Beyond Pick-and-Place: Tackling Robotic Stacking of Diverse Shapes

Anonymous Author(s)
Affiliation
Address
email

Abstract: We study the problem of robotic stacking with objects of complex geometry. We propose a challenging and diverse set of such objects that was carefully designed to require strategies beyond a simple "pick-and-place" solution. Our method is a vision-based reinforcement learning (RL) approach combined with interactive policy distillation and simulation-to-reality transfer. Our learned policies can efficiently handle multiple object combinations in the real world and exhibit a large variety of stacking skills. In a large experimental study, we investigate what choices matter for learning such general vision-based agents in simulation, and what affects optimal transfer to the real robot. We then leverage data collected by such policies and improve upon them with offline RL.

Keywords: CoRL, sim-to-real, offline RL, manipulation, robot learning

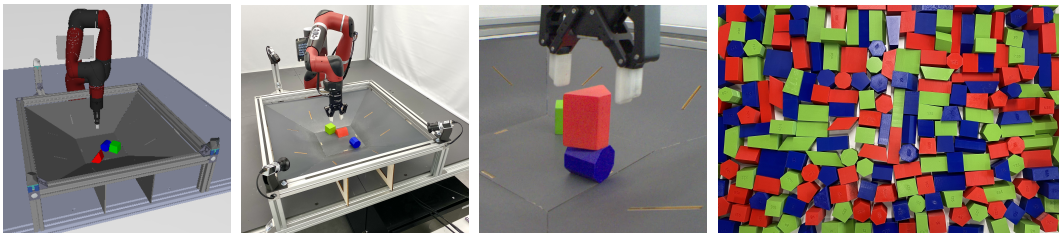
1 Introduction

In this work we study, in a principled way, robotic manipulation in a real-world setting where there is interaction between multiple objects that involve complex contact dynamics. We study this question via the task of stacking objects with diverse geometries. In particular, we go beyond simple pick-and-place tasks and focus on the interaction between objects by introducing a new benchmark, *RGB-Stacking*, see Figure 1. It consists of a set of 152 procedurally-generated and 3D printed rigid geometric objects that require different stacking strategies. *RGB-stacking* tasks involve three objects, colored red, green, and blue—hence the name—and require stacking one object on top of the other, while the third acts as a distractor. The objects were designed to have different degrees of grasping and stacking difficulty for a parallel gripper on an arm with 4-DoF control. The benchmark is standardized and we release all relevant information for replicating it as supplementary material.

To highlight the challenges in stacking these objects, we instantiated two RGB-stacking tasks (with corresponding versions in simulation and real world) designed to investigate a set of research questions: In the first task we consider mastering stacking for a set of 5 specific combinations of objects. The 5 combinations were chosen in terms of the challenges they present: precision in grasping and stacking, balancing, and using the top object as a tool to flip a bottom object that has a slanted face pointing up. For this task we were interested to see whether it is possible to learn a single vision-based policy that can reliably stack all 5 triplets in the real world. For the second task we considered the challenge of learning general stacking strategies from a large set of training objects and test how well they transfer to held-out objects—to assess generalization—we use the 5 triplets from the first task for this. In both cases, we find that using an agent trained in simulation is the most efficient way to bootstrap data collection for offline RL.

Our contributions are as follows. *(a)* We present and release a benchmark for stacking that features a diverse set of geometric objects and tens of thousands of possible stacks. *(b)* We show that it is possible to learn a single vision-based policy that can handle multiple combinations of objects and can demonstrate a plethora of stacking strategies, emergent from RL training. We demonstrate zero-shot transfer of these strategies to the real robot. *(c)* We provide a large ablation study to evaluate

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.



(a) Simulation environment **(b)** Real-robot environment **(c)** Successful stack example **(d)** RGB-objects

Figure 1: The RGB-stacking tasks involve three objects colored **R**ed, **G**reen, and **B**lue to signal to a vision-based agent which one should be stacked on top of which. We present tasks for stacking red objects on top of blue ones, with the green ones as distractors. The variations in shape require successful agents to reason about contact dynamics and object orientation to form stable stacks. The benchmark defines a parametric family of RGB-objects, which allows designating held-out objects that are quantifiably different from the training objects.

what matters for sim-to-real transfer and for imitation of a simulation-trained RL policy (that uses privileged state information) with a vision-based general policy. We train vision-based agents via an interactive distillation approach—decoupling learning the required stacking skills with RL training from mapping the skills to perception with imitation. This allowed faster experimentation and iteration. We find that for the distillation process to succeed the state and action distributions used for training are of paramount importance. *(d)* Finally, we demonstrate that a single offline RL step based on data gathered with sim-to-real policies on the real robots can boost performance; while training based on real episodes collected by a scripted agent was worse than zero-shot sim-to-real.

