Sim-to-Real Transfer of Automatic Extinguishing Strategy for Firefighting Robots

Chenyu Chaoxia, Weiwei Shang, *Senior Member, IEEE*, Junyi Zhou, Zhiwei Yang, and Fei Zhang

***Abstract*—The automatic extinguishing strategy (AES) is the core of the decision-making system for intelligent firefighting robots. Inspired by the fire extinguishing action of firefighters, designing a vision-based end-to-end AES aligns with human intuition. However, the cost of training agents to learn AES in reality is high. Moreover, training agents in simulation face a gap between simulation and reality, the trained agents often fail in the real world. To solve this problem, we propose a novel AES based on sim-to-real transfer for firefighting robots. This method uses JetGAN, an innovative application of generative adversarial networks (GANs), to translate the simulated jet images into the real domain and uses deep reinforcement learning to construct an AES. First, a genetic algorithm is used to find the simulated jet that closely resembles the input jet image in the real domain, thereby constructing a paired sim-real image dataset. Subsequently, we devise a jet consistency loss and employ the focal frequency loss for JetGAN, which is trained on the paired image dataset. Finally, agents are trained in the simulated environment constructed in Unity3D using jet images translated by JetGAN. The learned AES is capable of transferring to the real world. The experimental results on an actual firefighting robot demonstrate the effectiveness of the proposed sim-to-real transfer. The transferred AES achieved the highest success rate compared with other methods.**

***Index Terms*—Transfer learning, reinforcement learning, automatic extinguishing strategy, firefighting robots.**

I. Introduction

In THE past decade, the field of firefighting robots has witnessed substantial advancements, with various types of robots being researched and deployed in practical applications to replace human firefighters in hazardous environments. However, existing firefighting scenarios heavily rely on remote control from humans and lack the autonomous capability and intelligent decision to combat fires. Therefore, the development of an automatic extinguishing strategy (AES) has emerged as pivotal for the practical application of intelligent firefighting robots.

The main task of the AES is to guide the jet trajectory toward the flames. In real-world scenarios, the jet trajectory is influenced by various physical parameters, such as the water pressure, the wind speed, the initial velocity, and the sectional area of the jet. Thus, accurately predicting the jet trajectory remains challenging [1]. Human firefighters generally adjust the water cannon by visual cues, thus visual feedback is often introduced to the AES. McNeil et al. [2] segmented the flame in infrared images and aimed at the base of the flame for firefighting. Subsequent research [3] focused on locating points along the jet trajectory using stereo vision to estimate the optimal angle for a water cannon at each time step. Kim [4] achieved flame positioning and tracking via infrared stereo vision. However, existing methods have limitations in complex scenes because stereo vision is difficult to implement for measuring jets, and it is difficult to calculate the disparity of the jet area through feature matching due to the sparse image features of the jet area.

Stereo vision relies on complex feature design and feature matching, and an end-to-end motion strategy from vision to action alleviates this challenge. Moreover, deep reinforcement learning is an effective approach for handling the complexity of image data in such a strategy. Unlike conventional control methods, deep reinforcement learning trains agents from real-time data, thus providing robots with greater flexibility, adaptability, and intelligence. Various algorithms have been developed based on the three classic frameworks of DQN [5], DDPG [6], and A3C [7]. Notably, PPO [8] has been widely applied in many practical scenarios because of its high training stability. Nevertheless, directly training an agent for firefighting in the real world is essentially impractical. First, interacting with the environment and completing a single episode can consume tons of water and burn liters of fuel, regardless of the time cost. Second, unsafe behavior during training may damage the hardware of the firefighting robot or surrounding components. Thus, training the agent in simulation and then transferring it to the real world is more feasible. To our knowledge, there is currently no sim-to-real transfer method specifically for AES, but there are studies on AES via simulation. Lee et al. [9] built a virtual scene using the Unity3D engine and introduced reinforcement learning into flame aiming tasks within a cabin, but their investigation was confined to simulation and did not use visual information. The gap between simulation and reality is the main challenge for transferring agents trained in simulation to the real world [10]. Disparities in visual effects between simulation and reality can readily cause simulation optimization bias or lead to overfitting [11]. Consequently, strategies learned in simulations frequently fail in the real world. Making a simulation look similar to reality seems to be a simple way to narrow the gap between simulation and reality and can be a sim-to-real transfer method. However, detailed scanning of real scenes, a process that demands significant time, is often required for creating a realistic simulation. Additionally, scanning and capturing the texture of the jet as a moving fluid is challenging in the creation of a

