arXiv:1803.04996v1 [cs.RO] 13 Mar 2018

Flexible Robotic Grasping with Sim-to-Real Transfer based Reinforcement Learning

Michel Breyer, Fadri Furrer, Tonci Novkovic, Roland Siegwart, and Juan Nieto

***Abstract*— Robotic manipulation requires a highly flexible and compliant system. Task-specific heuristics are usually not able to cope with the diversity of the world outside of specific assembly lines and cannot generalize well. Reinforcement learning methods provide a way to cope with uncertainty and allow robots to explore their action space to solve specific tasks. However, this comes at a cost of high training times, sparse and therefore hard to sample useful actions, strong local minima, etc. In this paper we show a real robotic system, trained in simulation on a pick and lift task, that is able to cope with different objects. We introduce an adaptive learning mechanism that allows the algorithm to find feasible solutions even for tasks that would otherwise be intractable. Furthermore, in order to improve the performance on difficult objects, we use a prioritized sampling scheme. We validate the efficacy of our approach with a real robot in a pick and lift task of different objects.**

I. INTRODUCTION

Robust manipulation of objects is a fundamental, yet challenging topic for current and future robotic applications as machines increasingly leave the well-structured environments of assembly lines and research labs. Grasping is probably one of the most active fields in robotic manipulation research since it is a key component in many tasks. Manually designing policies that can cope with a variety of different scenarios and high dimensional sensory input is extremely difficult. Thus, while early approaches focused on analytic methods, motivated by advances in machine learning, recent efforts have shifted towards more data driven approaches [1]. Most notable are end-to-end deep neural network models trained to map camera images to robust grasp configurations [2]. While first attempts relied on human labeled images [3], latest works have devised automated data collection schemes [4], [5], [2] and achieved high grasp success rates on both known and novel objects. A drawback of many of these approaches is their supervised nature, in particular the need for task specific labeled input/output pairs. In general, it is easier to describe the desired outcome of a task rather than providing the necessary actions to complete it. Reinforcement Learning (RL) is a powerful framework for enabling machines to learn a large set of behavioral skills from trial-and-error. Deep Reinforcement Learning (DRL) combines RL with the expressive power of deep neural networks and has led to promising results in various decision problems [6], [7]. However, there still remain many challenges to overcome. Especially in robotics, the inherent high sample inefficiency of these methods is expensive in time and require complex hardware setup. Nevertheless, some notable results have been achieved in continuous control and manipulation tasks, both on real robots [8], [9] and simulated environments [10], [11].

M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto are with the Autonomous Systems Lab, ETH, 8092 Zurich, Switzerland e-mail: {michel.breyer, fadri.furrer, tonci.novkovic}@mavt.ethz.ch, {rsiegwart, nietoj}@ethz.ch.

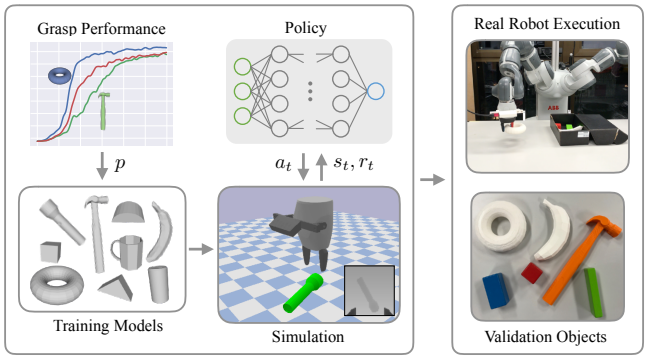

Fig. 1. Overview of our approach. We use a set of 3D models to iteratively train a policy in simulation on a pick and lift task where the probability of sampling a given object is inversely proportional to its recent performance. Afterwards, we transfer this policy to the real robot to achieve the same task with similar objects as in the training set.

