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

Ritvik Singh*, Arthur Allshire†, Ankur Handa*, Nathan Ratliff*, Karl Van Wyk*
*NVIDIA Corporation
†University of California, Berkeley

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

***Abstract*—One of the most important, yet challenging, skills for a dexterous robot is grasping a diverse range of objects. Much of the prior work has been limited by speed, generality, or reliance on depth maps and object poses. In this paper, we introduce DextrAH-RGB, a system that can perform dexterous arm-hand grasping end-to-end from RGB image input. We train a privileged fabric-guided policy (FGP) in simulation through reinforcement learning that acts on a geometric fabric controller to dexterously grasp a wide variety of objects. We then distill this privileged FGP into a RGB-based FGP strictly in simulation using photorealistic tiled rendering. 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. DextrAH-RGB is competitive with depth-based dexterous grasping policies, and generalizes to novel objects with unseen geometry, texture, and lighting conditions in the real world. Videos of our system grasping a diverse range of unseen objects are available at https://dextrah-rgb.github.io/.**

I. Introduction

Controlling multi-fingered robotic hands to naturally and quickly grasp objects has remained a longstanding challenge in robotics. Ideally, techniques must leverage visual, tactile, and proprioceptive sensory streams to generalize to novel objects and quickly react in dynamic, anthropocentric environments. With such capabilities, multi-fingered robots would become a pertinent technology to society with many relevant applica-

tions.

Recently, significant progress has been made in leveraging reinforcement learning in simulation for manipulation and locomotion. Vectorized physics and sensor simulation have allowed practitioners to easily scale robot experience in dynamic environments, enabling reactive behavior informed by both proprioceptive and vision-based inputs. Moreover, techniques like domain randomization facilitate learning robust and adaptive policies that successfully transfer from simulation to the real world.

Despite these powerful tools, the efficacy of many current strategies for learning manipulation skills remains limited. Existing systems almost always factorize the problem of creating a grasping policy which avoids training direct end-to-end RGB-to-action visuomotor policies. One common approach is to cast the problem as a kinematic one of finding a static grasp configuration and leveraging motion planning tools to acquire the targeted grasp. These methods are not continuously reactive and struggle with disturbances, partial observability, and unseen objects. Other methods that do continuous vision-based grasping via simulation are almost always restricted to depth cameras as most simulators are unable to render high-fidelity images efficiently at scale. Although methods relying on depth readings are fairly successful, IR-based depth cameras have issues with translucent and reflective objects and direct sunlight. They are also a source of various geometric distractors that are not easily simulated at scale.

To address these challenges, we present DextrAH-RGB, a dexterous arm-hand grasping policy that computes high-rate actions directly from RGB input. Our main **contributions** are: **1)** simulation-only training of an RGB-based FGP, **2)** including significant scaling of the perceptual architecture to handle the substantially more challenging (stereo) RGB processing, and **3)** successful deployment of our RGB policy in the real world to achieve safe, reliable, and reactive grasping behaviors. **To our knowledge, this is the first work to demonstrate successful sim2real transfer of RGB policies for a robot hand-arm system for such complex, contact-rich tasks.** Our method allows for training of both monocular and stereo RGB policies and can straightforwardly be extended to an arbitrary number of cameras in the scene.

II. Related Work

Dexterous grasping has been studied extensively with a rich history of prior work. Classical methods typically involve optimizing analytic grasp metrics [**?** **?** **?**]. 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 [**?**]. Data driven methods to dexterous grasping involve leveraging grasp datasets to train grasp planning networks. Some examples of these datasets include MultiDex [**?**], DexGraspNet [**?**], Grasp'D-1M [**?**], and Get a Grip [**?**]. 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 [**?**] and UniDexGrasp++ improved upon this method by introducing a geometry-aware curriculum [**?**]. Both of these works only demonstrate results in simulation. DexRepNet [**?**] learns a representation that combines 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 [**?**]. Agarwal et al. [**?**] 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 [**?**] 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 ours is DextrAH-G [**?**]. This work also uses a student-teacher distillation framework, but distill instead to a simpler depth processing policy. They first train a state-based teacher policy to pick up objects from the Visual Dexterity [**?**] 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 [**?**] to ensure safe and reactive controllers. The teacher is then distilled into the 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. 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. In fact, we use exactly the same geometric fabric controller as in [**?**]. With this vectorized controller, 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 does not receive any object state information and, instead, receives either one or two RGB images from a set of cameras in a stereo configuration as input. This student FGP is trained purely from simulated data and deployed to the real world. The teacher policy took 62 hours to train on a single 8-GPU H100 node

and the student policy took 53 hours to train on a single 4-GPU L40S node. We use L40S GPUs for distillation as they contain the RTX cores necessary for high-fidelity rendering.

