Robust Visual Sim-to-Real Transfer for Robotic Manipulation

Ricardo Garcia, Robin Strudel, Shizhe Chen, Etienne Arlaud, Ivan Laptev, Cordelia Schmid

Robust visual sim-to-real transfer for robotic manipulation

Ricardo Garcia[1], Robin Strudel[1], Shizhe Chen[1], Etienne Arlaud[1], Ivan Laptev[1] and Cordelia Schmid[1]

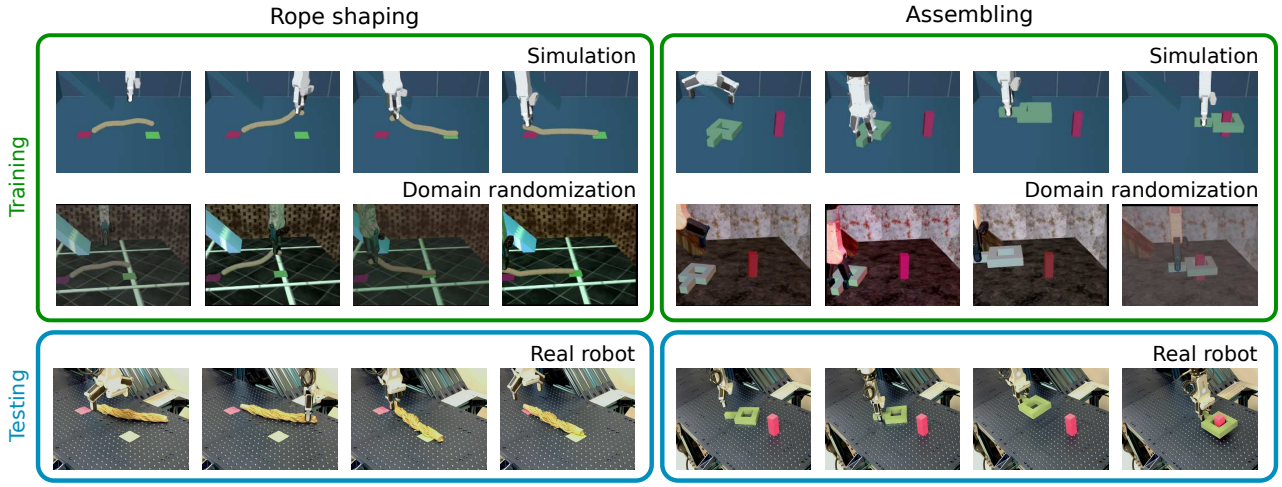


Fig. 1: **An overview of our approach.** We learn visuomotor manipulation policies in simulation (row 1) with domain randomization by sampling high-quality textures, lighting, object colors and camera parameters (row 2). We analyze different sampling options and demonstrate that simulator-trained policies can be directly deployed on a real robot for diverse and challenging manipulation tasks (row 3), such as rope-shaping (left) and assembling (right).

***Abstract*— Learning visuomotor policies in simulation is much safer and cheaper than in the real world. However, due to discrepancies between the simulated and real data, simulator-trained policies often fail when transferred to real robots. One common approach to bridge the visual sim-to-real domain gap is domain randomization (DR). While previous work mainly evaluates DR for disembodied tasks, such as pose estimation and object detection, here we systematically explore visual domain randomization methods and benchmark them on a rich set of challenging robotic manipulation tasks. In particular, we propose an off-line proxy task of cube localization to select DR parameters for texture randomization, lighting randomization, variations of object colors and camera parameters. Notably, we demonstrate that DR parameters have similar impact on our off-line proxy task and on-line policies. We, hence, use off-line optimized DR parameters to train visuomotor policies in simulation and directly apply such policies to a real robot. Our approach achieves 93% success rate on average when tested on a diverse set of challenging manipulation tasks. Moreover, we evaluate the robustness of policies to visual variations in real scenes and show that our simulator-trained policies outperform policies learned using real but limited data. Code, simulation environment, real robot datasets and trained models are available at `https://www.di.ens.fr/willow/research/robust_s2r/`.**

I. Introduction

Data-driven policy learning allows to acquire reactive robotic skills and has recently shown impressive results both in simulation and in the real world for tasks such as dexterous manipulation [1], [2], [3], [4], robotic arm manipulation [5], [6], [7], quadruped locomotion [8], [9], [10], [11] and navigation [12], [13], [14], [15]. Despite much progress in recent years, deploying simulator-trained policies in the real world remains challenging due to the visual and physical sim-to-real gap [1], [3], [8], [16], [10]. As result, current methods often rely on real-robot training data [6], [7], [16], [17], [18] and are hard to scale due to the prohibitive cost of large-scale real data collection.

To reduce the visual sim-to-real gap, existing methods typically apply domain adaptation (DA) [16], [19], [20] or domain randomization (DR) [3], [6], [21], [22] techniques. While DA attempts to create more realistic synthetic images, DR merely randomizes scene parameters such as textures and lighting, and is more common due to its simplicity and good performance in practice. Nevertheless, most existing DR methods study sim-to-real transfer for disembodied tasks such as pose estimation [21], [23] and object detection [24] as they are easier to evaluate than real robot tasks. Hence, it remains unclear (i) whether DR generalizes well to diverse and challenging robotic tasks, (ii) how to select optimal DR parameters for learning robotic policies, and (iii) how robotic policies trained with DR in simulation compare to policies trained on real data when tested in real scenes with varying visual appearance.

In this work we address the above questions and systematically study visual domain randomization on a suite of seven challenging robotic manipulation tasks, see Figure 4. Our initial experiments show that policies trained in simulation without adaptation fail and achieve zero accuracy when transferred to a real robot. To improve sim-to-real transfer, we propose a cube localization proxy task and use it for off-

[1]Inria, École normale supérieure, CNRS, PSL Research University, 75005, Paris, France.

line optimization of DR parameters such as textures, lighting, object colors and camera parameters. We then use DR with the obtained set of parameters and train policies for seven robotic manipulation tasks, namely stacking, box-retrieving, assembling, pushing, pushing-to-pick, sweeping and rope-shaping, see examples in Figure 1.

