Robot Vision-Based Autonomous Navigation Method Using Sim2Real Domain Adaptation

1st Mengjiao Zhang
dept. Computer Science
University of Science and Technology Beijing(USTB)
Beijing, China
email mengjiao2023@xs.ustb.edu.cn

2nd Shihong Duan
dept. Computer Science
USTB
Beijing, China
email duansh@ustb.edu.cn

3rdCheng Xu
dept. Computer Science
USTB
Beijing, China
email xucheng@ustb.edu.cn

***Abstract*—Vision-based autonomous navigation is pivotal in mobile robot technology, involving autonomous movement, path planning, obstacle avoidance, and target reaching in unknown environments. Reinforcement Learning (RL) offers a trial-and-error-driven learning method for autonomous navigation. However, training RL models directly on physical robots is time-intensive and poses potential risks. Simulators accelerate RL training efficiency and reduce costs, but they typically provide only approximate models of robot dynamics and their interactions with the environment. This leads to a simulation-to-reality gap (Sim2Real Gap), where strategies trained in simulation underperform in real-world applications, sometimes resulting in task failures. To address the issue of reducing the Sim2Real Gap in visual sampling training, this paper establishes a bridging plugin between the simulator and ROS, enabling the subscription to interaction data between the ROS-based robot and its real-world environment. We propose a Sim2Real domain adaptation method based on CycleGAN, which generates effective visual observations for autonomous navigation learning. The experimental results demonstrate that this domain adaptation-based method, by utilizing minimal real-world data, significantly reduces the Sim2Real Gap compared to approaches relying solely on simulation data, achieving a performance improvement of 62.38%.**

***Index Terms*—Sim2Real; Domain adaptation; Visual navigation; Reinforcement learning.**

I. INTRODUCTION

In recent years, Deep Reinforcement Learning (DRL) has shown advantages in robotic visual navigation research. However, due to the inefficiency and high cost of training in the real world, researchers commonly choose to train and test robots in simulated environments [1], [2], [3]. In simulated environments, data collection is more efficient, safe, and scalable. By conducting extensive training in simulated environments, agents can perform well in unknown and complex environments [3], such as efficiently navigating indoor environments [4], controlling vehicles in dynamic urban settings [5], following natural language instructions, and even answering complex questions [6]. These advancements demonstrate the bridging role of simulated environments to the real world, providing a solid foundation for the successful deployment of agents from simulation to reality. Despite the advantages of training agents in simulated environments, challenges remain when transferring them to real-world navigation, primarily due to inconsistencies in data distribution and the adaptability of visual perception models between simulation and reality [2].

Given that robots trained in high-fidelity simulators often perform poorly when transferred to autonomous navigation in the real world, how to effectively bridge the gap between simulation and reality (Sim2Real Gap) becomes a key challenge [7], [8]. [9] conducted a large-scale evaluation, and the results demonstrated that creating high-fidelity simulators does not aid in learning due to the slow simulation speed and overfitting to inaccurate simulation physics. In contrast, using real-world data can improve learning and generalization. Therefore, the main issue addressed in this paper is how to acquire data from real environments and how to transfer the autonomous navigation models on few-shot real data.

In existing work, researchers have also proposed Domain Randomization [10], [11], [12], which introduces randomness into the simulation to approximate the complexity of the real world and improve the physical reality of the simulation environment, including more accurate physical engines and sensor models. However, improving the reality of the simulation environment requires a lot of computing resources and time. Even if the simulation environment is realistic, it is difficult to fully cover all the complex situations in the real world.

In order to effectively narrow the Sim2Real Gap, this paper proposes a method based on domain adaptation. The main purpose is to build a Sim2Real platform to realize the data conversion between the simulation environment and the real environment, so as to achieve the goal of mutual fusion of environmental data. In the process of reinforcement learning training, CycleGAN is integrated to achieve unsupervised domain adaptation, which not only reduces the difference between simulation and reality, but also significantly improves the migration effect of the model.