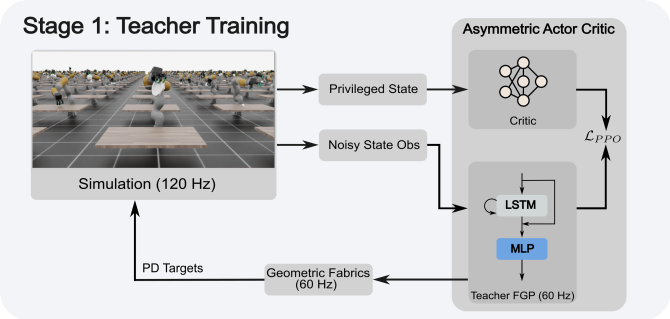


Fig. 2: The first stage of our pipeline involves training a state-based teacher policy in simulation using PPO. We adopt an asymmetric actor critic framework whereby the teacher policy receives noisy state observations whereas the critic receives privileged (and perfect) state observations. This is done to ensure that the policy is not overly reliant on behaviors that require accurate state estimates as this can make it harder to distill into a vision-based policy. The teacher policy uses an LSTM layer to enable reasoning over historical context, enabling adaptation to current dynamics. We add dense skip connections similar to [**?**] to improve stability and performance of the policy.

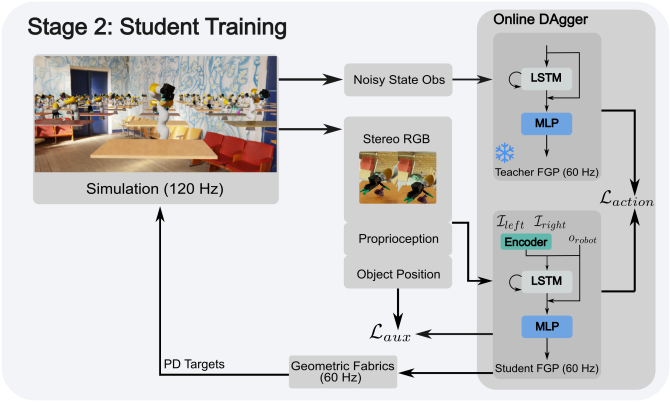


Fig. 3: The second stage of our pipeline involves distilling the previously trained state-based teacher policy into a vision-based student policy. We use an online implementation of DAgger [**?**] where at each step, the observations for the teacher and student are queried and fed into the respective networks. The student is supervised to minimize the KL-divergence between its action distribution and that of the teacher. Furthermore, we add an auxiliary loss for predicting the 3D object position. The student architecture consists of a encoder which takes as input images from the left and right camera and outputs a stereo embedding which is then concatenated with the standard robot proprioceptive observations and passed into an LSTM followed by an MLP. Similar to the teacher, dense connections are used to improve the performance of the policy.

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

Geometric fabrics are second-order dynamical systems that generalize classical mechanics and can be used to design safe, reactive, and stable controllers. Desired behaviors of a robot can be specified by the geometric fabric controller and then realized on a real robot via an appropriate torque law (e.g. joint-level PD control) [**?**]. We create these behaviors through a combination of *geometric* and *forcing* fabric terms. Geometric terms are used for specifying the nominal behavior of the controller, and crucially, create speed-independent paths. This ensures that no matter how fast or slow the robot moves, it still follows the same path. On the other hand, forcing terms serve to perturb the robot from its nominal behavior to complete a specific task. For example, we may want the robot to reach a target, or in a more advanced setting, have an FGP complete a grasping 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 use the same geometric fabric and action space as in DextrAH-G, and briefly outline the geometric fabric (refer to [**?**] for in-depth details).