As one of our main contributions, we demonstrate that improvements on our proxy task consistently transfer to all considered manipulation policies under the same DR parametrization. Our policies trained on simulation-only data using off-line optimized DR parameters achieve 93% average success rate on seven real-world manipulation tasks. Moreover, we show that our simulation-trained policies demonstrate robustness to appearance variations of real scenes and outperform policies trained on real but limited data.

In summary, our contributions are three-fold: (i) We propose a proxy task and demonstrate that DR parameters optimized on this task generalize to a rich set of manipulation policies. (ii) We present a diverse set of seven manipulation tasks and use them to benchmark sim-to-real policy transfer. (iii) We ablate and show high success rate for policies trained only in simulation across a diverse set of manipulation tasks on a real robot. We also demonstrate our approach to outperform policies trained on real but limited data.

II. Related Work

The discrepancies between simulation and the real world mainly come from the visual appearance of the scene [25], [26], [27] and the physical dynamics [28], [29], [30], [31]. Here, we focus on reducing the visual sim-to-real gap and review the two dominant approaches, namely domain adaptation and domain randomization.

Domain adaptation methods aim to map images rendered in simulation to realistic images, for example using Generative adversarial networks (GANs) [16], [19], [20]. Alternatively, such methods attempt to map simulated and real data to a common domain either in the image space [32] or feature space [33]. Other methods [34] perform adaptation by fine-tuning the model during deployment using self-supervision. While GANs are expressive models, they often generate artifacts [16] such as spurious objects and scene structures. To preserve the 3D structure of scenes in generated images, domain adaptation methods often resort to manually-defined constraints such as segmentation masks [16], [20].

Domain randomization methods, also known as domain augmentation, focus on adding variation to the simulated data to enforce robustness of the learned model to image perturbations. Tobin *et al.* [21] change the visual appearance of the simulation by randomizing the material textures, objects pose, shape and color, background, lighting properties and camera parameters. While this method can successfully transfer robotic policies trained in simulation to real robots [35], [36], [37], [26], it requires a significant amount of manual tuning and expensive iterations of simulation-based policy training followed by real-robot validation. For this reason, systematic and efficient domain randomization is still an open question [38]. In this work, we address this question by examining how domain randomization parameters should be chosen to make them usable across different tasks and scene conditions. Alghonaim and Johns [23] study a set of design choices for visual sim-to-real. However, the evaluation is limited to one off-line task, object pose estimation. Our work goes beyond this task studying design choices for a varied set of on-line robotic control tasks. Pashevich et al. [39] exploit object localization as a proxy task to optimize 2D image augmentations for depth images with Monte-Carlo tree search (MCTS). In contrast, we address sim-to-real transfer for RGB images where our experimental results in Table II confirm that 3D domain randomization is by far more effective than 2D image augmentations. The cost of 3D rendering prohibits MCTS optimization. We empirically show that a simple greedy search strategy on a proxy task enables successful sim-to-real transfer of visuomotor policies for diverse and challenging manipulation tasks.

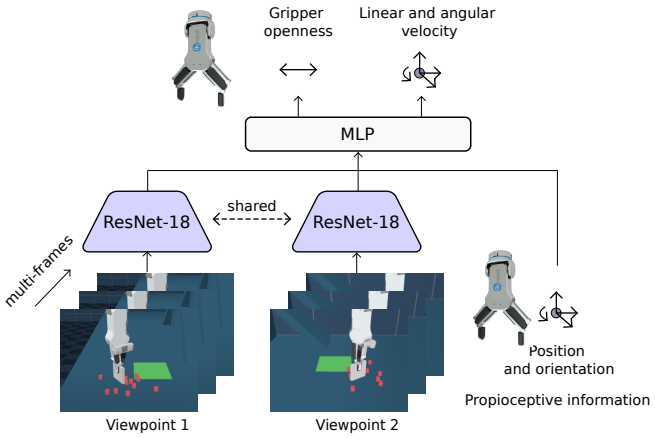


Fig. 2: **Overall architecture of our policy network.** The network predicts the next action given a history of multiple frames from two viewpoints and the proprioceptive information of the gripper.

III. Approach

Our goal is to train robust visuomotor policies in simulation and to deploy them to manipulation tasks on a real robot. We first describe our visual policies in Section III-A and introduce domain randomization for visual sim-to-real transfer in Section III-B. We then present our approach for selecting parameters of domain randomization using a proxy task in Section III-C.

A. Visuomotor policies for robotic manipulation

Problem formulation. We learn a policy $\pi_\theta(\mathbf{a}_t \,|\, \mathbf{o}_t)$ that maps environment observation $\mathbf{o}_t$ to a robot action $\mathbf{a}_t$ at timestep t, see Figure 2. The action $\mathbf{a}_t = (\mathbf{v}_t, \boldsymbol{\omega}_t, g_t)$ is composed of the linear velocity $\mathbf{v}_t \in \mathbb{R}^3$, the angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$ of the robot end-effector and the gripper openness state $g_t \in \{0, 1\}$. The policy is executed in a closed-loop setting to perform a manipulation task.