In summary, our main contributions are as follows:

1) Combining the weighted difference between the navigation metrics of the simulated and real environments, this paper defines the Sim2Real Gap using a comprehensive normalization method.
2) The Sim2Real platform is designed to simultaneously re-

The Guangdong Provincial Fund for Basic and Applied Basic Research (2023A1515140071) and the National Natural Science Foundation of China (62101029)

ceive simulation and real data and realize data conversion between them, so that the robot can obtain consistent data input for training in both environments and realize the migration from simulation to reality.

3) Based on the Sim2Real platform, the reinforcement learning algorithm is integrated with CycleGAN to comprehensively optimize strategies and data conversion, generate more and more effective training sample data, reduce data differences between different domains, and improve the migration effect of the model.

The rest of this paper is organized as follows. Section 2 introduces the related work in this field. Section 3 introduces the Sim2Real platform built in this paper. Section 4 introduces the domain adaptive training method proposed in this paper. Section 5 introduces the experiments conducted and their results. Section 6 concludes this paper.

II. Related Work

The difficulty of Sim2Real is that the model trained in the simulation environment is difficult to directly apply to the real environment, mainly focusing on the differences in sensing and actuation. In order to make up for this gap, existing research focuses on the accuracy and dynamic variability of simulated environments in terms of execution, so that the behavior of intelligent agents can be more accurately reflected in more diverse environments [13]. In terms of sensing, due to the complexity and variability of scenarios in real environments, situations that have not been encountered in simulated environments often occur, making the model perform poorly in practical applications [14].

Domain Adaptation is one of the effective methods to solve the Sim2Real problem [15]. As a subclass of transfer learning, domain adaptation usually targets scenarios where the source domain has sufficient labeled data, while the target domain data is scarce or even nonexistent. In order to improve the performance of the model in the target domain, the core of the research is how to reduce the difference in feature distribution between the source domain and the target domain [16].

Existing domain adaptation methods can be divided into three categories: difference-based, adversarial learning, and reconstruction methods. Difference-based methods reduce distribution differences and make feature spaces more consistent by minimizing metrics of feature distributions between source and target domains (such as maximum mean difference, MMD) [17] [18]. However, such methods rely on precise metrics and have difficulty capturing features in complex environments. Adversarial methods are widely used in domain adaptation [16] [19]. Through adversarial training of generators and discriminators, they learn the similarities between source and target domains and improve target domain performance. However, it usually requires a lot of computing resources, and the generated features are not necessarily optimal. Reconstruction methods reconstruct source and target domain samples through an encoder-decoder structure to reduce differences [20]. However, they are prone to losing details in high-dimensional data, resulting in unstable results.

Another approach to solving the Sim2Real problem is Domain Randomization. This approach introduces a wide range of randomness in the simulation environment to cover a variety of situations that may be encountered in the real world, thereby enhancing the generalization ability of the model [11] [21]. There are two types of domain randomization: visual randomization and dynamic randomization. The former is mainly used for visual tasks such as target detection [22], while the latter focuses more on changes in dynamic environments. However, the randomness design of domain randomization is more difficult, and its mechanism of action is not easy to explain.

Based on the above methods, this paper designs a Sim2Real platform, which not only supports the two-way conversion of simulation and real data, but also combines reinforcement learning with the CycleGAN algorithm. Unlike traditional domain adaptation methods, our method uses CycleGAN to generate more effective cross-domain training data, thereby enriching the diversity of samples and reducing the feature distribution differences between the source domain and the target domain. On this basis, through the strategy optimization of the reinforcement learning algorithm, the decision-making and migration capabilities of the model are further improved.

III. PROPOSED METHED

In this section, we will introduce how to define the Sim2Real Gap used in this paper and how to deploy the Sim2Real platform to achieve simulation-to-reality migration.

