Adaptive Curriculum Generation from Demonstrations for Sim-to-Real Visuomotor Control

Lukas Hermann*, Max Argus*, Andreas Eitel, Artemij Amiranashvili, Wolfram Burgard, Thomas Brox

***Abstract*— We propose Adaptive Curriculum Generation from Demonstrations (ACGD) for reinforcement learning in the presence of sparse rewards. Rather than designing shaped reward functions, ACGD adaptively sets the appropriate task difficulty for the learner by controlling where to sample from the demonstration trajectories and which set of simulation parameters to use. We show that training vision-based control policies in simulation while gradually increasing the difficulty of the task via ACGD improves the policy transfer to the real world. The degree of domain randomization is also gradually increased through the task difficulty. We demonstrate zero-shot transfer for two real-world manipulation tasks: pick-and-stow and block stacking. A video showing the results can be found at `https://lmb.informatik.uni-freiburg.de/projects/curriculum/`**

I. INTRODUCTION

Reinforcement Learning (RL) holds the promise of solving a large variety of manipulation tasks with less engineering and integration effort. Learning continuous visuomotor controllers from raw images circumvents the need for manually designing multi-stage manipulation and computer vision pipelines [1]. Vision-based closed-loop control can increase robustness of robots performing real-world fine manipulation tasks. Prior experience has shown that learning-based methods can cope better with complex semi-structured environments that are hard to model in a precise manner due to contacts, non-rigid objects or cluttered scenes [2], [3], [4].

A major challenge preventing deep reinforcement learning methods from being used more widely on physical robots is exploration. RL algorithms often rely on random exploration to search for rewards. This hinders the application to real-world robotic tasks in which the reward is too sparse to ever be encountered through random actions. Additionally, in the real world random exploration can also be dangerous for the robot or its environment.

A common strategy to address the exploration problem is reward shaping, in which distance measures and intermediate rewards continuously provide hints on how to reach the goal [5]. Reward shaping is typically task specific and requires manual setup as well as careful optimization. This contradicts the purpose of using RL as a general approach and can bias policies to satisfy shaped rewards.

Poor exploration can also be compensated by providing demonstration trajectories and performing Behavior Cloning (BC) [6]. Pre-training with behavior cloning can guide the RL algorithm to initial rewards, from where it can be optimized further [7]. Behavior cloning usually requires a substantial amount of expert demonstrations.

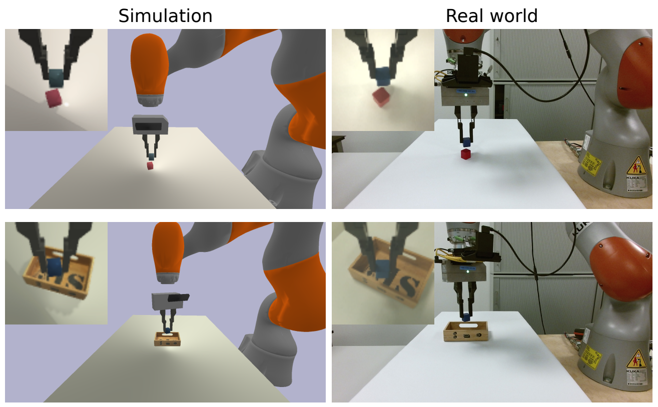

Fig. 1: Adaptive Curriculum Generation from Demonstrations utilizes only 10 demonstration trajectories to enable learning visuomotor policies for two fine manipulation tasks: block stacking (top) and pick-and-stow (bottom). The visuomotor policies are trained in simulation and transferred without further training to a physical robot. The image-in-image shows the egocentric view of the arm-mounted RGB camera. The third person view is not provided to the robot.

This paper tackles the exploration problem with curriculum learning based on a few manual demonstrations. We present an adaptive curriculum generation algorithm that controls difficulty by controlling how initial states are sampled from demonstration trajectories as well as controlling the degree of domain randomization that is applied during training. The algorithm continually adapts difficulty parameters during training to keep the rewards within a predetermined, desired success rate interval. Consequently, the robot can always learn at the appropriate difficulty level, which speeds up the convergence of the training process and the final generalization performance. The method is simple and applicable to a variety of robotic tasks. We demonstrate its performance on real world pick-and-stow and block stacking tasks. We apply the curriculum learning in conjunction with policy learning in a physics simulator. Afterwards, the learned policies are transferred without further re-training to the physical robot. The advantages of training in simulation include drastically increased learning speed through distributed training, no human involvement during training, improved safety, and added algorithmic possibilities due to access to simulator state information.

We present a novel perspective on domain randomization

* First two authors contributed equally. All authors are with the University of Freiburg, Germany. Wolfram Burgard is also with the Toyota Research Institute, USA. This work has been supported partly by the BMBF grant No. 01IS18040B-OML and the DFG grant No. BR 3815/10-1. {hermannl,argusm,eitel}@cs.uni-freiburg.de

inside our curriculum learning framework. Our algorithm automatically learns when to increase the amount of visual domain randomization and dynamics randomization during the training process, until the policy exhibits the desired degree of domain invariance required for a successful simulation to reality transfer.