Model architecture. Our model takes input from two RGB cameras as frames $\mathbf{I}_t = \{\mathbf{I}_t^1, \mathbf{I}_t^2\}$ at each timestep t where $\mathbf{I}_t^* \in \mathbb{R}^{H \times W \times 3}$. The cameras are placed on a wide baseline with 90 degrees relative angle to alleviate depth ambiguities

and occlusions. The observation o_t includes the last three RGB frames $(\mathbf{I}_{t-2}, \mathbf{I}_{t-1}, \mathbf{I}_t)$ and the last three gripper proprioceptive values $\mathbf{P}_t = [\text{pos}_t, \sin(\phi_t), \cos(\phi_t)]$, where pos_t is the gripper position with respect to the robot base, and ϕ_t is the rotation angle of the gripper around the end-effector axis. We stack frames from each viewpoint in the channel dimension and use ResNet-18 network [40] to generate a feature vector of dimension 512 for each viewpoint. We then concatenate feature vectors from the two viewpoints together with the proprioceptive information $\mathbf{P}_t$. The action $\mathbf{a}_t$ is finally predicted by a multi-layer perceptron (MLP) composed of two layers with 512 hidden units and a ReLU activation function each.

Training with behavior cloning. Given a dataset of observation-action pairs $\{(\mathbf{o}_t, \mathbf{a}_t)\}_t$ obtained from expert trajectories, we randomly initialize our policy network and train it using a combination of the mean-squared error (MSE) loss for velocity control $\mathbf{v}_t$, $\boldsymbol{\omega}_t$, and the binary cross entropy (BCE) loss for gripper state probability g_t defined as:

$$L = \lambda L_{MSE}((\hat{\mathbf{v}}_t, \hat{\boldsymbol{\omega}}_t), (\mathbf{v}_t, \boldsymbol{\omega}_t)) + (1-\lambda) L_{BCE}(\hat{g}_t, g_t) \quad (1)$$

where λ is a hyper-parameter to balance the MSE and BCE terms, $(\mathbf{v}_t, \boldsymbol{\omega}_t, g_t)$ is the expert action and $\pi_\theta(o_t) = (\hat{\mathbf{v}}_t, \hat{\boldsymbol{\omega}}_t, \hat{g}_t)$ is the predicted action for observation o_t.

B. Domain Randomization

In order to improve sim-to-real transfer, we augment synthetic images with domain randomization (DR) for policy learning. We investigate different visual DR components, including textures, lightning conditions, object colors and camera parameters as described below.

Texture randomization. To obtain robustness to scene appearance, we randomize the textures of the robot, table, wall and floor. As the ability to distinguish object colors matters in our tasks, we do not randomize the object textures. We compare two types of textures as illustrated in Figure 3. The first type is procedural textures [21] (Figure 3 top), which have random colors and follow one of the four patterns - checkers, gradient, noise and a plain color. The second type is a set of $1,203$ high quality and realistic textures from ambientCG assets [41] (Figure 3 bottom).

Lighting randomization. To achieve robustness to lighting conditions, we sample the light position uniformly in a portion of a sphere for rendering images. We define the sphere in spherical coordinates with a distance of the light source to the workspace center in the range $[1.0, 3.0]$ meter, azimuthal angle and polar angle in the range $[0, \pi/2]$ and $[\pi/10, 4\pi/10]$ radians. This range of parameters allows us to sample realistic light source positions around the table, e.g., we avoid sampling lights inside the table or under it. Besides light positions, we randomize the light properties with diffuse, specular and ambient coefficients around a nominal value set to 0.3 by adding an offset sampled from the range $[-\psi_l, \psi_l]$ where ψ_l is a parameter to be optimized.

Variation of object colors. We treat object colors in a special way since object manipulation in some of our tasks depends on the object color. We therefore randomize object colors by sampling the Hue, Saturation and Value (Brightness) (HSV) around their nominal value. We sample an offset from range $[-\phi_o, \phi_o]$ and add it to the object color expressed using HSV coordinates in $[0, 1]$. The range ϕ_o is selected such that it is sufficiently large to cover possible color discrepancies between real and simulated scenes, and not too large to avoid confusion between different objects with initially different colors.

svg/textures_comp .pdf

Fig. 3: **Two types of textures in texture randomization.**

Variation of camera parameters. Camera calibration is often imprecise especially in terms of extrinsic parameters. To improve robustness to slight viewpoint changes, we randomize camera positions in simulation by sampling camera angles, locations and the field of view (FOV) around default values.

C. Localization as a proxy task

DR parameters could be selected for each task by optimizing the accuracy of the learned policies for a real robot scenario. Such an approach, however, is highly time-consuming as it requires policy retraining and real-world policy evaluation for each set of DR parameters. In this work we demonstrate that DR parameters can be efficiently chosen off-line using real images recorded for a proxy task. For this purpose we propose the following task of cube localization.

Given RGB images of a scene with three cubes of different colors, our proxy task aims to predict 3D locations of each cube relative to the gripper, see Figures 3 and 6.

We chose this proxy task because it requires to distinguish different colors and localize multiple objects, which enables to study DR parameters for color-dependent and multi-object manipulation policies.

To predict 3D locations of the cubes, we use a similar neural network architecture as in Section III-A but with RGB frames $\mathbf{I}_t = \{\mathbf{I}_t^1, \mathbf{I}_t^2\}$ for one time step instead of three and no proprioceptive information at the input. We train the model