The goal of this work is to apply DRL to the task of grasping and lifting isolated objects of different shapes. We aim to explore the feasibility of model-free learning for manipulation skills based on Sim-to-Real transfer learning. We propose a problem formulation suitable for learning a map between depth images captured from a wrist-mounted camera to gripper displacements and finger actions. In order to reduce model sizes and training time, we separate perception and control and learn a compressed image representation using the latent space of an autoencoder. To further improve training efficiency and final model performance, we introduce techniques that dynamically adapt task parameters to the robot's performance during training. These include improved selection of objects based on past grasp attempts and progressively increasing the robot's workspace. We overcome the high data complexity by performing training in simulation and report the necessary adaptations to transfer learned policies to a physical system. We train and evaluate our proposed method in simulation, and report results of our final model in real world grasp experiments. To summarize, the main contributions of this work are:

- A working RL setup for the combined task of locating, reaching, grasping and lifting different objects.
- Strategy for improved object selection during training.
- Adaptive task parameters that allow scaling to large workspaces while maintaining feasible exploration.

II. RELATED WORK

Early works of RL in robotics relied on hand-engineered, task-specific policy representations [12], [13] to enable training on physical systems with high-dimensional continuous action spaces. However, recent algorithmic developments have allowed to scale RL to more expressive function approximators, such as deep neural networks. Most notable algorithms are Trust Region Policy Optimization (TRPO) [14] and Deep Deterministic Policy Gradients (DDPG) [15] which have been applied to a variety of different continuous control tasks [10], [11]. Due to the large amount of system interaction required for training these models, many applications have been restricted to simulated environments, with some exceptions such as [9].

Some previous works have explored the application of RL to the task of grasping. Lampe and Riedmiller [16] applied a combination of value-based RL and supervised learning to learn a controller with visual feedback for a combined reaching and grasping task. However, their perception pipeline was restricted to a single object and they relied on separate short- and long-range controllers to scale to larger working areas. James and Johns [17] used Deep Q-Network (DQN) [6] to learn to grasp a cube from full image observations, using distance information to guide the robots towards the object. Compared to the previous two works, our approach allows to train a single controller that can cope with different objects and scale to large workspaces. To achieve this, we employ autoencoders to learn a compressed state representation, similarly to [18] and [19].

There exist several techniques for dealing with exploration, including providing expert demonstration [12], [20] and reward shaping [17], [9]. We use a combination of reward shaping and a technique similar to curriculum learning [21] to enable efficient learning by progressively adapting workspace parameters. Our prioritized sampling scheme for selecting objects is most similar to Prioritized Experience Replay [22], a technique used for improved experience selection in DQN.

III. BACKGROUND

Reinforcement learning considers the problem of an agent interacting with its environment to maximize some long-term measure of reward. At each time step t, the agent observes the current state s_t of its environment and decides to take an action a_t according to a parametrized policy $\pi_\theta(a_t|o_t)$. The execution of this action causes the system to transfer to a new state s_{t+1} and the agent receives a reinforcement signal $r(s_t, a_t)$, often called reward. We follow [23] and model our RL problem as a Markov Decision Process (MDP) defined by the tuple $\langle S, A, r, p, \rho_0, T\rangle$, where S denotes the set of admissible states, A the set of valid actions, $r : S \times A \to \mathbb{R}$ a real-valued reward function and $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ the (unknown) transition probability distribution. We consider the episodic RL setup, in which episodes are terminated after a finite time horizon T and a new initial state is sampled from the initial state distribution ρ_0. The goal of RL is to find the optimal policy parameters θ^* that maximize the return,

$$J(\theta) = \mathrm{E}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t)\right], \tag{1}$$

where the expectation is computed over the distribution of all possible trajectories $\tau = \{s_0, a_0, \ldots, s_T\}$ with probabilities $p_\theta(\tau)=p(s_0)\prod_t p(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$ and $\gamma \in [0, 1]$ denotes a discount factor. Policy gradient methods are a class of RL algorithms that directly maximize the return J with respect to the policy parameters θ by iteratively following the gradient of the expected return, $\theta_{i+1}=\theta_i+\alpha\nabla_\theta J|_{\theta=\theta_i}$. Various empirical gradient estimators have been proposed [12], but scaling to high-dimensional, nonlinear policy representations has remained challenging due to high variance and sensitivity to hyperparameters, such as the step size α [10]. TRPO approaches these issues by constraining the maximum change in the policy distribution, resulting in more stable updates.

