Sim2Real for Peg-Hole Insertion with Eye-in-Hand Camera

Fedor Chervinskii[§], Alexander Rybnikov[§], Damian Bogunowicz[§] and Komal Vendidandi[§]
Λ Γ Γ I V Λ L

***Abstract*—Even though the peg-hole insertion is one of the well-studied problems in robotics, it still remains a challenge for robots, especially when it comes to flexibility and the ability to generalize. Successful completion of the task requires combining several modalities to cope with the complexity of the real world. In our work, we focus on the visual aspect of the problem and employ the strategy of learning an insertion task in a simulator. We use Deep Reinforcement Learning to learn the policy end-to-end and then transfer the learned model to the real robot, without any additional fine-tuning. We show that the transferred policy, which only takes RGB-D and joint information (proprioception) can perform well on the real robot.**

***Index Terms*—Deep Reinforcement Learning, Robotic Control, Visual Servoing, Peg-Hole Insertion, Sim2Real Transfer**

I. Introduction

A successful peg-hole insertion includes a series of sub-tasks like aligning the peg w.r.t hole and inserting in the hole. The task involves concurrent understanding of vision, depth and haptic modalities. This has been shown to pose significant difficulties for the robots. [1] claims that having a single modality, like vision, does not solve the peg-hole insertion problem accurately and precisely. This is the motivation behind our multi-modal perception system. In this work focus on using vision and optionally proprioception for robotic control, as well as trying to understand precision limits of a trainable vision-based controller.

Reinforcement Learning (RL) allows learning control policies that are difficult to model explicitly. These policies can generalise with respect to the geometry of the peg and the hole, as stated in [1].

A UR5e robot used in our work is trained in the CoppeliaSim [2] to learn a peg-hole insertion policy. We are using different model-free RL algorithms embedded in the Catalyst [3] framework. In our experiments TD3 [4] converged to higher returns than DDPG [5] or SAC [6] as shown in Fig. 2 in section V. Our deterministic control policy is learned from a multi-modal representation consisting of RGB-D images and proprioception. The policy generalises to different colours, as well as the 3D position of the peg and the block.

II. Related Work and Background

[7] uses a hybrid approach of model-free deep RL and imitation learning to achieve zero-shot Sim2Real transfer on six different tasks with a single agent. The method shows that pre-training on expert demonstrations leads to faster training than learning the policy from scratch. [8] learns to refine synthetic images (domain adaptation) to achieve good grasping success rate using less real-world data. [9] trains a network that maps input image to the robot's motor torque. [1] uses self-supervision to learn a compact and multimodal representation of the sensory inputs for the peg-hole insertion task. While all aforementioned works incorporate vision, none of them is using eye-in-hand approach which we consider much more suitable specifically for RL, where a policy can learn to optimize observations along the way, making the vision ”active”.

III. Problem Statement and Method Overview

The main objective of our work is to make the robot learn a peg-hole insertion policy, which is invariant to the pose and color of the mating part or peg. This is done by training a robot in the simulation and then directly transferring the model to the real robot. For a better Sim2Real transferability, we experiment with image augmentation and include different modalities. We benchmark our approach against a proposed baseline model, introduced in section IV.

IV. Baseline and RL Models

A. Baseline Model

Our baseline for the peg-in-hole insertion uses visual feedback together with classical computer vision tools to control the insertion of the peg. Firstly, the robot uses color information to segment out the mating part from the image. This allows to filter all the irrelevant visual noise. Secondly, the centre of the hole is being estimated. Finally, the end effector is guided to minimize the distance between the pose of the peg and the estimated pose of the hole.

B. Reinforcement Learning

We employ the actor-critic method, where both the actor and the critic are parameterized using neural networks.

$$R_{total} = R_{distance} + R_{success} + R_{collision} + R_{time} \quad (1)$$

The total reward stated in Eq. 1 for each step consists of following components.

- **Distance Reward:** Based on the distance between the the peg and the target (the hole of the mating part). The smaller the distance, the higher the reward.
- **Success Reward:** A sparse reward based on the distance between the peg and the desired position in the hole. The

[§]Equal contribution.
[chervinskii, rybnikov, bogunowicz, vendidandi]@arrival.com

agent is being rewarded once the peg gets very close to the target (specified by a threshold).