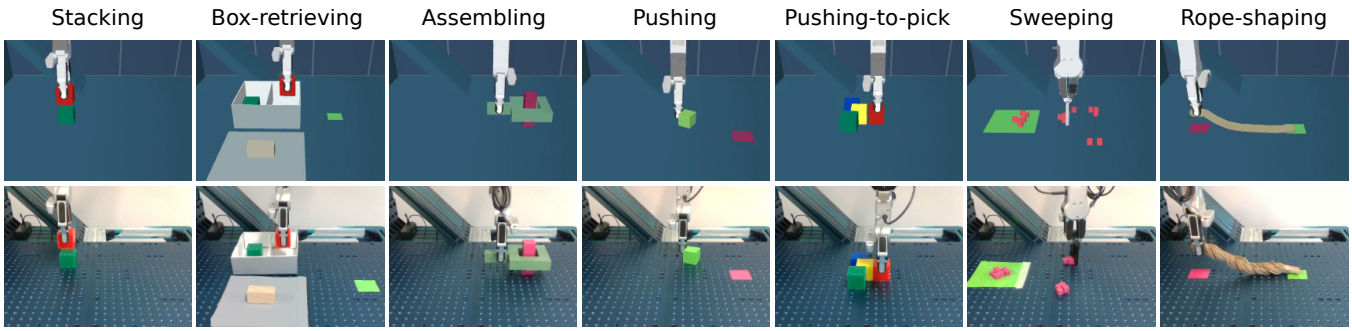


Fig. 4: **The seven manipulation tasks considered in our experiments.** Images from the simulated environment (top) and our real robot environment (bottom) demonstrate the gap between the two visual domains. Our tasks exhibit a variety of challenges: pick-and-place requires precise localization of objects and control of the end-effector; pushing is a contact-rich task that requires closed-loop reaction to adapt to object motion; box-retrieving is long-horizon; and rope-shaping involves manipulation of a deformable object.

by minimizing the distance between the predicted and the ground truth positions of the cubes.

To evaluate sim-to-real transfer for the proxy task, we train the cube localization network in simulation using different DR parameters and measure localization precision on pre-recorded images of real scenes. Our empirical experiments in Section V show that the performance of this proxy task is well aligned with the real-world accuracy of diverse manipulation policies.

IV. Tasks and Datasets

A. Manipulation tasks

To assess the quality and robustness of our proposed sim-to-real pipeline, we propose a rich set of robotic manipulation tasks with different challenges, see Figure 4.

Stacking. The robot has to pick up a red cube on the table and stack it on top of a green cube. Both cubes have a side length of 5 cm. Stacking is most similar to our proxy task and serves as a starting point to validate our sim-to-real pipeline.

Box-retrieving. A rectangular gray box of size $20.5 \times 21.5 \times 8$ cm^3 has two compartments containing a green and a red cube respectively. The cubes have a side length of 5 cm. The box is on the table and its lid is closed. The task requires a gripper to open the box lid, take out the red cube and place it on top of a green marker of size 6×6 cm^2 outside of the box. This task is challenging because it is a long-horizon task that requires a decision which object to pick once the box is opened.

Assembling. The task requires grasping a nut with a square hole located at a random position and orientation on the table, and then putting it on a pink rectangular parallelepiped of size $3.5 \times 3.5 \times 10.5$ cm^3 fixed on the table. It requires gripper rotation and high-precision motion to put the nut on the screw.

Pushing. The goal is to push a green cube of 4 cm side length with the closed tip of the gripper to reach a square pink marker of 6×6 cm^2 on the table. Pushing requires vision-based reactive behavior and closed-loop control to handle uncertainty of contacts between the gripper, the cube and the table.

Pushing-to-pick. A red cube is surrounded by three obstacle cubes with random colors. The goal is to pick the red cube which cannot be performed directly. The gripper needs to first unlock the red cube by pushing two cubes on each side of the red cube. All the cubes are of 5 cm side length. Pushing-to-pick combines challenges from pushing and picking into a long-horizon task.

Sweeping. The goal is to sweep 14 tiny foam cubes of 1.5 cm side length lying on the table onto a 16×16 cm^2 green marker. To do so a squared broom of 7×7 cm^2 is attached to the robot gripper. Sweeping is significantly harder than pushing as it requires to sweep many small objects towards the goal while they can spread. To successfully perform the task, a policy has to perform several sweeps and recover the missing objects that did not reach the target zone, a behavior only obtained with a closed-loop system.

Rope-shaping. A deformable rope of length 30 cm and diameter of 3 cm is in a random configuration on the table. The gripper needs to manipulate the rope to shape it into a straight line so that the ends of the rope are placed on top of two 6×6 cm^2 pink and green markers. This task is challenging due to the manipulation of a deformable object.

B. Dataset of expert demonstrations

We leverage object information available in simulation to generate expert trajectories solving the tasks. Stacking, pushing-to-pick, box-retrieving and assembling can be solved with an open-loop oracle computing the solution trajectory before execution given the initial gripper and object configuration. However, we need to design a closed-loop oracle for pushing, rope-shaping and sweeping tasks, where trajectories need to be computed at each step to deal with the dynamics of the environment. The closed-loop oracles are finite-state machines whose states define a sub-goal to finish the task, e.g., move next to the cube, push the cube, rotate the cube, move to the initial gripper position, etc. At each step, the oracle on a particular state uses the current gripper and object configuration to compute both the next action and next oracle state until the task is completed. The training datasets consists of $2,000$ demonstrations per task for stacking, box-retrieving, pushing and pushing-to-pick,

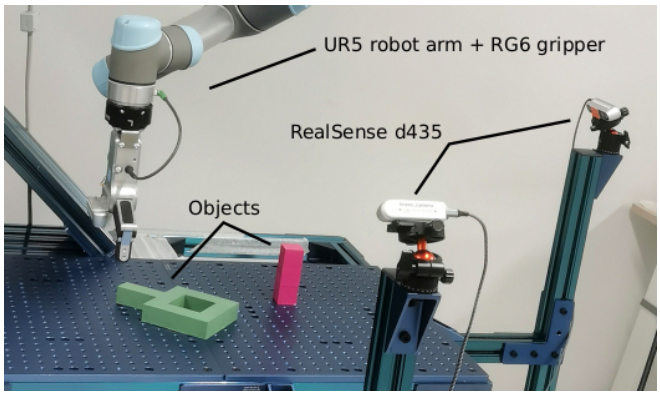


Fig. 5: **Robotic platform.** Our setup includes two RealSense D435 cameras, some objects and the UR5 robotics arm equipped with a RG6 gripper.

and $4,000$ demonstrations per task for assembling, sweeping and rope-shaping in simulation. We additionally collect 150 real robot demonstrations for stacking.

