End-to-end RL Improves Dexterous Grasping Policies

Ritvik Singh[1,2], Karl Van Wyk[1], Jitendra Malik[2], Pieter Abbeel[2], Nathan Ratliff[1], Ankur Handa[1]

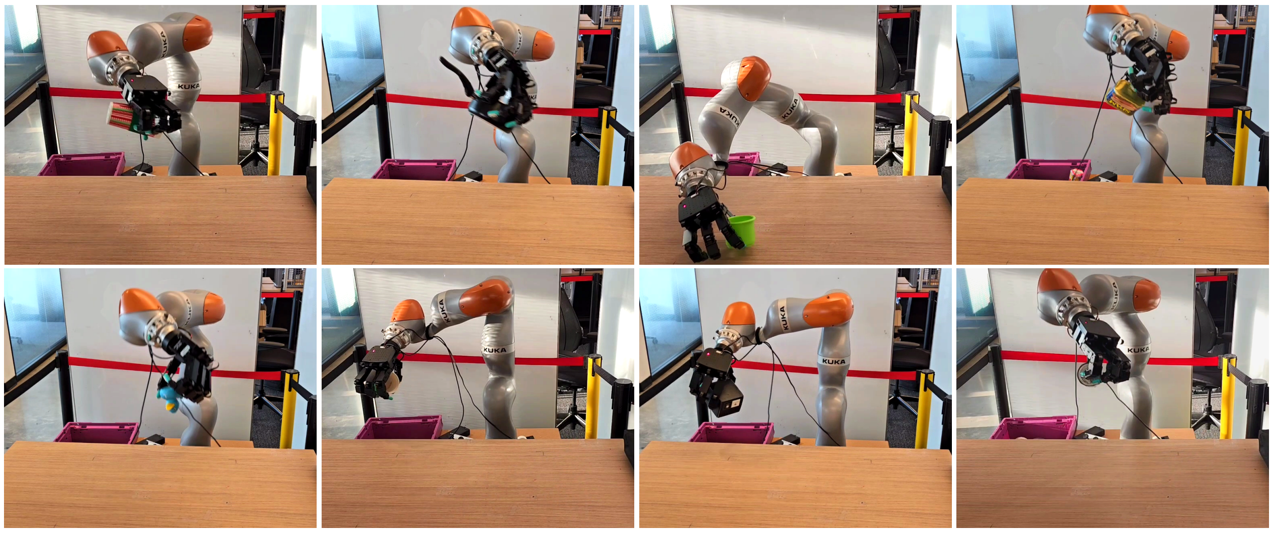


Fig. 1: We deploy our policy in the real world with varying lighting conditions. All of the objects in the above figure were unseen during training.

***Abstract*— This work explores techniques to scale up image-based end-to-end learning for dexterous grasping with an arm + hand system. Unlike state-based RL, vision-based RL is much more memory inefficient, resulting in relatively low batch sizes, which is not amenable for algorithms like PPO. Nevertheless, it is still an attractive method as unlike the more commonly used techniques which distill state-based policies into vision networks, end-to-end RL can allow for emergent active vision behaviors. We identify a key bottleneck in training these policies is the way most existing simulators scale to multiple GPUs using traditional data parallelism techniques. We propose a new method where we disaggregate the simulator and RL (both training and experience buffers) onto separate GPUs. On a node with four GPUs, we have the simulator running on three of them, and PPO running on the fourth. We are able to show that with the same number of GPUs, we can double the number of existing environments compared to the previous baseline of standard data parallelism. This allows us to train vision-based environments, end-to-end with depth, which were previously performing far worse with the baseline. We train and distill both depth and state-based policies into stereo RGB networks and show that depth distillation leads to better results, both in simulation and reality. This improvement is likely due to the observability gap between state and vision policies which does not exist when distilling depth policies into stereo RGB. We further show that the increased batch size brought about by disaggregated simulation also improves real world performance. When deploying in the real world, we improve upon the previous state-of-the-art vision-based results using our end-to-end policies. More information can be found at `https://e2e4robotics.com/`.**