2 Related Work

Robotic Stacking. Several works have recently dealt with vision-based real-robot stacking either by learning a curriculum of the different stages of the task [1], by combining human demonstrations and RL [2, 3], or by using some flavor of sim-to-real transfer [2, 4, 5]. What differentiates our work are 2 primary characteristics: the large variety in our benchmark and our extensive evaluations. In all prior work, the focus has almost exclusively been on cube stacking, a task that does not require reasoning about orientation, shape, or stack stability. Even then, the low task success rates reported points towards the difficulty of the stacking objects in the real world even when evaluated on a limited set of initial configurations. In contrast, RGB-stacking provides a general, reproducible, and significantly more difficult benchmark for robotic manipulation. It involves more than a million stacking combinations and our evaluations are significantly more extensive than in prior work.

Large-Scale Deep RL for Robot Manipulation. Deep learning for robotic manipulation was in part popularized by large-scale data collection for grasping in the real world [6, 7, 8, 9], a task that allows for automated evaluation and resetting. However, these efforts considered problems with short time horizons and limited dynamical effects (e.g. position-controlled robots with actions taking up to a second). As a result they often used simple Q-learning from collected data with direct optimization for action selection [8]. In contrast we consider velocity-based control at a rate of 20 Hz leading to long episodes (400 timesteps). This is a more challenging scenario in which learning with, e.g., QT-Opt [8] from recorded data alone would exhibit problems due to wrongly-estimated Q-values [10]. Additionally, large-scale RL-based learning in the real world can be difficult to reproduce as the entire real-world process (which can last months [7, 8]) is a function of the initial conditions. In contrast, our simulation-based training is inherently reproducible. In-hand object manipulation provides an alternative route for a difficult challenge, but automating episodic resets become more challenging as the objects easily fall out of the hand [11, 12].

Imitation Learning and (Offline) Reinforcement Learning. For training in simulation, and subsequent training of sim-to-real policies, we use techniques from off-policy RL followed by an interactive imitation learning approach. For training in simulation we consider RL with full state-information, in order to avoid problems with partial observability. In this setting we use the MPO algorithm [13]; a sample efficient, state-of-the-art, off-policy actor-critic method. Initial training is then followed by our pipeline for obtaining a (general) vision policy via a DAgger [14] style interactive imitation learning approach to distill the MPO policy. We found this to signifi-

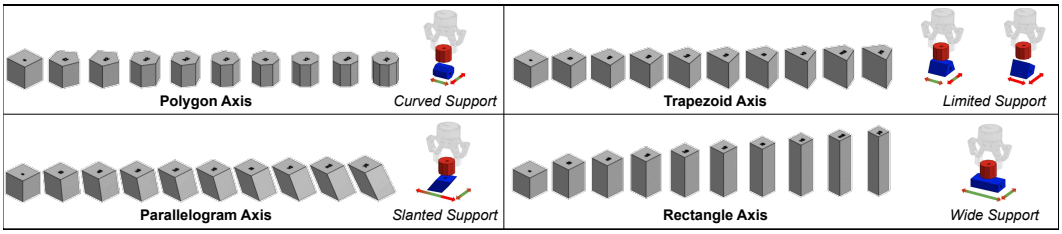


Figure 2: Illustration of the deformations applied for each of the 4 major axes of the RGB-Objects parametric family. Each deformation changes the stacking affordance of RGB-objects; for bottom objects the stacking support is illustrated with red-green lines showing where a stack is possible (green).

cantly outperform Behavior Cloning [15] / policy distillation [16]; due to the fact that DAgger-style training provides corrections for mistakes of the vision-based policy. We use Domain Randomization [17, 18, 19, 20, 21, 22] to obtain good sim-to-real transfer in the absence of real-world data; more details on this are given in the supplementary. Finally, when improving policies with offline RL from real data we consider: i) a filtered behavior cloning loss that is equivalent to CRR-exp [23] but with data from a single policy source (analogous to work on combined offline and online RL [24, 25]); ii) Behavior Cloning based on only successful trajectories. Both allow for stable offline learning without the problems standard RL algorithms exhibit in this setting [10].

