Zero-Shot Sim-to-Real Transfer of Reinforcement Learning Framework for Robotics Manipulation with Demonstration and Force Feedback

Yuanpei Chen[1], Chao Zeng[2], Zhiping Wang[3], Peng Lu[4] and Chenguang Yang[1]

***Abstract*— In the field of robot reinforcement learning, the reality gap has always been a problem that restricts the robustness and generalization of algorithms. We propose Simulation Twin (SimTwin) : a deep reinforcement learning framework that can help directly transfer the model from simulation to reality without any real-world training. Simulation Twin consists of a reinforcement learning module and an adaptive correct module. We train the policy using the soft actor-critic algorithm only in a simulator with demonstration and domain randomization. In the adaptive correct module, we design and train a neural network to simulate the human error correction process using force feedback. Subsequently, we combine the above two modules through digital twin to control real-world robots, correct simulator parameters by comparing the difference between simulator and reality automatically, then generalize the correct action through the trained policy network without additional training. We demonstrate the proposed method in an open cabinet task, the experiments show that our framework can reduce the reality gap without any real-world training.**

I. INTRODUCTION

In recent years, deep reinforcement learning has shined in the field of robot manipulation, especially of learning continuous control in the real world. However, it is still difficult to collect enough data for training in reality. While some researchers train robots directly in reality, others train the policy in simulation first and then transfer the trained model to reality [1]. But due to the difference between simulation and reality, the so-called reality gap, often does not work well by directly transferring the learned model to reality. Therefore, it is necessary to find a good sim-to-real transfer method to close the reality gap.

To achieve sim-to-real transfer, some researchers use the transfer learning methods like domain adaption to training the robust models [2], reducing the impact of the difference in data distribution between simulation and reality, so as to reduce the number of training required for transfer from simulation to reality. But this method still requires trial-and-error in reality to train the model. For some task scenarios such as medical treatment, high-risk, it is still difficult to expect in the real-world, the pursuit of generalization also makes us hope that the model can directly perform well in similar scenarios that have never been seen before. Therefore, zero-shot sim-to-real transfer learning is proposed [3], which can achieve good results without any fine-tune in reality.

Although several fine results have been achieved (see, e.g., [4], [5]), the zero-shot sim-to-real transfer is still quite challenging. In our work, we propose a new framework that can directly transfer the policies trained in simulation to the real environment without additional training. Our method uses the force information collected from the end-effector to achieve automatic error correction to reduce the reality gap. And our method can use partial reality state information, and only compute reward in the simulation, which benefits some scenarios where it is difficult to obtain accurate states.

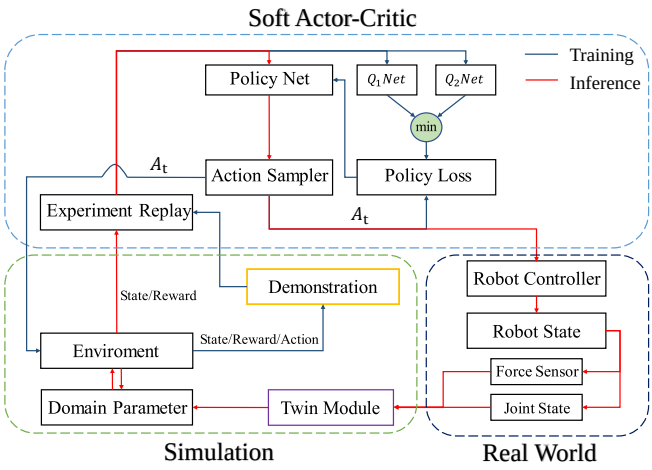

Fig. 1. An overview of the SimTwin framework. During training, the replay buffer obtains training data from the environment and demonstration to training policy. During inference, observations are obtained from the simulation environment, and the trained policy output actions are passed to the actual robot controller for execution. The controller then transmits the force and joint information of the reality back to the simulation and corrects the domain parameters through Twin-Module, which plays an error-correction role.

II. RELATED WORK

A. Deep Reinforcement Learning for Robot Manipulation

Deep Reinforcement Learning allows for end-to-end dexterous manipulation learning, with continuous high-dimensional action and state space [6] [7]. For the problem of low sample utilization, [8] used the Deep Deterministic

This work was supported in part by the National Nature Science Foundation of China (NSFC) (U20A20200, 62003096), in part by the Guangdong Basic and Applied Basic Research Foundation (2019B1515120076, 2020B1515120054), and in part by the Fellowship of the China Postdoctoral Science Foundation under Grant 2020M682613.