IV. APPROACH

We consider the combined task of locating, reaching, grasping and lifting objects using a parallel-jaw gripper and a depth camera. Our goal is to train a policy through model-free RL that takes observations from the camera as input and outputs controls for the end effector position and the gripper. Due to the inherent high data complexity of DRL approaches, we perform training in simulation and evaluate the models in both virtual and real-world experiments. Fast dynamic simulation not only allows quicker iterations over different design ideas and parameters, but also avoids the need for supervision during training, e.g. monitoring the hardware and/or manually resetting experiments.

A. Problem Formulation

At the beginning of each episode, one object is randomly selected from a set of training objects and placed at a random pose within a predefined area on a flat surface. The gripper is positioned close to the object in order to keep exploration within the area of interest.

State observations: At each time step, the agent receives information about the current finger joint positions and observes the scene through a wrist-mounted depth camera. Depth images were chosen in favor of simple RGB inputs since they provide more geometric information and models trained on synthetic depth information have been shown to transfer to the real-world [2], [5]. A general challenge with full image observations is their high dimensionality. In order to keep model sizes and training times within reasonable bounds, we separate perception and control by first learning a compressed image representation using an autoencoder, which is then used as efficient state representation for training agents to manipulate objects.

Autoencoders are a technique from unsupervised learning used for dimensionality reduction. They are a special case of deep neural networks with matching input and output dimensions and the goal of reconstructing their own input. Adding a low-dimensional latent space and training

the model to minimize the mean squared error MSE $= \frac{1}{n}\sum(\hat{y}_i - y_i)^2$ between original and reconstructed images y_i and $\hat{y}_i$, respectively, forces the network to learn a compressed representation of the input. In this work, we used a convolutional neural network with a latent space of 60 units, as shown in Figure 2.

In order to gather representative images for training the autoencoder, we need a policy to explore relevant parts of the state space. However, to train such a policy, a trained autoencoder is needed. This problem was addressed in [19] by optimizing a simple controller without visual input for collecting data before training the full model. In this work, we simply rendered synthetic images of objects placed randomly on the flat surface or between the gripper's fingers. Varying the poses of the gripper and objects ensured a rich training set. Since mostly the location of the object, its shape and distance to the camera are of interest, the fingers of the gripper and table surface were filtered from the image with simple masks and plane detection using a Random Sample Consensus (RANSAC) [24] based approach.

***Actions**:* The agent's action space includes continuous changes in position and yaw of the gripper in the end effector's coordinate system as well as opening or closing the fingers. This approach has two benefits. First, combined with the positioning of the camera, learning policies for configurations with similar relative poses between object and gripper is easier compared to learning joint actions. Second, position control should transfer easier to the real-world compared to velocity or torque control which would require highly accurate physical models. This formulation also makes enforcing safety constraints on the robot's workspace straightforward. Controlling the gripper was modeled with an additional discrete variable $p \in [-1, 1]$, where positive and negative values are mapped to opening and closing hand commands, respectively. To summarize, the action vector chosen by the agent at every step is given by

$$a_t = [\Delta x, \Delta y, \Delta z, \Delta\psi, p] \in [-\delta_{\text{pos}}, \delta_{\text{pos}}]^3 \times [-\delta_{\text{rot}}, \delta_{\text{rot}}] \times [-1, 1], \quad (2)$$

where δ_{pos} and δ_{rot} were set to $0.01\,\text{m}$ and $0.15\,\text{rad}$.