A. Sim2Real Gap

In the field of robot visual navigation, SPL (Success-weighted Path Length) and success rate (Success) are often used as evaluation indicators of navigation performance [23], [24], [25]. SPL comprehensively considers the success rate of the robot in performing navigation tasks and the effectiveness of the path. The value range of SPL is between [0,1]. When SPL is closer to 0, the navigation ability of the robot is worse; when SPL is close to 1, it indicates that the navigation ability of the robot is better [9]. Success reflects the frequency of successful navigation of the robot. In each episode, if the navigation is successful, Success is 1, otherwise is 0.

In this section, we will use a comprehensive normalization method to define the Sim2Real Gap, comprehensively considering Success and SPL, balancing the deviation of a single indicator, make different evaluation indicators comparable, by focusing on both the completion of the task and the efficiency of the path, and thus providing a more complete performance evaluation [1], [26], [27].

To calculate the sim2real gap, we need to compare the SPL difference and Success difference between the simulation environment and the real environment.

The SPL difference between the simulation environment and the real environment is defined as follows:

$$SPL_{\text{gap}} = |\text{SPL}_{\text{sim}} - \text{SPL}_{\text{real}}| \tag{1}$$

The difference between the Success in the simulation environment and the real environment is defined as follows:

$$Success_{\text{gap}} = |\text{Success}_{\text{sim}} - \text{Success}_{\text{real}}| \tag{2}$$

The SPL of different tasks or environments may have large scale differences. Directly using the absolute difference value may lead to biased results. In order to make the results of different experiments or tasks comparable, we need to normalize SPL and Success to eliminate this scale difference and more realistically reflect the performance gap between the agent in simulation and the real environment.

The normalized SPL_{gap} is defined as follows:

$$\text{Normalized\_SPL}_{\text{gap}} = \frac{\text{SPL}_{\text{gap}}}{\text{SPL}_{\text{sim}}} \tag{3}$$

The normalized $Success_{gap}$ is defined as follows:

$$\text{Normalized\_Success}_{\text{gap}} = \frac{\text{Success}_{\text{gap}}}{\text{Success}_{\text{sim}}} \tag{4}$$

Finally, combining the SPL and Success in the simulation environment and the real environment, the comprehensive normalization method is used to define the Sim2Real Gap as follows:

$$\begin{aligned}\text{Normalized\_Sim2Real}_{\text{Gap}} = {} & w_{\text{spl}}\text{Normalized\_SPL}_{\text{gap}} \\ & + w_{\text{success}}\text{Normalized\_Success}_{\text{gap}}\end{aligned} \tag{6}$$

This study focuses on indoor navigation. Since the indoor environment is small and complex, giving SPL a higher weight not only considers whether the robot reaches the target, but also evaluates the degree of path optimization. A more efficient path reduces resource consumption and time waste and improves user experience. Therefore, in the formula, the weights assigned are w_{spl}=0.6 and $w_{success}$=0.4 respectively.

B. Sim2Real Platform

This paper uses Habitat as the simulation platform [28]. Habitat is an open simulation platform designed for the research and development of robot visual navigation and intelligent perception capabilities. Its highly realistic environmental simulation can reflect complex indoor and outdoor scenes, providing researchers with a cost-effective, safe and controllable data collection and experimental platform. Flexible configuration options and rich customization capabilities enable users to design experiments and optimize algorithms according to specific needs [28]. The real environment uses the Gazebo platform. Gazebo not only provides a highly customizable and realistic experimental platform, but also has rich sensor simulation capabilities and powerful physical simulation support. It is an ideal choice for real-world environments [29].

The Sim2Real platform is shown in Fig. 1. The `Gazebo->Habitat` and `Habitat->Gazebo` nodes mainly implement information conversion functions. Specifically, the `Habitat->Gazebo` node is responsible for converting discrete actions in the simulation environment into continuous actions that can be executed in the real environment; the node subscribes to the action topic, and after receiving the discrete action instructions issued by the Habitat agent, it obtains the posture of the current agent. If it is an instruction corresponding to linear motion, the linear velocity in the world coordinate system is calculated and assigned; if it is a turning instruction, the angular velocity is set to make the agent turn left or right; if it is a stop instruction, the stop instruction is issued accordingly. Specifically, the `Habitat->Gazebo` node is responsible for converting discrete actions in the simulation environment into continuous actions that can be executed in the real environment. The `Gazebo->Habitat` node is responsible for converting data in the real environment into data in the simulation environment. The node receives information published by the real environment, packages this information into information that can be read by the Habitat Agent, and publishes it to topics such as rgb, depth, pointgoal_with_gps_compass, etc.