The geometric fabric possesses a few different behaviors. First, collision avoidance behavior is embedded into both the underlying geometric and forcing fabric terms to prevent self and environmental collision. 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-curled configuration. This term guides the overall posture of the arm without preventing goal-reaching behavior in the palm and PCA action spaces. 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. In these spaces, we add two forcing terms, one for the palm pose, and another for the PCA finger space which allow goal reaching in these spaces. Consequently, FGPs emit parametric forces on the geometric fabric by issuing targets in these spaces. Finally, we add forcing terms to ensure that the robot remains within its joint position limits as this is a hardware safety critical aspect. These fabric terms, and their associated priority metrics, allow us to solve for the net fabric acceleration which is further modulated to meet acceleration and jerk limits of the robot. The desired acceleration is forward integrated at 60 Hz using an approximate second-order Runge-Kutta scheme and the time-integrated position and velocity states are passed as targets to the underlying joint PD controller.

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. We then distill the teacher policy into an RGB-based student policy. The teacher FGP is trained via PPO with the same hyperparameters as in [**?**] with the policy network consisting of a single 512 unit LSTM layer followed by two MLP layers of 512 units. The critic consists of a 2048 unit LSTM with an MLP with [1024, 512] units. We also add skip connections

around the LSTM before passing through the final layers. This architecture is similar to the connections in DenseNet [**?**] and previous work [**?**] has shown that dense architectures are better for policy learning. 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 an asymmetric actor-critic formulation where the full privileged observations are passed into the critic whereas the teacher receives a noisy subset of the critic observations. This is to ensure that the teacher does not learn behaviors that rely on accurate state estimation as this can impact how well the teacher can be distilled into the student. Finally, the critic also receives measured robot joint torque and measured forces at the fingertip and palm points. Figure 2 illustrates our teacher training pipeline.

We use a simplified reward function compared to [**?**]. 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 positions of the fingertip and one position of the palm) 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 is 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 β coefficients are positive scalars. The final reward is defined as a weighted sum of these reward terms: $r = w_{hand\_obj} r_{hand\_obj} + w_{obj\_goal} r_{obj\_goal} + w_{lift} r_{lift} + w_{curl} r_{curl}$.

Similar to [**?** **?**], 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 in advance. Unlike [**?**], 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 IV for more details about the parameters ADR controls and their initial and terminal ranges. Figure 4 illustrates how the ranges are adjusted towards the terminal ranges.

Finally, we adopt four important teacher FGP training settings that depart from [**?**]. First, ADR scales the velocity targets for the PD controller from 1 to 0. Non-zero velocity targets allow a faster dynamic response of the robot which is beneficial to RL exploration. Ultimately zeroing the velocity targets allow us to train an FGP conditioned on these dynamics, which is in line with the FGPs in [**?**]. Next, ADR scales all velocity and acceleration inputs to the FGP from 1 to 0. This forces the FGP to leverage its recurrency to reason over position-only dynamics to avoid relying on higher-order signals that are state estimated. Instead, FGPs will only rely on direct observables, avoiding various nuances and errors in state estimation. Third, to further promote effective RL exploration through faster motion, we time-integrate the fabric differential equation two timesteps for every simulation step while ADR increases fabric damping from 10 to 20. The former increases the speed of motion, while the latter dials the speed back in a more continuous fashion that is deployable in the real world. Finally, we modify the control logic for the disturbance wrench applied to the object for grasping. In [**?**], the wrench activates when the object has been lifted. In this work, the wrench activates when the hand is 0.3 m away from the object center. This ensures that the object can start moving before the hand secures a grasp, which promotes even more reactive policies. These settings are carried over for training the subsequent student FGP as well.