The main contributions of this paper are: 1) a curriculum generation method for learning with sparse rewards that only requires a dozen demonstrations, 2) an algorithm for automatic and controlled scaling of task difficulty, 3) a unified treatment of demonstration sampling and domain randomization as task difficulty parameters of the curriculum, 4) zero-shot transfer from simulation to real-world for two robot manipulation tasks.

II. RELATED WORK

A. Training Curricula

The seminal paper by Bengio *et al.* [8] shows that curriculum learning provides benefits for classical supervised learning problems such as shape classification. Florensa *et al.* [9] propose a reverse curriculum for reinforcement learning that requires only the final state in which the task is achieved. Their curriculum is generated gradually by sampling random actions to move further away from the given goal and thus reversely expanding the start state distribution with increasingly more difficult states. The key difference of our method is that we extend the idea of curriculum learning to task difficulty parameters, which goes beyond sampling start states nearby the goal. Further, we sample backwards from the demonstration trajectory in comparison to randomly sampling backwards in action space, which is likely to fail for difficult states that are unlikely to be encountered randomly. Another approach is to generate a linear reverse curriculum, a baseline that we compare against in our experiments [10]. A more evolved approach generates a curriculum of goals using a Generative Adversarial Network (GAN) but cannot handle visual goals [11]. Learning by playing also generates a form of curriculum, where auxiliary tasks are leveraged to learn increasingly more complicated tasks [12]. Nevertheless, it is not straightforward to design auxiliary tasks that can benefit the final goal task.

B. Learning from Demonstrations and Imitation

The most common approach for learning from demonstrations is supervised learning, also known as behavior cloning. Behavior cloning has shown to enable training of successful manipulation policies [6], but usually has to cope with the compounding-error problem and requires hundreds or even thousands of expert demonstrations. Several methods have combined behavior cloning with RL, either in a hierarchical manner [13] or as a pre-training step to initialize the RL agent [7]. Other methods leverage demonstration data for training off-policy algorithms by adding demonstration transitions to the replay buffer [14]. In comparison to the those works our method does not try to copy the demonstrated policy. Only the visited states are used as initial conditions and the actions taken during demonstrations are not required. Therefore, our method is more robust against suboptimal demonstrations.

Demonstration data can also be provided in form of raw video data [15], [16], [17], [18]. These approaches can work if sufficient demonstration data is provided, while our method requires only around a dozen demonstrations. Nair *et al.* [19] combine off-policy learning with demonstrations to solve block stacking tasks. They use Hindsight Experience Replay [20] for multi-goal tasks, but this method is not applicable to visual domains where the desired goal configuration is not given, as in our tasks. Zhu *et al.* [21] combine imitation learning and RL for training manipulation policies in simulation. They report preliminary success for transferring the learned policies into the real world but also challenges, due to different physical properties of simulation and the real world. We address this problem by leveraging our curriculum generation approach for gradually training with more realistic simulation parameters.

C. Sim-to-Real and Domain Randomization

For sim-to-real transfer of our policy we build upon an efficient technique called domain randomization [22]. It has been successfully applied to transfer imitation learning policies for a pick-and-place task on a real robot [23]. Since our visuomotor controller operates in a first-person view with sensor data from a camera-in-hand robot setup, there is significantly less background clutter, so we generally use a smaller degree of domain randomization. Nevertheless, as we show in experiments domain randomization can harm convergence during RL training, so we circumvent this problem by incorporating domain randomization into our adaptive curriculum. Domain randomization has also been extended to dynamics randomization [24], which we also incorporate into our approach. A recent method proposes to close the sim-to-real gap by adapting domain randomization with experience collected in the real world [25]. Experiments with a real robot show impressive results for a drawer-opening and a peg-in-hole task, but the learned policies are not based on visual input and it is unclear if the method also works with sparse rewards. In concurrent work, OpenAI *et al.* [26] have also considered adaptive domain randomization for sim-to-real dexterous manipulation.

Sim-to-real transfer can also be viewed from a transfer learning or domain adaptation perspective [27]. Here, methods that use GANs have been successfully applied for instance grasping on a real robot [28]. Nevertheless, training the GAN requires a great amount of real-world training data (on the order of 100,000 grasps). Recent meta-learning methods seem to be suitable to reduce the amount of data needed in the real world for efficient domain adaptation [29]. Those transfer learning methods are outside of the scope of our work because we focus on zero-shot sim-to-real transfer.

D. Motor Control for Manipulation

Model-based reinforcement learning techniques with probabilistic dynamics models have been proposed to learn control policies for block stacking [30] and multi-phase manipulation tasks [31]. Guided Policy Search has been applied to visuomotor control tasks [32]. The mentioned approaches work well in the local range of trained trajectories, but generalize less to larger variations in goal and robot state positions, which is required for our tasks.