3 The RGB-Stacking Benchmark

Despite stacking being addressed as a robotic learning task in prior work, previous approaches have been limited to a reduced set of objects, typically cubical in shape. We focus on the problem of stacking a variety of objects, characterized by different shapes. This is obtained by defining a parametric family of *RGB-objects*. The design principle is to vary the grasp and stack affordances of these objects for a parallel gripper. Our choices significantly change the difficulty of the stacking task by requiring an agent to exhibit behaviors that go beyond a simple pick-and-place [26] strategy. We do so while keeping the benefits of automated learning and evaluation as in [27, 7]. In Appendix A, we analyze the difficulty of the task in various ways. We qualitatively evaluate the grasping affordance based on general principles on force closure and object funnels [28], and quantitatively based on the Ferrari-Canny grasp metric [29, 30]. Similarly, we qualitatively depict the stacking affordance (Figure 2), which is varied by having bottom objects, whose top flat surfaces differ in area, shape, and orientation. We also quantitatively evaluate both affordances by evaluating the stacking performance of human teleoperators in simulation, and of a carefully scripted agent.

3.1 The RGB-Objects Family

Our objects are all obtained by applying a deformation to a *seed* cube, a 2D vertical extrusion of a planar square. We defined 4 major axes of deformation, illustrated in Figure 2, resulting in different shapes. These shapes can be thought of as the vertical extrusion of a 2D shape. The **Polygon Axis** transforms the planar square into a regular polygon. The **Trapezoid Axis** morphs the planar square to an isosceles trapezoid. The **Parallelogram Axis** changes the orientation of the extrusion axis, from vertical to progressively more slanted axes. Lastly, the **Rectangle Axis** uniformly scales the object along the x, y or z-axis.

These deformations and their combinations form a parametric family of objects. We obtain the final set of objects by uniformly sampling, *for both the major axes and all pairwise combinations*, 8 objects between the seed object and a maximally deformed object. Figure 1(d) shows a subset of these in the real world, and Appendix A contains a complete depiction.

3.2 RGB-Stacking Tasks

The RGB-stacking tasks we tackle in this paper involve three RGB-objects in a basket in front of the robot. Those are colored **R**ed, **G**reen, and **B**lue to signal to a vision-based agent which one should be stacked on top of which, and which one is just a distractor. In all our vision-based experiments, red is assigned to the top object, blue to the bottom object, and green to the distractor. We note that the role of the latter is not just to distract visually but also to serve as an obstacle during stacking.

A successful stack is achieved when the red object is above the blue one and their centroids are vertically aligned within some thresholds. A detailed description on how we formally define this, as well as reward functions, can be found in Appendix A.

3.2.1 Skill Mastery on 5 Specific Triplets

The first task involves 5 RGB-object triplets, with each object in a triplet being from a *major* axis. The triplets were chosen to have different kinds and degrees of difficulty for a stacking agent. The task is to achieve skill mastery on these 5 fixed combinations of objects, shown in Figure 3, with a single vision-based agent. We explain the challenges for each triplet in a loosely descending order of difficulty:

Triplet 1. The main challenge is grasping the top object, a grasp that involves the gripper closing on the slanted sides will fail. The bottom object also provides difficulty, as it has a limited stacking surface and can easily roll.

Triplet 2. In this triplet, the bottom object has sloped sides and may need to be reoriented by using the the top object as a tool before stacking.

Triplet 3. The challenge with this triplet is to have a secure central grasp for the elongated object and balance it on top of the slanted bottom object.

Triplet 4. An easy triplet, it has rectangular prisms for both red and blue objects; the challenge is primarily to align their centroids, required for a stack to be considered successful.

Triplet 5. In this triplet, the top object can easily roll off the bottom object as it has a large number (10) of faces and is nearly cylindrical.

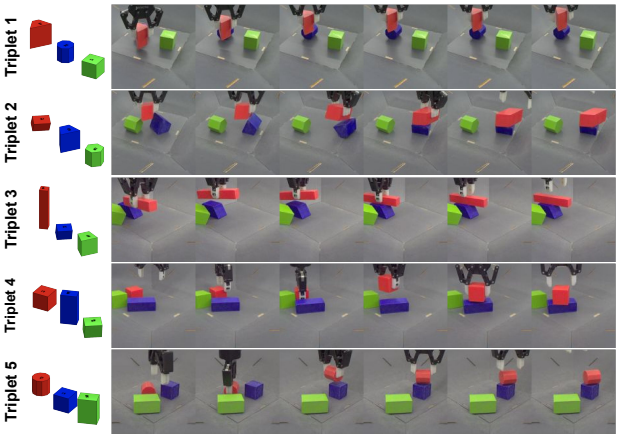


Figure 3: Stacking Skill Mastery for our 5 Specific Triplets. All the 5 stacks shown here are performed by a single agent trained solely in simulation with domain randomization and transferred to the real robot in a zero-shot fashion. Each triplet poses its own unique challenges to the agent: Triplet 1 requires a precise grasp of the top object; Triplet 2 often requires the top object to be used as a tool to flip the bottom object before stacking; Triplet 3 requires balancing; Triplet 4 requires precision stacking (the object centroids need to align); and the top object of Triplet 5 can easily roll off if not stacked gently.

