2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

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

Ritvik Singh[1,2], Arthur Allshire[2], Ankur Handa[1], Nathan Ratliff[1], and Karl Van Wyk[1]

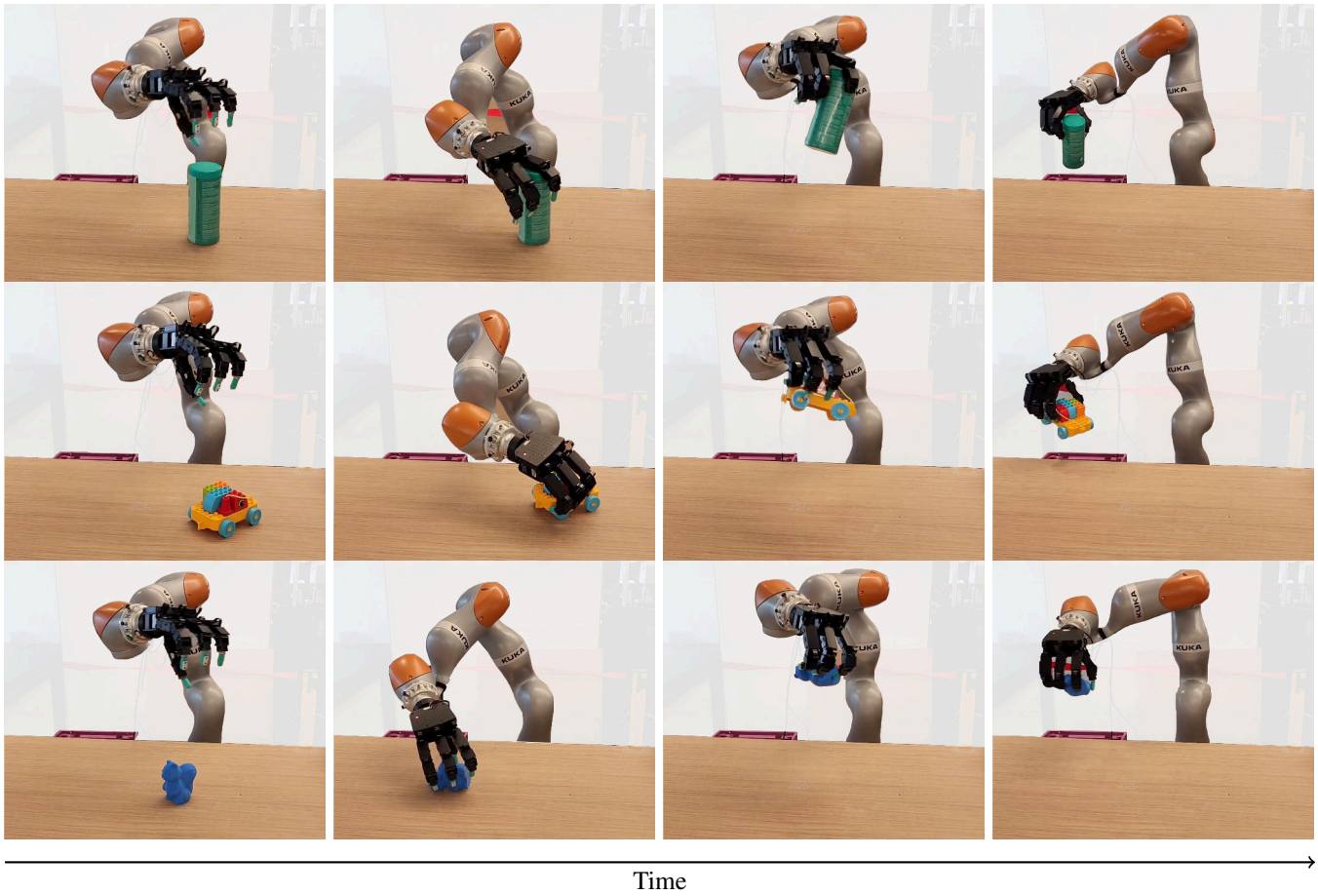


Fig. 1: DextrAH-RGB (Dexterous Arm-Hand RGB) is an end-to-end RGB-based policy that can dexterously grasp a wide variety of objects that were unseen during training.