III. ADAPTIVE CURRICULUM GENERATION FROM DEMONSTRATIONS

In this section we present Adaptive Curriculum Generation from Demonstrations (ACGD), a method to overcome exploration difficulties of reinforcement learning from sparse rewards in simulated environments. Depending on the current success rate, ACGD automatically schedules increasingly difficult subtasks by shaping the initial state distribution and scaling a set of parameters that control the difficulty of the environment, such as the degree of domain randomization.

A. Reinforcement Learning

In reinforcement learning an agent makes some observation (o_t) of an underlying environment state, which is used by a policy to compute an action $a_t = \pi(o_t)$. This produces transitions consisting of (o_t, a, r, o_{t+1}) for discrete timesteps. In a sparse reward setting a correct sequence of actions produces rewards (r_t). The policy is optimized to maximize the discounted future rewards, $R = \sum_{i=t}^{T} \gamma^{(i-t)} r_i$, called return.

A number of different RL algorithms exist, they can be categorized as either off-policy algorithms or on-policy algorithms based on if they make use of transitions that are not generated by the current policy being optimized. In our work we use the Proximal Policy Optimization (PPO) [33] algorithm, as on-policy algorithms are more suited to gradually changing the environment parameters.

B. Reverse-Trajectory Sampling

During training in simulation it is possible to initialize episodes from arbitrary states of demonstration trajectories. We bootstrap exploration by using reverse-trajectory sampling, meaning we initially sample states from the end of the demonstration trajectories. These states are close to achieving sparse rewards, making these tasks much easier and solvable using RL. As the training progresses we successively sample states closer to the beginning of demonstration trajectories. This creates a curriculum in which the agent learns to solve a growing fraction of the task.

However, sampling states from the demonstration trajectories restricts the policy to observing initial states present in the recorded demonstrations. Especially if we want our algorithm to work with few demonstrations this presents a potential source of bias. To overcome this problem we mix resets from demonstrations with regular resets of the environments which have automatic randomization of initial states. The choice of sampling demonstration or regular resets is made by a mixing function depending linearly on the success rates sr_r of regular resets episodes and the overall progress of the training. As opposed to [10], who claim that it is important for reverse curriculum learning to include previously sampled sections throughout the training in order to prevent catastrophic forgetting, our experiments suggest that this is not the case.

Algorithm 1: Adaptive Curriculum Generation from Demonstrations

```
Input  : Iterations $N$, initial policy $\pi_0$, increment $\varepsilon$,
         task params $\mathscr{H}^j = \{\mu^j_{init}, \sigma^j_{init}, \mu^j_{end}, \sigma^j_{end}\}$,
         reward interval $[\alpha, \beta]$
Output: final policy $\pi_N$
1  $sr_d,\ sr_r,\ \delta_d,\ \delta_r \leftarrow 0$;
2  for $i \leftarrow 1$ to $N$ do
3      with probability $p = 0.5(sr_r + i/N)$ do
4          sample_regular_restart($\mathscr{H},\ \delta_r$);
5          $sr,\ \delta \leftarrow sr_r,\ \delta_r$;
6      otherwise
7          sample_demonstration_restart($\mathscr{H}_d, \delta_d$);
8          $sr,\ \delta \leftarrow sr_d,\ \delta_d$;
9      $rollouts \leftarrow$ generate_rollouts($\pi$);
10     $\pi \leftarrow$ update_policy($rollouts,\ \pi$);
11     $sr \leftarrow$ update_success_rates($rollouts$);
12     $\delta \leftarrow \delta + \varepsilon \cdot \mathbb{1}_{(sr>\beta)} - \varepsilon \cdot \mathbb{1}_{(sr<\alpha)}$;
   end
```

C. Task and Environment Parameter Adaptation

Apart from the distance between start states and goal states, the difficulty of a task also depends on a set of factors, such as the degree of domain randomization, intrinsics of the physics simulator or criteria that define the task completion. As an example, the complexity of block stacking depends significantly on the bounciness of the blocks. Therefore, we design our tasks such that their difficulty can be controlled by a set of parameters $\mathscr{H}$. Examples can be found in Table II, most of these are not task specific. At the beginning of the training all parameters are set to the intuitively easiest configuration (e.g. less bouncy objects or smaller initial distance between objects). The parameters that determine the degree of appearance or dynamics randomization are initially set to a minimum. During training we scale the variance of the sampling distributions and thus increase the difficulty by linearly interpolating between 0 and the maximal value chosen based on what is realistic. Our experiments clearly suggest, that it is beneficial to gradually increase the degree of randomization over time since too much domain randomization from the beginning slows down training or might even prevent the policy from learning the task at all. To our knowledge, we are the first to apply curriculum learning to sim-to-real transfer. When sampling initial states from demonstration it is not possible to randomize all parameters because some configurations are pre-determined by the demonstration. This results in a different set of parameters being randomized for each type of reset, see Table II.