[1]NVIDIA
[2]University of California, Berkeley

I. INTRODUCTION

In robotics, crafting behaviours which exhibit a mix of agility, reactivity, and dexterity when interacting with the environment remains a longstanding goal. Recently, reinforcement and imitation learning have emerged as powerful paradigms for the training of robot policies. For the vast majority of everyday tasks, endowing policies with vision is a required component to sense the environment and achieve the desired behaviour. In recent years, visuomotor policy learning - learning policies which take in a mix of visual and proprioceptive inputs - has emerged as a powerful paradigm for the creation of robotic policies for a wide variety of tasks [1, 2]. As part of this, so-called end-to-end methods of direct processing of image to action have emerged as a useful technique for representing policies, sidestepping the need for explicit representations of the environment state [3]. Such methods usually take in raw RGB or depth camera observations, and produce an action output that is the result of processing using a neural network.

One popular approach to producing visual policies is via distillation in simulation [4, 5, 6, 7, 8, 9]. In such approaches, an expert policy, commonly known as the "teacher", is learned, usually via Reinforcement Learning

(RL), and then a downstream "student" policy is learned by rolling out the policy and supervising its actions based on the expert policy's output. The advantage of this method is that the student and the teacher do not need to have the same inputs, allowing the practitioner great flexibility in designing the input space and network architecture for both the teacher RL training (where it is possible to include privileged information only available in the simulator, e.g. ground truth states) and in the student during distillation. This way, the sample inefficiency of RL is limited to just the privileged teacher which is much easier than visual RL where the state space is far more complex. This leads to a "factorization" of the visuomotor learning process into behavior learning in the first stage and representation learning in the second. While convenient, this approach has several limitations. Namely, it results in vision-based policies that learn state-based behaviors. This is problematic because this can create a partial observability problem whereby the student has difficulty replicating the teacher actions.

A solution to this would be to simply train the vision policy with RL. While this has been employed for primitive grasping tasks [3, 10], it has yet to be done for complex, dexterous manipulation tasks as the sample complexity in pixel space is a lot higher. This requires scaling up the number of simulated environments. However, simulating batched rendering environments in parallel is very memory intensive. This means that without access to a large number of GPUs, it becomes difficult to scale up the number of environments in order to train such vision policies from scratch with RL. In order to more efficiently utilize GPU memory, we take inspiration from the disaggregated prefill and decode setups that are used for modern day LLM inferencing [11]. We introduce a disaggregated simulation and RL framework, which separates RL experience buffers/training and simulation environments onto separate GPUs. This allows us to simulate twice as many environments compared to the previous data parallelism baseline on the exact same hardware setup. With this, we can train end-to-end depth policies for a Kuka-Allegro robot setup to perform the task of dexterous grasping. We distill these depth policies into RGB policies which avoids the teacher-student distillation gap previously seen with state-based teachers. Ultimately, we find that in both simulation and the real world, policies distilled from vision-based teachers are more performant than policies distilled from state-based ones. Figure 1 shows snippets of the robot grasping various objects unseen during training.

II. Related Work

Scaling frameworks for vision-based RL. There are many existing simulators for robot learning that support rendering such as Isaac Gym [12], Isaac Lab [13], ManiSkill [14], and MuJoCo [15]. However, the current paradigm with which they scale up RL is by using data parallelism, which can be inefficient when it comes to memory use as shown in Section III-B. Impala [16] is a scalable distributed RL framework which hosts the learner on a separate machine from the environment. The limitation with this approach is that they also store the trajectory on the same device that contains the environment which becomes expensive for vectorized simulators. When applied to modern robotics simulators, this would result in large experience buffers that cannot fit on the same GPU without lowering the number of environments. SEED RL [17] is a follow up work which stores the trajectory on the same device as the learner. However, their implementation is not tailored towards modern robotics simulators which are vectorized for GPUs as they built their distributed RL infrastructure around large CPU clusters. Concretely, this means that their framework is used in scenarios with very large number of CPUs, each one simulating a few environments, whereas our method for disaggregated simulation is meant to run on GPU-accelerated simulators which require few replicas as each GPU can simulate many more environments. Lastly, they also do not demonstrate end-to-end learning on robotics tasks that transfer to the real world.