***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 policy in simulation through reinforcement learning that acts on a geometric fabric controller to dexterously grasp a wide variety of objects. We then distill this into an RGB-based policy strictly in simulation using photorealistic tiled rendering. To our knowledge, this is the first work that is able to demonstrate robust sim-to-real transfer of an end-to-end (monocular or stereo) RGB-based policy for complex, dynamic, contact-rich tasks such as dexterous grasping with multi-fingered hands. Unlike previous methods, DextrAH-RGB requires no explicit depth or CAD models, making it significantly more practical and robust in varied real-world lighting and texture conditions. It generalizes to novel objects and scenes, offering a strong step toward deployable, vision-based dexterous manipulation.**

[1]NVIDIA, Santa Clara, CA, USA.
[2]University of California, Berkeley, Berkeley, CA, USA.

I. INTRODUCTION

Controlling multi-fingered robotic hands to naturally and quickly grasp objects is a longstanding challenge in robotics. Ideally, techniques must integrate visual, tactile, and proprioceptive inputs to generalize to novel objects and react dynamically in unstructured environments. Achieving this would allow dexterous robots to perform complex tasks in homes, factories, and outdoor settings.

Reinforcement learning (RL) in simulation has recently made impressive strides in manipulation and locomotion.

979-8-3315-8160-2/26/$31.00 ©2026 IEEE

High-throughput simulators and domain randomization have allowed reactive policies to scale, achieving successful sim-to-real transfer in some cases. However, most approaches to dexterous grasping either (i) factorize the problem into static grasp planning followed by motion execution — limiting reactivity and robustness — or (ii) depend heavily on depth sensing to infer object geometry. Depth-based methods struggle in real-world settings, particularly with translucent or reflective surfaces and under high dynamic range (HDR) lighting such as direct sunlight. Moreover, while end-to-end vision-based grasping policies exist for simpler grippers, extending this paradigm to high-DoF dexterous hands has remained elusive.

To address these challenges, we present DextrAH-RGB, a dexterous arm-hand grasping policy that computes high-rate actions directly from stereo RGB images. Unlike prior work, our method entirely avoids depth sensing or pose estimation, relying solely on raw RGB input. This allows DextrAH-RGB to operate in environments where depth maps are unreliable such as brightly lit rooms or with small, transparent/reflective objects. Our main contributions are: (1) simulation-only training of a state-based policy distilled into an RGB-based policy; (2) scaling a novel stereo vision architecture with cross-attention transformers to enable implicit depth inference purely from images; and (3) robust real-world deployment demonstrating safe, reliable, and reactive dexterous grasping without reliance on depth maps or object CAD models. Figure 1 shows a montage of our policy grasping objects unseen during training.

To our knowledge, this is the first system to achieve robust sim-to-real transfer of an end-to-end RGB-based policy for complex, dynamic, contact-rich dexterous manipulation. DextrAH-RGB generalizes to unseen objects, textures, and lighting conditions — including challenging HDR environments — and is readily extensible to arbitrary multi-camera configurations, paving the way for practical real-world deployable dexterous robots.

II. RELATED WORK

Classical Analytic Grasp Planning. Dexterous grasping has been studied extensively with a rich history of prior work. Classical methods typically involve optimizing analytic grasp metrics [1]–[3]. 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 [4].