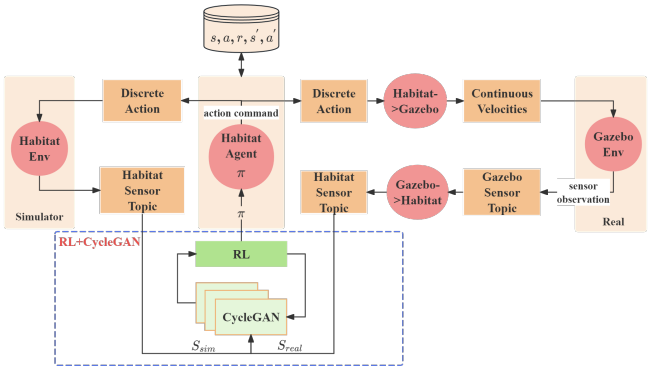


Fig. 1. The framework diagram of the Sim2Real platform.

As a bridge node connecting the simulation environment and the real environment, the Habitat Agent node will continuously load the reinforcement learning policy model. The state information s_{real} from the real environment and the state information s_{sim} from the simulation environment will be input into the RL+CycleGAN module together. The agent selects actions based on these state information, and the environment responds and feeds back new states and rewards, thereby evaluating and adjusting the strategy to maximize the cumulative reward. As learning deepens, the policy model is continuously optimized to guide the agent in the simulation environment to make more reasonable action choices.

IV. Domain Adaptation

Through the Sim2Real migration platform, we can effectively collect real environment data, such as sensor readings, environment layout, and dynamic elements. However, the data of the real environment is limited after all. In this section, we will generate more effective sample data related to the decision-making of the intelligent agent based on the domain

adaptation method, integrating PPO and CycleGAN, effectively improving the performance of the intelligent agent in the simulation and real environment, and narrowing the gap between the simulation and the real environment.

A. PPO

The PPO algorithm is the key reinforcement learning algorithm used for robot autonomous navigation training in this experiment. It ensures the stability and speed of training by optimizing the policy network [30], [31], [32]. A specially designed objective function is used in the PPO algorithm. The form of the objective function is usually defined as follows:

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \quad (5)$$

$$r_t(\theta) = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \quad (6)$$

In this context, π_θ is the new policy, $\pi_{\theta_{\text{old}}}$ is the old policy, $\hat{A}_t$ is the advantage function estimate, and ϵ is a hyperparameter. PPO uses a gradient ascent method to update the policy parameters θ, i.e., $\theta \leftarrow \theta + \alpha\nabla_\theta L(\theta)$, where α is the learning rate.

The key feature of the PPO algorithm lies in restricting the magnitude of policy updates, thereby stabilizing the learning process. During each update, the probability ratio $r_t(\theta)$ is constrained within the range $[1-\epsilon, 1+\epsilon]$, preventing extreme policy updates due to individual data points. Additionally, PPO allows for multiple policy updates using the same data within each iteration.

B. CycleGAN

As an unsupervised image-to-image translation method, CycleGAN is able to perform effective feature translation and alignment between different domains [33]. CycleGAN learns the mapping between two image domains X and Y from unpaired examples $\{x_i\}_{i=1}^N \in X$ and $\{y_i\}_{i=1}^M \in Y$. In this paper, X and Y are the simulation domain and the real domain, respectively. CycleGAN involves learning two generators: $G : X \to Y$ and $F : Y \to X$. The two adversarial discriminators D_X and D_Y distinguish between the simulated environment image $\{x\}$ and the image after mapping from the real environment $\{F(y)\}$, and distinguish between the image after mapping from the simulated image $\{G(x)\}$ and the real environment image $\{y\}$.