Vision Based Grasping with Hands. There have been several prior works for vision-based grasping with multi-fingered hands. They can either learn from depth/pointclouds [8, 18, 19] or through RGB [9]. While the various implementation details differ among the papers, the one common aspect is that they all leverage a teacher-student distillation pipeline. The teacher policies all operate on privileged information which is then distilled into vision policies that operate in the real world. This means that vision policies are trained to mimic the behavior of privileged policies, which can ultimately lead to suboptimal policy performance. This is because the inability to reproduce teacher actions can result in trajectories that diverge from those of the teacher, which results in distributional shifts that can lead to unrecoverable failures. Although there has been previous work aiming to reduce this partial observability gap [20, 21], there has not been much focus investigating this for complex environments such as vision-based dexterous grasping.

End-to-End Visuomotor Policies for Parallel Jaw Grippers. Several works in the past have demonstrated end-to-end grasping from RL using simple parallel jaw grippers [22, 23, 3, 10, 24]. The reduced action space brought by parallel jaw grippers makes the RL process much easier than using hands. However, this morphology can be limiting because they cannot achieve the same types of stable grasps as multifingered hands [25].

III. Method

A. End-to-End RL

Vision-based policies are incredibly important for real-world manipulation. However, training them directly from RL has historically been challenging due to the high sample complexity of image space. This has led to two-stage methods gaining prominence whereby a state-based teacher policy is trained with RL and then distilled into a vision-based student policy. While this has led to training successful policies, they fundamentally do not learn vision-aware behaviors. For example, imagine a robot arm trying to grasp an object. Suppose the arm is currently occluding said object. The teacher policy, which has access to the groundtruth object position, will simply just pick it up. However, the student policy may struggle to recreate this behavior as it has not learned to move the arm out of the way in order to see the object. In such cases, the student is trying to mimic state-based behaviors while only having access to vision information which causes it to act sub-optimally with respect to its inputs. Therefore, end-to-end training, where the RL policy learns directly from images, will lead to policy behaviors that lend themselves better to their sensory modalities. However, RGB end-to-end RL is much slower than depth-based RL as the rendering process for accurate light transport simulation is far more time consuming. Thus, a suitable middle ground that meets the requirements above while also being able to run on a reasonable hardware budget is to train a depth based policy with RL, and distill this into a stereo RGB-based policy in order to deploy in the real world. This way, there is no theoretical information gap between the student and the teacher.

B. Disaggregated Simulation and RL

When training end-to-end RL, it is important to scale up the number of environments in order to get a reliable learning signal. Scaling up reinforcement learning is typically done through naive data parallelism. This means that every GPU runs the simulator, stores the RL experience buffers, and computes the gradients for the actor and critic. After each instance calculates the gradients, they are averaged across all GPUs in order to obtain a less-noisy gradient estimate (see Fig 2a). When performing end-to-end vision RL, the experience buffers can balloon in size. For example, assuming depth maps are represented using standard fp32 precision, storing the experience across just 512 environments at 320x240 resolution for a relatively short horizon length of 16 steps can take up to 2.5GB alone. The simulator will also start to consume more memory as more environments are added. Crucially, however; the simulator will take up a non-trivial amount of memory that is not a function of the number of environments which is typically used for the asset cache. Therefore, naively running the simulator on every GPU is a suboptimal use of memory as each copy will bring with it the same large asset cache.

Algorithm 1 Simulation Replica (GPU $s \in \{0,1,2\}$)

```
1: Given: learner id $\ell \leftarrow 3$, environment Env
2: obs ← Env.Reset()
3: SendTo($\ell$, obs)              ▷ send initial obs to learner
4: while true do
5:     actions ← Recv($\ell$)
6:     (obs, rew, dones) ← Env.Step(actions)
7:     SendTo($\ell$, (rew, dones, obs))
```

A more optimal use of memory would involve limiting the amount of GPUs that run the simulation, and ensuring that the ones that are running simulation do not have to also store RL experience buffers or network gradients. We present our proposed disaggregated simulation and RL setup in Fig 2b. In a node of 4 GPUs, we run the simulator on 3 of them, in order to fully maximize the number of environments we can simulate, and on the 4th GPU, we run RL training and store the experience buffers. Psuedocode for how the different GPUs communicate with each other is shown in Algorithms 1 and 2.