3.2.2 Skill Generalization

The second RGB-stacking task we are tackling is a transfer task. For this we designate the axes of all pairwise combined deformations (see Appendix A for details) as the *training axes* and the ones of single deformation—the major axes above—as the *held-out axes*. Based on these, we created a *training object set*, which consists of 103 different shapes, and a *held-out set*, containing 40 shapes. The 5 fixed triplets chosen for the previous task belong to the held-out axes. While final performance is evaluated on these 5 fixed test triplets, during training we hold out not just these objects but the entire 4 axes of deformation they belong to.

4 Stacking a Diverse Set of Objects in the Real World with a General Policy

Since we have access to a simulator with a reasonably accurate model of our real robot and its cell, our approach is to solve the RGB-stacking tasks in three stages: 1) RL training of expert in simulation using state information, 2) imitation of RL expert with a vision-based policy (utilizing both visual data augmentation and domain randomization), and 3) optionally a policy improvement step using data gathered on the real robots. We explicitly decouple these stages into separate learning problems which provides distinct advantages—it allows faster and cheaper iteration for obtaining a vision-based policy (specifically with respect to finding the right network architecture), state-based experts allow us to vary visual characteristics in step 2, and offline learning allows us to re-use any data collected during evaluation of sim-to-real policies.

Notation. We consider the Markov Decision Process (MDP) in which a policy $\pi(a|s)$—a probability distribution over actions $a \in \mathcal{A}$ that is conditioned on states $s \in \mathcal{S}$—acts to maximize the discounted sum of rewards, with sparse rewards $r(s,a)$ and discount factor $\gamma \in (0,1]$. We use $Q^\pi(s,a) = \mathbb{E}_{\rho_\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)|s_0 = s, a_0 = a]$ to denote the Q-function, where ρ_π is the trajectory distribution induced by π, and $A^\pi(s,a) = Q^\pi(s,a) - \mathbb{E}_{a'\sim\pi}[Q(s,a')]$ for the advantage when executing π. In the real world, we cannot assume access to full state information (e.g. object poses and labels), so both the policy and advantage are defined based on observations $o(s) \in \mathcal{O}$ (actuator positions of the robot and two camera images) as $\pi(a|o(s))$ and $A^\pi(o(s),a)$, respectively.

4.1 Training Expert Policies from State Features in Simulation

We train expert policies via off-policy RL in simulation with a shaped reward. We used the MPO algorithm [13] for this purpose (details in Appendix D), but any RL algorithm could have been used in this stage. We found training directly from state s to be significantly faster than training from vision and thus trained expert policies $\pi_e(a|s,y)$—where y denotes a triplet number in $\{1,\ldots,5\}$ for the *Skill Mastery* task, or the parameters of object deformation for the *Skill Generalization* task.

4.2 Interactive Imitation Learning of a Vision-Based Policy for Sim-to-Real Transfer

We use imitation in simulation to distill the state-based expert(s) into a single "student" vision-based policy $\pi_\theta^{\text{s2r}}(a|o(s))$ that can be executed in the real world, where accurate object state is unavailable, and that is designed for maximizing sim-to-real transfer using components described in Section 5.2. We could attempt to train this policy via supervised Behavior Cloning (BC) [15] on a dataset of trajectories from the teacher, but we find this suffers from covariate shift, where the student visits different states than the teacher (in which it fails). We instead perform DAgger-style [14] interactive imitation learning (IIL) of teacher actions on states from trajectories obtained by executing the *student* policy. We refer to this as IIL, in which we perform optimization alongside data-collection (rather than alternating collection and supervised learning as in [14]). Specifically, our sim-to-real student policy π_θ^{s2r} is continuously executed in simulation to collect trajectories stored in a replay buffer $\mathcal{D}_{\text{s2r}}$. This policy is trained to maximize the log-likelihood of actions from π_e:

$$\mathcal{L}_{\text{IIL-s2r}}(\theta) = \mathop{\mathbb{E}}_{(y,\tau)\sim\mathcal{D}_{\text{s2r}}} \left[\sum\nolimits_{s_t \in \tau} \log \pi_\theta^{\text{s2r}}(a_t|o(s_t)) \big| a_t \sim \pi_e(\cdot|s_t,y) \right] , \tag{1}$$

where (y,τ) is a pair of object information y and trajectory $\tau = (s_0, a_0, s_1, \ldots, s_T)$. Note that if we replace the collected data $\mathcal{D}_{\text{s2r}}$ with expert data $\mathcal{D}_e$ from executing the teacher π_e, then we recover standard BC but with infinite data. The form of the teacher and the data above depends on the task.