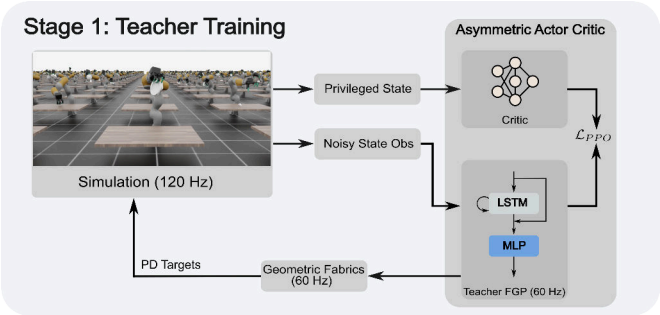
Learning-based Grasping with Depth or Pointclouds. Data driven methods to dexterous grasping involve leveraging grasp datasets to train grasp planning networks. Some examples of these datasets include MultiDex [5], DexGraspNet [6], Grasp'D-1M [7], and Get a Grip [4]. A lot of past work involves grasp synthesis based on pointcloud/depth information. Not only do they capture the geometry of the objects, but are also easier to simulate compared to high-fidelity RGB rendering. UniDexGrasp learns dexterous grasping policies from full pointclouds [8] and UniDexGrasp++ improved upon this method by introducing a geometry-aware curriculum [9]. Both of these works only demonstrate results in simulation. DexRepNet [10] learns a representation that combines spatial geometric features of the hand and object and demonstrate successful sim-to-real 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 sim-to-real transfer of their pointcloud policies. 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 [11]. Agarwal et al. [12] 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 [13] by reconstructing objects in simulation and retargeting human hand motions to robot hand actions. After pre-training, they are able to demonstrate impressive sim-to-real results by finetuning with PPO in simulation to train a policy conditioned on the depth image. The work most similar to ours is DextrAH-G [14], which also uses a student-teacher distillation framework but distills into a simpler depth-based policy. They first train a state-based teacher on the Visual Dexterity [15] dataset, then distill it into a depth-based student that achieves impressive sim-to-real transfer, even on unseen objects. However, their reliance on depth maps necessitates blocking sunlight and truncating depth readings to avoid sensor artifacts—significantly limiting generalization and robustness. In contrast, DextrAH-RGB requires no object CAD models, depth sensors, privileged pose estimation, or controlled lighting conditions, making it the most generalizable method. Other prior work such as [16] also do grasping from depth input, and inherit the same limitations as DextrAH-G.

End-to-End Visuomotor Policies for Simple Grippers. [17]–[21] demonstrated end-to-end RGB policies for grasping; however, they use parallel jaw grippers rather than dexterous hands which limits the types of objects that can be grasped (*e.g.* it is difficult to grasp large objects or objects with complex geometry with simple grippers). Furthermore, the action space is far simpler for parallel jaw grippers than for fully dexterous hands, thereby making the control problem easier.

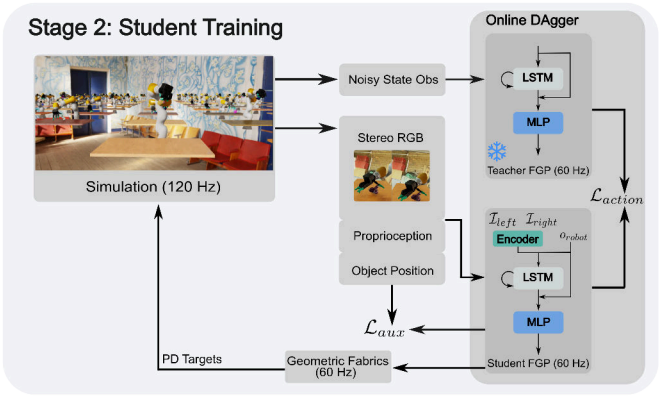
III. DEXTRAH-RGB

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. With this vectorized controller, we first train a state-based, teacher policy in simulation using reinforcement learning. We then use an online version of DAgger to distill the teacher into a student policy. Instead of receiving object state information,

the student receives either one or two RGB images from a set of cameras in a stereo configuration.