[1]Yuanpei Chen and Chenguang Yang is with College of Automation Science and Engineering, South China University of Technology, Guangzhou, China

[2]Chao Zeng is with School of Automation, Guangdong University of Technology, Guangzhou, China

[3]Zhiping Wang is with School of Electronic Engineering, Dongguan University of technology Dongguan, China

[4]Peng Lu is with the Department of Mechanical Engineering, The University of Hong Kong

*Corresponding author is Chenguang Yang (Email: cyang@ieee.org)

Policy Gradient (DDPG) which has the mechanism of experience replay to improve the sample efficiency. [9] use entropy regularization to balance exploration and exploitation. In our framework, we use the SAC algorithm. [10] save the demonstration data in the experience replay to speed up the learning process, our framework also use this method.

B. Sim-to-Real Transfer and Zero-Shot Learning

In previous studies, [2] use the domain adaption method to map the simulation and reality into a latent space to cross the sim-to-real gap. [11] learn an inverse transition probability matrix in the real environment to directly apply the trained model. However, they all need to use real-world data, which is very difficult for some specific tasks.

Zero-shot sim-to-real learning, instead, enables transferring the model from simulation to reality without using the data of the reality. Domain randomization [4] [5] randomize the visual information or dynamic parameters in the simulated environment, which is a popular method in zero-shot sim-2-real transfer. We also use the domain randomization method to train the model in the simulator.

In recent years, there are some other zero-shot transfer methods. [12] proposed a feedback policy that is conditioned on a sensory observation history to adapt the scenario. It also uses force information to adjust the trained policy, but a new student neural network is needed to be trained since we can directly correct the sim-to-real errors. Next, the idea of our work is similar to the MRAC-RL framework [13], which is a framework for online policy adaptation under parametric model uncertainty. Our error correction method is similar to their adaptive controller module, but the difference is that we can correct inference model through force feedback, at the same time we will have lag. It should be pointed out that our framework does not conflict with its framework and can be used at the same time.

Force control has shown promising results for robotic manipulation [14] [15]. Recently, several attempts have been made to combine force control and reinforcement learning [16] [17]. [16] proposed a RL method that can learn variable impedance control policy in the task space. However, there is still a gap between the physics engine in simulation and reality, it may not be a good choice to use it directly for training. [17] proposed a learning-based framework that combines RL and traditional force control. They also learned the corrected trajectory through the force information at the time of the collision, but they scheme to combine with traditional PID. In fact, in the use of force information, we innovatively map it to an abstract state in the task and use the digital twin technology [18] to achieve the automatical error correction during inference.

The important principle of our work is based on the digital twin [18], which is an industry 4.0 concept and has been recently used for RL [19] [20]. The digital twin establishes the model of the object in the digital space, realizes complete synchronization with the real state of the object through the sensors, performs real-time analysis and evaluation after the movement of the object. Similar to the digital twin, we have established a relation between reality and simulation for the robot. Our method could correct the errors in real-time through the force sensor to achieve a lower reality gap. To the best of our knowledge, our work is the first attempt to use this method similar to digital twin for zero-shot sim-to-real transfer.

III. METHOD

Our proposed framework SimTwin, as shown in Fig. 1, consists of three major modules: Simulation-to-Reality loop, Soft Actor-Critic, and Twin module. The zero-shot sim-to-real transfer will be implemented in the Twin module, These modules will be discussed in the following subsections.

A. Simulation-to-Reality Loop

The policy neural network is completely trained in simulation, and a sim-to-real loop is used to control the real robot. Like digital twin, we use a control method that maps robot states in simulation to reality. The specific method is: let the simulation robot send the current joint angle values to the real robot continuously at a faster frequency, and the real robot moves to the target joint angle after receiving the send joint angle, so as to realize the real robot following the simulation robot.

The original intention of using digital twin is that some tasks are not easy to observe, and the environment can be modeled through prior knowledge. In Simulation Twin, the simulation environment store information about the environment and make decisions based on this information.

