Sim-to-Real Transfer of Robotic Assembly with Visual Inputs Using CycleGAN and Force Control

Chengjie Yuan[1,2†], Yunlei Shi[3,2†], Qian Feng[1,2], Chunyang Chang[2],
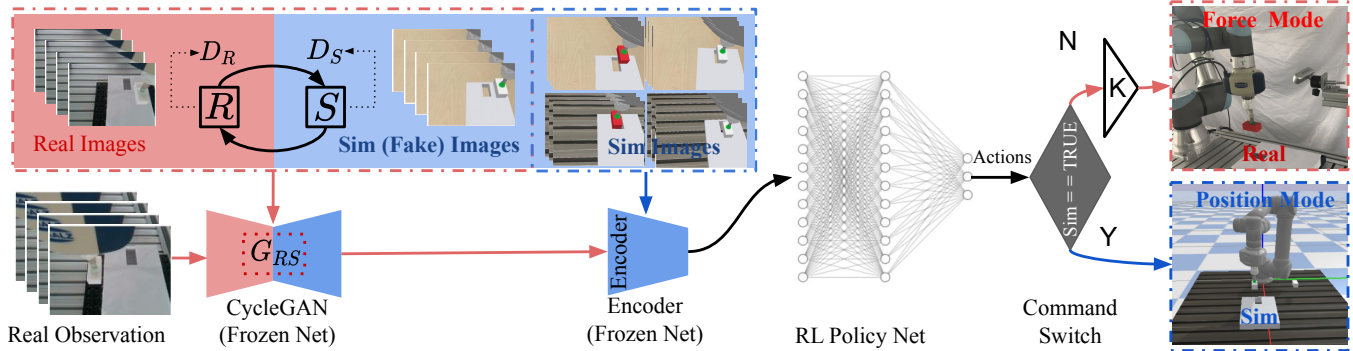Zhaopeng Chen[2], Alois Christian Knoll[1], Jianwei Zhang[3]

Fig. 1: Sim-to-real learning-based framework for a rectangular peg-in-hole insertion task. Sim part (in blue) is used to train the encoder (Frozen Net) and reinforcement learning (RL) policy net in a simulator. $G_{RS} : R \rightarrow S$ is a mapping function generated using a cycle-consistent generative adversarial networks (CycleGAN) to transfer an image from a real world style to a simulator style. A command switch is used to ensure safety in the contact-rich operation by changing the position control mode in the simulator to a force control mode in the real world.

Abstract— **Recently, deep reinforcement learning (RL) has shown some impressive successes in robotic manipulation applications. However, training robots in the real world is nontrivial owing to sample efficiency and safety concerns. Sim-to-real transfer is proposed to address the aforementioned concerns but introduces a new issue called the reality gap. In this work, we introduce a sim-to-real learning framework for vision-based assembly tasks and perform training in a simulated environment by employing inputs from a single camera to address the aforementioned issues. We present a domain adaptation method based on cycle-consistent generative adversarial networks (CycleGAN) and a force control transfer approach to bridge the reality gap. We demonstrate that the proposed framework trained in a simulated environment can be successfully transferred to a real peg-in-hole setup.**

I. INTRODUCTION

Industrial robots are commonly used in structured environments, such as car manufacturing factories and phone assembly lines. The requirement to push the border of the "Robot Zone" [1] toward the manual manufacturing domain is increasing rapidly. Humans can execute manual manufacturing tasks easily using visual and force feedback, whereas robotic conventional methods, such as position control or visual servoing, are difficult to accomplish. Reinforcement learning (RL) shows potential to solve complex robot manipulation problems because it allows an agent to interact with the environment for trial-and-error learning and accepts high-dimension feedback as the input. [2], [3], [4].

For contact-rich manipulations, it is nontrivial to establish a robotic system that can learn a task with a safety guarantee and avoid wear and tear problem. Thus, sim-to-real methods are proposed [5] to address the aforementioned concerns. Recently, style transfer methods based on generative adversarial networks (GANs) [6] have been proposed recently in the computer vision field, enabling the use of vision-based manipulation tasks for deploying visual sim-to-real methods; however, owing to poorly simulated dynamics, the sim-to-real reality gap could be an issue when transferred the simulated policies to physical setups [7].