***Reward function**:* We consider an episode as successfully completed if the object was lifted above a given target height z_{target}. Episodes are reset after achieving this goal or after reaching a maximum number of $T = 150$ time steps. A natural choice for the reward function of this task would thus be $r = I(z_{\text{object}} > z_{\text{target}})$ where $I(\texttt{cond})$ evaluates to 1 if `cond` is true and 0 otherwise. However, such sparse formulations are difficult to learn from scratch, requiring significant exploration, and often fail altogether [20]. For this reason, we construct our reward function from a set of sub-goals. Similarly to [17], the agent receives a reward signal upon grasping the object. Also, lifting the object gets rewarded with a monotonic increase in the episode return,

$$\begin{aligned} r = {} & I(z_{\text{object}} > z_{\text{target}}) \cdot c_1 \\ & + I(\text{object } \textbf{is} \text{ grasped}) \cdot (c_2 + c_3 \cdot \Delta z) \\ & - (c_2 + c_3 \cdot \delta_{\text{pos}}), \end{aligned} \quad (3)$$

with c_1=100, c_2=1 and c_3=1000. Grasps are naively detected by checking if the fingers stalled before fully closing. Note that compared to similar approaches [17], [11], we do not rely on privileged information such as distance measurements. These are not only hard to obtain outside of simulated environments but also bias the gripper to grasp at the point on the object from which this distance is computed and may introduce undesired local minima if not carefully crafted. The last term in equation (3) is a large time penalty shifting rewards to negative values encouraging the agent to complete the task as quickly as possible. Without this term, the agent might drop the object before reaching the target height allowing it to accumulate additional reward signals by picking up the object a second time.

B. Policy Representation and Algorithm

As suggested by Schulman et al. [14], we model our policies as a multivariate Gaussian distribution. A feed-forward neural network maps observations to the means of the distribution and the log-standard deviations are parameterized by a global, trainable vector. Figure 2 shows the architecture of our policy network, consisting of three hidden layers with Rectified Linear Unit (ReLU) activations. A hyperbolic tangent (tanh) nonlinearity is applied to the output layer to scale actions to their respective ranges. Policy parameters are randomly initialized and iteratively updated using TRPO.

C. Prioritized Object Sampling

At the beginning of each episode, we select an object for training the agent. A simple strategy is to uniformly sample objects from the training set. However, we argue that efficiency can be improved by sampling objects according to the agent's recent performance. Instead of spending episodes on objects that are easy to grasp, we should prioritize objects that the agent struggled with in the past. To achieve this goal, we introduce an improved sampling strategy, similar to [22], that selects objects inversely proportional to their number of past grasp successes in order to balance performance between individual objects. Formally, the probability of choosing object i is given by

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}, \quad (4)$$

where $p_i = 1/(1 + \text{successes}(i))$. The parameter α allows to tune the amount of prioritization, with $\alpha = 0$ corresponding to the uniform case. We set α to 0 at the beginning of training when only limited statistics are available and linearly increase its value over time.