D. Adaptive Curriculum Generation

The challenge of curriculum learning is to decide a good strategy to choose the appropriate difficulty of start states and task parameters in the course of the training. Previous

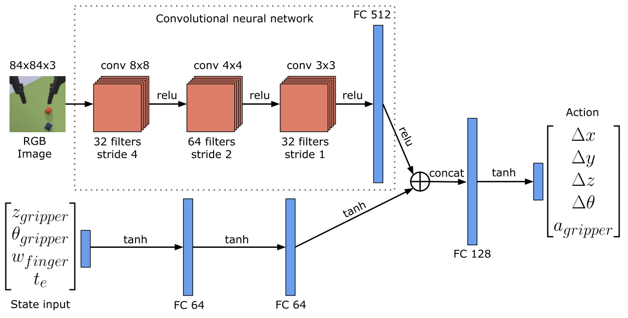

Fig. 2: Architecture of the policy network. The value function has the same architecture apart from having a single output for the value. Policy and value functions share the weights of the CNN (dotted box).

approaches sampled initial states uniformly [34], [19], [21] or linearly backwards [10].

However, sampling states from the end of the demonstration trajectories for too long unnecessarily slows down the training because the policy is trained on subtasks that it has already learned to master. On the other hand, sampling more difficult task configurations too fast may prevent the policy from experiencing any reward at all. Especially tasks with long episode lengths often do not have a constant difficulty at every stage. Consider for instance a stacking task: it consists of both easier parts that only require straight locomotion and more difficult bottleneck moments like grasping and placing the object. Our method adaptively generates a curriculum, such that the learning algorithm automatically dedicates more time on hard parts of the task, while not wasting time at straightforward sections.

Intuitively, we want the probability of experiencing a reward to be neither too high, nor too low. Instead, our goal is to confine the probability within a desired reward region $\alpha \leq \mathbb{P}(R_t > 0|\pi) \leq \beta$, where R_t denotes the return of a rollout started from a state sampled from the demonstration data at timestep t and the interval $[\alpha, \beta]$ are hyperparameters which we set to $[0.4, 0.6]$ after empirical testing. This is inspired by the *Goals of Intermediate Difficulty* of [11]. For sparse bi-modal tasks, the probability $\mathbb{P}(R_t > 0|\pi)$ corresponds to the expected success rate of the task.

We control all difficulty parameters with two coefficients δ_d and $\delta_r \in [0,1]$, which regulate the difficulty of resets from demonstrations and regular resets respectively by scaling the variances of $\mathscr{H}$ linearly w.r.t δ. A δ close to 0 corresponds to the easiest and close to 1 corresponds to the most difficult task setting (i.e. initializing episodes further away from the goal and sampling task parameters with higher variance). Our method tunes the difficulty during training to ensure that the success rates of regular and demonstration resets (sr_r and sr_d) stay in the desired region, as shown in the pseudo-code in Algorithm 1. An example of this optimization can be seen in Fig. 3.

IV. EXPERIMENTS

We demonstrate the effectiveness of the proposed curriculum generation via two manipulation tasks: pick-and-stow and stacking small blocks; see Fig. 1. We posed the following questions: 1) can curriculum learning with demonstrations enable learning tasks with sparse rewards that do not succeed without curriculum learning? 2) what is the generalization performance compared to the existing behavioral cloning and reinforcement learning with shaped rewards? 3) does our adaptive curriculum outperform other curriculum baselines? 4) can visuomotor policies trained in simulation generalize to a physical robot?

A. Experimental Setup

For training and evaluation of the policies we re-created the two tasks and the robot setup in a physics simulator. We aligned simulation and real-world as closely as possible; for a side-by-side comparison see Fig. 1.

Our KUKA iiwa manipulator is equipped with a WSG-50 two finger gripper and an Intel SR300 camera mounted on the gripper for an eye-in-hand view. The control of the end effector is limited to the rotation around the vertical z-axis, such that the gripper always faces downwards, it is parameterized as a reduced 5 DoF continuous action $a = [\Delta x, \Delta y, \Delta z, \Delta\theta, a_{gripper}]$ in the end effector frame. Here, $\Delta x, y, z$ specify a Cartesian offset for the desired end effector position, $\Delta\theta$ defines the yaw rotation of the end effector and $a_{gripper}$ is the gripper action that is mapped to the binary command open/close fingers.

The observations that the policy receives are a combination of the 84×84 pixels RGB camera image and a proprioceptive state vector consisting of the gripper height above the table, the angle that specifies the rotation of the gripper, the opening width of the gripper fingers and the remaining timesteps of the episode normalized to the range $[0,1]$, see Fig. 2.