V. Experimental Results

A. Experimental Setup

Our robotic platform is a UR5 6-DoF robotic arm equipped with a two-finger Robotiq RG6 gripper, see Figure 5. Input RGB images are captured with two Intel RealSense D435 cameras located in the front and the left side of the robotic arm. We perform control and obtain camera measurements together with proprioceptive information at the frequency of 10 Hz. We use the MuJoCo engine [42] to perform physics simulation and rendering of the robotic environment. To align real and simulated images, we carry out extrinsic camera calibration using AprilTag [43] and we use the intrinsic parameters provided by the RealSense camera. Images are resized and cropped into 180×240 pixels. We define the gripper workspace as a volume of $40 \times 40 \times 20$ cm^3. The gripper initial position is sampled from the set of possible positions inside this workspace while the initial object positions are sampled randomly inside this same workspace and on the robot table.

Evaluation under real-world variations. We evaluate the robustness of our models to natural variations in visual appearance caused by changes in surface textures, lighting and camera poses. We consider five scenarios by: (i) adding a textured table cloth, (ii) decreasing the intensity of the room lighting, (iii) adding a variable color lighting with two Olafus 50W RGB LED flood lights, (iv) offseting object colors and (v) adding noise to the camera pose. Figure 6 illustrates the default scenario as well as the five variations.

B. Implementation details

Cube localization proxy task. To evaluate the precision of cube localization, we use three cubes with side length of 5cm and colored green, red and yellow respectively. We collect a real dataset for the proxy task where we sample random positions for the cubes and the gripper inside the environment workspace. We repeat this 250 times and vary the viewing conditions, such as background texture, lightening and camera viewpoint, see Figure 6. While in simulation we know the cubes and gripper position, on the real robot we do not have direct access to this information. To overcome this limitation, we manually place the cubes in known positions, and the gripper moves each cube to a random position. Finally, the gripper moves to a random position, and we use the robot forward kinematics at each step to create the ground truth. We train our model for $400,000$ gradient steps using the AdamW optimizer [44]. We use a cosine scheduler with initial learning rate of 3×10^{-4} and the minimum learning rate of 1×10^{-6}.

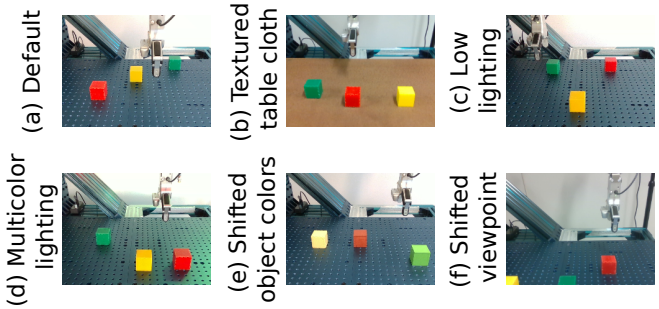


Fig. 6: **Scenarios with varying visual appearance.** We study the robustness of our models to variations in visual appearance caused by changes in surface texture (b), lighting (c, d), object colors (e) and cameras pose (f).

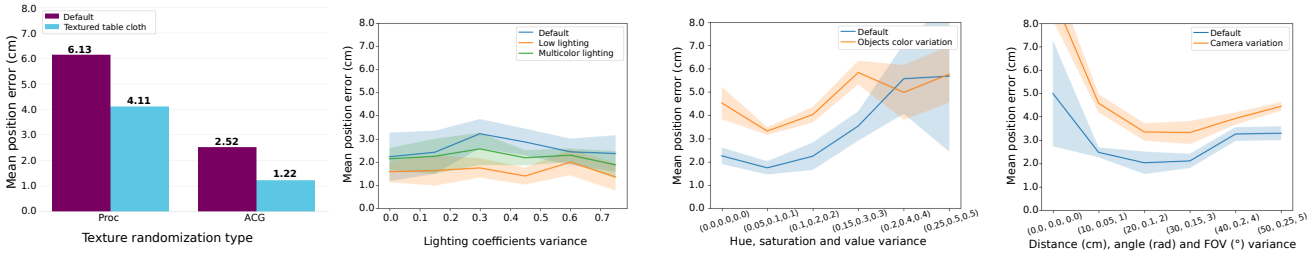
Policy learning. We optimize robotic manipulation policies using a batch size of 32. For the stacking, pushing, pushing-to-pick and box-retrieving tasks we perform $400,000$ gradient steps while the more difficult tasks of assembling, rope-shaping and sweeping are optimized for 1M steps. We evaluate the performance of our models on 250 episodes in simulation with domain randomization disabled, and run 20 trials for each model and task when performing real robot evaluation. We experimentally set the value of λ in (1) to 0.8 during training and train our policy models with the AdamW optimizer [44] and a cosine scheduler.

C. Policy learning in simulation

In this section, we evaluate the design choices of our policy network in simulation for all seven tasks, see Table I. We train our models on 2000 (4000) demonstrations in simulation using DR with the parameters defined in Section V-D and evaluate them in simulation without any augmentation. The baseline takes as input a single viewpoint and a single frame and reaches 44.97% success rate on average. Adding a second viewpoint improves the performance to 65.43% on average and results in 50.4% absolute gain for the assembling task which requires high precision. Adding a short-term history by taking as input three frames results in a further improvement of 28.51% on average. Finally, we add proprioceptive information to the input, which further improves the performance by 4.4% and leads to the best performance of 98.34% success rate. Propioceptive information is essential for tasks such as sweeping (+18%) where a high precision of the gripper position is required.

Model	Stacking	Box-retrieving	Assembling	Pushing	Pushing-to-pick	Sweeping	Rope-shaping	Average
Baseline	39.6	67.2	14.0	96.0	26.8	28.4	38.4	44.97
+ Second viewpoint	61.6	83.6	64.4	95.2	64.0	17.2	72.0	65.43
+ Multiple frames	96.4	100.0	98.8	94.8	98.8	80.0	88.8	93.94
+ Proprioceptive info.	100.0	100.0	99.6	95.6	100.0	98.0	95.2	98.34

TABLE I: **Comparison of different modeling choices for policy learning in simulation.** We train on 2000 demonstrations for stacking, box-retrieving, pushing and pushing-to-pick and 4000 for assembling, sweeping and rope-shaping. We evaluate models on 250 episodes for each manipulation task and report the success rate (%).