An advantage of this method is that it does not need to observe all states in reality. However, it must be pointed out that this control method is quite dependent on the accuracy of the simulation model. Therefore, the states of the simulated robot and the real robot will not be always identical due to the model difference. We will explain in the following chapters that how to use our proposed Twin Module to adaptively reduce this reality gap.

B. Soft Actor-Critic from Demonstration

Reinforcement learning from demonstration has better performance in complex tasks. Choosing an off-policy reinforcement learning algorithm can help reduce policy forgetting. Therefore, we selected Soft Actor-Critic (SAC) as the deep reinforcement learning algorithm.

$$\mathcal{J}(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} [r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathscr{H}(\pi(\cdot \mid \mathbf{s}_t))] \quad (1)$$

Where r is the step reward, $\mathbf{s}_t, \mathbf{a}_t$ are the state and action in the time step t. α is a weight parameter that determines the relative importance of the entropy term versus the reward. The update method of the Policy network (parameter θ) is to minimize the KL-divergence.

$$\mathcal{J}_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} [\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} [\alpha \log(\pi_\phi(\mathbf{s}_t \mid \mathbf{a}_t)) - \mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t)]] \quad (2)$$

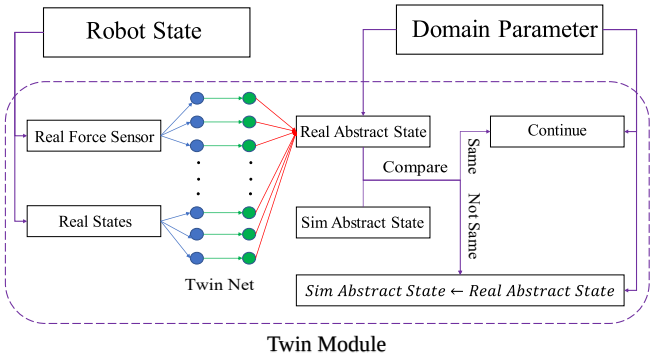

Fig. 2. Twin-Module overview. When real robot state and force information input, a neural network is used to determine which abstract state the reality is in, and compare with the simulation abstract state. If the same, do nothing. If they are not the same, let the simulation abstract state be reality abstract state

where $\mathcal{Q}(\mathbf{s}_t, \mathbf{a}_t)$ is the soft Q function given by the Q learning framework, which can be trained to minimize the soft Bellman residual:

$$\mathcal{J}_{\mathcal{Q}}(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}[\frac{1}{2}(\mathcal{Q}(\mathbf{s}_t,\mathbf{a}_t) - (r(\mathbf{s}_t,\mathbf{a}_t))+ \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p}[\mathbb{E}_{\mathbf{a}_{t+1}\sim\pi_\phi}[\mathcal{Q}(\mathbf{s}_{t+1},\mathbf{a}_{t+1})- \alpha\log(\pi_\phi(\mathbf{s}_{t+1}\mid\mathbf{a}_{t+1}))]])] \quad (3)$$

which γ is the reward discount factor. We use automating entropy adjustment for update α in SAC algorithm after solving for π_t^* and $\mathcal{Q}_t^*$.

$$\mathcal{J}(\alpha) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t^*)\sim\pi_t^*}[-\alpha_t\log\pi_t^*(\mathbf{a}_t\mid\mathbf{s}_t;\alpha_t) - \alpha_t\overline{\mathscr{H}}] \quad (4)$$

In origin SAC, random actions are started to interact with the environment, and transition(s, a, r, s') are obtained to fill the replay buffer $\mathcal{D}$. In our framework, the demonstration action $a*$ will be used to interact with the environment at the beginning to obtain a higher quality transition$(s, a*, r, s')$, fill it into the replay buffer and keep it forever. If the demonstration steps are done, SAC controls the rest of the training. In our framework, all the demonstration steps are completed in simulation, which is more convenient to obtain demonstration data than in reality.

C. Twin Module

Due to the sim-2-real gap, we propose an adaptive error correction method to achieve zero-shot transfer in simulation and reality. We think that this process is independent of the logic of opening the cabinet policy trained by reinforcement learning, so we use another neural network: Twin-Net to simulate this thinking process. An overview of Twin-Module is shown in Fig. 2.