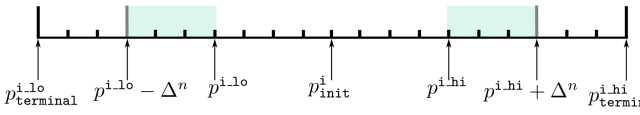


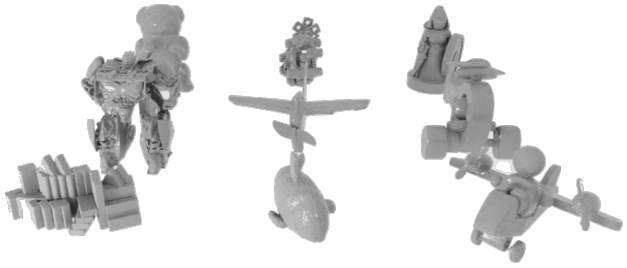
Fig. 4: For all physics parameters p^{i}, the initial values for $p^{\text{i\_lo}}$ and $p^{\text{i\_hi}}$ are initialized to $p^{\text{i}}_{\text{init}}$. As the policy starts performing better, $p^{\text{i\_lo}}$ is decremented by Δ^n and $p^{\text{i\_hi}}$ is incremented by Δ^n. The parameter value ranges are constantly increased until they reach the terminal values $p^{\text{i\_lo}}_{\text{terminal}}$ and $p^{\text{i\_hi}}_{\text{terminal}}$.

C. RGB Student FGP Training

To train the RGB-based policy, we leverage student-teacher distillation and use an online version of DAgger [**?**], similar to DextrAH-G [**?**]. Our training pipeline for this stage is depicted in Figure 3. 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 implicitly infer depth from the images. We use the Isaac Lab [**?**] simulation framework which offers ray-traced tiled rendering functionality to allow for fast and realistic rendering in each environment.

To create realistic scenes, we follow a similar method as Synthetica [**?**]. 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, so we bind textures from random, everyday objects found in the Omniverse Asset Library. Although the

textures may not semantically match the geometry because the UV mapping is completely off, the objects will still look somewhat realistic. Figure 5 shows a comparison between the original, textureless meshes, and the subsequently textured meshes. Along with these randomizations, we also add data augmentations such as random background, color jitter, and motion blur. Tables V and VI contains more details about the randomization ranges and probabilities. We set the ADR increment to the maximum when training the student. Figure 6 shows various examples of images rendered from the left camera. The top row shows the camera renderings from the 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.



(a) Example meshes from the Visual Dexterity dataset with no texture.

(b) Random texture maps of everyday objects binded to the meshes.

Fig. 5: (a) shows an example subset of object meshes with no texture. **(b)** Shows those meshes with random textures binded to them.

The architecture of the student is shown in Figure 3. We chose a stereo-based setup because we found that stereo policies performed better than monocular ones in simulation (detailed ablations can be found in Section IV-A). The policy takes in two images that are of shape 320×240. The left and right images are fed into a stereo encoder, which is shown in Figure 7a. The images are encoded in a Siamese manner through a pre-trained ResNet-18 encoder [**?**]. This encoder has the last two layers removed. For each image, the output of the ResNet encoder is a 40960-dimensional feature vector. Using an MLP, each of these feature vectors is projected down to a 16384-dimensional vector. This is then reshaped to produce 128 128-dimensional tokens per image. The tokens for the left and right images are then passed into a transformer that also takes as input a learnable `[embed]` token. The transformer consists of two layers, for which the `[embed]` token attends to all the other tokens, whereas tokens from one image are only allowed to attend to tokens of the other image or the `[embed]` token. This cross-attention between tokens in one image and the tokens in the other is meant to allow for some form of implicit stereo matching in order to determine the depth of objects in the scene. The cross-attention mask we use is depicted in Figure 7b. This design decision was inspired by DUSt3R [**?**] which performed cross-attention between tokens of different image features in order to perform multi-view stereo reconstruction. We then pass the output for the learnable `[embed]` token into an MLP layer to arrive at a 64-dimensional stereo embedding of the two images. This is similar to how ViT [**?**] uses a learnable `[CLS]` token and feeds its corresponding output into an MLP to ensure that the output isn't biased to any particular input token from the image. The embedding vector is concatenated with the proprioceptive input and fed into an LSTM with 512 units. The output of the LSTM is concatenated with the input to it and fed into an MLP. 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. During training, we make sure to finetune the ResNet encoder as this produces the most performant policies. Due to memory constraints, we cast all of the ResNet weights to bf16.