(a) **Texture types.** (b) **Light parameters ϕ_l.** (c) **Object color parameters ϕ_o.** (d) **Camera parameters.**

Fig. 7: **Impact of domain randomization parameters on the proxy cube localization task.** The performance (y-axis) is mean position error (cm) over 250 images in real scenes, where the lower the better.

D. Domain randomization ablation with a proxy task

In this section, we first determine the best domain randomization (DR) parameters for each augmentation type on the proxy localization task. Figure 7a compares the two different texture types and shows that ambientCG textures are much more effective than the procedurally generated ones. As illustrated in Figure 3, ambientGC textures are of high quality and have richer variations than procedurally generated textures which could explain this difference in error. In the following, we always use ambientCG texture randomization. For lighting randomization, Figure 7b shows that performance is not very sensitive to different randomization ranges for light ambiance, diffuse and specular coefficients. Hence, we select an intermediate value randomizing the previous three coefficients in the range $[0, 0.6]$. For the object colors, HSV color offset has a significant impact as shown in Figure 7c. A large offset significantly deteriorates the performance as it confuses colors of different objects. If the offset is too small, the model is not robust enough. We use an offset around the object nominal color of $\phi_o = (0.05, 0.1, 0.1)$ that achieves the best performance. Finally, as shown in Figure 7d a small randomization of camera parameters is important due to possible calibration errors and too much variation hurts the performance. We offset the values for camera position by $[-10, 10]$ cm, for camera angle by $[-0.05, 0.05]$ rad and for field of view by $[-1.0, 1.0]°$.

Table II compares combinations of different DR types using the selected parameters. We evaluate both the error on the proxy localization task for the default scene appearance and the average error of all the varied appearance scenes, see Figure 6. In addition, we compare DR with the standard 2D image augmentation; we follow [6] and modify synthetic images using a random offset $[-0.12, 0.12]$ for brightness, $[-0.05, 0.05]$ for hue, $[0.5, 1.5]$ for saturation, $[0.5, 1.5]$ for contrast and $[-4, 4]$ pixels for horizontal and vertical image translation.

Training data Synt	Training data Real	Img aug	Texture ACG	Light	Obj color	Camera	Error (cm) default	Error (cm) variations
		×	×	×	×	×	7.55	8.22
		✓	×	×	×	×	6.92	7.17
		×	✓	×	×	×	2.52	3.53
20k	-	×	✓	✓	×	×	2.66	3.65
		×	✓	✓	✓	×	1.62	2.92
		×	✓	✓	✓	✓	1.33	2.70
		✓	✓	✓	✓	✓	0.95	1.97
100k	-	✓	✓	✓	✓	✓	**0.48**	**1.39**
-	750	×	×	×	×	×	0.72	3.08
100k	750	✓	✓	✓	✓	✓	**0.14**	**1.00**

TABLE II: **Domain randomization ablation on the cube localization proxy task.** The training data include 750 real images, 20k and 100k synthetic images, as well as synthetic images followed by real images fine-tuning. The models are evaluated on 250 real images without (default) and with scene appearance variation (variations). We report the localization error (cm).

The first row shows the results for training with 20k synthetic images. We can observe that by adding image augmentations the performance improves only very slightly. Texture randomization significantly improves the performance and decrease the error from 7.55 to 2.52 cm. Adding lighting randomization does not improve performance, i.e., the error remains similar. With the variation of object colors, we further improve the error by 1 cm and achieve an error of 1.62 cm. Varying the camera parameters additionally improves performance to 1.33 cm. Further adding 2D image augmentations has a minor impact on the error and decreases it to 0.95 cm.

Since we are using synthetic data for training, we can easily generate additional data. We show a large boost in performance when increasing the dataset from $20,000$ to

img aug	Texture ACG	Light	Obj color	Camera	Success rate Stacking	Box-retrieving	Assembling	Pushing	Pushing-to-pick	Sweeping	Rope-shaping	Average
✓	×	×	×	×	0/20	0/20	0/20	0/20	0/20	0/20	0/20	0/20
✓	✓	×	×	×	15/20	16/20	7/20	13/20	11/20	9/20	8/20	11.3/20
✓	✓	✓	✓	×	17/20	15/20	13/20	18/20	13/20	11/20	11/20	14.0/20
✓	✓	✓	✓	✓	**20/20**	**17/20**	**16/20**	**18/20**	**17/20**	**19/20**	**18/20**	**18.6/20**

TABLE III: **Comparison of different domain randomizations.** We report performance on seven manipulation tasks with a policy trained in simulation and evaluated on a real robot. For each task we run 20 episodes on the robot and report the success rate.

$100,000$ images, resulting in an error of 0.48 cm. Notably, our model trained with DR and simulation-only data outperforms the model trained on real but limited data, and is also more robust to variations in the scene appearance. The error for such data with variations (last column of Table II is 1.39 cm for synthetic training data with DR in contrast to 3.08 cm obtained when training with real data. Finally, we show that additional fine-tuning of our model on real data is beneficial, i.e., the error decreases to 0.14 cm.

E. Ablation of domain randomization for policy learning

In this section, we measure the impact of domain randomization (DR) on policy learning beyond the proxy task. We train policies with our DR approach on a suite of seven manipulation tasks described in Section IV. The same DR parameters selected for the cube localization proxy task are used for all the manipulation tasks.