Firstly, we demonstrate the Twin-Net training method. We define some abstract states ($\mathbf{As}_1$, $\mathbf{As}_2$, ..., $\mathbf{As}_n$) which decompose the task into several abstract states, For example, we can decompose the open cabinet task into three abstract states: not grasping the handle, just grasping the handle, pulling the handle back. A condensed observation at time t is given by:

$$\mathbf{o}(t) = \Big\{\mathbf{F}_x(t),\, \mathbf{F}_y(t),\, \mathbf{F}_z(t),\, \mathbf{s}(t)\Big\} \quad (5)$$

which $\mathbf{F}_x(t)$, $\mathbf{F}_y(t)$, $\mathbf{F}_z(t)$ are the force values of the end-effector, $\mathbf{s}(t)$ is t time robot state, which does not have to be the same as the used one for RL training. Labels are defined as below:

$$\mathbf{A}(t) = \Big\{\mathbb{1}(\mathbf{As}(t),\, \mathbf{As}_1),\, ...,\, \mathbb{1}(\mathbf{As}(t),\, \mathbf{As}_n)\Big\} \quad (6)$$

$\mathbf{As}(t)$ is the abstract state of the robot, which can be obtained from the simulation. $\mathbb{1}(\mathbf{As}(t),\, \mathbf{As}_n)$ is the indicator function, which is 1 when the $\mathbf{As}(t)$ is equals to $\mathbf{As}_n$ and is 0 otherwise.

We aims to learn

$$T_\theta(\mathbf{o}(t-(T-1):t)) \longrightarrow \mathbf{A}(t) \quad (7)$$

which map the history of the last T condensed observationse $\mathbf{o}(t-(T-1):t)$ to current abstract state, where $T = 8$ is the fixed history length, θ is parameter of this network, it can be trained to minimize the negative log-probability:

$$\mathcal{L}(T_\theta) : = -\log(\mathbf{Prob}(T_\theta(\mathbf{o}(t-(T-1):t)))-\mathbf{A}(t)) \quad (8)$$

This neural network has 2 linear hidden layers with 64 neural units in each layer. Rectified Linear Unit (ReLU) activation function is used in all hidden layers.

During inference, we input the force and robot state in the reality into Twin-Net to obtain the abstract state in reality. When there is a difference between the state in reality and simulation due to the reality gap, SimTwin corrects the abstract state in simulation to make it consistent with reality. Therefore, new actions will be planned through reinforcement learning, to achieve automatic error correction, until the task is completed.

D. Domain Randomization

Domain randomization is an effective way to enhance the success rate of sim-to-real transfer[21]. In our framework, domain randomization plays an important role in error correction. It makes the model adaptive to noise by adding noise to the parameters during training. In our framework, after the twin module gives the correct parameters, domain randomization guarantees that the model can still give the correct action after correction.

IV. EXPERIMENTS

In our experiments we focus on answering the following questions:

- Does our framework can work for real-world tasks and robots?
- How does the Twin module reduce the translate gap?

We start by applying SimTwin to a cabinet opening task, showing that we can directly transfer trained policy to real robots such as Baxter for the complex articulated task, and without any other real data.

Fig. 3. Simulation and real-world settings. Left is the open-cabinet task trained by SimTwin with multiple agents in the simulation. Right is the experimental setup for the open-cabinet task in the real world.

Next, we will apply SimTwin in transferring policies between scenes with different initial poses of the cabinet in the cabinet opening task. We demonstrate that our Twin-Module improved the performance of the RL-trained policies applied to systems with such modeling error.

A. Task

The open-cabinet task requires the use of grippers to clamp the handle and pull the drawer out. We use a 6-DoF robot arm from Baxter in this task, Simulated and real-world settings are shown in Fig. 3.

Due to the advantage that the policy in our framework is completely run in simulation, we can choose as many states as possible to train the model, because the states are easier to obtain in the simulation. So we use a 16D observation space: 6-DoF joint angles, the 3D positions of the cabinet drawer handle, the robot left finger and robot right finger, 1D of drawer open length. The reward function consists of reward shaping from the end-effector to the handle, the opening distance of the cabinet, and a rotating reward that makes sure the two robot fingers are on the handle. A 6D action space directly outputs the speed of the 6-DoF joint angle values to control the baxter arm.

It should be noted that our framework has another advantage in that we do not need to obtain all state information in the real environment, such as the position of the handle. It is very convenient for some difficult-to-observe tasks.