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 $\Sigma_i \frac{1}{\sigma_i^2} (\mu^i_{student} - \mu^i_{teacher})^2$. 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 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 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

Fig. 6: The top row shows the left camera renderings for different environments in simulation. The bottom row shows various data augmentations applied to these sim renderings that are passed to 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. Finally, the student FGP is trained on the terminal ADR settings which are always achieved by the associated teacher FGP chosen for distillation.

IV. Experiments

A. Architecture Ablations

In this section, we conduct ablation studies to justify our design decisions. Specifically, we compare the performance of a monocular policy versus a stereo policy, with results summarized in Table I. Performance is measured as the success rate of the student policy relative to the teacher policy, averaged across three seeds. Additionally, we evaluate the impact of the auxiliary loss in the object position prediction and the effectiveness of incorporating a transformer into the encoder. All the different model types are distilled from the same three teacher policies. The results show a clear performance boost when using a stereo configuration compared to a monocular one. This is expected, as stereo vision provides additional information for depth perception and computing the object's 3D position. This advantage is further reflected in the positional error, which decreases by nearly 1 cm in the stereo setup. We also evaluate the importance of the transformer. In the stereo case, the transformer corresponds to the architecture in Figure 7a, whereas in the monocular case it is a full self-attention based transformer encoder. The architectures without the transformer have the embedding from the ResNet backbone concatenated directly with policy observations and fed into the LSTM. Adding the transformer module to the stereo model enhances performance, highlighting its role in improving policy performance. Lastly, using a frozen ResNet-18 encoder or one trained from scratch did not perform as well as the finetuned encoder. This highlights another benefit of RGB-based policies which is that they can leverage previously learned representations from pre-training on large-scale datasets.

Model Type	Performance ↑	Pos Error (cm) ↓
Finetune ResNet Mono w/o Attn	0.83	3.5
Finetune ResNet Mono w/ Attn	0.83	3.3
Frozen ResNet Stereo w/ Attn	0.71	4.3
Scratch ResNet Stereo w/ Attn	0.83	3.2
Finetune ResNet Stereo w/o Attn	0.86	2.5
Finetune ResNet Stereo w/ Attn	**0.89**	**2.5**

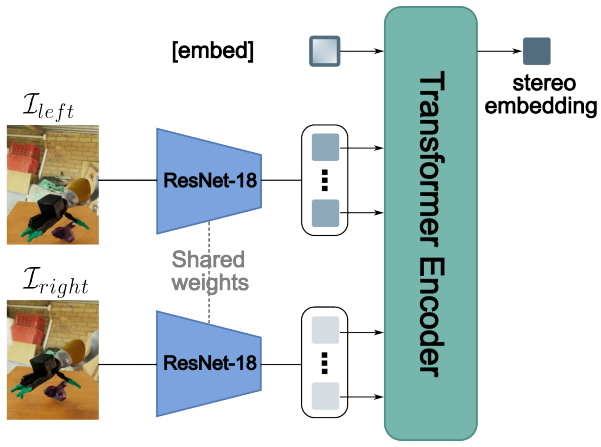
TABLE I: Comparison of different model configurations in simulation. Performance is normalized relative to the teacher's performance and averaged across multiple seeds. Positional error, reported in centimeters, is also averaged across multiple seeds.

B. Real World Experiments

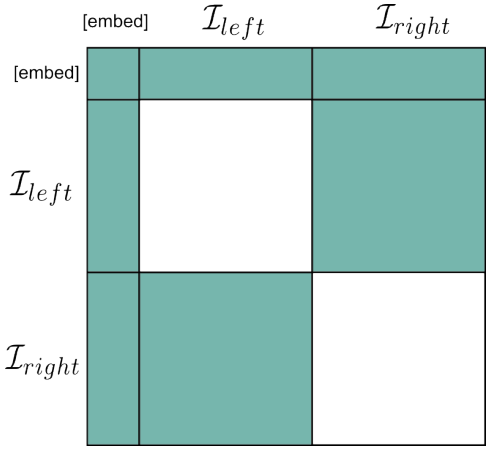
We deploy our policies on a 7 DoF Kuka LBR iiwa arm with a 16 DoF Allegro Hand v4 mounted on top. Additionally, we have two Intel Realsense D415 cameras mounted rigidly to the table. The setup of our robot system is shown in Figure 8a. The low-level PD control node runs at 1 kHz for the Kuka and at 333 Hz for the Allegro. The camera nodes stream RGB images at 60 Hz, and both the geometric fabric control node and FGP node run at 60 Hz. Considering that our FGPs are significantly larger with ResNet encoders, we are able to meet the 60 Hz control rate by lowering the kernel launch overhead with CUDA graphs of the FGP. Finally, a state machine node runs at 60 Hz and orchestrates manual fabric actions and FGP actions as in [**?**]. The PD controllers for both the arm and hand, fabric controller, and state machine run on an NVIDIA Jetson AGX Orion Developer Kit local to the robot. Camera nodes and FGP node run on a full desktop PC with an NVIDIA 3090 RTX card. All nodes communicate with each other via ROS 2. Figure 8b shows our pipeline for deploying the trained policies in the real world.

