}^i - \mu_{teacher}^i)^2$ . This prioritizes driving the error to zero along dimensions with lower variance which is more expressive than the standard  $l2$  loss which weighs the error in all dimensions equally.

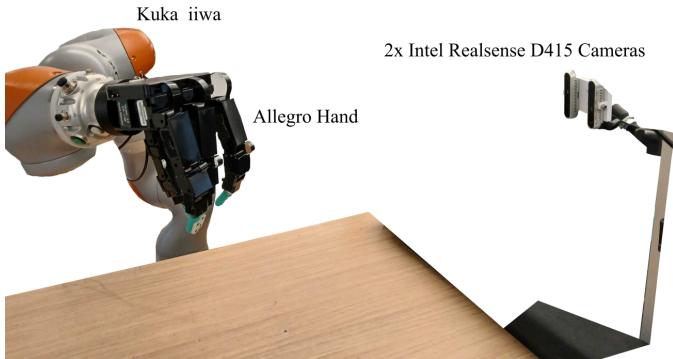
During teacher training, the maximum episode length is set to 10 seconds. This is to give the teacher sufficient time for exploration to lift the object, and to ensure that once the object is lifted, it remains firmly grasped. However, if we train the student with the same maximum episode length, then most of the episode will involve the object already being lifted up in the air. The main divergence between the student and teacher is likely in the beginning of the episode when the teacher agent is trying to grasp the object, and so with a longer episode, this portion of the trajectory will be proportionately de-emphasized. However, it is still imperative to have a sufficiently long episode so that the student policy can learn important recovery behaviors if it is unable to grasp the object on the first try. Thus, when distilling the student, we timeout the episode early if the object is held in the air for 2 seconds.

#### IV. EXPERIMENTS

We deploy our policies on a 7 DoF Kuka LBR iiwa arm with a 16 DoF Allegro Hand v4 mounted on top. Additionally, we have two Intel Realsense D415 cameras mounted rigidly to the table. The setup of our robot system is shown in Figure 6. The low-level PD controller for the Kuka runs at 1 kHz and for the Allegro runs at 333 Hz. The cameras stream RGB



**Fig. 5:** Stereo images from left camera (top) and right camera (bottom).



**Fig. 6:** Real robot setup

information at 60 Hz. We are able to lower the kernel launch overhead by doing CUDA graph captures of our policy to ensure that it also runs at 60 Hz. We noticed that in the real world, the policies were far more performant when running at 60 Hz than at 30 Hz. All policy and control processes run on a single NVIDIA Jetson Orin.

One of the most popular assessments of dexterous grasping ability involves quantifying the single-object grasp success rate. In order to assess the success rate, we evaluate our policies on 11 objects from common datasets such as [2] which have been used by other grasping research. Each object is placed in five different poses on the table and for each trial, we deploy our RGB grasping policy, and the fraction of poses that lead to a successful grasp of the object in the air is recorded as the success rate. We run our policy continually until either the grasp succeeds, or we experience a failure from which the robot cannot recover. One of the benefits of being able to run the policy continuously is that its recurrent architecture allows it to progressively adapt to the environment. Table I shows how our method compares with prior works. For many of the objects, we are able to achieve close to the state-of-the-art. It is important to note that the state-of-the-art method uses depth whereas we use stereo RGB, making our task more challenging.

## V. LIMITATIONS

In this work, we are able to demonstrate remarkable grasping ability; however, there are some important limitations to be discussed. Firstly, our usage of the PCA action space inherits the limitations from DextrAH-G, namely that the focus

on grasping behavior fundamentally limits the dexterity of the robot. Additionally we imbue behaviors such as collision avoidance with the table in the underlying geometric fabric to ensure safety of the system. This can lead to the robot having difficulty grasping smaller objects as they are closer to the table and in the future, it would be better to have this behavior be something that the policy can learn through sensory inputs. Our distillation method requires a two-stage pipeline for policy training which can be cumbersome to train. Further research on various exploration strategies can lead to a single-stage end-to-end RL pipeline which can lead to a more streamlined training framework. Another key limitation is that our grasping is not functional. As seen in Figure 1, the pot is grasped from its base rather than from its handle which is the intended design of the object. Lastly, our policies can only handle one object in the scene which means that it would not be able to perform the task in a cluttered scenario.