Consistent with results of [35], preliminary experiments showed that it is beneficial to include the proprioceptive features. The policy network is trained using PPO, but our approach can be used with any on-policy RL algorithm. For all experiments, training runs 8 environments in parallel for a total number of 10^7 timesteps, this takes approximately 11 hours on a system with one Titan X and 16 CPU cores. During training the performance of the policy is always evaluated on the maximum task difficulty in order to obtain comparable results between different methods. The policy output is the mean and standard deviation of a diagonal Gaussian distribution over the 5-dimensional continuous actions. The value function yields a scalar value as output. We use the *Adam* [36] optimizer with an initial learning rate of 0.00025, which is linearly decayed in the course of the training.

B. Experiments in Simulation

For experiments in simulation we use the PyBullet Physics simulator [37]. We compare our approach against several baselines: 1) training PPO with sparse and shaped rewards, 2) behavior cloning, 3) PPO with behavior cloning initialization, 4) several standard non-adaptive curriculum learning methods that use demonstrations. For both tasks, we recorded a set of 10 manual demonstrations for curriculum learning and

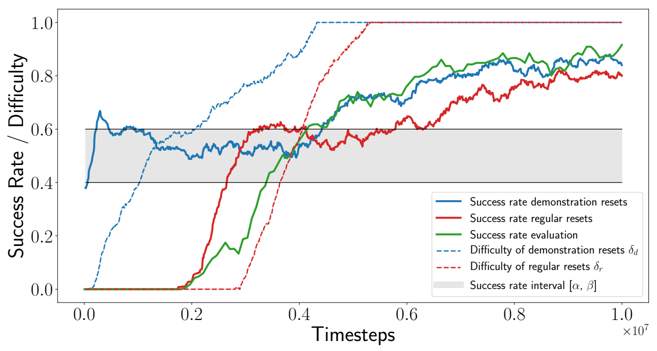

Fig. 3: Example training run showing evolution of success rates and difficulty coefficients δ over the course of training. First the success rate and thus difficulties increase for resets from demonstrations (blue curves), followed by increases for regular resets (red). The plot shows that the success rate is kept in the desired interval (grey area) by the difficulty coefficients until the highest difficulty is reached.

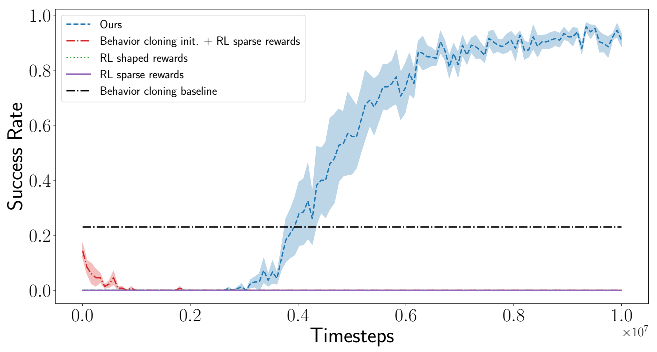

Fig. 4: ACGD vs. baseline methods for the pick-and-stow task, evaluated in simulation.

an additional set of 100 demonstrations for the BC baseline, both with a 3D mouse.

Pick-and-stow. The robot has to pick up a small block and place it within a box. The initial position of the robot and the block randomly change every episode. A sparse reward of $1-\phi$ is received only after reaching a goal state, where ϕ denotes a penalty for collisions of gripper and block with the edges of the box. The dense reward function for the RL baseline is composed of the Euclidean distance between gripper and block as well as the distance between block and a location above the box plus additional sparse rewards for successfully grasping and placing the block.

Fig. 4 shows the task success rates during training. The results show averages over five different random seeds for every experiment. Our method successfully solves the task with an average final success rate of 94%. The policy trained with BC achieves a notable success rate of 23%, but overall lacks consistency having overfitted to the small set of demonstrations. Interestingly, the policy that used BC as initialization for RL performed poorly and experienced catastrophic forgetting after few timesteps. Neither RL with sparse rewards nor RL with shaped rewards are able to completely finish the task. While the former is unable to learn any meaningful behavior, the latter learns to grasp the

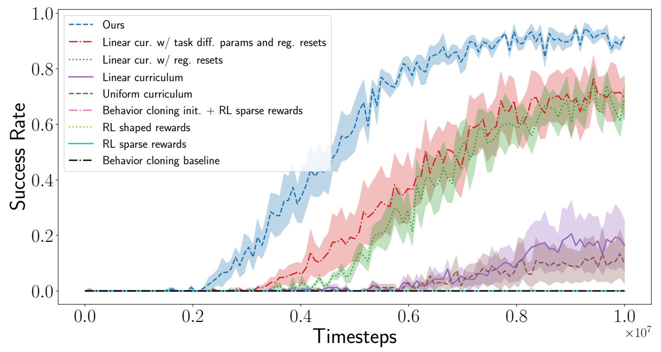

Fig. 5: ACGD vs baseline methods for the block stacking task, evaluated in simulation.