(a) Stage 1: Training a state-based teacher policy with PPO using asymmetric actor-critic. The actor receives noisy observations, while the critic has privileged (accurate) state data. An LSTM enables temporal context and dynamic adaptation.



(b) Stage 2: Distilling the teacher policy into a vision-based student policy using online DAgger [22]. The student minimizes the KL-divergence to teacher actions and includes an auxiliary object position prediction loss. Stereo embeddings combined with proprioceptive data are fed into the policy.

Fig. 2: Overview of the two-stage pipeline.

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

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) [23]. We create these behaviors through a combination of *geometric* and *forcing* fabric terms. Geometric terms specify 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 a policy complete a grasping task. For the purposes of this work, we use the same geometric fabric and action space as in DextrAH-G [14]. The action space comprises the 6-DoF pose of the palm and a reduced, 5-dimensional PCA action space for the hand.

B. State-based Teacher Policy Training

Due to the sample inefficiency of RL, we do not directly train an RGB-based policy from scratch. Instead, we first train a teacher policy which receives privileged state information and then distill the teacher policy into an RGB-based student policy. The teacher policy is trained via PPO. We also add skip connections, similar to those found in DenseNet [24], as previous work [25] has shown that dense connections are better for policy learning. The policy is tasked to lift up 140 objects from the Visual Dexterity Dataset [15] into the air. 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 previous 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 2a illustrates our teacher training pipeline.

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 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 [26], [27], 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 [27], 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 V for more details about the parameters ADR controls and their randomization ranges.

C. RGB Student Policy Training

To train the RGB-based policy, we leverage student-teacher distillation and use an online version of DAgger [22]. Our training pipeline for this stage is depicted in Figure 2b. 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 [28] simulation framework which offers ray-traced tiled rendering to allow for fast and realistic rendering in each environment.

To create realistic scenes, we follow a similar method as Synthetica [29]. 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. The objects we used initially came textureless, so we bind textures from random, everyday objects found in the Omniverse Asset Library. The resulting set of textured objects approximate the support of the wide distribution of real-world objects. Figure 3 compares the original textureless meshes with the textured versions. Along with these randomizations, we also add data augmentations such as random background, color jitter, and motion blur. Tables VI and IV contains more details about the randomization ranges and probabilities. During distillation, we set the ADR increment to the maximum in order to match the domain randomization ranges on which the teacher was trained. Figure 4 shows various examples of images rendered from the left camera.

Fig. 3: **(a)** Shows a subset of object meshes with no texture. **(b)** Shows meshes with random textures bound.

The student architecture is shown in Figure 2b. We adopt a stereo-based setup, as stereo policies outperform monocular ones in simulation (see ablations in Section IV-A). The policy receives stereo RGB images of size 320×240, processed through a stereo encoder illustrated in Figure 5a. Each image is independently fed through a pre-trained ResNet-18 backbone [30] in a Siamese manner with its last two layers removed. Outputs from the backbone are down-projected and reshaped into 128 tokens per image, each 128-dimensional. These tokens are passed into a two-layer transformer along with a learnable `[embed]` token. The transformer employs cross-attention, allowing tokens from one image to attend exclusively to tokens from the other image or the `[embed]` token (see mask in Figure 5b). The reason for this was because DUSt3R [31] demonstrated that cross-attention mechanisms can effectively recover 3D geometry from multi-view images, hence we adopted a similar architecture to allow for implicit depth inference. As shown in the ablations in Section IV-A, this was a key design decision that served to improve policy performance. The transformed `[embed]` token is then passed through an MLP, resulting in a 64-dimensional stereo embedding. This approach mirrors ViT [32], which uses a learnable `[CLS]` token to produce unbiased embeddings. The final embedding is concatenated with proprioceptive inputs before being fed into the policy. Compared to the teacher, the student replaces the object pose and one-hot encoding with stereo-RGB pairs.