B. Simulation

We use Isaac-Gym [22] as our simulation environment. It utilizes NVIDIA PhysX, a powerful physics engine, which can effectively reduce the reality gap. Another advantage is that model rendering and interaction with the environment can be fully loaded into the GPU to better support parallel computing. Both the training and testing stages run on a PC with Intel i7-9700K CPU @ 3.60GHz and NVIDIA RTX 2070Super GPU.

C. Real World Experiment

To prove the generalization ability of SimTwin, we choose to use handles different from simulations in real experiments. The size of the handle in the real world is smaller than in the simulation, which means it is more difficult to grasp.

We assume that in the open cabinet task, the reality gap is caused by the different positions of the cabinet. Before the training starts, we have an initial estimate of the positions of the cabinet. During the training, we use two types of randomization: observation randomization and dynamic randomization. The observation randomization represents the uncertainty of sensors, which is the focus of our framework. The dynamic randomization represents the model inaccuracy, it is a general method to improve sim-to-real ability. Noise is modeled as an additive or scaling noise sampled from Gaussian distributions $\mathcal{N} \sim (\mu, \rho)$. The experiment configuration is given in Table. I.

TABLE I
DOMAIN RANDOMIZATION CONFIGURATION

Parameters	μ	ρ	Operation
Cabinet Initial Position $\mathbf{p}$	0	0.05	Additive
Actions $\mathbf{a}_t$	0	0.05	Additive
Baxter Rotational inertia	0	0.15	Scaling
Baxter Friction	0	0.15	Scaling
Cabinet Friction	0	0.15	Scaling
Cabinet Mass	0	0.15	Scaling

At the beginning of the training, we manually set an open cabinet trajectory in the simulation and store this demonstration data in the replay buffer. Next, we use the SAC algorithm and gradient descent method to train the policy and Twin-Net until achieving good results in simulation.

After training, we use SimTwin to transfer our policy to reality. We use a PC mentioned above to run our simulation environment, use Baxter SDK to control Baxter, and they are connected via wifi. When the simulation is turned on, the PC size packages the joint angle values of the robot in the simulation into a ROS topic and sends it to the Baxter size. The Baxter size executes the angle value, and sends the force feedback to the PC size in the same way, enters the Twin module. After the Twin module has updated the domain parameters (such as X-coordinate), the policy net inference of the next action and repeat the above process. ROS topic receiving frequency, Baxter joint angle update frequency are both 100hz. Our control method is shown in Fig. 6.

We set the X-coordinate in the simulation and reality to be the same, and then use the above method to perform the open-cabinet task. Our experiments snapshot is shown in Fig. 4. We can see that when the ideal situation (reality gap is relatively small), our control method can well transfer the policy from the simulation to reality. Next, we will show that in the case of a large reality gap, how does our framework perform sim-to-real transfer with twin module. Assuming that the cabinet in the simulation is farther than the reality due to the reality gap, there will be the gripper that has already touched the cabinet in the reality but not touched in the simulation. At this time, the force in the simulation is quite different from that in reality. After the twin module detects this abnormality, it can judge that the gripper in the reality has touched the cabinet, so we correct the position of the cabinet in the simulation to make it contact with the gripper. After that, the SAC algorithm will generalize to a new trajectory and successfully open the cabinet. In the end,

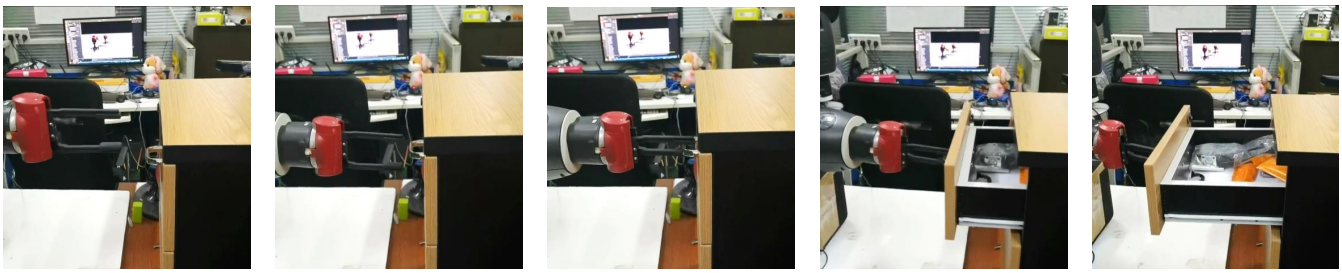

Fig. 4. Snapshots of the manipulator performed during the open-cabinet task.