In this experiment, adversarial loss is applied to two mapping functions. For the mapping function $G : X \to Y$ and its discriminator D_Y, the objective is expressed as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) = &\mathbb{E}_{y\sim p_{\text{data}}\,(y)}\left[\log D_Y(y)\right] \\ &+ \mathbb{E}_{x\sim p_{\text{daa}}\,(x)}\left[\log\left(1 - D_Y(G(x))\right)\right]\end{aligned} \quad (7)$$

From $X \to Y$, the generator G aims to generate realistic images by minimizing the objective, while the discriminator D_Y tries to maximize the objective, resulting in the update $\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$. The training from $Y \to X$ is similar, using $\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$. CycleGAN further adds a cycle consistency loss to encourage $x \to G(x) \to F(G(x)) \approx x$ and $y \to F(y) \to G(F(y)) \approx y$, as shown below:

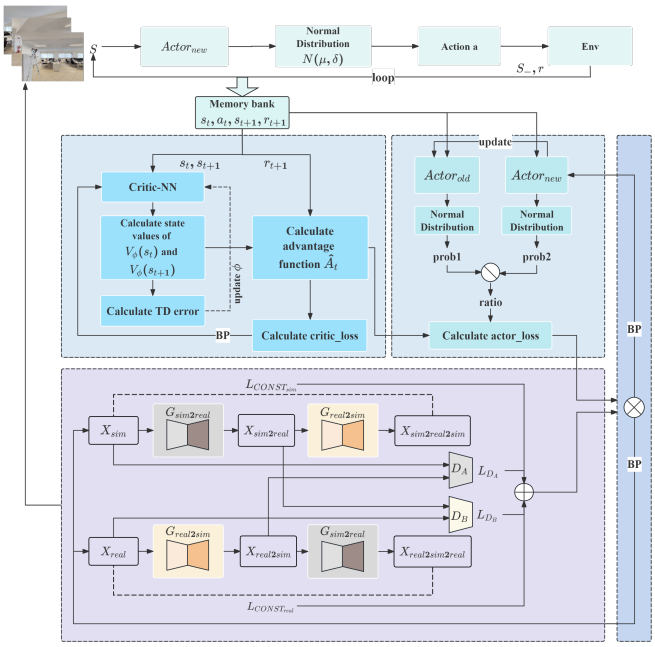


Fig. 2. The training process of integrating RL and CycleGAN.

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x\sim p_{\text{data}}\,(x)}\left[\|F(G(x)) - x\|_1\right] \\ &+ \mathbb{E}_{y\sim p_{\text{data}}\,(y)}\left[\|G(F(y)) - y\|_1\right].\end{aligned} \quad (8)$$

The complete formula is defined as follows:

$$\begin{aligned}\mathcal{L}\left(G, F, D_X, D_Y\right) = &\mathcal{L}_{\text{GAN}}\left(G, D_Y, X, Y\right) \\ &+ \mathcal{L}_{\text{GAN}}\left(F, D_X, Y, X\right) \\ &+ \lambda_{cyc}\mathcal{L}_{\text{cyc}}(G, F)\end{aligned} \quad (9)$$

C. RL+CycleGAN

Combining the PPO with CycleGAN in Sim2Real domain adaptation can bring significant benefits. CycleGAN can align the simulation environment with the real environment, maintain important spatial and positional information in robot navigation, and generate more and effective sample data related to the decision of the agent, so that the PPO algorithm can learn a more robust and generalizable navigation strategy in the simulation environment. The structural diagram of the RL+CycleGAN module proposed in this paper is shown in Fig. 2.