In this work, we verify our framework using the the most commonly used assembly task: peg-in-hole [8]. The framework training is performed in a simulated environment by employing only images captured using a camera as the input. When transferring the trained policy in our framework to a physical robot, the execution command is mapped from the position signal to the force signal to assist the peg-in-hole insertion task (we employ an admittance controller to perform the compliant movement).

Our framework can be described as follow. First, we train the RL policy net (we use soft actor-critic (SAC) algorithm in this work) in the simulator. Then, we use the images of insertion scenarios collected from a simulator and the real world to train a cycle-consistent generative adversarial networks (CycleGAN) [9]. Thereafter, the trained policy is driven by the real-to-sim transformed image style, obtained from the trained CycleGAN. The entire framework is shown

[1]Technische Universität München, [2]Agile Robots AG, [3]TAMS (Technical Aspects of Multimodal Systems), Department of Informatics, Universität Hamburg

[†] The first two authors contributed equally to this work.

in Figure 1.

Our primary contributions are listed:

- **C1:** A vision-based sim-to-real learning framework is proposed to perform assembly tasks.
- **C2:** A peg-in-hole task that effectively leverages visual information and force control using a simple reward function for a complete insertion, including hole searching, alignment, and insertion. The task performance is compared when training using different visual observation spaces.
- **C3:** The robustness of the framework to perturbations and sensor noise in real world is evaluated.

The remainder of the paper is structured as follows. In Section II, we describe the background and development of the deployed method. Section III introduces the problems. In section IV, we provide an overview of our method. A quantitative experiment of our methods and the experiment results in Section V. Section VI presents the conclusions and future work.

II. RELATED WORK AND BACKGROUND

A. Contact-Rich Assembly

The entire process of an assembly task can be considered as a constrained motion with geometrical and environmental constraints. Generally, we can decompose the existing peg-in-hole assembly strategies into two categories: contact model-based and contact model-free strategies [10]. Model-based strategies rely on the contact model analysis and compliant control. Two common examples of model-free strategies are learning from demonstration (LFD) and RL. For contact model analysis, analytical and statistical models are commonly used [11], [12], [13], [14], [15], [16]. Analytical models are based on the analysis of geometrical and environmental constraints, whereas statistical models rely on the estimation of the contact state by collecting samples. Among contact model-free strategies, the LFD approaches can be categorized into three phases: sensing, encoding, and reproducing [17], [18]. However, the sample efficiency of the aforementioned methods highly relies on the human operation and these methods introduce transparency problems in contact-rich teaching tasks [19].

B. Reinforcement Learning

The RL is a machine learning approach for teaching agents to solve different tasks based on trials and errors when interacting with environments. The RL agent aims to learn a policy $\pi(a_t|s_t)$, which selects the action a_t, and meanwhile the agent observes the environment s_t. The transition probability $p(s_{t+1}|a_t, s_t)$ is used to connect the state change over dynamics. The final trajectory can be represented as $\tau = (s_0, a_0, s_1, a_1, ...)$. The discount factor γ controls the sum of the reward. An optimal policy π^* should maximize the cumulative reward $r(s_t, a_t)$ during interactions with the environment, as shown in Equation (1).

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t)) \right] \qquad (1)$$

With the development of expressive function approximation such as neural networks (e.g., Deep RL), high-dimensional inputs such as raw images can be handled [20], [21]. Great success has been gained because of the advances in RL in many fields, for instance, the development of video games such as Atari [20], dexterous hand manipulation [22], robot grasping [23], and robot manipulation [24].

RL algorithms can be classified into two branches: model-based and model-free algorithms [3]. The main difference between the two branches is whether an agent gets access to or learns a model of the environment. Different assembly tasks are also solved using a model-based RL called mirror descent guided policy search (MDGPS) [25]. By combining the force information obtained from a force-torque sensor at the end-effector with a long short-term memory (LSTM), a high-precision assembly task was performed [26]. An operational space control framework was used in [27]. By combining visual inputs with natural rewards, different connector insertion tasks were demonstrated [28]. InsertionNet [29] was proposed to solve the general insertion problem by combining visual and force inputs, and it was trained in the real-word environment, which has safety risks [30].