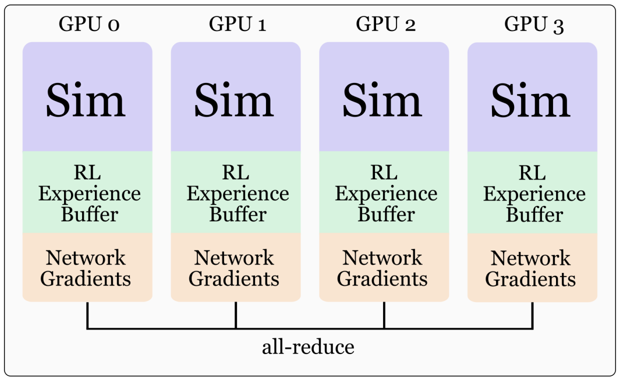
It is important to maximize the number of environments because unlike traditional state-based environments which can be scaled up to the point of PPO saturation long before the GPU is out of memory, vision environments can be far more memory intensive. For example, the state-based DextrAH task [8, 9], which involves object grasping with a Kuka and Allegro hand-arm system, can be reliably solved with 4096 environments which only takes up 14GB of memory. In contrast, the same task, when using 320x240 depth cameras, will take up 44GB just to simulate 256 environments. Thus, maximizing the number of environments becomes crucial when trying to solve vision tasks with RL. Our disaggregated method, as shown in Table I, is able to double the number of environments that are able to be simulated on the exact same hardware.

TABLE I: Maximum concurrent environments on one 4-GPU NVIDIA L40S node at various resolutions. Disaggregated Simulation is able to more than double the number of simulated environments on the same hardware compared to traditional data parallelism.

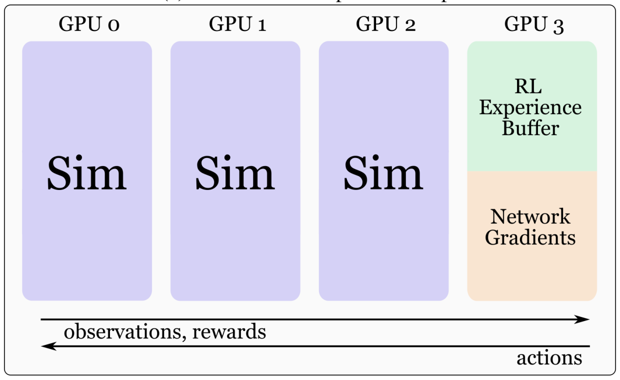
Input resolution	Data Parallel	Disaggregated Simulation
160×120	1024 / GPU (4096 total)	2800 / GPU (8400 total)
320×240	256 / GPU (1024 total)	700 / GPU (2100 total)

C. Training Environment

We use the same environment as in DextrAH-RGB [9] and provide a brief summary of it here. The environment



(a) Traditional data-parallel setup.



(b) Disaggregated RL setup.

Fig. 2: Disaggregated simulation/learning pipeline. (a) The standard data parallel setup when scaling up where every GPU runs both simulation and RL; (b) our disaggregated simulation setup where three GPUs purely run training and the last one stores the experience buffers and runs RL.

consists of a 7 DoF Kuka iiwa arm and a 16 DoF Allegro V4 hand. The policy is trained in simulation to grasp and lift 140 different objects from the Visual Dexterity dataset [26]. To help facilitate sim-to-real transfer, we employ domain randomization on a set of physics parameters such as joint friction, stiffness, damping, mass, etc. This is implemented through a method known as Automatic Domain Randomization (ADR) [27, 9] which sets an initial randomization range for the various physics parameters and gradually increases the range towards the terminal range as the policy becomes more proficient at grasping and lifting the object. This is done to induce a curriculum onto the environment that balances exploration early on in training while ensuring robustness of the policy.

We train our depth-based policies end-to-end with PPO [28]. Our network architecture comprises a 4-layer CNN with $[16, 32, 64, 128]$ filters, layer normalization, and ReLU activation function. The output of this is passed into a fully connected layer which outputs a 32 dimensional embedding for the depth. This is combined with the proprioception of the robot and fed