The Actor network of PPO samples states, actions, and rewards from the environment and stores data. The Critic network estimates the state value and updates the policy network. CycleGAN uses generators and discriminators to perform data conversion in simulation and real environments, improves the conversion quality through cycle consistency

and adversarial loss, and makes the generated data closer to the target domain. The Actor and Critic losses of PPO are combined with the generator and discriminator losses of CycleGAN. Through the combination and back propagation of loss functions, the gradients are calculated and the Actor and Critic networks of PPO and the generator and discriminator of CycleGAN are updated, thereby ensuring that the policy optimization of PPO and the data conversion of CycleGAN are carried out simultaneously, while considering both policy optimization and data conversion quality. In this method, we define the comprehensive function as follows:

$$\begin{aligned}\mathcal{L}_{\mathrm{RL-CycleGAN}}(G, F, D_X, D_Y) = \lambda_{PPO}\mathcal{L}(\theta) \\ + \lambda_{GAN}\mathcal{L}_{\mathrm{GAN}}(G, D_Y) \\ + \lambda_{GAN}\mathcal{L}_{\mathrm{GAN}}(F, D_X) \\ + \lambda_{GAN}\lambda_{cyc}\mathcal{L}_{\mathrm{cyc}}(G, F)\end{aligned} \tag{10}$$

Through comprehensive optimization policy and data conversion, the system is able to perform well in both simulation and real environments, achieving a smooth transition and effective application of reinforcement learning algorithms in different environments.

Integrating CycleGAN into the RL training pipeline may incur additional computational overhead, so this paper pre-trains CycleGAN separately before training begins, then integrates it into the RL pipeline and improves data utilization through experience replay.

V. EXPERIMENTS

A. Experimental Setup

- **Task:** In the PointGoal navigation task, the agent needs to reach a target point specified by a relative coordinate. The action space of the agent includes left turn, right turn, forward and stop actions. When the agent performs a stop action within 0.2 meters of the target point, the task is considered successful. We set the upper limit of the number of steps for a single task to 500 steps. If the agent still does not reach the target point after more than 500 steps, or issues a stop command without entering the 0.2 meter range of the target point, it is considered a navigation failure [24].
- **PPO Algorithm Parameter:** The PPO algorithm parameter settings used in this experiment are shown in Table I.

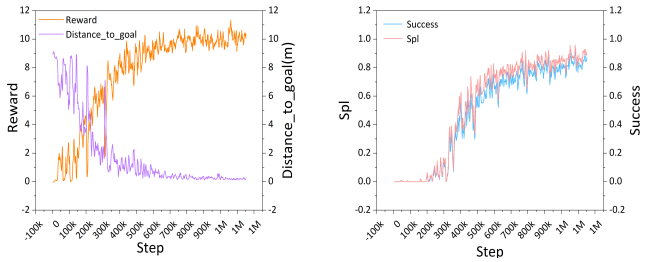
B. Experimental Result

We recorded four key performance metrics during the training process: Reward, SPL, Success, and Distance_to_goal. As shown in Fig. 3, as the number of training updates increases, these metrics all exhibit a gradual convergence trend, indicating a steady improvement in the training effectiveness of the agent.

The model is placed in the simulation environment and the real environment to test 1000 episodes, and each episode is a specific navigation task instance. The test results correspond to the row of RL+CycleGAN in Table II.

TABLE I
EXPERIMENTAL PARAMETER SETTINGS

Parameter	Value
Clip_param	0.2
Entropy_coef	0.01
Lr	$3e-4$
Eps	$1e-5$
Gamma	0.99
Max_grad_norm	0.2
Tau	0.95



(a) (b)

Fig. 3. The changes of various indicators of the agent during navigation training. Figure a shows the changes of reward and distance_to_goal, and Figure b shows the changes of spl and success.

The results of the RL+CycleGAN row in Table II show that the evaluation index value decreases slightly after the robot migrates from the simulated environment to the real environment. This is because the data difference between simulation and reality cannot be completely eliminated. According to the Sim2Real Gap defined in this paper, the Sim2Real Gap generated by the method in this paper is calculated to be 0.2081.