C. CycleGAN

CycleGAN is an extension of GANS [6]. GANs comprise two submodels: generator and discriminator models. The key idea of GANs is to train the two submodels in a zero-sum adversarial game, until the discriminator is fooled by the generated examples half the time. In CycleGAN, except for the original adversarial loss, a cycle consistency loss is proposed: this loss can be used to calculate the reconstruction error of the images. Furthermore, CycleGAN offers a considerably more efficient approach for training than common GANs because of using unpaired and unlabeled dataset.

D. Sim-to-Real Transfer

Training in a simulator can assuredly provide an infinite amount of data and alleviate certain safety concerns during training. However, a reality gap exists that often separates a simulated task and its real-world analog, leading to failure when working with physical robots. To bridge the reality gap, three main options for solving this problem: system identification (SI), domain adaptation (DA) and domain randomization (DR) [31]. Using SI, we can remove inaccessible states and apply state estimation during the training [32]. DR helps the trained policy to adapt to different dynamics and generalize to dynamics of the real world [5], allowing the randomization of parameters for visual or pixel-based inputs such as lighting and textures [33].

For dexterous hand manipulation, a trained policy combing both dynamic and visual randomization is deployed [22]. The DA approach aims to map the source domain to the target domain; in the robotic context, this approach usually exploits
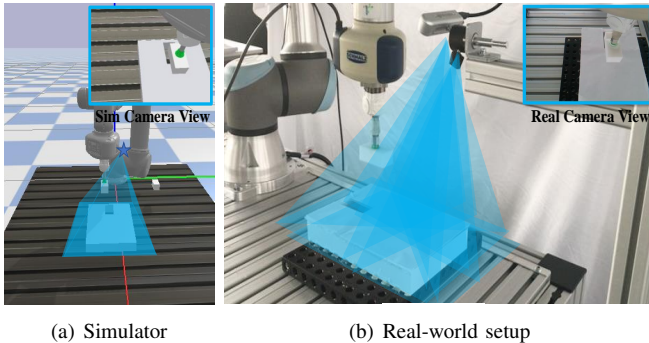
(a) Simulator   (b) Real-world setup

Fig. 2: Simulator and real-world setup: The blue area represents the view of the camera and the scene in camera can be seen in the insets of the two images (a) and (b).

the recent advances in visual domain adaptation [34], [35], [36].

III. PROBLEM STATEMENT

Owing to unknown contact mechanics, designing a feedback control mechanism for contact-rich tasks is challenging. RL has shown some progress in robotic contact-rich tasks in unstructured environments; however, sample efficiency and safety concerns are two main problems when performing policy training. Many RL algorithms require millions of steps to train policies for performing complex tasks [37], [2]. In other words, human supervision is always needed in resetting experiments, hardware status monitoring, and safety assurance, which is quite time-consuming and tedious [31].

The sim-to-real approach shows potential to solve the aforementioned problems; however, one significant difficulty associated with this approach is bridging the reality gap to address the mismatch in distinct distributions of rendered images and real-world counterparts. Another challenge is ascribed to force modeling in simulation as the force interactions will inevitably occur between the target object and environments when performing contact-rich tasks. Moreover, it is expensive to apply the system calibration due to the limitation of the simulation domain expert's ability [29] and accurate requirements [30].

IV. FRAMEWORK DESIGN

In this section, we present the overview of our sim-to-real framework proposed to solve the problems stated in Section III. We built our system in a Bullet simulator [38] and modeled the robot and task environment based on OpenAI Gym [39]. Figure 2(a) and Figure 2(b) shows the setup in the simulator and real world, respectively.

A. Learning Framework in Simulation

1) Policy: The soft actor-critic (SAC) algorithm [40] was employed in our framework. SAC introduces an entropy H in its objective function (Equation (2)), which is a significant characteristic, where α denotes a temperature parameter that determines the importance of the entropy term. The entropy
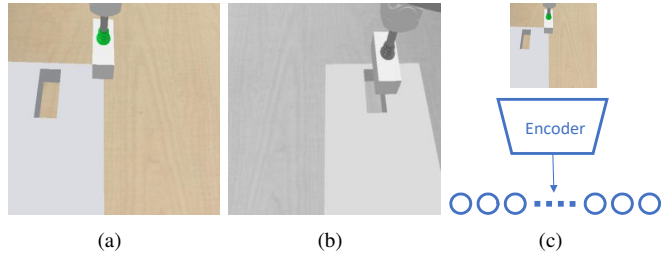

(a)   (b)   (c)