- **Collision Reward:** To discourage the robot from colliding with the block, a negative reward is obtained every time the agent collides with the mating part.
- **Time Reward:** To coerce the agent to complete the task as fast as possible, it obtains a small penalty per every step.

The scene in the CoppeliaSim and the real robot in starting position are shown in Fig. 1. On the real robot we use RealSense D435 RGB-D active stereo sensor. The mating part is being randomly placed on the table (always fully or partially visible for the robot in the starting position), imitating a flexible assembly process.

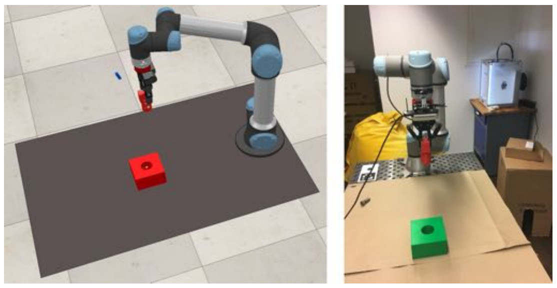
Fig. 1: *Left:* Rendered simulator scene from CoppeliaSim showing the UR5e robot with Robotiq gripper, a peg and a block. *Right:* Real UR5e robot in our lab.

V. Experiments: Results

First, we train the agent in the simulation. The learned policy is directly transferred to our real UR5e robot. Evaluation of the models is shown in Table I. We did a total of 30 rollouts for each model, in which the mating part pose is changed for every 5 rollouts. These 6 mating poses are predefined, with a maximum distance of 39 cm and 21 cm between the poses in x- and y-axis respectively (in the plane of the table). The robot starts at a predefined home position for every rollout in both simulation and real world. The comparison of different model-free RL algorithms can be found in Fig. 2. Fig. 3 shows which modalities are most important for the performance in the simulation.

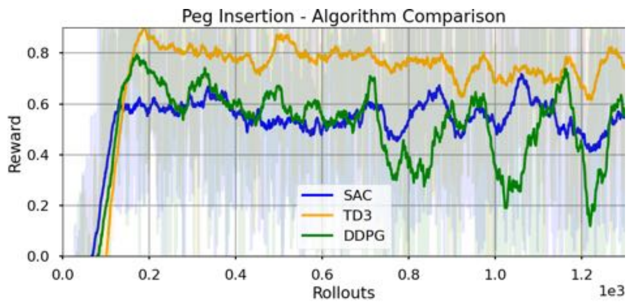

Fig. 2: TD3 yielding higher return than SAC and DDPG.

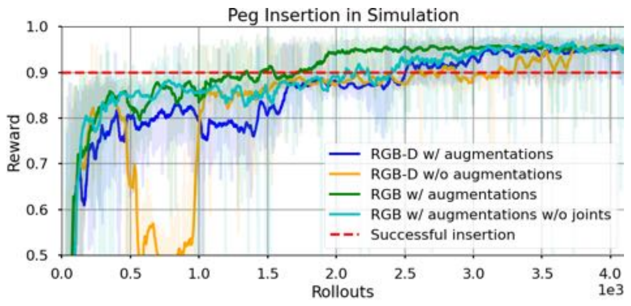

Fig. 3: Ablation study showing different configurations of training in learning a peg-hole insertion policy in simulation. It can be seen that all the configurations converged to successful insertion in simulation. The performance of these configurations on real robot can be seen in Table I.

Model Type	Mean Error [mm]		Insertion [%]
	x	y	
Baseline	2.4 ± 1.7	0.3 ± 3.0	100
RGB w/ augs	-2.0 ± 3.8	-9.8 ± 11.2	55
RGB-D w/ augs	17.7 ± 12.3	2.7 ± 1.8	0
RGB-D w/o augs	-124.1 ± 166.9	65.6 ± 94.4	0

TABLE I: **Mean insertion error and insertion rate on the real robot.** The mean error between the ground truth position of the inserted peg versus the actual position at the end of an episode. Insertion rate is the percentage of the rollouts which end with successful insertion. The mean is taken over 30 runs for different positions of the mating part on the table.