The training results were further verified by the application of the visualization model. The visualization navigation in the simulation environment is shown in Fig. 4. In this figure, the left and middle images are the rgb information and depth information received when the robot stops navigation, and the right is the robot navigation trajectory. The results show that the robot finally successfully reaches the target point, and its motion trajectory is close to the optimal path, which shows that in the simulation environment, the robot can complete the navigation task very efficiently.

In order to verify the performance of the model in a real environment, we also performed a visual analysis of the navigation in the real environment, as shown in Fig. 5. Through the analysis of three different navigation segments, it can be inferred that the robot also exhibits stable and reliable navigation capabilities in the real environment.

The above experimental results show that the robot can complete the navigation task relatively efficiently in both the simulation environment and the real environment. This verifies the effectiveness of the reinforcement learning training method for the intelligent agent by integrating RL and CycleGAN

TABLE II
AVERAGE EXPERIMENTAL RESULTS (1000 EPISODES)

Model	Sim				Real			
	Distance_to_goal	Reward	Success	SPL	Distance_to_goal	Reward	Success	SPL
RL+CycleGAN	**0.2139**	**10.1407**	**0.8909**	**0.8477**	**1.0745**	**8.2304**	**0.7150**	**0.6622**
Ablation Experiment	0.8762	7.4317	0.1762	0.1399	1.7132	5.9647	0.0820	0.0599

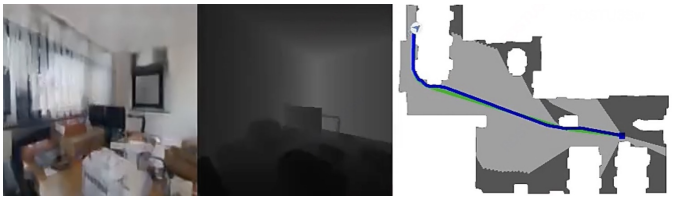


Fig. 4. The effect diagram of the intelligent agent navigating in the simulation environment. The green line represents the optimal route and the blue line represents the actual movement route of the robot.

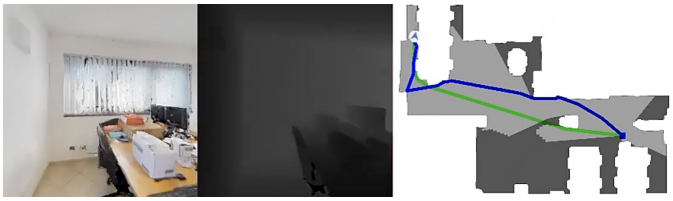
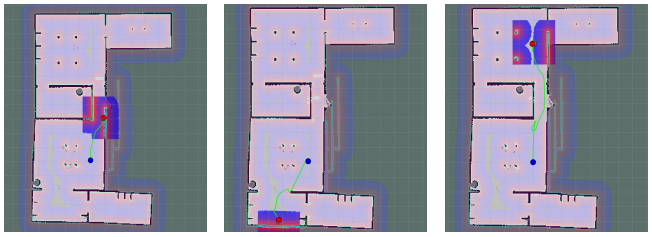


Fig. 6. Under the ablation experiments, the navigation result of the agent in the simulated environment showed that the distance from the target point when the agent stopped moving was 0.38 meters.

algorithms on the Sim2Real platform.



(a) (b) (c)

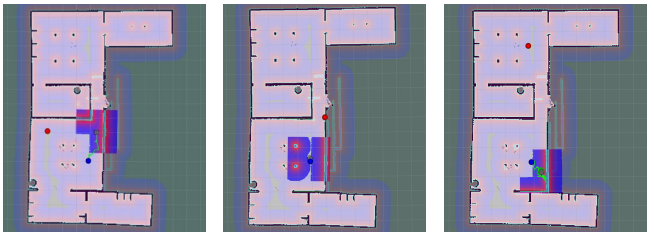
Fig. 5. The navigation effect of the intelligent agent in the real environment. The blue solid ball represents the starting point, the red solid ball represents the target point, and the green line represents the movement trajectory.