Fig. 3: Three different observation spaces: (a) a raw RGB image, (b) grayscale image, and (c) latent representation generated by an encoder.

is used to measure the randomness of a given policy. In this study, the policy is trained to maximize a value that relies on the expected return value as well as the entropy. It helps to reach a good trade–off between exploration and exploitation.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))) \right] \quad (2)$$

2) States: For a vision-based learning policy, the commonly used observation states are the RGB, grayscale, and latent representation [2], [28]. In this work, we select observation spaces as follow:

- RGB observation space: $3 \times 64 \times 64$ tensor
- Grayscale observation space : $1 \times 64 \times 64$ tensor
- Latent representation observation space: 128×1 vector.

For the RGB and grayscale observation spaces, the network conducts end-to-end learning; in other words, raw images are inputted to the network and the output command is obtained. For the latent representation observation space, an autoencoder is employed as a part of the network. This autoencoder comprises an encoder and a decoder, we exploit the encoder to compress the input image and generate the latent representation observation space.

3) Actions: Inspired by the literature [41], the necessary translation movement along the X-, Y-, and Z axes are considered and the orientation of the end-effector is fixed. We define a three-dimensional (3D) vector that contains the translation movement information of the robot. We use a position controller in the simulation, and the robot will move along a relative distance with respect to the current pose. The continuous 3D displacement action space ΔP

$$\Delta P = [\Delta x, \ \Delta y, \ \Delta z], \quad (3)$$

which considers translation movement along the X-, Y-, and Z-axes. The value in each axis is strictly in the interval of $[-0.02, 0.02]$ m.

4) Rewards: Some researchers set reward functions based on the different insertion phases such as reaching, alignment and insertion [41], [26], making the reward function hard to design; and need to distinguish the different phases. We only design one normal reward function that combines L1 and L2 distances for reaching, alignment and insertion phases and one reward for successfully insertion:

$$R(\mathbf{s}) = \begin{cases} 50, & \text{(Success)} \\ -(flag * 10 + 0.4 * (\|p_{obj} - p_{goal}\|) \\ \quad + 0.6 * (|p_{obj} - p_{goal}|)) & \text{(Otherwise)}, \end{cases}$$

where p_{obj} and p_{goal} represent the positions of the peg and hole, respectively, and *flag* is set to 1 if the robot moves to a distance exceeding a certain threshold (i.e., 15 cm away from the hole center); otherwise, it is set to 0. Here, *flag* works as a punishment when the robot makes unexpected movements.

B. Transfer Framework to Real-world Environment

1) Observation Space Transfer: To transfer our policy from the simulator to the real world, we must transfer the images from the domain of the real world to its counterparts in the simulator. Conventionally, training an image-to-image translation model requires a paired dataset. The requirement for paired examples is a limitation, it is challenging and expensive to prepare these datasets.

A successful approach for unpaired image-to-image translation is the CycleGAN. We commanded the robot to move randomly in the view of the camera and captured its random state each time. Approximately 200-300 images can be effortlessly obtained for training the model. Using the style transfer based on the CycleGAN, we map the view of the camera in the real-world environment to its counterpart, which we use to train our policy.

2) Action Space Transfer: In a study [29], researchers incorporated force augmentations by multiplying a random constant α with the force and the moment because they arrived at the conclusion that **the direction of the vectors** (F, M), but not the magnitude, **is the most important factor** in insertion operations. We extended this conclusion to our sim-to-real transfer process using a new method: we multiply gain K and the original position action output ΔP and then use the product C_{real} as the control command for the real robot force controller:

$$\begin{aligned} \boldsymbol{C}_{real} &= [F_x, F_y, F_z] \\ &= K\Delta P \\ &= K[\Delta x, \ \Delta y, \ \Delta z], \end{aligned} \tag{4}$$

where $K = 100$ N/m; thus, the force command values along the X-, Y-, and Z-axes are in the range $[-2, 2]$ N.