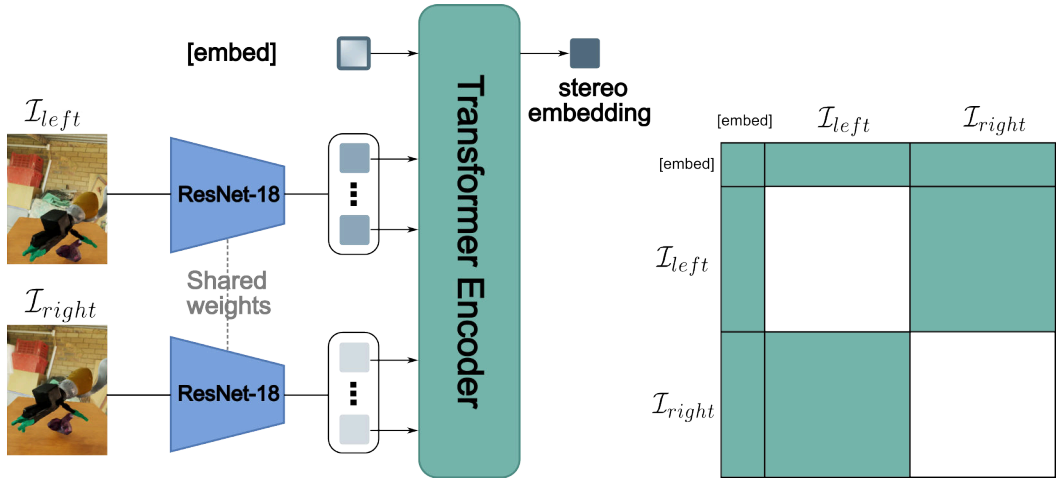
The student action space is the same as the teacher's. It is jointly supervised on the imitation loss and auxiliary loss with $\mathcal{L} = \mathcal{L}_{action} + \mathcal{L}_{aux}$, where $\mathcal{L}_{action} = D_{KL}(\pi_{student} \| \pi_{teacher})$, and $\mathcal{L}_{aux} = \|\hat{x}_{obj} - x_{obj}\|$. x_{obj} refers to the groundtruth object position and $\hat{x}_{obj}$ is the network's prediction of the object position. Unlike some prior works that use distillation [14], [33], we opt for a KL loss instead of an $l2$ loss on the actions because we noticed across 4 seeds, policies trained to explicitly minimize the KL divergence between the student and teacher always outperformed their $l2$ counterparts. We believe this is because the KL loss is more expressive as it provides a natural method to weigh errors in different dimensions of the action space.

IV. Experiments

A. Architecture Ablations

We conduct ablation studies to validate key design decisions. Specifically, we compare monocular and stereo policies, summarized in Table I. Performance is measured by the success rate relative to the teacher policy, averaged over three seeds. We also evaluate the impact of the auxiliary object position prediction loss and the transformer in the encoder. Stereo policies significantly outperform monocular ones, reducing positional error by nearly 1 cm due to improved depth perception. Incorporating the transformer further boosts stereo policy performance, highlighting its importance. Additionally, fine-tuning the pre-trained ResNet-18 encoder substantially outperforms using frozen or scratch-trained encoders. This highlights another benefit of RGB-

Fig. 4: 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.



(a) Stereo Encoder **(b)** Cross-Attention Mask

Fig. 5: (a) The stereo encoder uses a pre-trained ResNet-18 (with the last two layers removed) to encode each image independently into a high-dimensional vector. Each vector is projected and split into 128 tokens. Tokens from both images, along with a learnable `[embed]` token, are passed into a two-layer transformer that performs cross-attention. The output from the `[embed]` token is processed through an MLP, producing the final stereo embedding vector. **(b)** The turquoise regions illustrate cross-attention between the tokens. Each image's tokens attend to the other image's tokens and the shared `[embed]` token, which attends to all tokens.

based policies which is that they can leverage previously learned representations from pre-training on large datasets.