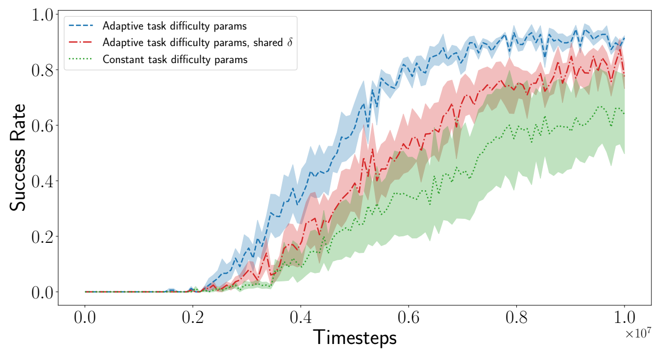

Fig. 6: Here we investigate the different methods for choosing the task difficulty for all parameters $\mathscr{H}$ besides the location of demonstration resets. We compare adaptively increasing the difficulty of those parameters during training against always training on the highest difficulty.

block but fails to place it inside the box.

Block stacking To solve the task, the agent has to stack one block on top of the other one, which involves several important subproblems such as reaching, grasping and placing objects precisely. The task features complex, contact-rich interactions that require substantial precision. We further increased the difficulty of the task compared to prior work by using smaller blocks. Since it is a task of long episode length that requires at least around 70 steps to be solved with our choice of per-step action ranges, it is particularly well suited for curriculum learning.

We define a successful stack as one block being on top of the other one for at least 10 timesteps, which indicates stable stacking. A sparse reward of $1-\phi$ is given after a successful stack where ϕ denotes a penalty for the movement of the bottom block during the execution of the task. The shaped reward function consists of a mixture of distance functions and sparse rewards similar to the pick-and-stow task.

The training progress and the evolution of difficulty coefficients are shown in Fig. 3. We see that the success rate is kept in the desired interval through the adaption of the difficulty coefficients until the highest difficulty is reached. The success rate of evaluation runs (green) is higher due to execution of a deterministic policy. Fig. 5 shows the results compared to baselines. As it is considerably harder than the previous task, none of the curriculum-free baselines

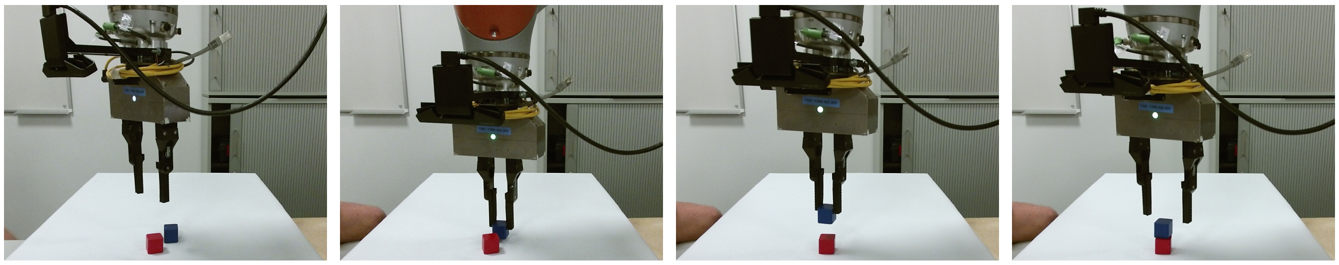

Fig. 7: Each image shows the progress of the robot attempting to stack a small blue block on top of a red block.

Task	Train	Test	Trials	Success Rate
Block stacking	Simulation	Simulation	91/100	91%
Pick-and-stow			94/100	94%
Block stacking	Simulation	Real	12/20	60%
Pick-and-stow			17/20	85%

TABLE I: Success rates in simulation and real-world.

are able to solve the stacking task. In comparison to the uniform and linear reverse curriculum learning variants, our method learns faster and achieves a better final performance, with the final success rate being more than 20% higher. For the linear curriculum variant start states are sampled linearly further away from the goal during the course of the training. Our method shows less variance across the seeds, which indicates that the adaptive curriculum improves the stability of learning. The experiment also demonstrates the importance of not learning exclusively from demonstrations, especially if the amount of demonstration data is limited. Linear curricula with regular resets, similar to [10], clearly outperform uniform and linear curriculum learning that are trained only on initial states sampled from demonstrations.

Another advantage of our method is that it can learn substantially more efficient solutions than those provided by the demonstrations. This results from the use of demonstration states, but not the actions taken. For the stacking task, manual demonstration episodes had a mean duration of 164 ± 17 transitions, while the learned solution had a mean duration of 73 ± 15 transitions. This means that in simulation our trained policy solves the task twice as fast as a human expert operating the robot with a 3D mouse.

We further evaluated how adaptively changing the task parameters $\mathscr{H}$[1] for the task difficulty and domain randomization improves the training speed and performance. In Fig. 6 we compare our full model with the following ablations: 1) *shared* δ for demonstration resets and regular resets i.e. $\delta_r = \delta_d$, 2) *constant task difficulty*, i.e. adaptive curriculum from demonstrations, but without changing the difficulty of the task parameters $\mathscr{H}$, which were set to the maximum difficulty for the complete training.