V. EXPERIMENTS

In this section, we introduce the peg-in-hole insertion task to validate our framework and explain the experimental results for both the simulated and real-world environments, in which we address the following questions:

1) Will all observation spaces work well in our framework?
2) Can our trained policy be transferred to a real-world environment successfully?
3) How does our framework perform compared with other insertion methods in terms of the success rate?
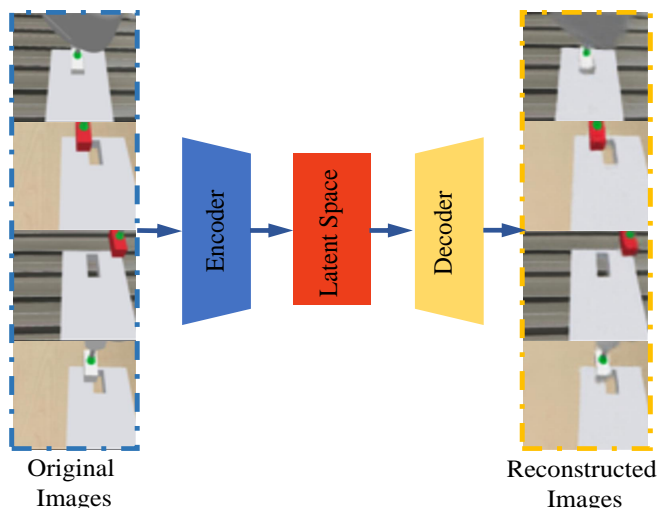

Fig. 4: Results obtained using an autoencoder: where the left panels show the original images and the right panels show the reconstructed images.

TABLE I: SUCCESS RATES OF 3 OBSERVATION SPACES

Total episodes	Observation space	Success rate
	Gray $1 \times 64 \times 64$	0%
3000	RGB $3 \times 64 \times 64$	0%
	Latent 128×1	**96%**
	Gray $1 \times 64 \times 64$	0%
10000	RGB $3 \times 64 \times 64$	0%
	Latent 128×1	**96%**

4) What is the robustness of our framework under external perturbations and target uncertainties?

A. Simulation Experiment

1) Observation Space Comparison: To compare the performances of the policy with different observation spaces, we test the scene of a white block with a metallic texture. We use the success rate of a complete insertion task in the simulation as a criterion to evaluate the performance of the learned policy.

A convolutional neural network is utilized as a part of the SAC network for training using the inputs from the RGB and grayscale observation spaces. An autoencoder is used to obtain a latent representation of the input image. The encoder part allows the compression of the original image to a lower-dimension vector that contains the important information.
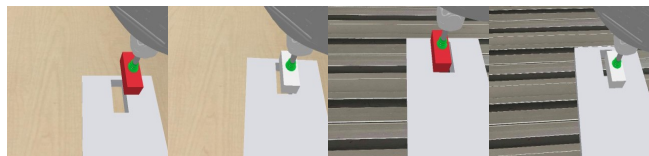

Fig. 5: Four environmental scenes. From left to right: a red block with a wooden texture, a white block with a wooden texture, a red block with a metallic texture, and a white block with a metallic texture.

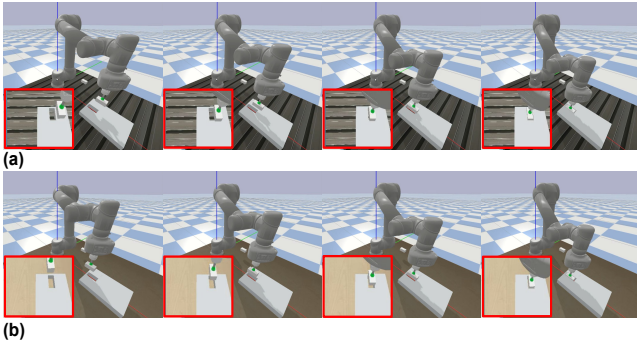In the simulation, we first generate a series of images of

Fig. 6: (a) Execution phases of a scene of a white block with a metallic texture scene and (b) scene of a white block with a wooden texture from the initial pose to the target hole.
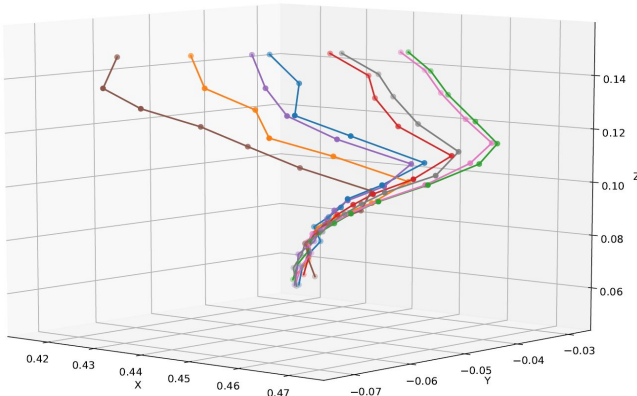

Fig. 7: Successful insertion trajectories starting from 8 different initial positions in the simulation. Eventually, the blocks are all moved into the target hole.

the robot state by executing random actions in the simulator as the training dataset and then train the autoencoder using this dataset. Thereafter, we extract the encoder as a part of the SAC network. We use the generate simulated images of the RGB observation space (size=$3 \times 64 \times 64$) to train the autoencoder.