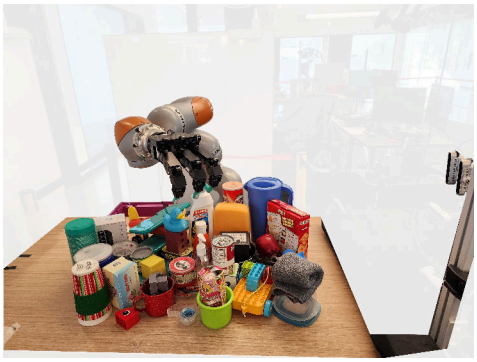
Model Type	Performance ↑	Pos Error (cm) ↓
Finetune ResNet Mono w/o Attn	0.83	3.5
Finetune ResNet Mono w/ Attn	0.83	3.3
Frozen ResNet Stereo w/ Attn	0.71	4.3
Scratch ResNet Stereo w/ Attn	0.83	3.2
Finetune ResNet Stereo w/o Attn	0.86	2.5
Finetune ResNet Stereo w/ Attn	**0.89**	**2.5**

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.

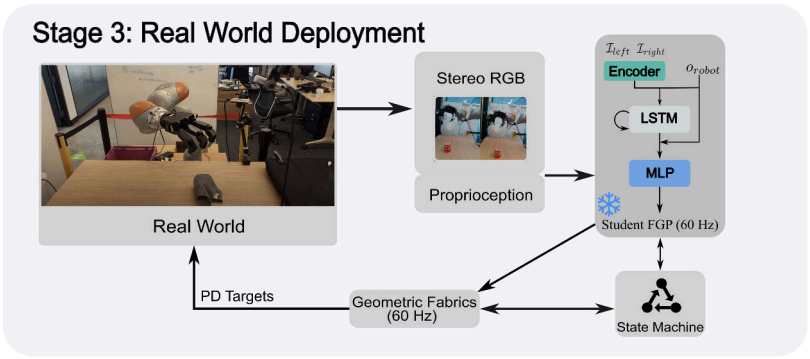
B. Real World Experiments

We deploy our policies in the real world on a 7 DoF Kuka LBR iiwa arm with a 16 DoF Allegro Hand v4 mounted on top. The setup of our robot system is shown in Figure 6a.

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 [37] 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. A grasp is considered successful if the object is lifted and held stably for two seconds. We run our policy continually until either the grasp succeeds, or we experience a failure from which the robot cannot recover. One of the



(a) Real robot setup



(b) State machine for bin packing

Fig. 6: (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.

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 [34]	-	60%	-	-	60%	80%	**100%**	-	-	-	20%
ISAGrasp [35]	-	60%	-	40%	-	80%	-	-	-	80%	-
Matak [36]	67%	**100%**	67%	-	0%	0%	**100%**	**100%**	0%	-	-

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

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; however, it uses depth maps instead of RGB (which is a more challenging setting). Furthermore, the depth maps for DextrAH-G were specially truncated to avoid background distractors and incoming sunlight was blocked to avoid eroding the quality of depth readings. DextrAH-RGB does not have this limitation and is thus a more generalizable and robust solution as it can work in many lighting conditions and background settings as shown in the next section and in the supplementary video.

Bin Packing Assessment: We also follow the bin packing assessment protocol in [14] to more holistically evaluate the grasping performance. In this assessment, the robot is tasked with continuously grasping 36 different objects (unseen during training) one-at-a-time and depositing them into a bin. The 36 objects varied widely in size, geometry, texture, and mass, ranging from small Lego bricks to large 2.5L pitchers. For this task, we employ a state machine which decides when to pass policy actions to the robot. Specifically, it queries the policy'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 move the arm to the bin and deposit the object. Afterwards, another fixed action is issued to return the robot to the forward position after which the policy is re-engaged. The full, real-world pipeline is shown in Figure 6b. 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 evaluate DextrAH-RGB (stereo) and DextrAH-RGB (mono) for this protocol. To evaluate generalizability, we also evaluate our policies under 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 stereo camera pair creating a bright background and dark foreground. Such settings were explictly avoided in DextrAH-G [14] since natural light erodes depth readings and ruins policy performance. We conducted three trials for each combination of lighting condition and vision modality for a total of 12 experiments.