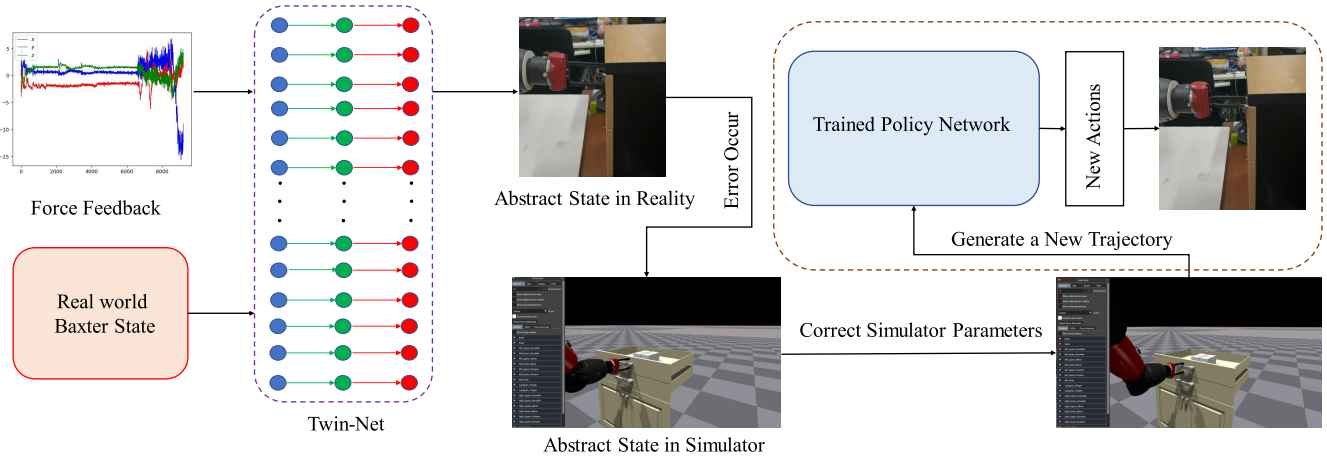

Fig. 5. Our framework is used in open cabinet tasks. Input the force feedback and baxter state into Twin-Net to get the abstract in the real world. When there is an error (collision in the figure) occurred, correct the parameters in the simulator to make it the same as in reality, and get new correct actions through the trained policy.

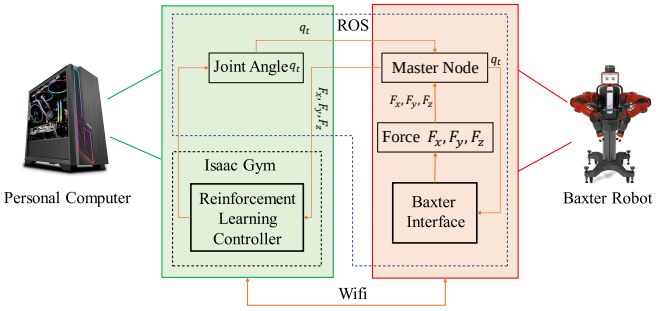

Fig. 6. The experimental configuration of our real-world experiment. The IsaacGym is running in a personal computer, sent and receive data from baxter robot by ROS across two machines.

the position error was corrected, and the reality gap was reduced. This process is shown in Fig. 5.

Our code is released at: https://github.com/cypypccpy/Isaac-ManipulaRL

D. Performance Without Demonstration

We demonstrate the smoothed episodic reward in Fig. 7, the smooth function is $r'_{t+1} = 0.375r'_t + 0.625r_{t+1}$. The blue curve corresponds to the demonstration enabled, while the purple curve corresponds to the result with demonstration removed. It can be seen that after the demonstration is removed, the agent can not successfully learn a useful policy. We think that it is because the probability of finding the correct action by random exploration for complex tasks is low, and it is easy to obtain the demonstration data in the simulation, so we have also added the demonstration part to our framework. It can be seen that demonstration can both improve the learning speed and stability.