Received 17 April 2024; accepted 25 September 2024. Date of publication 19 November 2024; date of current version 26 November 2024. This article was recommended for publication by Associate Editor D. Park and Editor H. Moon upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grant U22A2056 and Grant 62173316, and in part by the Anhui Science Fund for Distinguished Young Scholars under Grant 2108085J32. *(Corresponding author: Weiwei Shang.)*

The authors are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: chaoxia@mail.ustc.edu.cn; wwshang@ustc.edu.cn; speed@mail.ustc.edu.cn; zhwyang@mail.ustc.edu.cn; zfei@ustc.edu.cn).

Digital Object Identifier 10.1109/LRA.2024.3502059

firefighting simulation. Therefore, a cost-effective sim-to-real transfer method is necessary for narrowing the gap between simulation and reality.

Domain randomization (DR) is an approach for sim-to-real transfer that involves randomly altering parameters in the simulation throughout training. This method grants the agent the ability to extract generalizable and robust features, allowing it to adapt to different scenes, including the real world. Chebotar et al. [12] used real-world experimental data to adjust the distribution of simulation parameters, replacing manual tuning of simulation randomness. Tiboni et al. [13] proposed DROPO, which models parameter uncertainty based on a likelihood-based approach using precollected offline trajectories, thus estimating the domain randomization distribution for sim-to-real transfer. However, DR remains task specific, and its applicability is limited [14]. Domain adaptation (DA) is another approach that aims to bridge the gap between simulation and reality by translating images from one domain to another. Ho et al. [15] proposed RetinaGAN for sim-to-real transfer, which introduced consistency loss of object detection. They ensure consistent prediction of bounding boxes for original and translated images. Lin et al. [16] used GANs to achieve sim-to-real transformation of tactile sensor images, which are applicable to various sensor types, and conducted experiments on a robotic arm for different manipulation tasks. Scheikl [17] applied GANs to deep reinforcement learning for surgical robots, translating images from simulations into real images and demonstrating their effectiveness through tissue retraction experiments. GANs are responsible for image translation in these DA works, learning features from one domain and transferring them to another. The sim-to-real translation of jet images presents a conditional generative adversarial network (cGAN) problem [18], where the generator uses simulated jet images as conditional inputs to generate jet images in the real domain. Ensuring the shape consistency of the jet images before and after translation is crucial for narrowing the gap between simulation and reality, and using paired jet images during training can improve the ability of GANs to learn to generate jets consistent with the shape of the input jet images. However, unpaired image-to-image translation framework such as CycleGAN [19], CUT [20] and DCLGAN [21] would lose this advantage, thus special image translation methods need to be designed. Additionally, obtaining paired sim-to-real jet images is challenging because the physical characteristics of jets in the simulation differ from reality, making it difficult to directly generate simulated jets with consistent shapes based on real jets. Therefore, it is necessary to use a parameter search to find jets that match the shape of real jets in the simulation.

To employ a vision-based end-to-end AES in the real world, we propose a sim-to-real transfer method of AES. First, we design a genetic algorithm to automatically construct a dataset of paired sim-real jet images by searching for jet parameters in the Unity3D scene. Then, JetGAN is proposed to bridge the gap between simulation and reality. Jet consistency loss is defined to ensure the shape consistency of jet images across domains, while focal frequency loss (FFL) is introduced to enhance training stability and mitigate the influence of the image background. Finally, we design a fire suppression agent (FSA) based on deep reinforcement learning to accomplish the sim-to-real transfer of AES. By training on visual data in the simulation, the FSA uses jet images translated by the JetGAN for AES learning. In the experiments, we transfer the trained AES from the simulation to an actual firefighting robot and conduct automatic firefighting experiments, thereby validating the effectiveness of our method. The main contributions of this work can be summarized as follows:

1) We construct a simulated scene for firefighting tasks in Unity3D and develop an automated method for obtaining paired sim-real jet images by leveraging a genetic algorithm, thus constructing the dataset.
2) We propose a method for sim-to-real translation of jet images employing JetGAN, which integrates jet consistency loss and focal frequency loss designed for jet images, ensuring the shape consistency of the original and translated jet images.
3) We design a deep reinforcement learning approach in the Unity3D scene, train the FSA using jet images translated by JetGAN, and achieve the sim-to-real transfer of AES by implementing the AES on an actual firefighting robot in the real world.

II. System Overview

This section provides an overview of the proposed AES framework based on sim-to-real transfer. As illustrated in Fig. 1, the system comprises four components. First, the simulated and real jet images are obtained, where simulated jet images are acquired from the scene built in Unity3D, and real images are collected using an actual firefighting robot. We define the simulated and real jet images obtained through jet region extraction as $\boldsymbol{J}_{\mathrm{s}}$ and $\boldsymbol{J}_{\mathrm{r}}$. Second, a genetic algorithm is employed to search for jet parameters in the simulation in order to identify the simulated jet images that best match the real jet images. Third, the JetGAN is trained following the establishment of the paired sim-real jet dataset, ensuring the shape consistency of the translated and original jet images via jet consistency loss while maintaining training stability through FFL. Finally, the jet images generated by the JetGAN, referred to as $\boldsymbol{J}_{\mathrm{G}}$, serve as inputs for the training of the FSA in the Unity3D scene. The AES learned by the FSA can then be transferred to the real world.

III. Data Acquisition and Sim-to-Real Image Translation

Paired jet images from two domains are required in the sim-to-real translation, thus necessitating the creation of a paired image dataset. The Obi fluid component in Unity3D offers capabilities for jet simulation. However, this component can only simulate fluids to a certain extent, leading to discrepancies in the physical characteristics between simulated and real jets. Namely, simulated jets that match the shape of real jets cannot be directly obtained. Hence, it is essential to adjust the parameters governing the jets in the simulation to align the simulated jets with their real counterparts. We found that parameters such as the position $\boldsymbol{p}_{\mathrm{j}0}$ and orientation $[\theta_p\,\theta_y]$ of the jet emitter, as well as the initial velocity $v_{\mathrm{j}0}$ and radius r_{j} of the jet particles, are controllable and have a significant influence on the shapes of jets. Among these, $\boldsymbol{p}_{\mathrm{j}0}$ represents the distance between the jet emitter and the camera imaging plane, $[\theta_p\,\theta_y]$ represents the pitch and yaw angles of the simulated water cannon, $v_{\mathrm{j}0}$ affects the range of the jet, and r_{j} affects the width of the jet. Manually adjusting these parameters to match simulated jet images with each real jet image requires a considerable amount of time and labor. Therefore, an automated search method for jet parameters becomes imperative.