VI. Discussion and Conclusion

Our initial hypothesis was that image augmentation is crucial for successful transfer from simulation to the real-world. Table I shows that augmentations improve the precision significantly. However, the RGB-D model still does not achieve steady, successful insertions. That could be because the depth image has a very specific structural noise (depth values are computed from stereo cameras) that can hardly be modelled with standard augmentation algorithms. Thus, RGB-only model shows much better performance. Additionally, the ablation study in Fig. 3 shows that proprioceptive features are not critical for model performance for this task, neither in simulation, nor in real-life. Finally, we show that a purely eye-in-hand, image-based controller can be trained in simulation to perform peg-hole insertion with sub-centimetre accuracy. However, we still experience instabilities caused by changing light conditions and other environmental factors, that can be possibly improved by enhancing rendering in simulation and richer augmentation. For the future work we consider enhancing the vision-based controller with force-based one (trained end-to-end, as a single model), that can perform precise insertions with large initial uncertainty.

References

[1] Lee, Michelle A., et al. "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks." 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019.

[2] E. Rohmer, S. P. N. Singh, M. Freese, ”CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework”, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013. www.coppeliarobotics.com

[3] Kolesnikov, Sergey, and Oleksii Hrinchuk. ”Catalyst. RL: A Distributed Framework for Reproducible RL Research.” arXiv preprint arXiv:1903.00027 (2019).

[4] Fujimoto, Scott, Herke Van Hoof, and David Meger. ”Addressing function approximation error in actor-critic methods.” arXiv preprint arXiv:1802.09477 (2018).

[5] Silver, David, et al. ”Deterministic policy gradient algorithms.” 2014.

[6] Haarnoja, Tuomas, et al. ”Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.” arXiv preprint arXiv:1801.01290 (2018).

[7] Zhu, Yuke, et al. ”Reinforcement and imitation learning for diverse visuomotor skills.” arXiv preprint arXiv:1802.09564 (2018).

[8] Bousmalis, Konstantinos, et al. ”Using simulation and domain adaptation to improve efficiency of deep robotic grasping.” 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018.

[9] Levine, Sergey, et al. ”End-to-end training of deep visuomotor policies.” The Journal of Machine Learning Research 17.1 (2016): 1334-1373.

[10] *albumentations* python module https://albumentations.readthedocs.io/en/latest/api/augmentations.html

Appendix

Simulation sensor data augmentations

Different sets of augmentations are used for the RGB image and the depth image. Augmentations for the latter are stronger due to the significant noise in depth values (computed from RealSense camera stereo vision). *albumentations* [10] library is used to implement of all augmentations used in this work. RGB image is initially converted to grayscale and then the following augmentations are used: random brightness-contrast, motion blur, Gaussian blur, median blur, solarize filter, embossing, histogram equalization, sharpening, CLAHE, image compression, multiplicative noise, Gaussian noise. For depth images the following augmentations are used: thresholding and dilation, sharpening, embossing, motion blur, Gaussian blur, median blur, multiplicative noise, Gaussian noise, coarse ”pepper” dropout, coarse ”salt” dropout.

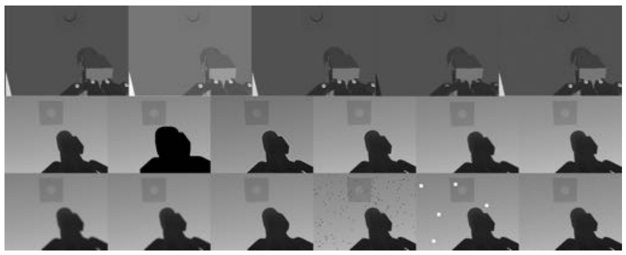

Fig. 4: Examples of used augmentations. *Top row, left to right:* original grayscale image, random brightness-contrast, motion blur, solarize filter, Gaussian noise. *Middle row, left to right:* original depth image, threshold and dilation, sharpening, embossing, motion blur, multiplicative noise. *Bottom row, left to right:* Gaussian blur, blur, median blur, coarse ”pepper” dropout, coarse ”salt” dropout, Gaussian noise.