C. Experiments with Real Robot

We applied the trained policies without any additional fine-tuning on the real robot. Our results are shown in Table I. We see that despite incurring a performance penalty by evaluating on the real robot, the policies transfer with a good success rate of 85% for pick-and-stow and 60% for block stacking. Both task were evaluated with a constant episode length of 300 timesteps (15 seconds). Within the given time frame, the policy can attempt a second trial after a failed first execution of the task. This shows the advantages of a policy learned in closed-loop as it implicitly aims to re-grasp the block in case of a failed stacking or stowing attempt. Using small wooden blocks with an edge length of only 2.5cm requires a highly precise actuation. as the block tends to fall over if not placed precisely or dropped from a too large height. In contrast, polices learned with non-adaptive curriculum learning baselines were unable to achieve the sim-to-real transfer. It is difficult to compare performance with previous approaches due to the lack of clear benchmark setups. In a related approach, Zhu *et al.* [21] performed zero-shot sim-to-real transfer of a block stacking task, however, they use large deformable foam blocks for stacking. These are easier to grasp because they are made of foam and easier to stack because they are larger and have more friction. They report a success rate of 35% for stacking on the real robot, which is lower than ours.

[1]not including the trajectory sampling position

Difficulty Parameters: $\mathscr{H}_d$
bounciness of objects, num. steps stacked for task success, gripper speed, position offset of relative Cartesian position control, camera field of view, camera position and orientation, block color, table color, camera image brightness, camera image contrast, camera image hue, camera image saturation, camera image sharpness, camera image blur, light direction
Additional Regular Reset Parameters: $\mathscr{H}_a$
initial gripper height, lateral gripper offset, distance between blocks, min. final block vel. for task success, table height, height of the robot base, initial gripper rotation, block size

TABLE II: Task parameters used to adapt the difficulty of the stacking task. When initializing from demonstration states only a subset $\mathscr{H}_d$ can be randomized, regular resets randomize $\mathscr{H}_r = \mathscr{H}_d \cup \mathscr{H}_a$.

V. CONCLUSIONS

The proposed Adaptive Curriculum Generation from Demonstration (ACGD) method enables robust training of policies for difficult multi-step tasks. It does this by adaptively setting the appropriate task difficulty for the learner by controlling where to sample from the demonstration trajectories and which set of simulation parameters to use. This unified treatment of demonstration sampling and domain randomization as task difficulty improves training. In combination with domain randomization the method can train policies in simulation that achieve good success rates when evaluated on a real robot.

References

[1] F. Wirnshofer, P. S. Schmitt, W. Feiten, G. v. Wichert, and W. Burgard, "Robust, compliant assembly via optimal belief space planning," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, May 2018, pp. 1–5.

[2] A. Zeng, S. Song, K. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Qu Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, May 2018, pp. 1–8.

[3] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4238–4245.

[4] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," in *Proc. of the International Symposium on Robotics Research (ISRR)*, 2017.

[5] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. of the International Conference on Machine Learning (ICML)*, 1999, pp. 278–287.

[6] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[7] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proc. of Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, June 2018.

[8] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. of the International Conference on Machine Learning (ICML)*. New York, NY, USA: ACM, 2009, pp. 41–48. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553380

[9] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2017, pp. 482–495.

[10] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning (CoRL)*, 2018, pp. 767–782.

[11] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 1515–1528. [Online]. Available: http://proceedings.mlr.press/v80/florensa18a.html

[12] 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*, vol. abs/1802.10567, 2018.

[13] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid, "Combining learned skills and reinforcement learning for robotic manipulations," *arXiv preprint arXiv:1908.00722*, 2019.

[14] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[15] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[16] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. de Freitas, "Playing hard exploration games by watching youtube," in *Advances in Neural Information Processing Systems*, 2018, pp. 2930–2941.

[17] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, May 2018, pp. 1111–1117.

[18] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard, "Adversarial skill networks: Unsupervised robot skill learning from video," 2019.

[19] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2018, pp. 6292–6299.

[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.

[21] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. KramÃ¡r, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," in *Proc. of Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, June 2018.

[22] 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," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[23] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Conference on Robot Learning (CoRL)*, vol. 78. PMLR, 13–15 Nov 2017, pp. 334–343. [Online]. Available: http://proceedings.mlr.press/v78/james17a.html

[24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[25] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2019, pp. 8973–8979.

[26] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv preprint*, 2019.

[27] A. A. Rusu, M. Večerík, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," in *Conference on Robot Learning (CoRL)*, vol. 78. PMLR, 13–15 Nov 2017, pp. 262–270. [Online]. Available: http://proceedings.mlr.press/v78/rusu17a.html

[28] 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 *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. IEEE, 2018, pp. 4243–4250.

[29] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine, "One-shot imitation from observing humans via domain-adaptive meta-learning," in *Proc. of Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, June 2018.

[30] M. Deisenroth, C. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Proc. of Robotics: Science and Systems (RSS)*, June 2011.