We train the agent using cumulative episodes, and the results are shown in Table I. With the latent representation as policy input, the policy converged and the success rate could reach 96% at checkpoints 3000 and 10000 episodes. However, the raw RGB and grayscale observation spaces cannot train a feasible policy, which is consistent with the results reported in a study [28].

We can achieve the answer to question 1 from Table I. We conclude that end-to-end learning is not as efficient as the approach that uses the latent representation observation space. Hence, we perform the remaining experiments using the latent representation observation space.

2) Different Scene Evaluation: Although we demonstrate the policy performance before using a latent representation observation space in the scene of a white block with a metallic texture, it is unclear whether the difference in the scene will influence the performance. A high success rate must be achieved in the simulation environment to perform

TABLE II: SUCCESS RATES OF DIFFERENT SCENES (SIMULATION)

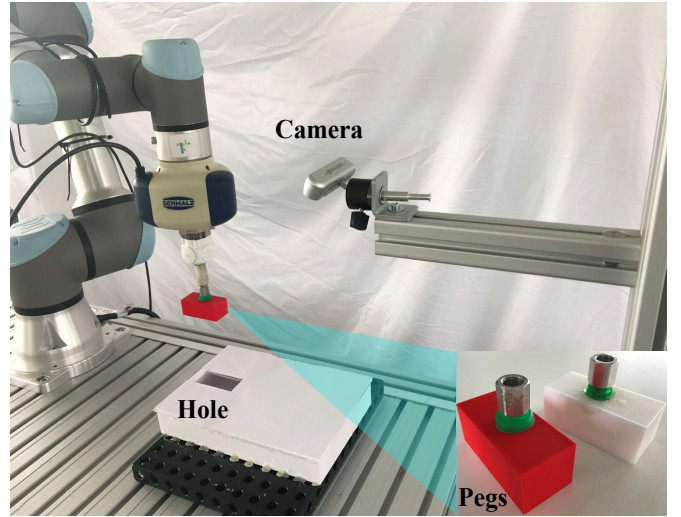| Evaluate Trials	Scene	Success Rate
	red block with wooden texture	96%
500	red block with metal texture	70.5%
	white block with wooden texture	99%
	white block with metal texture	96%

Fig. 8: Hardware settings for real environment experiment. The realsense D415 camera was installed manually without calibration, because we deliberately introduce uncertainties for the sim-to-real transfer to test it's robustness.

further real experiments. Additionally, to verify the generalization of the framework, we consider the permutation of four environmental scenes with two blocks and two textures: a red block with a wooden texture, a white block with a wooden texture, a red block with a metallic texture, and a white block with a metallic texture (Figure 5). Every scene is trained 3000 episodes in the simulation environment with a Dell Precision 5510 laptop CPU.

We compare the performance of this approach in different scenes, and the execution phase is presented in Figure 6. Additionally, a visualization of successful insertion trajectories is shown in Figure 7. Table II shows the success rate of the framework obtained under different scenes. Three scenes achieved a success rate higher than 96%. The white block with a wooden texture reached a 99% success rate.

B. Real-World Environment Experiment

Although we perform training and evaluate the success rate in the simulation environment, our goal is to transfer the trained policy to the physical environment.