Algorithm 2 Learner / Trainer (GPU 3)

```
1:  Given: sim ids $\mathcal{S} \leftarrow \{0, 1, 2\}$, horizon $H$, policy $\pi$
2:  for all $s \in \mathcal{S}$ do
3:      obs[$s$] ← RECV($s$)                                  ▷ initial obs
4:  while true do
5:      $\mathcal{D} \leftarrow \emptyset$                     ▷ trajectory buffer
6:      for $t = 1$ to $H$ do
7:          actions ← $\pi\big(\text{stack}(\{\text{obs}[s]\}_{s\in\mathcal{S}})\big)$
8:          for all $s \in \mathcal{S}$ do
9:              SENDTO($s$, actions[$s$])
10:             rew[$s$], dones[$s$], nextObs[$s$] ← RECV($s$)
11:         $\mathcal{D} \leftarrow D \bigcup \{(\text{obs, actions, rew, done})\}$
12:         obs ← nextObs
13:     TRAINPPO($\pi, \mathcal{D}$)
```

into two LSTM layers with 1024 units, the output of which is fed into a 3 layer fully-connected network with $[512, 512, 256]$ hidden units. This is in contrast to IMPALA [16] where the fully connected network is placed before the LSTM. The architecture for the policy is shown in Figure 3.

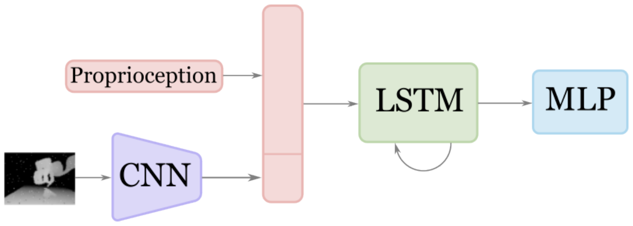


Fig. 3: The image is first passed into a 4-layer CNN which outputs a 32-dimensional embedding. This is concatenated with the rest of the robot proprioceptive data and then fed through to the LSTM and then MLP.

We choose to not train direct end-to-end RGB policies as realistic RGB rendering can be significantly more time consuming than simple tiled depth rendering. However, we still want to retain the benefits of RGB over depth as shown in [9]. Thus, we chose to distill the depth policy into a stereo RGB policy with the same architecture as [9]. The overall pipeline is shown in Figure 4. Because depth is recoverable from a stereo RGB pair, the behaviors learned by a depth policy can theoretically be completely replicated by a distilled stereo RGB policy. This is in contrast with trying to imitate the behavior of a state-based policy which has a much larger observation gap. As shown in Section IV our method leads to improved performance in both simulation and reality.

IV. RESULTS

We perform various experiments to demonstrate the benefit of end-to-end RL to produce vision-based teachers over the traditional factorization of state-based teachers

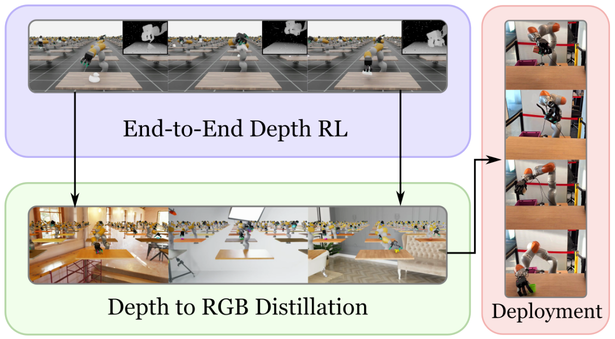


Fig. 4: We first train our depth-based teacher policies end-to-end using reinforcement learning. We then distill these depth policies into stereo RGB. Lastly, these policies are then deployed into the real world.

TABLE II: Data Parallel (DP) vs Disaggregated Simulation (Disagg) at 160×120 and 320×240 runs averaged across five seeds. The first metric is the portion of the terminal DR ranges that were achieved by ADR. The second metric is percentage of seeds that reached the terminal DR ranges. The last metric is the success rate, which measures the portion of environments that have the object grasped in the air (note: if the object has been grasped for 10 seconds, the environment is reset and the object is placed back onto the table).

Res.	Method	ADR Inc. ↑	% Full ADR ↑	SR ↑
160×120	DP	0.38	20%	0.37
	Disagg	**1.0**	**100%**	**0.42**
320×240	DP	0.0	0%	0.00
	Disagg	**0.90**	**20%**	**0.35**

and vision-based students. We empirically verify both in simulation and in reality the benefit of end-to-end RL on success rates. We further show that our design choice of disaggregated simulation helps improve batch size and thus, downstream performance. All experiments are done on a 4-GPU NVIDIA L40S node.