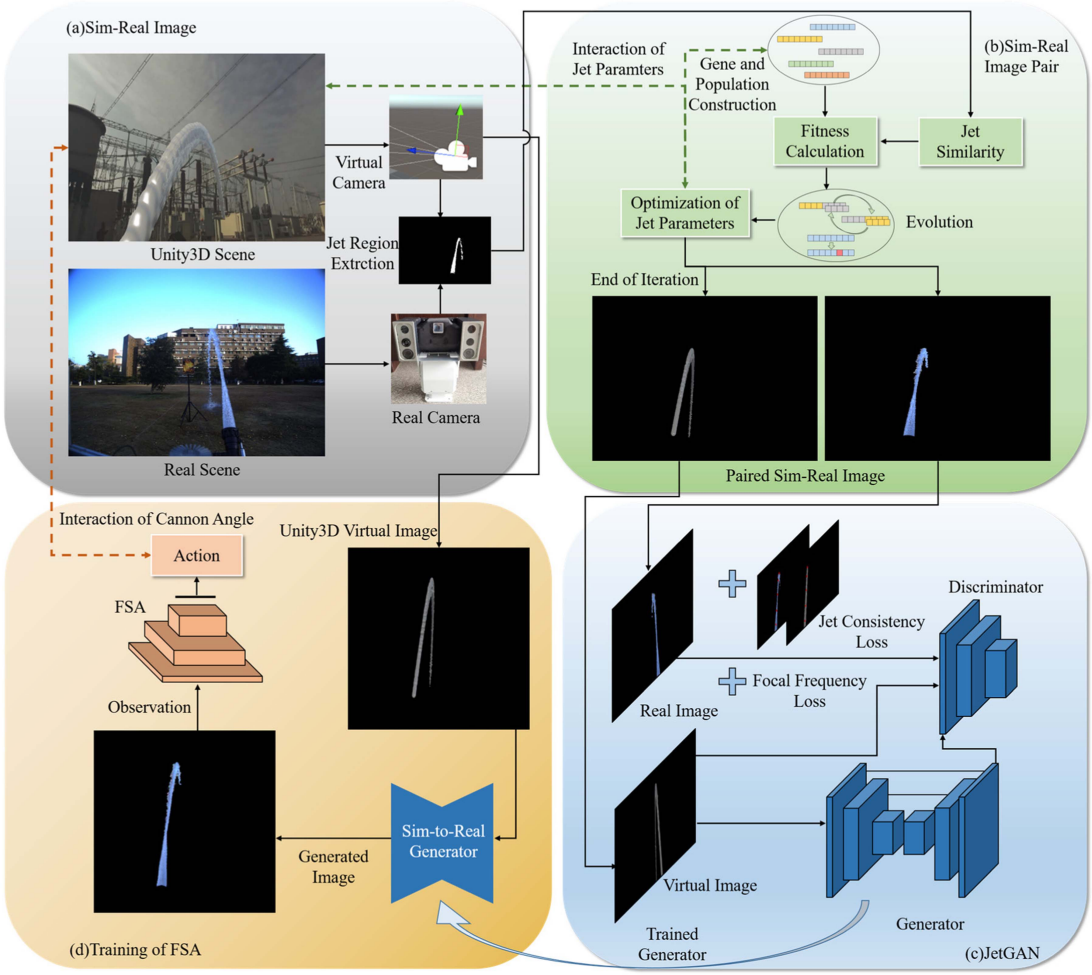

Fig. 1. Overview of the AES. The framework is divided into four parts: (a) Sim-real image acquisition, (b) creation of a paired sim-real image dataset, (c) JetGAN and (d) training of the FSA.

We employ masks to compute the similarity between the shapes of two jet images. In the calculation, we penalize the size of the simulated jet and encourage the matching of pixels between the two jets. The similarity calculation is outlined as follows:

$$\mathrm{S} = \sum\nolimits_{\mathrm{x,y}} (\boldsymbol{M}_{\mathrm{s}} \wedge \boldsymbol{M}_{\mathrm{r}}) - \sum\nolimits_{\mathrm{x,y}} (\boldsymbol{M}_{\mathrm{s}} - (\boldsymbol{M}_{\mathrm{s}} \wedge \boldsymbol{M}_{\mathrm{r}})), \quad (1)$$

where S denotes the similarity between two images, (x, y) denotes the pixel coordinate, and $\boldsymbol{M}_{\mathrm{s}}$ and $\boldsymbol{M}_{\mathrm{r}}$ represent the masks of the simulated and real jet images, respectively. Since the relationships between $\boldsymbol{p}_{\mathrm{j0}}$, $[\theta_p\, \theta_y]$, v_{j0}, and r_{j} and the similarity S are implicit, traditional gradient descent methods are unsuitable for optimizing the parameters. Therefore, a genetic algorithm is employed to search directly in the parameter space. The computational load of the genetic algorithm primarily lies in computing S for each individual in the population. We design a two-step genetic algorithm to improve the efficiency of finding the optimal jet parameters. The most influential parameters on the jet shape are $\boldsymbol{p}_{\mathrm{j0}}$ and $[\theta_p\, \theta_y]$. Therefore, we search for these two parameters in the first step. Moreover, parameters v_{j0} and r_{j} are fixed in this step to reduce the dimensionality of the search space and avoid exponential growth in computational complexity. Once the first step converges, we use the optimal