## VI. CONCLUSION

We present DextrAH-RGB, a method for dexterous grasping from RGB-input. In order to achieve this, we first train a teacher policy in simulation that receives state-information. We then distill this into an RGB-based student policy. We leverage the real-time ray-tracing capabilities to offer fast and realistic tiled rendering for the student. We further make use of geometric fabrics that exposes an action space to both the teacher and student policies. This action space is one that allows for safety and reactivity while also providing a strong inductive bias for dexterous grasping behaviors. We are able to demonstrate successful sim-to-real transfer of our end-to-end RGB policies. Future work includes improving the performance of DextrAH-RGB along with performing a bin-packing assessment.

## REFERENCES

- [1] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping, 2023. URL <https://arxiv.org/abs/2312.02975>.
- [2] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, September 2015. ISSN 1070-9932. doi: 10.1109/mra.2015.2448951. URL <http://dx.doi.org/10.1109/MRA.2015.2448951>.
- [3] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84), November 2023. ISSN 2470-9476. doi: 10.1126/scirobotics.adc9244. URL <http://dx.doi.org/10.1126/scirobotics.adc9244>.
- [4] 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, 2022. URL <https://arxiv.org/abs/2210.13638>.

- [5] Matei T. Ciocarlie, Corey Goldfeder, and Peter K. Allen. Dexterous grasping via eigengrasps : A low-dimensional approach to a high-complexity problem. 2007. URL <https://api.semanticscholar.org/CorpusID:6853822>.
- [6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL <https://arxiv.org/abs/2010.11929>.
- [7] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, 1992. doi: 10.1109/ROBOT.1992.219918.
- [8] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviychuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality, 2024. URL <https://arxiv.org/abs/2210.13702>.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL <https://arxiv.org/abs/1512.03385>.
- [10] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. URL <https://arxiv.org/abs/1608.06993>.
- [11] Puha Li, Tengyu Liu, Yuyang Li, Yiran Geng, Yixin Zhu, Yaodong Yang, and Siyuan Huang. Gendexgrasp: Generalizable dexterous grasping, 2023. URL <https://arxiv.org/abs/2210.00722>.
- [12] Qingtao Liu, Yu Cui, Qi Ye, Zhengnan Sun, Haoming Li, Gaofeng Li, Lin Shao, and Jiming Chen. Dexrepnet: Learning dexterous robotic grasping network with geometric and spatial hand-object representations, 2023. URL <https://arxiv.org/abs/2303.09806>.
- [13] Tyler Ga Wei Lum, Albert H. Li, Preston Culbertson, Krishnan Srinivasan, Aaron D. Ames, Mac Schwager, and Jeannette Bohg. Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer, 2024. URL <https://arxiv.org/abs/2410.23701>.
- [14] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D. Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics, 2024. URL <https://arxiv.org/abs/2407.02274>.
- [15] Martin Matak and Tucker Hermans. Planning visual-tactile precision grasps via complementary use of vision and touch, 2022. URL <https://arxiv.org/abs/2212.08604>.
- [16] A.T. Miller and P.K. Allen. Graspit! a versatile simulator for robotic grasping. *IEEE Robotics & Automation Magazine*, 11(4):110–122, 2004. doi: 10.1109/MRA.2004.1371616.
- [17] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034.
- [18] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik’s cube with a robot hand, 2019. URL <https://arxiv.org/abs/1910.07113>.
- [19] Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Hao Su, and Xiaolong Wang. Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation, 2022. URL <https://arxiv.org/abs/2211.09423>.
- [20] Nathan Ratliff and Karl Van Wyk. Fabrics: A foundationally stable medium for encoding prior experience, 2023. URL <https://arxiv.org/abs/2309.07368>.
- [21] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL <https://arxiv.org/abs/1011.0686>.
- [22] Himanshu Gaurav Singh, Antonio Loquercio, Carmelo Sferrazza, Jane Wu, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Hand-object interaction pretraining from videos, 2024. URL <https://arxiv.org/abs/2409.08273>.
- [23] Ritvik Singh, Jingzhou Liu, Karl Van Wyk, Yu-Wei Chao, Jean-Francois Lafleche, Florian Shkurti, Nathan Ratliff, and Ankur Handa. Synthetica: Large scale synthetic data for robot perception, 2024. URL <https://arxiv.org/abs/2410.21153>.
- [24] Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. D2rl: Deep dense architectures in reinforcement learning, 2020. URL <https://arxiv.org/abs/2010.09163>.
- [25] Dylan Turpin, Tao Zhong, Shutong Zhang, Guanglei Zhu, Jingzhou Liu, Ritvik Singh, Eric Heiden, Miles Macklin, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Fast-grasp’d: Dexterous multi-finger grasp generation through differentiable simulation, 2023. URL <https://arxiv.org/abs/2306.08132>.
- [26] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning, 2023. URL <https://arxiv.org/abs/2304.00464>.
- [27] Ruicheng Wang, Jiali Zhang, Jiayi Chen, Yinzen Xu, Puha Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation, 2023. URL <https://arxiv.org/abs/2210.02697>.
- [28] Zehang Weng, Haofei Lu, Danica Kragic, and Jens Lundell. Dexdiffuser: Generating dexterous grasps with