Object	Pitcher	Pringles	Coffee	Container	Cup	Cheezit	Cleaner	Brick	Spam	Pot	Airplane
DextrAH-RGB (Stereo)	**80%**	**100%**	80%	**100%**	**80%**	40%	40%	**100%**	**100%**	**100%**	**60%**
DextrAH-RGB (Mono)	40%	**100%**	80%	**100%**	60%	80%	20%	**100%**	80%	**100%**	**60%**
DextrAH-G	**80%**	**100%**	**100%**	**100%**	**80%**	**100%**	**100%**	**100%**	**100%**	**100%**	**60%**
DexDiffuser [?]	-	60%	-	-	60%	80%	**100%**	-	-	-	20%
ISAGrasp [?]	-	60%	-	40%	-	80%	-	-	-	80%	-
Matak [?]	67%	**100%**	67%	-	0%	0%	**100%**	**100%**	0%	-	-

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



(a) Stereo Encoder



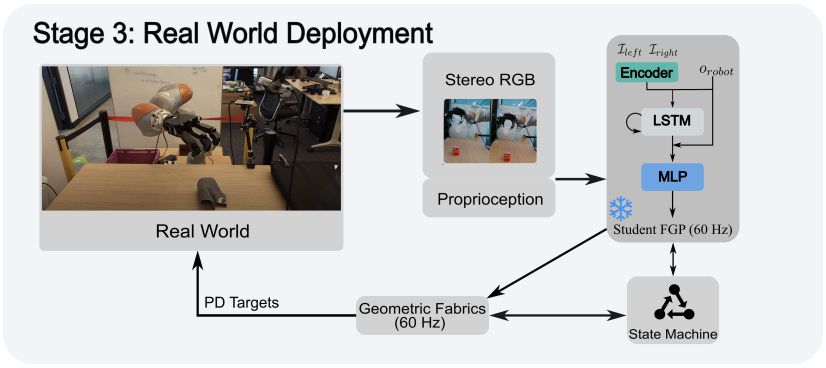
(b) Cross-Attention Mask

Fig. 7: (a) Our stereo encoder first starts with a pre-trained ResNet-18 encoder where the last two layers are removed. Each image is passed into the encoder and outputs a 40960-dimensional vector, which is projected down to a 16384-dimensional layer. These are then split into 128 tokens of 128 dimensions each. The tokens from the left and right images are passed into a transformer that also receives a learnable `[embed]` token. This transformer (2 layers) performs cross-attention between the two images. The output for the `[embed]` token is passed into an MLP layer, yielding the final stereo embedding vector. **(b)** The turquoise sections show the cross-attention between tokens. The `[embed]` token attends to all other tokens in the sequence. The tokens from the left image attend to the `[embed]` token and the tokens from the right image. Likewise, the tokens from the right image attend to the `[embed]` token and the tokens from the left image.

Single Object Grasping Assessment: One of the most popular assessments of dexterous grasping ability involves quantifying the single-object grasp success rate. To estimate the success rate, we evaluate our policies on 11 objects from common datasets such as [**?**] 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. The fraction of poses that lead to a successful grasp and lift of the object 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 II shows how our method compares with prior works. DextrAH-RGB (stereo) typically outperforms DextrAH-RGB (mono) for all but one object, indicating some level of benefit for stereo perception. Overall, DextrAH-RGB is able to achieve state-of-the-art performance for most objects. DextrAH-G does outperform in some cases, but that FGP uses depth images instead of RGB (which is a more challenging setting). Furthermore, the depth images for DextrAH-G were specially truncated to avoid background distractors and incoming sunlight was blocked to avoid eroding the quality of depth images. DextrAH-RGB does not have this limitation and is thus a more generalizable solution as it can work in many lighting conditions and background settings as shown next and in the accompanying video.