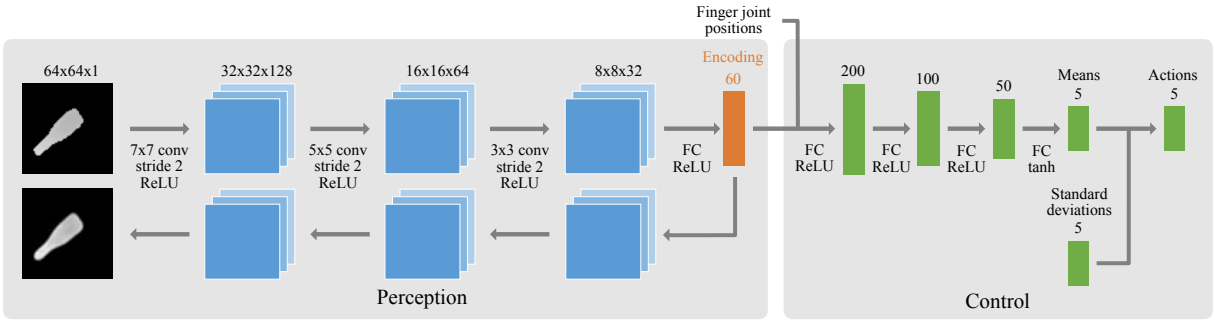

Fig. 2. We separate perception and control by first learning a compressed image representation using an autoencoder. The autoencoder consists of three convolutional layers connected to a low-dimensional bottleneck and is trained to reconstruct its own input in an unsupervised manner. In a second stage, we train a separate policy network to map the generated encoding concatenated with finger joint positions to the mean actions of a Gaussian policy.

D. Adaptive Task Parameters

Limited prior knowledge makes model-free RL approaches very flexible, but also renders exploration of task-relevant parts of the state space challenging. For this reason, we propose to adapt task parameters to the current performance of the agent. In particular, we dynamically increase the area within which objects can be placed as well as the initial height of the gripper. We implemented a heuristic that performs updates of parameters in discrete steps after a certain threshold of success rate of recent grasp attempts has been reached. Gradually increasing the difficulty of the task enables the agent to transfer experience from past trials to the new setup allowing to efficiently scale to larger workspaces.

E. Sim-to-Real Transfer

Performing training in simulation is attractive due to the high costs of collecting data on real robots. In this work, we explore directly transferring policies trained in simulation to the physical world without any additional fine-tuning. However, real world depth images are noisy and have missing information compared to the rendered images. We apply an elliptic mask to filter out missing points and curvatures on the image's edges and dilate the finger masks to remove noise and shadows cast around the robot's hand.

We also noticed that policies might display undesirable behavior for real robot execution, including excessive yaw rotation and pushing down on objects leading the system to halt. To discourage these actions, during experiments targeting execution on the real robot, we included the following penalties to the reward function (3):

$$r \mathrel{-}= c_4 \cdot \Delta\psi + c_5 \cdot I(\mathbf{n} \cdot \mathbf{z} > \cos\tfrac{\pi}{6}), \tag{5}$$

where we detect the gripper pushing down on objects by projecting the collision normal $\mathbf{n}$ onto the unit Z vector. However, adding these penalties leads to strong local minima hurting exploration of grasps. For this reason, we set the constants c_4 and c_5 to 0 at the beginning of the experiment and increase them using the adaptive framework introduced in the previous section.

V. EVALUATION

In order to evaluate the presented methods, we designed a series of simulated and real-world experiments. First, the effect of prioritized object sampling and adaptive task parameters are analyzed in a well controlled virtual environment. Second, we train agents on a larger set of objects and evaluate their performance when trained in simulation and transferred to the real-world. To compare performances of different agents, we report results in the form of the following metrics: episode return, success rate and average number of steps until completion of the task.

A. Experimental Setup

The platform used for evaluation consists of an ABB YuMi with parallel-jaw gripper. Depth images are captured using a time-of-flight camera of type Camboard pico flexx. The camera is attached to the robot's wrist at a tilt angle of $15°$ as depicted in Figure 1 and images are cropped and scaled down to 64×64 pixels.

We use PyBullet [25] to construct a simulated world of the system. Within the virtual world, the gripper is controlled by directly enforcing constraints on its pose, avoiding the computation of inverse kinematics. A step size of $5\,\mathrm{ms}$ provided plausible physical behavior. Synthetic depth images were generated using a renderer which is a part of the physics engine and camera parameters were modeled to match the real setup, though no precise hand-eye calibration was performed.

Training objects include a set of blocks with different geometric shapes, such as cuboids and cylinders, as well as a subset of models from 3DNet [26], including objects from 10 categories such as cup, donut and hammer. For each category, we selected one model and generated several variations with different friction parameters and scales, ensuring they fit into the robot's hand with an opening width of $5\,\mathrm{cm}$, resulting in a total of 69 models.