In our real-world environment setup (Figure 8), a UR5e robot[1] is used to perform a peg-in-hole insertion task. This 6-axis robot features a 5 kg payload and a working radius of 850 mm. It is equipped with a 6 degrees of freedom force/torque sensor on the end effector. The robot uses an
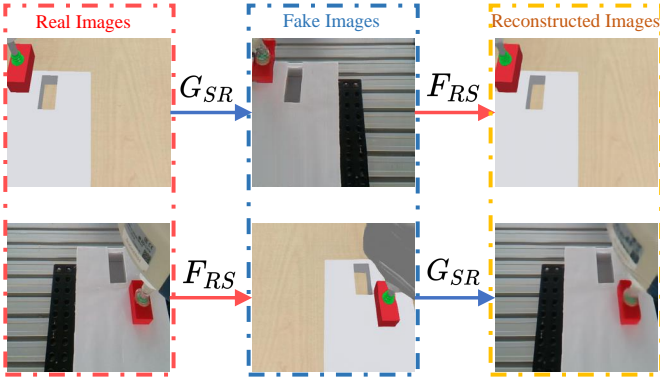
[1]https://www.universal-robots.com/products/ur5-robot/

Fig. 9: Domain adaptation by CycleGAN: Mapping function G_{SR}, which maps the simulated distribution to a real world distribution. Mapping function F_{RS}, which maps the real world distribution to simulated distribution.

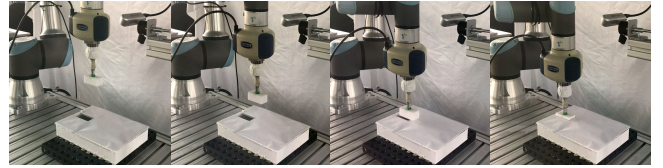Fig. 10: White block with metal texture scene real setup execution phases from initial pose to target hole.

operational space admittance controller [42] with 500 Hz control rate. The blocks are mounted behind the force/torque sensor to ensure the detection of the contact force with the environment.

An Intel RealSense D415 camera[2] is fixed on the platform to observe the operation. The position and orientation of the camera are selected to ensure the block and hole are visible during most of the training time. Figure 8 shows our hardware setup in the experiment. In our experiment, we use a white and a red block with same dimensions of $65 \times 30 \times 25$ mm, and a white block with a hole size of $70 \times 35 \times 30$mm. The clearance in each direction (i.e., length and width) is 5 mm. We select this setup because we aim to establish a potential scenario in which a packed data cable is inserted into a phone box in the mobile phone assembly line [43].

As described in Section IV-B, the CycleGAN is introduced to perform the domain adaptation process to transfer the image distribution from the real world to the simulation. We capture 200 images of the robot state in the real world and then generated a training dataset along with 3000 simulated images to train the CycleGAN. We train the CycleGAN model on four Nvidia 1080 Ti GPUs. The domain adaptation results of the trained CycleGAN with our setup inputs is shown in Figure 9.

Based on the previous results listed in Table II, we can conclude that the scene with wooden texture achieve the highest success rate with our policy. Hence, we transfer the real-world image to a scene of a block with a wooden texture using the DA method to evaluate our framework in a real-world setup. We define three situations when testing the policy in the real world as [2].

TABLE III: PERFORMANCES IN REAL ENVIRONMENT SETUPS

Scene	Complete insertion	Touched the box	Failed
Red block	86/100	10/100	4/100
White block	88/100	12/100	0/100

[2]https://www.intelrealsense.com/zh-hans/depth-camera-d415/

Complete Insertion means that the robot accomplishes the insertion task completely. *Touched the box* implies that the the peg was moved in the right direction, but the insertion is not completed. *Failed* indicates a situation in witch the robot moves far away from the target in the wrong direction or performs unexpected movements.

During the execution (Figure 10), we randomly occlude the camera's field of view for several seconds and push the robot in the wrong direction to the target hole to evaluate the system robustness to external perturbations. The performance of the physical robot in the real-world setup is summarized in Table III. We obtain an average success rate equal to method reported in the literature [2] with a safer sim-to-real framework because we limit the force command amplitude during the control.

VI. CONCLUSIONS AND DISCUSSIONS

In this work, we proved that our sim-to-real framework is a valid approach to solving the peg-in-hole task both in simulated and real-world environments. By employing DA and force controller, we can directly transfer the policy that was trained in a simulator to a real-world setup. Moreover, we evaluated different observation spaces and proved that the latent representation (i.e., low dimension) can accelerate the convergence of policy learning and afford a higher success rate for the task than end-to-end learning using raw image input. The importance of force control is shown by the fact that in real-world experiments, the blocks often needs to contact the environment and "slide" into the hole.