- diffusion models, 2024. URL <https://arxiv.org/abs/2402.02989>.
- [29] Karl Van Wyk, Ankur Handa, Viktor Makoviychuk, Yijie Guo, Arthur Allshire, and Nathan D. Ratliff. Geometric fabrics: a safe guiding medium for policy learning, 2024. URL <https://arxiv.org/abs/2405.02250>.
- [30] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, Tengyu Liu, Li Yi, and He Wang. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy, 2023. URL <https://arxiv.org/abs/2303.00938>.

## APPENDIX

### A. Randomization Parameters

Parameter	Initial Setting	Terminal Setting
Robot Static Contact Friction Coefficient	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.3, 1.2)$
Robot Dynamic Contact Friction Coefficient	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.2, 1)$
Robot Collision Restitution	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.8, 1)$
Robot Joint PD Stiffness Multiplicative Scaling	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.5, 2)$
Robot Joint PD Damping Multiplicative Scaling	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.5, 2)$
Robot Joint Friction Coefficient	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(-10, 10)$
Object Static Contact Friction Coefficient	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.3, 1.2)$
Object Dynamic Contact Friction Coefficient	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.2, 1)$
Object Collision Restitution	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.8, 1)$
Object Mass Multiplicative Scaling	$\sim \mathcal{U}(1, 1)$	$\sim \mathcal{U}(0.5, 3)$
Object Disturbance Acceleration	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 10)$
Object Spawn Width	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.8)$
Object Spawn Height	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 1)$
Object Measured Position Noise	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.3)$
Object Measured Position Bias	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.2)$
Object Measured Rotation Noise	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.1)$
Object Measured Rotation Bias	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.08)$
Robot Initial Joint Velocity	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 1)$
Robot Measured Position Noise	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.08)$
Robot Measured Position Bias	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.08)$
Robot Measured Velocity Noise	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.18)$
Robot Measured Velocity Bias	$\sim \mathcal{U}(0, 0)$	$\sim \mathcal{U}(0, 0.08)$
$\beta_{\text{obj\_goal}}$	-15	-20
$\beta_{\text{curl}}$	-0.01	-0.05

**TABLE II:** Various physics parameters controlled by automatic domain randomization during learning progression.

Parameter	Probability Distribution
<b>Lighting</b>	
HDRI Texture Map	$\sim \mathcal{U}(\text{texture\_maps})$
Rotation	$\sim \mathcal{U}(SO(3))$
Intensity	$\sim \mathcal{U}(1000, 4000)$
<b>Object</b>	
Texture Map	$\sim \mathcal{U}(\text{texture\_maps})$
Texture Scale	$\sim \mathcal{U}(0.7, 5)$
Diffuse Tint	$\sim \mathcal{U}(0, 1)$
Roughness	$\sim \mathcal{U}(0, 1)$
Metallic	$\sim \mathcal{U}(0, 1)$
Specular	$\sim \mathcal{U}(0, 1)$
<b>Robot</b>	
Roughness	$\sim \mathcal{U}(0.2, 1)$
Metallic	$\sim \mathcal{U}(0, 0.8)$
Specular	$\sim \mathcal{U}(0, 1)$
<b>Table</b>	
Texture Map	$\sim \mathcal{U}(\text{texture\_maps})$
Texture Rotate	$\sim \mathcal{U}(0, 2\pi)$
Diffuse Tint	$\sim \mathcal{U}((0.3, 0.2, 0.1), (0.6, 0.4, 0.2))$
Roughness	$\sim \mathcal{U}(0.3, 0.9)$
Specular	$\sim \mathcal{U}(0, 1)$

**TABLE III:** Various visual domain randomization parameters and their probability distributions

Data Augmentation Type	Probability
Random Background	0.5
Color Jitter	1
Random Blur	0.1

**TABLE IV:** Various data augmentations and their associated probabilities