[31] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. IEEE, 2015, pp. 1503–1510.

[32] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016. [Online]. Available: http://jmlr.org/papers/v17/15-522.html

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[34] I. Popov, N. Heess, T. P. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. A. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *CoRR*, vol. abs/1704.03073, 2017.

[35] D. Kalashnikov, A. Irpan, P. P. Sampedro, 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/pdf/1806.10293

[36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.

[37] E. Coumans, “Bullet physics simulation,” in *ACM SIGGRAPH 2015 Courses*, ser. SIGGRAPH ’15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2776880.2792704

APPENDIX

A. Hyperparameters

Table III shows a list of hyperparameters that were used in the experiments.

Hyperparameter	Value
Adam learning rate	2.5×10^{-4}
Adam ε	1×10^{-5}
Discount γ	0.99
GAE τ	0.95
Entropy coefficient	0.01
Value loss coefficient	0.5
Max grad. norm	0.5
Number of actors	8
Minibatch size	8×512
Num. epochs	4
Clip param.	0.1
Training steps	10^7
Interval $[\alpha, \beta]$	$[0.4, 0.6]$
Increment ε	0.002
Number of demonstrations	10

TABLE III: Default hyperparameters that were used in the experiments unless stated otherwise.

B. Additional Experiments

Additional experiments were conducted for the block stacking task described in Section IV-B. We investigate the choice of input modalities (Fig. 8) and the impact of different values for the hyperparameters reward interval $[\alpha, \beta]$ (Fig. 9) and difficulty increment ε (Fig. 10).

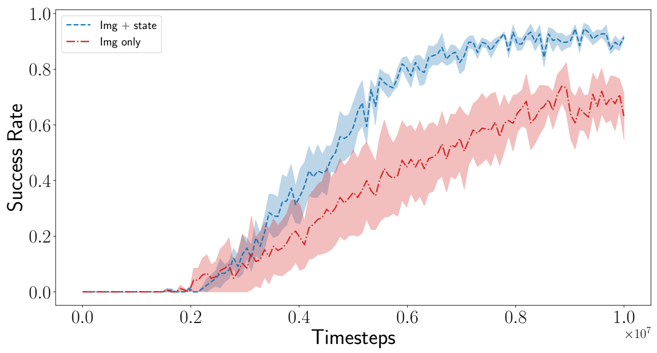

Fig. 8: Observation experiment. We evaluate how the performance depends on the input modality. *Img only* learns purely from the camera images. We reduced the network architecture to the CNN part of the default network. The full model clearly outperforms the *Img only* network. We assume, that this is because the state vector contains valuable information that is not or only insufficiently contained in the images. The plot shows averages and standard deviations of the evaluation success rate over five runs with different random seeds.

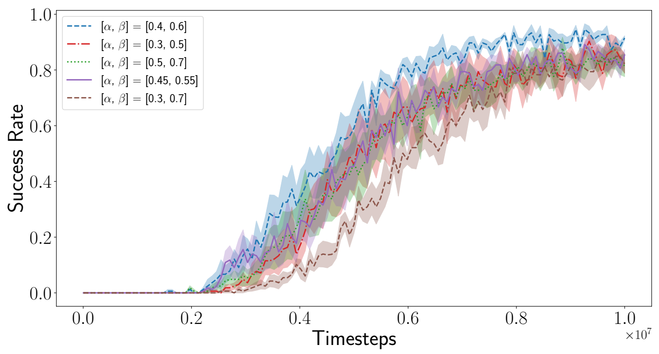

Fig. 9: Reward Interval Experiment. In this experiment, we evaluate the influence of difference values for $[\alpha, \beta]$. The interval of $[0.4, 0.6]$ shows the best performance and is used in the paper, but the other runs also achieve good scores. The plot shows averages and standard deviations of the evaluation success rate over five runs with different random seeds.

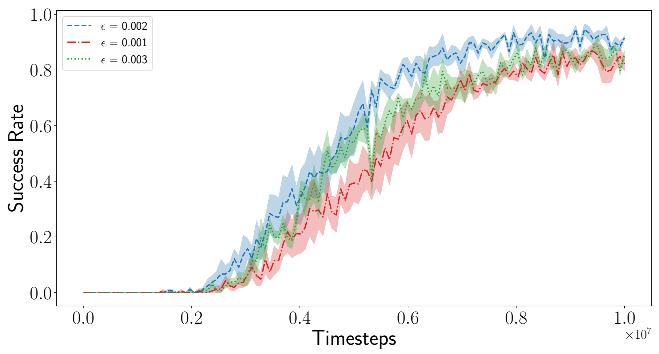

Fig. 10: Difficulty Increment Experiment. Here, we compare how different values for ε influence the training speed and final performance. All three values show similar training curves with our default value of $\varepsilon = 0.002$ being slightly better than the rest. The plot shows averages and standard deviations of the evaluation success rate over five runs with different random seeds.