Skill Mastery. We have access to 5 teachers indexed by the object triplet number $y \in \{1,\ldots,5\}$. Each teacher $\{\pi_e(a|s,y=1),\ldots,\pi_e(a|s,y=5)\}$ is independently trained only on the respective triplet y. The student collects trajectories on all triplets, chosen at random for every episode.

Skill Generalization. We use the same technique for the generalization task, except that now we have access to a single general teacher $\pi_e(\cdot|s,y)$. This generalist is conditioned on parameters y of object deformation (see Figure 2) and can specialize to different objects. During IIL, trajectories in $\mathcal{D}_{\text{s2r}}$ are from the training objects, as the test objects will not be seen until final evaluation.

4.3 One-Step Offline Policy Improvement for Sim-to-Real Policies

Once we obtained a sim-to-real policy, we consider improving its behavior on the real robots. We perform a single step of policy improvement in which we: 1) deploy π_θ^{s2r} on the robots to collect a dataset $\mathcal{D}_{\text{real}}$, 2) train a new policy π_θ^{imp} offline, using a filtering function $f(s,a,\tau)$, and 3) deploy the improved policy on the robot again for evaluation (1-3 could be iterated to obtain a policy improvement loop). The offline training step aims to maximize the following objective:

$$\mathcal{L}_{\text{IMP}}(\theta; f) = \mathop{\mathbb{E}}_{(y,\tau)\sim\mathcal{D}_{\text{real}}} \left[\sum\nolimits_{s_t,a_t \in \tau} f(s_t,a_t,\tau) \log \pi_\theta^{\text{imp}}(a_t|o(s_t)) \right] . \tag{2}$$

We use the filter $f(s,a,\tau) = \exp(A^{\pi_\theta^{\text{imp}}}(o(s),a)/\alpha)$ with temperature α (a hyperparameter) and learned Q-function (to calculate advantage), referred to as CRR-IMP, which recovers the CRR-exp [23]/AWAC [24] objectives, but with data from the deployed policy only. Unlike in Section 4.1, *we here use a binary, sparse, reward* for success per step, which is available on our real system. We also consider $f(s,a,\tau) = r(s_T)$, referred to as BC-IMP, which corresponds to BC on successful trajectories only (a stack at the end of the episode).

5 Experiments: Solving RGB-Stacking

We outline our efforts to solve the two stacking tasks we proposed in Section 3.2. In both tasks we evaluate our agents on 5 fixed object triplets, and we assume access to unlimited simulated data but only to a fixed budget ($< 100{,}000$) of real episodes. In *Skill Mastery* we have access to the fixed object triplets during all stages of training. In *Skill Generalization* our agents only have access to the training object set; the 5 fixed triplets are now held out both in simulation and the real world.

In all experiments we use a standardized evaluation protocol: the positions of the objects are randomized at the beginning of each episode, and the agent has 20 seconds at $20\,\mathrm{Hz}$ to stack the red object on the blue one. We define task success as having a stack at the *end* of the episode. This definition was chosen to exclude cases in which a stack is briefly achieved incidentally in the middle of an episode when the red object is above the blue one but subsequently falls off. In simulation, we evaluate on the training object set by running each policy on $5{,}000$ random object triplets for 2 episodes each. For the 5 fixed test triplets, we evaluate each policy for $1{,}000$ episodes per triplet. In the real world, we only evaluate on the test triplets, as evaluating on a sufficiently large sample of training objects to obtain statistics is impractical. We use one robot per triplet (evaluating for 200 episodes), for a total of $1{,}000$ episodes per policy. Mean performance is calculated by averaging the success rate across the triplets. For state-based policies and vision-based policies trained on real data we report the average across 2 training runs (i.e. random seeds), and for the vision-based policies distilled from a teacher we report the average across 4 runs: 2 distillation runs for each of the 2 teacher seeds. Results for all runs individually are in Appendix E.

5.1 Evaluation on the Simulation Stacking Benchmark

We first evaluate several baselines on our simulated benchmark task. We are interested in the following questions: *(a)* How difficult is the task in simulation? *(b)* Would it be preferable to train directly from vision? *(c)* How does interactive imitation and the design choices for it affect performance compared to the state-based teacher?