The bin packing performance of DextrAH-RGB compared to the state-of-the-art DextrAH-G is presented in Table III. DextrAH-RGB notably reduces cycle times by 1-2 seconds

on average, approaching human-level speeds (3.63 s as estimated in [14]). Stereo policies consistently outperform their monocular counterparts in success rates. Although DextrAH-RGB's success rates are lower than those reported for DextrAH-G, our evaluation used a far more challenging and realistic setup, including distractor objects and varied lighting conditions. DextrAH-G's experiments were specifically conducted under highly controlled conditions, excluding background distractors and challenging lighting such as HDR scenarios. In their experiments (which do not have any HDR conditions), they have truncated depth maps which removes any external distractors. Thus, even our non-HDR experiments are still more challenging than the experiments conducted in DextrAH-G. Furthermore, we are able to show that our methods perform well in HDR scenarios and crucially, do not experience a significant drop in success rate, desipte any external distractors or varying lighting conditions. Thus, our direct end-to-end based method sets a new benchmark in state-of-the-art generalization for dexterous manipulation. The supplementary video showcases the robustness of our grasping policies, under both physical and visual perturbations.

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. 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 further present a novel end-to-end stereo architecture for our RGB policies. DextrAH-RGB demonstrates successful sim-to-real transfer of our end-to-end RGB policies. Furthermore, our methods do not suffer from similar limitations as depth policies and are better at generalizing to novel scenes. To our knowledge, this is the first work to demonstrate robust sim-to-real transfer of end-to-end RGB policies for such a complex task. 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

[1] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[2] M. T. Ciocarlie, C. Goldfeder, and P. K. Allen, "Dexterous grasping via eigengrasps : A low-dimensional approach to a high-complexity problem," 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:6853822

[3] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295 vol.3.

[4] T. G. W. Lum, A. H. Li, P. Culbertson, K. Srinivasan, A. D. Ames, M. Schwager, and J. Bohg, "Get a grip: Multi-finger grasp evaluation at scale enables robust sim-to-real transfer," 2024. [Online]. Available: https://arxiv.org/abs/2410.23701

[5] P. Li, T. Liu, Y. Li, Y. Geng, Y. Zhu, Y. Yang, and S. Huang, "Gendexgrasp: Generalizable dexterous grasping," 2023. [Online]. Available: https://arxiv.org/abs/2210.00722

[6] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," 2023. [Online]. Available: https://arxiv.org/abs/2210.02697

[7] D. Turpin, T. Zhong, S. Zhang, G. Zhu, J. Liu, R. Singh, E. Heiden, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, "Fast-grasp'd: Dexterous multi-finger grasp generation through differentiable simulation," 2023. [Online]. Available: https://arxiv.org/abs/2306.08132

[8] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, T. Liu, L. Yi, and H. Wang, "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," 2023. [Online]. Available: https://arxiv.org/abs/2303.00938

[9] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang, "Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning," 2023. [Online]. Available: https://arxiv.org/abs/2304.00464

[10] Q. Liu, Y. Cui, Q. Ye, Z. Sun, H. Li, G. Li, L. Shao, and J. Chen, "Dexrepnet: Learning dexterous robotic grasping network with geometric and spatial hand-object representations," 2023. [Online]. Available: https://arxiv.org/abs/2303.09806

[11] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, "Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation," 2022. [Online]. Available: https://arxiv.org/abs/2211.09423