Table III presents the results. We compare our full DR approach with three variations of Table II. The first variation uses only 2D image augmentation; the second one adds ambientCG texture randomization to the 2D image augmentation; and the third one includes lighting and object color randomization on top of the previous variation. Using only 2D image augmentation is not enough to succeed in any task on the real robot, i.e., the success rate is 0. Adding ACG texture randomization helps transferring the policy with an average success rate of $11.3/20$. Additionally adding light and object color randomization improves the performance further and results in an average success rate of $14.0/20$. Adding small camera variations to the training further improves the performance, i.e., our full DR setup obtains a success rate of $18.6/20$ on average. When comparing results in Tables II and III, we can see that *the performance of the proxy task is well aligned with the performance of policies for all considered manipulation tasks on the real robot*. This validates the effectiveness of our proxy task for choosing DR parameters for the visual sim-to-real policy transfer. We illustrate successful runs of our policies on the real robot in Figure 8. Additional qualitative results for all seven manipulation tasks are presented in the supplementary video.

Failure cases analysis. Figure 9 illustrates two type of failure cases. The first type of failure (top row) is due to unseen states. During pushing the cube rotates by 90 degrees, a state never observed in training. This causes the gripper to leave the pushing surface and continue the pushing to the target place without the cube. The second one originates from the physical sim-to-real gap (bottom row). During assembling the gripper grasps the nut by the end of the handle and the nut rotates due to its weight and the lack of sufficient friction between the gripper fingers and the nut. This issue does not occur in simulation due to the insufficient realism in friction modeling.

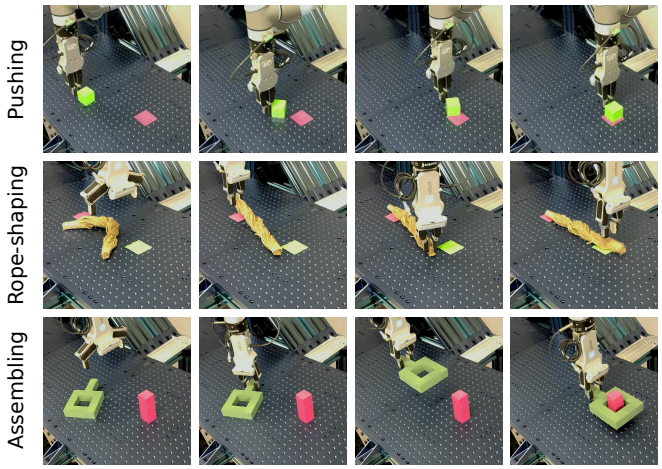


Fig. 8: **Real robot experiments.** We illustrate successful runs for pushing, rope-shaping and assembling.

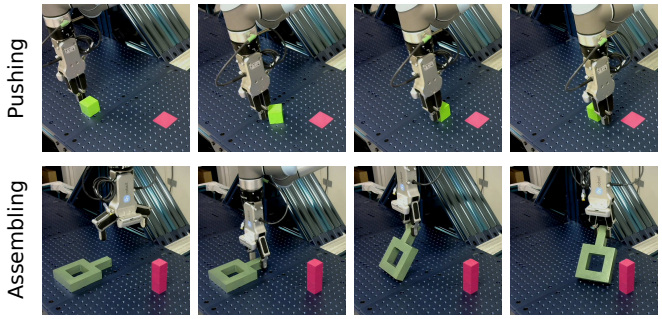


Fig. 9: **Real robot experiments.** We illustrate failure cases for pushing and assembling.

F. Robustness of policies to visual variations

In this section, we evaluate the robustness to visual variations for the stacking task on the real robot. Table IV compares our policy trained with DR, a policy trained on real data and a policy trained on DR and fine-tuned on real data for the different variations described in Section V-A. We use $2,000$ demonstrations in simulation and 150 real robot demonstrations. While simulation allows us to gather 2000 demonstrations in minutes, gathering 150 demonstrations on the real robot is time-consuming (approx. 4.5 hours).

In the default setting without visual appearance variation

	DR	Real	DR+Real
Default	20/20	20/20	20/20
Textured table cloth	20/20	1/20	20/20
Low lighting	19/20	17/20	20/20
Multicolor lighting	16/20	14/20	17/20
Object colors variation	18/20	19/20	20/20
Camera variation	11/20	8/20	12/20
Average	17.3/20	13.2/20	**18.2/20**

TABLE IV: **Success rate for stacking on a real robot under default and five variations of the scene appearance.** We compare policies trained with synthetic data with domain randomization (2000 demonstrations + DR), real data (150 real demonstrations), and their combination (2000 demonstration + DR and fine-tuned on 150 real demonstrations).

of real scenes, all the policies achieve 100% success rate. However, our DR policy exhibits much stronger robustness to changes in the table texture and achieves 100% success rate compared to 5.0% success rate obtained by the real data policy. Similarly, variations of the lighting conditions caused by lowering light intensity and changing light color result in larger decrease of performance for policies trained on the real data compared to our policies. When we modify cube colors, we can see that both the DR and real data policy perform well achieving a success rate of 90.0% and 95.0% respectively. Finally, the variation of camera parameters results in a large degradation of both policies. Overall we conclude that policies trained on synthetic data with DR are more robust compared to policies trained on the real data, when such data is available in limited amounts. Additionally, using the real data to fine-tune the policies trained with DR results in an additional increase of the average success rate of $+4.5\%$.

VI. Conclusion

In this paper we address visual sim-to-real transfer of manipulation policies using domain randomization. We explore multiple visual domain randomization components and propose a proxy task of cube localization to choose appropriate DR parameters. We demonstrate that under the same DR settings, the real-world performance on our proxy task strongly correlates with the performance of policies trained for diverse and challenging manipulation tasks. We introduce a rich set of seven manipulation tasks to benchmark visual sim-to-real transfer and demonstrate that our method significantly and consistently outperforms other approaches without DR, achieving 93% average success rate on a real robot. Our method also demonstrates increased robustness to visual appearance changes in real scenes.