E. Performance Without Twin Module

We also test the generalization ability of our framework when the Twin module is not used. Taking X-coordinate as a test, in the case of only domain randomization is used, the tolerable error is about $-1 \sim 1cm$. In the case of using Twin Module and domain randomization at the same time, the maximum error tolerance can reach about $-4.5 \sim 5cm$, and the cabinet can still be opened in this error. It should be pointed out that the error tolerance of the Twin module is related to the generalization ability of the trained RL model. When the error is too large, the reason for the generalization failure is that the RL model cannot plan a new path successfully after correcting the domain parameters. Therefore, to improve the generalization ability of the RL model, domain randomization is still very important in SimTwin.

V. CONCLUSIONS

Zero-shot sim-to-real transfer is an important component for the generalization of reinforcement learning in robotics.

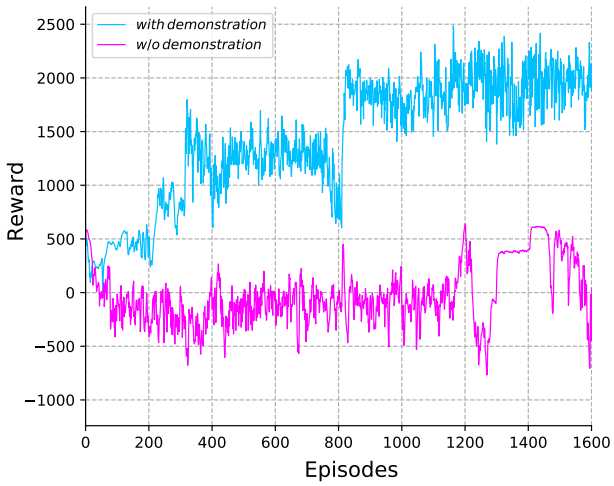

Fig. 7. Comparison of smoothed episodic reward of begin with and without demonstration. Except for this, the other configurations are the same at the 1600th episode. But only the former case can find the policy successfully.

In this work, we indicate that SimTwin can directly transfer the policy sim-to-real without additional training. SimTwin uses the force information of the real-world robot to correct the errors between simulation and reality. We designed a neural network to learn the difference between reality and simulation. When there is a difference between reality and simulation, SimTwin modifies the domain parameters in the simulation, then the RL model can generalize to a new trajectory. SimTwin uses Baxter robots for experiments and has achieved good results on open cabinet tasks.

In this work, our framework is proven to be used for open cabinet tasks. Next, we will try to use SimTwin for other tasks that are complex and hard to collect real-world data. We also plan to develop more general error monitoring methods based on force information to increase the generalization of our framework. Now our framework is still more reliant on the establishment of the simulation environment. In the future, we plan to combine SimTwin with perception algorithms to reduce the reliability of the simulation environment.

REFERENCES

[1] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[2] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[3] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[4] 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.

[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] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 590–595.

[7] 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8973–8979.

[8] 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.

[9] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[10] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[11] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[12] T. Bi, C. Sferrazza, and R. D'Andrea, "Zero-shot sim-to-real transfer of tactile control policies for aggressive swing-up manipulation," *IEEE Robotics and Automation Letters*, 2021.

[13] A. Guha and A. Annaswamy, "Mrac-rl: A framework for on-line policy adaptation under parametric model uncertainty," *arXiv preprint arXiv:2011.10562*, 2020.

[14] C. Zeng, H. Su, Y. Li, J. Guo, and C. Yang, "An approach for robotic leaning inspired by biomimetic adaptive control," *IEEE Transactions on Industrial Informatics*, 2021.

[15] C. Zeng, Y. Li, J. Guo, Z. Huang, N. Wang, and C. Yang, "A unified parametric representation for robotic compliant skills with adaptation of impedance and force," *IEEE/ASME Transactions on Mechatronics*, 2021.

[16] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1010–1017.

[17] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contact-rich manipulation tasks with rigid position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.

[18] J. Wu, Y. Yang, X. Cheng, H. Zuo, and Z. Cheng, "The development of digital twin technology review," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 4901–4906.

[19] K. Xia, C. Sacco, M. Kirkpatrick, C. Saidy, L. Nguyen, A. Kircaliali, and R. Harik, "A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence," *Journal of Manufacturing Systems*, vol. 58, pp. 210–230, 2021.

[20] M. Mayr, K. Chatzilygeroudis, F. Ahmad, L. Nardi, and V. Krueger, "Learning of parameters in behavior trees for movement skills," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 7572–7579.

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

[22] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.