[12] A. Agarwal, S. Uppal, K. Shaw, and D. Pathak, "Dexterous functional grasping," 2023. [Online]. Available: https://arxiv.org/abs/2312.02975

[13] H. G. Singh, A. Loquercio, C. Sferrazza, J. Wu, H. Qi, P. Abbeel, and J. Malik, "Hand-object interaction pretraining from videos," 2024. [Online]. Available: https://arxiv.org/abs/2409.08273

[14] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. V. Wyk, "DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics," 2024. [Online]. Available: https://arxiv.org/abs/2407.02274

[15] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand reorientation of novel and complex object shapes," *Science Robotics*, vol. 8, no. 84, Nov. 2023. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.adc9244

[16] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu, "Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids," 2025. [Online]. Available: https://arxiv.org/abs/2502.20396

[17] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," 2017. [Online]. Available: https://arxiv.org/abs/1707.02267

[18] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," 2018. [Online]. Available: https://arxiv.org/abs/1806.07851

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," 2016. [Online]. Available: https://arxiv.org/abs/1504.00702

[20] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," 2019. [Online]. Available: https://arxiv.org/abs/1812.07252

[21] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018. [Online]. Available: https://arxiv.org/abs/1806.10293

[22] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," 2011. [Online]. Available: https://arxiv.org/abs/1011.0686

[23] K. V. Wyk, A. Handa, V. Makoviychuk, Y. Guo, A. Allshire, and N. D. Ratliff, "Geometric fabrics: a safe guiding medium for policy learning," 2024. [Online]. Available: https://arxiv.org/abs/2405.02250
[24] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018. [Online]. Available: https://arxiv.org/abs/1608.06993
[25] S. Sinha, H. Bharadhwaj, A. Srinivas, and A. Garg, "D2rl: Deep dense architectures in reinforcement learning," 2020. [Online]. Available: https://arxiv.org/abs/2010.09163
[26] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," 2019. [Online]. Available: https://arxiv.org/abs/1910.07113
[27] A. Handa, A. Allshire, V. Makoviychuk, A. Petrenko, R. Singh, J. Liu, D. Makoviichuk, K. V. Wyk, A. Zhurkevich, B. Sundaralingam, Y. Narang, J.-F. Lafleche, D. Fox, and G. State, "Dextreme: Transfer of agile in-hand manipulation from simulation to reality," 2024. [Online]. Available: https://arxiv.org/abs/2210.13702
[28] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.
[29] R. Singh, J. Liu, K. V. Wyk, Y.-W. Chao, J.-F. Lafleche, F. Shkurti, N. Ratliff, and A. Handa, "Synthetica: Large scale synthetic data for robot perception," 2024. [Online]. Available: https://arxiv.org/abs/2410.21153
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385
[31] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," 2024. [Online]. Available: https://arxiv.org/abs/2312.14132
[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929
[33] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986
[34] Z. Weng, H. Lu, D. Kragic, and J. Lundell, "Dexdiffuser: Generating dexterous grasps with diffusion models," 2024. [Online]. Available: https://arxiv.org/abs/2402.02989
[35] Z. Q. Chen, K. V. Wyk, Y.-W. Chao, W. Yang, A. Mousavian, A. Gupta, and D. Fox, "Learning robust real-world dexterous grasping policies via implicit shape augmentation," 2022. [Online]. Available: https://arxiv.org/abs/2210.13638
[36] M. Matak and T. Hermans, "Planning visual-tactile precision grasps via complementary use of vision and touch," 2022. [Online]. Available: https://arxiv.org/abs/2212.08604
[37] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, p. 36–52, Sep. 2015. [Online]. Available: http://dx.doi.org/10.1109/MRA.2015.2448951

APPENDIX

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

TABLE IV: Various data augmentations and their associated probabilities

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 V: 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 VI: Various visual domain randomization parameters and their probability distributions.