Training was conducted on a machine running Ubuntu 16.04 with an Intel Core i7-6700K and NVIDIA GeForce GTX 980 Ti. The autoencoder was trained using the Adam optimizer [27] with a learning rate of 2×10^{-4} and batch sizes of 128 while policies were trained with an implementation of TRPO based on Rllab [10] with batch sizes of 20000 samples and a step size of 0.01.

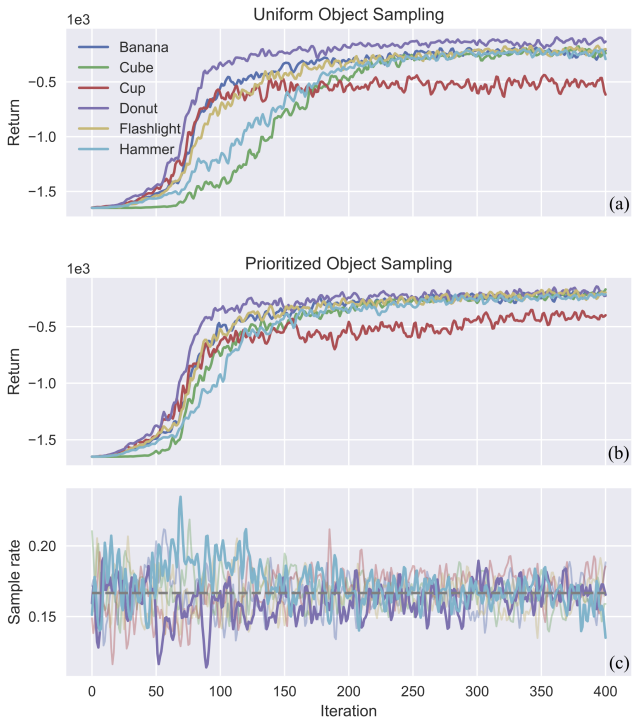

Fig. 3. Learning curves for individual objects with (a) uniform object sampling and (b) prioritized object sampling, including history of object sample rates (c) chosen by the adaptive strategy. The dotted line represents the constant uniform sampling rate of $1/6$.

B. Prioritized Object Sampling

In order to analyze the effects of prioritized object sampling on training and final model performance, we compare results using uniform object sampling against the strategy discussed in Section IV-C. To facilitate comparisons, we restricted ourselves to a set of 6 objects. Both experiments were run with the same sequence of random numbers and we linearly increased the prioritization factor α from 0 to a maximum value of 0.5 over 30000 episodes, corresponding to approximately 200 iterations of TRPO depending on the exact performance of the agent. Figure 3 shows learning curves and sampling rates for individual objects against the number of policy iterations. We can see that the prioritized sampling scheme successfully balances experience over objects, leading to quicker convergence of individual curves. In particular, it appears that the agent had initial difficulties lifting the hammer, leading to higher sampling rates and quicker learning compared to the uniform case. Similarly, the agent learned to grasp the donut quicker compared to the other objects, leading to smaller than average sampling rates. Over the course of training, performance on other objects increases and sampling rates settle around the uniform value of $1/6$. Table I shows results of the final models averaged over 100 episodes. We can see that both policies achieved high success rates on the trained objects, with the final performance of the agent trained with prioritized sampling being slightly higher in most cases, in particular grasping the cup.

Object	**Uniform Sampling**		**Prioritized Sampling**	
	Success rate	Steps	Success rate	Steps
Banana	0.99	48.1	0.97	43.8
Cube	0.98	44.6	0.98	44.8
Cup	0.77	61.2	0.85	54.7
Donut	0.99	34.6	0.99	37.9
Flashlight	0.94	43.5	0.98	38.7
Hammer	0.97	47.4	0.98	48.6

TABLE I

RESULTS FOR PRIORITIZED OBJECT SAMPLING

Training Setup	**Small Workspace**		**Large Workspace**	
	Success rate	Steps	Success rate	Steps
Small workspace π	1.00	22.1	0.81	65.7
Large workspace π	0.00	150.0	0.00	150.0
Adaptive workspace π	0.98	29.6	1.00	40.3