To assess the task difficulty, we tuned a scripted agent with access to the object positions and evaluated it on both our training set and our test triplets. We also hired 4 individuals with no relation to the research team to attempt the sim task via teleoperation and a game pad for 846 episodes. For details on these indicative baselines, see Appendix C. These demonstrate the difficulty of the task as in both cases the success was lower than 50% (see Table 1).

Next, we evaluate our benchmark tasks in the dense reward setting with MPO from state. A first finding was that the state agents, which are given the full pose of each of the three objects, learn the task 10 times faster than the vision agent; as illustrated in Appendix E, using a vision agent to train from scratch on the robot would be impractical for these tasks. The MPO state-based agents obtain 79.3% on the test triples in the skill mastery setting and 68.8% on the training objects in the skill generalization setting, significantly outperforming the scripted agent. However, *generalization* from training to test objects is only slightly higher than the scripted agent's performance (47.8%). This is because these state-based agents are conditioned on object parameters, which are out of distribution for the test triplets. After obtaining vision agents via IIL-s2r we observe (Table 1, bottom) that although the training set performance drops, performance improves for the test triplets, as the agents can now generalize from visual cues to make inferences about the shape.

We next compare the interactive imitation learning setting (IIL) to learning from teacher trajectories with a behavior cloning loss (BC). Rather than explicitly constructing a fixed size dataset, we generate the data on-the-fly by executing the teacher continuously during training. As shown in Table 1, learning from data generated by the student (interactive imitation) is crucial: IIL performed significantly better than the alternative in both task settings.

5.2 Evaluation on the Real-World Stacking Benchmark

We then investigate the following questions related to RGB-stacking in the real world: *(a)* Is it possible to solve the challenging tasks on real robots with a single vision policy, and is that better than a scripted baseline tuned for the test triplets? *(b)* How well does zero-shot transfer of sim-trained agents solve the tasks, and which of our decisions were important for zero-shot performance? *(c)* Do we get a single-step improvement with BC-IMP and CRR-IMP from using data collected in the real world? *(d)* And how does the data distribution affect the performance of these algorithms?

Method	Simulation Success	
	Train Set	Test Triplets
Human teleoperator	-	46.6%
Scripted agent	45.0%	43.1%
Skill Mastery		
State teacher	N/A	79.3%
BC	N/A	52.4%
IIL-s2r	N/A	71.7%
Skill Generalization		
State teacher	68.8%	47.8%
BC	49.4%	41.2%
IIL-s2r	64.7%	56.0%

Table 1: Simulation Success. Comparison of distillation methods for solving, in simulation, our RGB-stacking tasks from vision and proprioception. These methods assume we have either a large dataset generated by a teacher policy (BC) or the teacher policy itself to supervise the student's interaction (IIL). Interactive learning leads to much higher performance than learning only from the teacher's behavior.

Method	Sim Success	Real Success
	Test Triplets	Test Triplets
Skill Mastery		
IIL-s2r (deterministic)	71.7%	67.9%
IIL-s2r (stochastic)	71.7%	54.3%
No Transformer	73.3%	69.3%
No image augmentation	72.8%	65.7%
No action delay	73.5%	68.7%
No binary gripper	74.4%	30.0%
MSE & no binary gripper	72.3%	27.6%
Skill Generalization		
IIL-s2r	56.0%	51.9%
No Transformer	51.3%	45.4%
No object parameters	53.7%	41.1%

Table 2: Sim-to-Real Transfer Success. Ablations of the components of the sim-to-real policy. The simulation and real-world evaluations use 20,000 and 4,000 episodes for the triplet average, respectively. The choice of using binary gripper actions for the policy seemed to be the most important factor for sim-to-real transfer success, followed by the choice of executing the deterministic policy on the robots.

As described in Section 4.2, we distill a vision-based policy from a single or multiple state-based experts in simulation. These are 5 specialists for the *Skill Mastery* task and a single generalist for the *Skill Generalization* task. We can then directly execute this distilled vision policy on the real robots. As discussed previously, this decoupling allowed us to quickly iterate and investigate what aids sim-to-real transfer for these tasks. We compare, in Table 2, the simulation performance and the zero-shot sim-to-real transfer performance when ablating these various choices for our method. Policies were trained up to 1 million learner steps, and the sim-to-real policy was selected to be the one with the highest performance in the training setting for each task. As our policies are stochastic, for inference we have the option to execute a random sample from the action distribution or its mode (i.e. the action with highest probability). We execute the **stochastic** policy in simulation and the **deterministic** mode of the policy on the real robots, unless otherwise specified. We are able to achiev high performance of 67.9% on the Skill Mastery task and 51.9% on the Skill Generalization task in the real world. Qualitatively, we also see in Figure 3 that we can, with a single vision-based agent, show a diverse set of skills to address the challenges needed to solve the task for the 5 specific triplets we have chosen. Videos of the behavior are in the supplementary.