However, our method can be optimized further in terms of the performance and generalization ability. For example, the scene of a red block with a metallic texture in the simulation achieves a success rate of only 70.5% considerably worse than those achieved using the other three scenes; hence, further research is needed. Moreover, in the experiment setup, we assumed that the target orientation is known for the insertion task, which simplifies the task. For more general tasks, for example, when a robot starts with a random pose, our method must be improved to output the orientation of the end-effector. A more complex action space with both translation and rotation, $[\Delta x, \ \Delta y, \ \Delta z, \Delta r_x, \Delta r_y, \Delta r_z]$, can be designed for training. In our future roadmap, investigating the application of our sim-to-real approach to more industrial robotic tasks will be interesting.

REFERENCES

[1] S. Zei, "Manipulation skill for robotic assembly," Master's thesis, Technischen Universität Darmstadt, 2014.

[2] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8943–8950.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[4] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[5] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[7] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[8] H. Park, J.-H. Bae, J.-H. Park, M.-H. Baeg, and J. Park, "Intuitive peg-in-hole assembly strategy with a compliant manipulator," in *IEEE ISR 2013*. IEEE, 2013, pp. 1–5.

[9] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[10] J. Xu, Z. Hou, Z. Liu, and H. Qiao, "Compare contact model-based control and contact model-free learning: A survey of robotic peg-in-hole assembly strategies," *arXiv preprint arXiv:1904.05240*, 2019.

[11] D. E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *Trans. ASME, J. Dyn. Sys., Mes. Cont*, vol. 104, 1982.

[12] Z. Jakovljevic, P. B. Petrovic, and J. Hodolic, "Contact states recognition in robotic part mating based on support vector machines," *International Journal of Advanced Manufacturing Technology*, vol. 59, no. 1-4, pp. 377–395, 2012.

[13] J. Xiao and L. Liu, "Contact states: Representation and recognizability in the presence of uncertainties," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, 1998.

[14] L. M. Brignone and M. Howarth, "A geometrically validated approach to autonomous robotic assembly," *Nottingham Trent University*, vol. 2, pp. 1626–1631, 2002.

[15] I. F. Jasim and P. W. Plapper, "Contact-state monitoring of force-guided robotic assembly tasks using expectation maximization-based gaussian mixtures models," *International Journal of Advanced Manufacturing Technology*, vol. 73, no. 5-8, pp. 623–633, 2014.

[16] G. E. Hovland and B. J. McCarragher, "Hidden markov models as a process monitor in robotic assembly," *The International Journal of Robotics Research*, vol. 17, no. 2, pp. 153–168, 1998.

[17] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, 2018.

[18] M. Kyrarini, M. A. Haseeb, D. Ristić-Durrant, and A. Gräser, "Robot learning of industrial assembly task via human demonstrations," *Autonomous Robots*, vol. 43, no. 1, pp. 239–257, 2019.

[19] Y. Shi, Z. Chen, Y. Wu, D. Henkel, S. Riedel, H. Liu, Q. Feng, and J. Zhang, "Combining learning from demonstration with learning by exploration to facilitate contact-rich tasks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1062–1069.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[21] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[22] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. Mc-Grew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[23] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[24] 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.

[25] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2062–2069.

[26] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 819–825.

[27] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4383–4388.

[28] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," *arXiv preprint arXiv:1906.05841*, 2019.

[29] O. Spector and D. Di Castro, "Insertionnet-a scalable solution for insertion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, 2021.

[30] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, F. Golemo, M. Mozifian, C. Atkeson, D. Fox, K. Goldberg, J. Leonard *et al.*, "Perspectives on sim2real transfer for robotics: A summary of the r: Ss 2020 workshop," *arXiv preprint arXiv:2012.03806*, 2020.

[31] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, 2021.

[32] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[33] 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 *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[34] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[35] K. Arndt, M. Hazara, A. Ghadirzadeh, and V. Kyrki, "Meta reinforcement learning for sim-to-real domain adaptation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2725–2731.

[36] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[37] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[38] "Bullet physics engine," https://pybullet.org/wordpress/.

[39] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[40] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[41] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," *arXiv preprint arXiv:1810.10191*, 2018.

[42] N. Hogan, "Impedance control: An approach to manipulation: Part itheory," 1985.

[43] R. Song, F. Li, T. Fu, and J. Zhao, "A robotic automatic assembly system based on vision," *Applied Sciences*, vol. 10, no. 3, p. 1157, 2020.