TABLE II

RESULTS FOR ADAPTIVE TASK PARAMETERS

C. Adaptive Task Parameters

We train policies for grasping a single object, namely a cup, with three different setups. First, training is performed within a $10\,\mathrm{cm}\times10\,\mathrm{cm}$ workspace and the gripper is positioned close to the surface the object rests on. In a second experiment, the object is randomly placed within a larger $30\,\mathrm{cm}\times30\,\mathrm{cm}$ area and the gripper is placed $25\,\mathrm{cm}$ above the surface, corresponding to a more realistic grasping scenario. In the third experiment, we use adaptive task parameters to start training with the small working area and linearly increase its size in 10 discrete steps every time the agent achieves a success rate of at least 50% in the last 1000 episodes. Figure 4 shows learning curves for the three setups. We observe that the initial random policy fails to explore grasping objects when the working area is too large. The other two agents rapidly learn to grasp the cup. Note that the performance of the agent trained with the adaptive environment converges to a lower return value, which is to be expected since the workspace is larger and therefore more steps are required to complete the task. Table II reports results averaged over 100 trials for the converged policies evaluated on both small and large workspaces. Not surprisingly, the success rate of the policy trained on the large workspace is 0 even after 200 policy iterations due to failure of exploration. The policy trained on the small workspace achieved a perfect score on the same task it was trained on, but also manages to generalize to the larger workspace. An explanation for this result could be the fact that the target height z_{target} used in the experiment was actually higher than the initial gripper position leading the agent to gather additional experience in these cases. However, it was not able to reach the performance of the agent that was gradually guided to learn coping with larger workspaces.

D. Real Robot Grasping of Isolated Objects

In our final experiment, we train a policy using both prioritized object sampling and adaptive task parameters

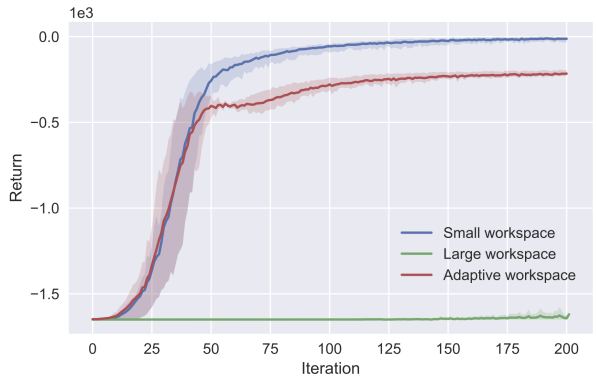

Fig. 4. Learning curves for agents trained on a small, large and adaptive workspace. The lines represent averages computed over 4 runs with different seeds while the shaded area depicts the best and worst performer.

on the full set of training objects. In simulation, the converged policy managed to grasp 185 out of 200 randomly selected objects, corresponding to a success rate of 0.925, and presented balanced performance over individual object categories. To evaluate policy transfer to the physical system, we perform real robot experiments on a set of 6 objects, including three wooden blocks and three printed models from 3DNet, as shown in Figure 1. For each object, we run 10 trials and use the same success metric as defined in Section IV-A, with the addition of considering actions that lead the robot to halt, for example due to exerting large forces on the objects, as failures. Table III summarizes the results from these experiments and compares success rates to values measured with similar setups in simulation. We can see that the policy trained only in simulation was able to complete between 6 and 9 of the 10 attempts for each of the tested objects leading to a total average success rate of 0.78. An example execution of our policy on the robot is shown in Figure 5. It may be noted that both in simulation and real world experiments, the reactive nature of the learned policy allowed to recover from failed grasps and still be able to lift the object within the allowed time. Unfortunately, even with the additional penalties for pressing down on objects, this was still the major cause for failure on the real robot. In most cases, this behavior is due to our reward formulation. In order to minimize time penalties, the agent learned to grasp objects as high as possible to shorten the number of steps required to complete the task. In simulation, accurate vision and approximate contact models encourage this behavior, especially for the cuboids due to their simple geometry. However, these grasps are not particularly robust and lead to safety issues on the real system.