Ablations. We ablate different algorithmic choices in Table 2 (extended table in the supplementary). One of our decisions was on the BC loss for distillation, which can either be the **negative log-likelihood** or **the mean squared error (MSE)**, which respectively correspond to $-\log \pi_\theta^{\text{s2r}}(a_e|o(s_t))$ or $\mathbb{E}_{a\sim\pi_\theta^{\text{s2r}}}[||a - a_e||^2]$, where a_e are actions from the teacher (the two are not equivalent for the Gaussian case as the policy variance is state dependent). We also decided to use a hybrid action space, where the 3D Cartesian and angular velocities are modeled as continuous actions (Gaussian), but we have a **binary gripper action** (Bernoulli). This applies to both state-based and vision-based policies. This choice seemed to be the most important factor for sim-to-real transfer success. However, note that the effects are only visible when transferring (corresponding simulation performance does not change). For the *Skill Generalization* task, the state-based expert is conditioned on the **object parameters** y. However, the object parameters are not passed to π_θ^{s2r}— the agent needs to infer physical object properties from image observations. We also chose to use a **Transformer** model [31] after the ResNet vision encoder for the policy's network architecture, which gives the agent access to temporal information with an attention mechanism. The Transformer does not have a significant effect on *Skill Mastery*. However, it seems to be able to utilize the skill variety of a state-based expert conditioned on the object parameters a lot more successfully, when compared to an MLP model with the same number of weights. This is clear both in simulation and the real world (and we hence used it in all experiments). Random **action execution delays** of $0, 1$ or 2 timesteps, and standard **image augmentation** (random color perturbations and image translations) did not seem to affect transfer but we decided to include them in our training procedure to increase our agents' robustness to natural perturbations.

Method	Real-Robot Success					
	Triplet Avg.	Triplet 1	Triplet 2	Triplet 3	Triplet 4	Triplet 5
Scripted agent	51.2%	36.3%	23.0%	34.4%	84.9%	77.6%
Skill Mastery						
BC (scripted agent data)	53.4%	34.5%	26.8%	38.3%	84.5%	82.8%
CRR (scripted agent data)	43.4%	20.5%	28.0%	38.3%	64.0%	66.3%
IIL-s2r (sim-to-real)	67.9%	72.8%	48.8%	61.0%	85.1%	71.9%
BC-IMP (sim-to-real agent data)	74.6%	75.5%	60.8%	70.8%	87.8%	78.3%
CRR-IMP (sim-to-real agent data)	81.6%	87.3%	68.3%	75.3%	88.3%	88.8%
Skill Generalization						
IIL-s2r (sim-to-real)	51.9%	25.0%	39.6%	36.5%	91.1%	67.0%
—Suboptimal agent for training set data	32.6%	21.5%	16.5%	17.0%	60.0%	48.0%
BC-IMP (suboptimal agent data)	49.0%	23.0%	39.3%	39.3%	77.5%	66.0%
CRR-IMP (suboptimal agent data)	53.8%	32.0%	33.0%	47.0%	88.0%	69.0%

Table 3: Real-Robot Success. Different approaches for solving our RGB-stacking tasks in the real world. The kind of data used matters significantly for BC and CRR: using scripted agent data does not result in improved performance, but using data from a sim-to-real agent does. Note that the sim-to-real data used were collected by a single agent per task. For Skill Generalization, as data collection with multiple triplets from the training set was exceptionally time-consuming, we used a suboptimal agent trained earlier in our investigation. This agent was not trained with the best settings we now have for sim-to-real. See text for more details.

Our sim-to-real approach also outperforms a scripted agent tuned on the test triplets, as shown on Table 3. For the Skill Mastery task, we further improve results via offline RL with BC (BC-IMP) on $32{,}651$ successful real episodes collected by the scripted agent and with CRR (CRR-IMP) on a total of $67{,}446$ episodes, which included the unsuccessful ones collected by the same agent. Hyperparameters and architecture choices were selected after such an investigation in simulation. As evident in the table, learning from scripted agent data is not better than sim-to-real. Qualitatively, the policies learned seem to exhibit the same "robotic" movements as the scripted policy that generated the data they were trained on. However, a single policy improvement step based on data collected by a sim-to-real agent with an average success of 69.7%: $85{,}213$ episodes ($58{,}979$ successful) yields 74.6% (BC-IMP) and 81.6% (CRR-IMP), resulting in policies with remarkable stacking consistency.