Bin Packing Assessment: We follow the bin packing assessment protocol and metrics in [**?**] to more holistically evaluate the grasping performance. In this assessment, the robot is tasked with continuously grasping 36 different objects one-at-a-time and depositing them into a bin. For this task, we employ a state machine which decides when to pass FGP actions to the robot. Specifically, it queries the FGP's prediction of the object position and if the object is located in the air, thereby indicating a successful grasp, the state machine will issue fixed actions to the fabric controller to move the arm to the bin and deposit the object. Afterwards, another fixed action is issued to the fabric controller to return the robot to the forward position after which the FGP is re-engaged. The metrics that are tracked are the consecutive successes (CS) - the number of consecutively successful object transports, cycle time (CT)- the amount of time required for the robot to pick up the object, deposit it in the bin, and then return back to the ready position, and success rate (SR) - the percentage of objects that were successfully transported to the bin. We eval-

(a) Real robot setup



(b) State machine for bin packing

Fig. 8: (a) Our real-world robot setup: an Allegro Hand mounted onto a Kuka iiwa robot arm and two Intel RealSense D415 cameras in a stereo configuration (55 mm baseline). **(b)** The stereo image pair from the real world, along with robot proprioceptive states, are fed into the trained student policy. The student' actions and object-position prediction are passed into the state machine for bin packing. By default, the state machine passes the student actions into the underlying geometric fabric. However, if the object position prediction is sufficiently high (indicating a successful grasp), it transitions from using policy actions to a fixed behavior that moves the object above the bin and deposits it. Afterwards, the state machine returns the robot to its original state and resumes the student policy for the next grasp attempt.

uate DextrAH-RGB (stereo) and DextrAH-RGB (mono) for this protocol in both controlled indoor lighting conditions and high-dynamic range conditions (HDR). HDR conditions were imposed by completely opening window shades and removing light-blocking boards around the robot. Consequently, natural sunlight directly streamed into the camera stereo pair creating a bright background and dark foreground. Such settings were explictly avoided in [**?**] since natural light erodes depth readings and ruins policy performance.

The bin packing performance results for DextrAH-RGB are shown in Table III along with state-of-the-art DextrAH-G. DextrAH-RGB significantly improves upon cycle time being 1-2 seconds faster on average over DextrAH-G. In many cases, objects were successfully transported and the robot returned in 4 seconds, very close to the estimated human cycle time of 3.63 s in [**?**]. However, the increased solution speed came at a cost of less reliability. CS is less by about 2-3 objects on average SR decreased by about $10\% - 14\%$ on average when compared to DextrAH-G. Encouragingly, there does not appear to be a substantial difference in performance between no-HDR and HDR settings, providing empirical evidence that DextrAH-RGB is, in fact, trained to be robust to lighting conditions. Finally, discrepancies in numerical performance between DextrAH-RGB (stereo) and DextrAH-RGB (mono) are not strikingly apparent. This suggests that the generalization error from deploying policies in the real world dominates the performance gain of stereo over mono in simulation.

Model Type	CS (objects) ↑	CT (s) ↓	SR (%) ↑
DextrAH-G	$\mathbf{6.56 \pm 2.41}$	10.66 ± 0.84	**87**
DextrAH-RGB (mono, no HDR)	3.24 ± 1.58	8.63 ± 1.62	73
DextrAH-RGB (stereo, no HDR)	4.53 ± 1.75	8.22 ± 1.10	77
DextrAH-RGB (mono, HDR)	3.83 ± 1.35	$\mathbf{8.18 \pm 1.82}$	73
DextrAH-RGB (stereo, HDR)	3.24 ± 0.91	9.07 ± 1.40	74

TABLE III: Consecutive Successes (CS), Cycle Time (CT), and Success Rate (SR) for the task of bin picking measured across depth, monocular RGB, and stereo RGB.