C. Ablation Experiment

In order to further verify the effectiveness of the domain adaptation method proposed in this paper, we designed a set of ablation experiments. Specifically, we trained a model using only simulated environment data and then migrated it to the real environment. During the testing phase, all test conditions were kept consistent with the previous experiments to ensure the fairness and comparability of the results. The relevant experimental results are detailed in the Ablation Experiment row in Table II.

The navigation performance of the robot in the simulated environment was unsatisfactory under the ablation experiment. Therefore, it can be inferred that the navigation performance will be even worse when the model is transferred to the real environment. The results in the Ablation Experiment row of Table II also confirm this idea.

In addition, we also visualized the model. The visualization results in the simulation environment and the real environment are shown in Fig. 6 and Fig. 7 respectively. In the simulation environment, when the robot stopped moving, the distance from the target point was 0.38m, and the navigation failed. The actual walking route deviated greatly from the planned optimal route, and the spl was low. In the real environment, we also set three different target points, but the robot failed to successfully reach the target navigation point. This further reveals the limitations of the model in practical applications.



(a) (b) (c)

Fig. 7. Visualization of the navigation result of the agent in the real environment under the ablation experiments.

In this study, we calculated the Sim2Real Gap value for the ablation experiment to be 0.5532. In contrast, the Sim2Real Gap obtained by using the domain adaptation method proposed in this paper was significantly reduced to 0.2081, achieving a relative improvement of 62.38% in performance, effectively narrowing the gap between simulation and real environments, and verifying the effectiveness of this method in promoting the ability of the model to migrate across environments.

VI. CONCLUSION

This paper deploys and builds the Sim2Real platform, and effectively integrates CycleGAN in the reinforcement learning training process to optimize the navigation strategy of the agent. Based on this method, the performance of the agent migrating from simulation to real environment is significantly improved. Compared with the method that only uses simulation environment data for training, the performance of the method

proposed in this paper is improved by 62.38%, effectively reducing the Sim2Real Gap.

Although the performance of this method has been improved, in the future, more attention will be paid to the acquisition of real environment data, and the performance improvement of the CycleGAN method is limited. To this end, we will explore more efficient data collection systems to solve the problem of scarce real data, and improve the model structure to more effectively narrow the gap between simulation and reality.

REFERENCES

[1] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9339–9347.

[2] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9068–9079.

[3] 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.

[4] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[6] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3674–3683.

[7] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, "Sim-to-real transfer with neural-augmented robot simulation," in *Conference on Robot Learning*. PMLR, 2018, pp. 817–828.

[8] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard, "Vr-goggles for robots: Real-to-sim domain adaptation for visual control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1148–1155, 2019.

[9] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. S. Chaplot, O. Maksymets *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *Advances in neural information processing systems*, vol. 34, pp. 251–266, 2021.

[10] 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.

[11] 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.

[12] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2100–2110.

[13] 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.

[14] W. Zhao, J. P. Queralta, L. Qingqing, and T. Westerlund, "Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning," in *2020 5th International conference on robotics and automation engineering (ICRAE)*. IEEE, 2020, pp. 7–12.

[15] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[16] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3722–3731.

[17] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[18] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[19] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016.

[20] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1637–1645.

[21] S. Tang, J. Peterson, and Z. Pardos, "Predictive modelling of student behaviour using granular large-scale action data," *The Handbook of Learning Analytics*, pp. 223–233, 2017.

[22] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 969–977.

[23] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[24] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[25] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2018.

[26] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, and M. Savva, "On evaluation of embodied navigation agents," 2018.

[27] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," 2020.

[28] R. Ye, W. Xu, H. Fu, R. K. Jenamani, V. Nguyen, C. Lu, K. Dimitropoulou, and T. Bhattacharjee, "Rcare world: A human-centric simulation world for caregiving robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 33–40.

[29] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ros and gazebo," in *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2016, pp. 96–101.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[31] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[32] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[33] 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.