E. Discussion and Limitations

Currently, our method was only tested on known, isolated objects using a simple gripper model. It remains to be investigated how well the trained policies generalize to unseen objects with significantly different geometry. Indeed, in our experiments we observed that the robot favors grasps with similar observations. As an example, the hammer was usually grasped at the end of its shaft, which looks comparable to a cube from the wrist-mounted camera's perspective. A general challenge in DRL approaches is that the end-to-end nature and underlying random processes make it hard to reason about some behaviors of the agent, rendering designing and testing different ideas difficult and time-consuming. Especially when transferring policies to the real world, tracing failures to different components, e.g. perception in presence of noisy, real world sensors or model limitations is hard.

Object	Simulation	Real Robot
Cube	0.97	0.80
Cuboid	0.89	0.60
Flat cuboid	0.93	0.80
Banana	0.98	0.80
Donut	0.96	0.90
Hammer	0.87	0.80

TABLE III
SUCCESS RATES OF SIMULATED AND REAL WORLD EXPERIMENTS

VI. CONCLUSION

In this work, we presented a method for training a robotic system to grasp and lift a set of simple objects. The policy trained in simulation was successfully transferred to the real robot with only small modifications to filtering. We have shown that gradually adapting the task difficulty can significantly improve the learning rate of more complex tasks that might otherwise be intractable. Furthermore, by prioritizing more difficult objects during sampling in training, we show that faster convergence can be achieved. We compared the performance in simulation and in real-world experiments and discussed about the limitations of our system.

In future work, we plan to investigate how such a system can generalize to unseen objects, deal with multiple objects and even cluttered scenes. Also, including additional sensing modalities is another important future research avenue.

ACKNOWLEDGMENTS

This work was supported in part by the Swiss National Science Foundation (SNF) through the National Centre of Competence in Research (NCCR) Digital Fabrication.

REFERENCES

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[2] 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," *arXiv preprint arXiv:1703.09312*, 2017.

[3] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[4] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3406–3413.

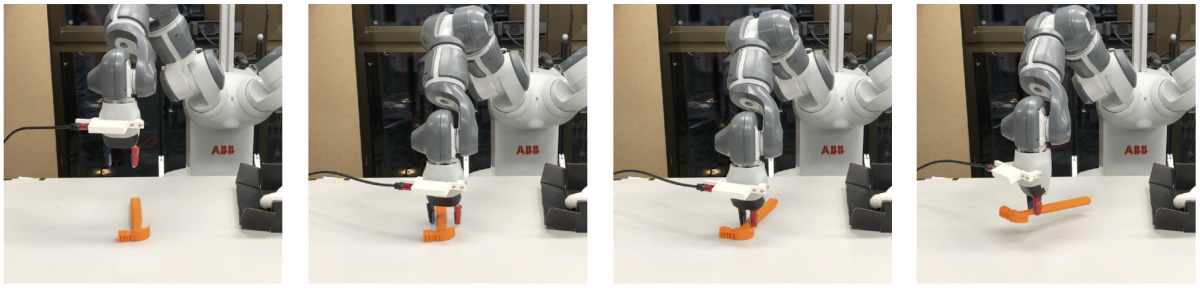

Fig. 5. Real robot execution of a policy trained in simulation that successfully approaches, grasps and lifts a hammer.

[5] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4461–4468.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[9] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 3389–3396.

[10] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[11] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[12] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[13] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[15] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[16] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–8.

[17] S. James and E. Johns, "3d simulation for robot arm control with deep q-learning," *arXiv preprint arXiv:1609.03759*, 2016.

[18] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012, pp. 1–8.

[19] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 512–519.

[20] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[21] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[22] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[23] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[24] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[25] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2018.

[26] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, "3dnet: Large-scale object class recognition from cad models," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5384–5391.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.