A. *Disaggregated Simulation vs Data Parallelism*

We verify the efficacy of our disaggregated simulation setup by comparing it with data parallelism. For each type of parallelism method, we run experiments at two sets of resolutions: 160×120 and 320×240. For each experiment, we run five different seeds and take the average. For each resolution, we use the corresponding number of environments as reported in Table I. The results of this experiment are shown in Table II. The first metric we report is the average portion of the terminal domain randomization (DR) ranges that was achieved by ADR. Since ADR increases the DR range towards the terminal range when the success rate is at least 0.4, a higher value indicates that the policy was more performant, causing ADR to increase the ranges more. The second metric we track is the percentage of runs that managed to reach the terminal ADR range. This metric measures how reliable the training setup is in producing performant policies. The last metric that we report is the average success rate for all seeds that reached the terminal ADR range. The success rate is defined as the percentage of objects that have been grasped. After being held in the air for 10 seconds, the environments will then reset. For the 320×240 data parallel case, none of the seeds reached the terminal DR range, so we report zero.

We see that across both resolutions, our disaggregated simulation setup is able to offer superior performance across all metrics when compared to the standard data-parallelism solution that the other simulators utilize. For the 160×120 resolution, not only are runs more likely to reach their terminal DR ranges, but once they do, they also display better grasping performance. For 320×240, it was impossible to train any policy to grasp the objects when using data parallelism. This is likely because the number of simulated environments goes down by a factor of four which significantly constrains the PPO batch size. Although our method is unable to reliably train policies to reach their terminal ranges at higher resolutions, it is still able to significantly improve over the baseline. These results show that existing end-to-end tasks are heavily constrained by the number of environments that can be simulated. Thus, our method, which doubles the number of environments that can be simulated on the same hardware, directly contributes to the improved performance of end-to-end vision-based RL policy training.

B. *Distilling State Teachers vs Depth Teachers*

In order to test the hypothesis that vision teachers offer superior performance to state teachers, we distill both into stereo RGB students to compare the policy performance. For each modality, we train 3 different seeds. The depth teachers were trained as described in Section III-C with disaggregated simulation at 160×120 resolution to maximize the batch size for RL. The state teachers were trained similar to previous work [8, 9]. In those papers, the state teachers were given access to the object pose and a one-hot vector that corresponds to the object category as the observation. The success metric is the percentage of objects that have been grasped into the air. It is important to note that this is an instantaneous metric and is not the success rate of picking up all objects. Rather, this instantaneously measures how many environments have lifted the object successfully in the air. Once the object has been lifted in the air for two seconds, the environment resets. A higher instantaneous success rate means that the policy is faster and more adept at grasping the object. The results of this experiment are shown in Figure 5. It is clear from the plot that vision-based teachers lead to better performance than state-based

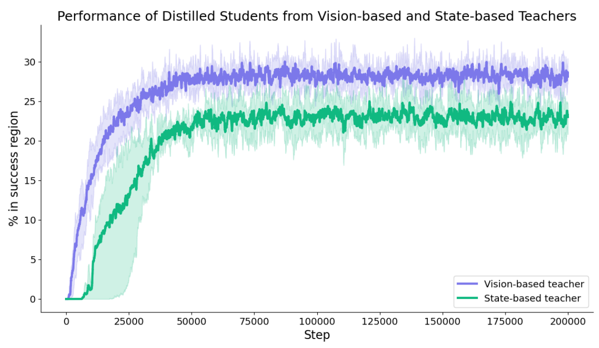


Fig. 5: The average performance of RGB students distilled from vision-based teacher policies (blue) and state-based teacher policies (red). It is clear from the plot that students perform better when the teacher policy is also vision-based.

ones. This is because there is less of an information asymmetry between the student and the teacher when it also has to operate on vision input. This leads to the students learning behaviors that are more congruent with their modality. For example, with a depth teacher, the student can learn to better deal with occlusions so as to manipulate the object without significantly occluding it from view of the camera.