Acknowledgements. This work was partially supported by the HPC resources from GENCI-IDRIS (Grant 20XX-AD011012122). It was funded in part by the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA-0001 (PRAIRIE 3IA Institute), the ANR project VideoPredict (ANR-21-FAI1-0002-01) and by Louis Vuitton ENS Chair on Artificial Intelligence.

References

[1] I. Akkaya *et al.*, "Solving Rubik's Cube with a Robot Hand," *CoRR*, vol. abs/1910.07113, 2019.
[2] A. Allshire *et al.*, "Transferring Dexterous Manipulation from GPU Simulation to a Remote Real-World TriFinger," *CoRR*, vol. abs/2108.09779, 2021.
[3] A. Handa *et al.*, "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," *CoRR*, vol. abs/2210.13702, 2022.
[4] T. Chen *et al.*, "Visual Dexterity: In-hand Dexterous Manipulation from Depth," *CoRR*, vol. abs/2211.11744, 2022.
[5] S. Levine *et al.*, "End-to-End Training of Deep Visuomotor Policies," *JMLR*, 2016.
[6] A. X. Lee *et al.*, "Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes," in *CoRL*, 2021.
[7] P. Florence *et al.*, "Implicit Behavioral Cloning," in *CoRL*, 2021.
[8] T. Miki *et al.*, "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild," *Sci. Robotics*, 2022.
[9] J. Frey *et al.*, "Locomotion Policy Guided Traversability Learning using Volumetric Representations of Complex Environments," *CoRR*, vol. abs/2203.15854, 2022.
[10] A. Agarwal *et al.*, "Legged Locomotion in Challenging Terrains using Egocentric Vision," *CoRR*, vol. abs/2211.07638, 2022.
[11] M. Aractingi *et al.*, "Controlling the Solo12 Quadruped Robot with Deep Reinforcement Learning," *Sci. Rep.*, 2022.
[12] D. S. Chaplot *et al.*, "Object Goal Navigation Using Goal-Oriented Semantic Exploration," in *NeurIPS*, 2020.
[13] S. Chen *et al.*, "History Aware Multimodal Transformer for Vision-and-Language Navigation," in *NeurIPS*, 2021.
[14] S. Chen *et al.*, "Think Global, Act Local: Dual-Scale Graph Transformer for Vision-and-Language Navigation," in *CVPR*, 2022.
[15] A. Khandelwal *et al.*, "Simple but Effective: Clip Embeddings for Embodied AI," in *CVPR*, 2022.
[16] K. Bousmalis *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *ICRA*, 2018.
[17] P. R. Florence *et al.*, "Self-Supervised Correspondence in Visuomotor Policy Learning," *RA-L*, 2020.
[18] T. Zhang *et al.*, "Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation," in *ICRA*, 2018.
[19] K. Rao *et al.*, "RL-CycleGAN: Reinforcement Learning Aware Simulation-to-Real," in *CVPR*, 2020.
[20] D. Ho *et al.*, "RetinaGAN: An Object-aware Approach to Sim-to-Real Transfer," in *ICRA*, 2021.
[21] J. Tobin *et al.*, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *IROS*, 2017.
[22] R. Strudel *et al.*, "Learning to Combine Primitive Skills: A Step Towards Versatile Robotic Manipulation," in *ICRA*, 2020.
[23] R. Alghonaim and E. Johns, "Benchmarking Domain Randomisation for Visual Sim-to-Real Transfer," in *ICRA*, 2021.
[24] S. Valtchev and J. Wu, "Domain Randomization for Neural Network Classification," *JBD*, 2020.
[25] M. Denninger *et al.*, "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering," in *ICRA*, 2020.
[26] F. Sadeghi *et al.*, "Sim2Real Viewpoint Invariant Visual Servoing by Recurrent Control," in *CVPR*, 2018.
[27] Y. Qin *et al.*, "DexPoint: Generalizable Point Cloud Reinforcement Learning for Sim-to-Real Dexterous Manipulation," in *CoRL*, 2022.
[28] X. Peng *et al.*, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *ICRA*, 2018.
[29] E. Valassakis *et al.*, "Crossing the Gap: A Deep Dive into Zero-Shot Sim-to-Real Transfer for Dynamics," in *IROS*, 2020.
[30] B. Mehta *et al.*, "Active Domain Randomization," in *CoRL*, 2020.
[31] Y. Tsai *et al.*, "DROID: Minimizing the Reality Gap Using Single-Shot Human Demonstration," *RA-L*, 2021.
[32] S. James *et al.*, "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks," in *CVPR*, 2019.
[33] Y. Ganin *et al.*, "Domain-Adversarial Training of Neural Networks," *JMLR*, 2016.
[34] N. Hansen *et al.*, "Self-Supervised Policy Adaptation during Deployment," in *ICLR*, 2021.
[35] S. James *et al.*, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," in *CoRL*, 2017.
[36] O. M. Andrychowicz *et al.*, "Learning Dexterous In-Hand Manipulation," *IJRR*, 2020.

[37] J. Matas *et al.*, "Sim-to-Real Reinforcement Learning for Deformable Object Manipulation," in *CoRL*, 2018.
[38] S. Hofer *et al.*, "Perspectives on Sim2Real Transfer for Robotics: A Summary of the R:SS 2020 Workshop," *CoRR*, vol. abs/2012.03806, 2020.
[39] A. Pashevich *et al.*, "Learning to Augment Synthetic Images for Sim2Real Policy Transfer," in *IROS*, 2019.
[40] K. He *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.
[41] L. Demes, "AmbientCG Textures," 2017-2022.
[42] E. Todorov *et al.*, "MuJoCo: A physics engine for model-based control," in *IROS*, 2012.
[43] E. Olson, "AprilTag: A Robust and Flexible Visual Fiducial System," in *ICRA*, 2011.
[44] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *ICLR*, 2019.