As data collection for the training object set is particularly time-consuming, we only did a single collection ($38{,}446$ episodes, 37.4% of which were successful) with an earlier iteration of a sim-to-real agent for the Skill Generalization task (performance of 32.6% when evaluated on the test triplets). As before we trained BC-IMP and CRR-IMP on this data and again observed improved performance on the test triplets, this time using only the episodes collected from the training object set. Note that even though the suboptimal sim-to-real agent trained on the training set of objects was performing worse than the scripted agent, and the data in this case was more limited and of different objects, BC-IMP and CRR-IMP trained on this more diverse data lead to a significant improvement. Finally, Table 3 showcases the variation in difficulty of the 5 triplets: all methods perform best on triplets 4 and 5, as these can sometimes be solved with a stereotypical "pick and place" behavior. In contrast, triplets 1, 2, and 3 each require specialized, advanced, strategies as indicated by the large gap between sim-to-real performance and scripted performance in the Skill Mastery task.

6 Conclusion

We studied the problem of vision-based stacking with a large variety of geometric objects that require a diverse set of stacking strategies. We propose a benchmark for studying this problem in a principled way, and make significant progress in solving two tasks involving both skill mastery on specific objects and generalization across them. We use simulation-trained agents to collect data in the real world, which in turn can be used for further performance improvement with offline RL. Our best agent is a single vision-based policy that is capable of a variety of stacking strategies and achieves high performance. Finally, we provide thorough real-world evaluations and discuss what is important for solving our tasks. Despite this success, our benchmark still poses many open challenges as, for example, shown by the gap between Skill Mastery and Skill Generalization results and we hope that it can help development of new methods for learning general policies (e.g. by further adaptation at robot deployment time).

References

[1] M. A. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. V. de Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by playing - solving sparse reward tasks from scratch. *CoRR*, abs/1802.10567, 2018. URL http://arxiv.org/abs/1802.10567.

[2] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and imitation learning for diverse visuomotor skills. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi:10.15607/RSS.2018.XIV.009.

[3] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Żołna, Y. Aytar, D. Budden, M. Vecerik, et al. A framework for data-driven robotics. *arXiv preprint arXiv:1909.12200*, 2019.

[4] R. Jeong, Y. Aytar, D. Khosid, Y. Zhou, J. Kay, T. Lampe, K. Bousmalis, and F. Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation. *arXiv preprint arXiv:1910.09470*, 2019.

[5] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox. Adaptive curriculum generation from demonstrations for sim-to-real visuomotor control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6498–6505. IEEE, 2020.

[6] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. *CoRR*, abs/1509.06825, 2015. URL http://arxiv.org/abs/1509.06825.

[7] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *CoRR*, abs/1603.02199, 2016. URL http://arxiv.org/abs/1603.02199.

[8] 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. *CoRR*, abs/1806.10293, 2018. URL http://arxiv.org/abs/1806.10293.

[9] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE, 2018.

[10] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2052–2062. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/fujimoto19a.html.

[11] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[12] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[13] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. 2018.

[14] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[15] M. Bain and C. Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129. Oxford University Press, 1996.

[16] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[17] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

[19] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *CoRR*, abs/1710.06537, 2017. URL http://arxiv.org/abs/1710.06537.

[20] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *CoRR*, abs/1707.02267, 2017. URL http://arxiv.org/abs/1707.02267.

[21] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *CoRR*, abs/1710.06542, 2017. URL http://arxiv.org/abs/1710.06542.

[22] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[23] Z. Wang, A. Novikov, K. Zolna, J. S. Merel, J. T. Springenberg, S. E. Reed, B. Shahriari, N. Siegel, C. Gulcehre, N. Heess, and N. de Freitas. Critic regularized regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] A. Nair, A. Gupta, M. Dalal, and S. Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[25] R. Jeong, J. T. Springenberg, J. Kay, D. Zheng, A. Galashov, N. Heess, and F. Nori. Learning dexterous manipulation from suboptimal experts. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[26] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, Jan 2018. ISSN 1558-3783. doi:10.1109/TASE.2016.2600527.

[27] M. Ahn, H. Zhu, K. Hartikainen, H. Ponte, A. Gupta, S. Levine, and V. Kumar. Robel: Robotics benchmarks for learning with low-cost robots, 2019.

[28] M. Mason. The mechanics of manipulation. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 544–548, March 1985. doi:10.1109/ROBOT.1985.1087242.

[29] C. Ferrari and J. Canny. Planning optimal grasps. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2290–2295 vol.3, 1992. doi:10.1109/ROBOT.1992.219918.

[30] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, abs/1703.09312, 2017. URL http://arxiv.org/abs/1703.09312.

[31] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.