C. Real World Benchmarking

We follow the bin packing evaluation protocol used in [8, 9]. This benchmarking protocol involves placing 30 objects of varying size, shape, and weight on the table and having the policy grasp it. The goal of this is to assess the continuous performance of this task. We leverage the same state-machine as prior work. In this, the distilled policy also has an extra MLP head that predicts the position of the object. During rollout, when the object prediction head predicts the object to not be sufficiently lifted in the air, the state machine executes the policy's actions onto the real robot in order to grasp the object into the air. Once the predicted object position is sufficiently high, the state-machine transitions to a fixed motion to deposit the object in a bin before resetting the pose of the arm and running the policy again. The main metric we track is the success rate, which measures what percentage of objects were successfully grasped and deposited into the bin.

The results of this experiment are shown in Table III. The models trained from depth teachers outperformed all other models using state-based teachers, demonstrating that existing policies are often held back by the observability gap between the student and teachers. This shows end-to-end RL as a promising step towards improving robot policies. Furthermore, our policy with disaggregated simulation performed the best, which shows that increasing the batch size helps these vision policies. This is likely because tiled rendering environments are incredibly memory intensive which results in us saturating GPU memory before saturating PPO with a very large batch size.

TABLE III: Success rate by model.

Model	**Success Rate** ↑
DextrAH-G (state teacher)	87%
DextrAH-RGB (state teacher)	77%
Ours (depth teacher)	87%
Ours (depth teacher, disagg)	**93%**

V. Conclusion

In this work, we demonstrate a method for end-to-end reinforcement learning with depth for dexterous grasping. Contrary to the standard paradigm of training state-based policies with RL and distilling them into vision-based ones, we are able to train depth policies with RL and distill them into RGB policies. We show that the lack of an information gap when distilling these depth policies results in better performance both in simulation and real-world experiment. We also present a method for efficiently training end-to-end RL policies with a low number of GPUs.

References

[1] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," 2024. [Online]. Available: https://arxiv.org/abs/2402.10329 1

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023. [Online]. Available: https://arxiv.org/abs/2304.13705 1

[3] 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 1, 2

[4] 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 1

[5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, Jan. 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abk2822 1

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots,"

2021. [Online]. Available: https://arxiv.org/abs/2107.04034 1

[7] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, “Legged locomotion in challenging terrains using egocentric vision,” 2022. [Online]. Available: https://arxiv.org/abs/2211.07638 1

[8] 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 1, 2, 3, 5, 6

[9] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. V. Wyk, “Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands,” 2025. [Online]. Available: https://arxiv.org/abs/2412.01791 1, 2, 3, 4, 5, 6

[10] 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 2

[11] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, “Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving,” 2024. [Online]. Available: https://arxiv.org/abs/2401.09670 2

[12] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, “Isaac gym: High performance gpu-based physics simulation for robot learning,” 2021. [Online]. Available: https://arxiv.org/abs/2108.10470 2

[13] 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. 2

[14] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, “Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai,” 2025. [Online]. Available: https://arxiv.org/abs/2410.00425 2

[15] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, “Mujoco playground,” 2025. [Online]. Available: https://arxiv.org/abs/2502.08844 2

[16] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, “Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures,” 2018. [Online]. Available: https://arxiv.org/abs/1802.01561 2, 4

[17] L. Espeholt, R. Marinier, P. Stanczyk, K. Wang, and M. Michalski, “Seed rl: Scalable and efficient deep-rl with accelerated central inference,” 2020. [Online]. Available: https://arxiv.org/abs/1910.06591 2

[18] H. Zhang, Z. Wu, L. Huang, S. Christen, and J. Song, “Robustdexgrasp: Robust dexterous grasping of general objects,” 2025. [Online]. Available: https://arxiv.org/abs/2504.05287 2

[19] 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 2

[20] S. Levine and P. Abbeel, “Learning neural network policies with guided policy search under unknown dynamics,” in *Neural Information Processing Systems (NIPS)*, 2014. 2

[21] Y. Kim, N. Chin, A. Vasudev, and S. Choudhury, “Distilling realizable students from unrealizable teachers,” 2025. [Online]. Available: https://arxiv.org/abs/2505.09546 2

[22] 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 2

[23] 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 2

[24] 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 2

[25] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik, “Learning visuotactile skills with two multifingered hands,” 2024. [Online]. Available: https://arxiv.org/abs/2404.16823 2

[26] 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 4

[27] 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 4

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347 4