V. Discussion

We have empirically shown that generalizing RGB-based dexterous grasping policies can be successfully trained in simulation and deployed in the real world. We believe this creates a new frontier in end-to-end simulation-based policy learning for dexterous robots, a highly scalable approach. Evaluation in the real world revealed that such policies are not only viable, but also competitive with current state-of-the-art. DextrAH-RGB expeditiously grasped and transported over 30 objects of novel geometry and texture against unseen backgrounds. Moreover, DextrAH-RGB yielded consistent performance through adverse lighting conditions and succeeded with both monocular and stereo RGB setups, a strong showcase for training robust and flexible RGB-based policies in simulation. While DextrAH-RGB was competitive with DextrAH-G, admittedly DextrAH-RGB did not uniformly improve over DextrAH-G as we hoped. What we have found empirically is, sim2real transfer of dexterous manipulation policies is highly variable and it is currently very hard to predict why some policies transfer better than others given similar or exactly the same training pipelines. Sources that contribute to this variance include reward underspecification, locality of RL, nuanced changes in teacher training pipeline, changes in simulation physics, and varying proficiency levels of teacher reconstruction by student policies. For instance, repeated runs of the distillation pipeline on the same teacher FGP using different seeds transfer differently to the real world despite attaining similar metrics. Given the high-variance, training more teacher and student FGPs for DextrAH-RGB, is likely to result in an improved top performer. Thus, the reduction in DextrAH-RGB's reliability *cannot* be squarely attributed to FGPs consuming RGB over depth. Despite the current high-variance reality of sim2real for DextrAH, DextrAH-RGB does perform quite well and sets a new reference point for RGB-based dexterous grasping policies trained entirely in

simulation, and we expect further improvements in the near future as we scale the models and training runs.

VI. 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 that results in more "vision-aware" policies. Another key limitation is that our grasping is not functional. For example, when the policy tries to grasp the pot, it tries to grasp 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.

VII. Conclusion

We present DextrAH-RGB, end-to-end dexterous grasping policies from RGB-input trained entirely in simulation. To achieve this, we first train a teacher policy in simulation that receives privileged 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 in the single object setting and initiate the multi-object setting as well. We believe our approach is real-world relevant on its own, can be used to develop more complex skills, and serve as a source of data for larger pixels-to-action foundational policies.

References

[] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping, 2023. URL https://arxiv.org/abs/2312.02975.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, 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.

[] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. URL https://arxiv.org/abs/1608.06993.

[] Puhao 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] 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.

[] Nathan Ratliff and Karl Van Wyk. Fabrics: A foundationally stable medium for encoding prior experience, 2023. URL https://arxiv.org/abs/2309.07368.

[] 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.

[] 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.

[] 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.

[] Samarth Sinha, Homanga Bharadhwaj, Aravind Srinivas, and Animesh Garg. D2rl: Deep dense architectures in reinforcement learning, 2020. URL https://arxiv.org/abs/2010.09163.

[] 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.

[] 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.

[] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao 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.

[] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy, 2024. URL https://arxiv.org/abs/2312.14132.

[] Zehang Weng, Haofei Lu, Danica Kragic, and Jens Lundell. Dexdiffuser: Generating dexterous grasps with diffusion models, 2024. URL https://arxiv.org/abs/2402.02989.

[] 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.

[] 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
β_{curl}	-0.01	-0.05
PD Velocity Target	1	0
Fabric Damping Gain	10	20
Observation Annealing	1	0

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

Parameter	Probability Distribution
Lighting	
HDRI Texture Map	$\sim \mathcal{U}(\texttt{texture\_maps})$
Rotation	$\sim \mathcal{U}(SO(3))$
Intensity	$\sim \mathcal{U}(1000, 4000)$
Object	
Texture Map	$\sim \mathcal{U}(\texttt{texture\_maps})$
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	$\sim \mathcal{U}(0, 1)$
Robot	
Roughness	$\sim \mathcal{U}(0.2, 1)$
Metallic	$\sim \mathcal{U}(0, 0.8)$
Specular	$\sim \mathcal{U}(0, 1)$
Table	
Texture Map	$\sim \mathcal{U}(\texttt{texture\_maps})$
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 V: 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 VI: Various data augmentations and their associated probabilities