Fig. 2. Example of paired sim-real jet image. Simulated jet (a) and real jet (b).

individual from the first step to construct a new population, add v_{j0} and r_{j} to the genes, and converge again to determine the final result. The process of the genetic algorithm is shown in Algorithm 1, where the superscript numbers of the jet parameters indicate the step in which they are generated. Additionally, we randomly select a few individuals from the population and choose the one with the best fitness as the winner. This process is repeated until a sufficient number of individuals are selected. We use fuzzy crossover as the crossover strategy, generating offspring genes by linearly blending the genes of parents. The mutation strategy involves adding random values drawn from a Gaussian distribution to the individual genes.

Algorithm 1: Sim-Real Jet Image Matching.

Input: real jet image mask $\boldsymbol{M}_{\mathrm{r}}$, simulated jet image mask $\boldsymbol{M}_{\mathrm{s}}$, jet parameters$\boldsymbol{p}_{\mathrm{j0}}$, $[\theta_p\,\theta_y]$, v_{j0}, r_{j}.
Output: optimal jet parameters
1: Initialize the population with gene $(\boldsymbol{p}_{\mathrm{j0}}[\theta_{\mathrm{p}}\theta_{\mathrm{y}}])$
2: **while not** reach the maximum number of generations
3: Evaluate the fitness of individuals $\mathrm{S}(\boldsymbol{M}_{\mathrm{r}}, \boldsymbol{M}_{\mathrm{s}})$
4: Selection, crossover and mutation operations
5: Update the population
6: **end while**
7: **if** length of gene $= 2$
8: Add $(v_{\mathrm{j0}}\, r_{\mathrm{j}})$ to the genes and initialize a new population based on the current best individual $(\boldsymbol{p}^1_{\boldsymbol{J}0}[\theta_{\mathrm{p}}\theta_{\mathrm{y}}]^1)$
9: **go to step 2**
10: **end if**
Return individual with highest fitness $(\boldsymbol{p}^2_{\boldsymbol{J}0}[\theta_{\mathrm{p}}\theta_{\mathrm{y}}]^2 \mathrm{v}^2_{\mathrm{j0}} \mathrm{r}^2_{\mathrm{j}})$

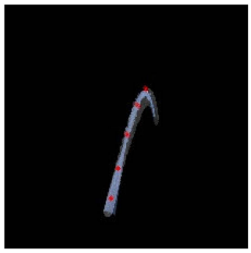

Fig. 3. Visualization of the central points of the matched simulated and real jet images. ($K = 5$).

The jet parameters derived from the genetic algorithm make the shape of the simulated jet tend to be consistent with that of the real jet. The paired sim-real jet images resulting from the genetic algorithm are shown in Fig. 2. Moreover, to improve the efficiency of the similarity calculation, the resolution of the images is downscaled to 224 $\times$ 224. The scaling does not compromise the quality of the training data because the subsequent image translation will maintain the same resolution for the input.

Ensuring the consistency of jet shapes before and after sim-to-real translation is important during FSA training. The FSA extracts position data of the jet and target from the images, and determines the next action based on this information. Therefore, we propose JetGAN to address jet consistency during sim-to-real transfer. JetGAN is built upon the Pix2pix [18] framework because the task involves one-to-one correspondence between input images during training. The goal of JetGAN is to enable generator G to produce a jet $\boldsymbol{J}_{\mathrm{G}}$ based on $\boldsymbol{J}_{\mathrm{s}}$ that closely resembles the texture of $\boldsymbol{J}_{\mathrm{r}}$ while preserving the shape of $\boldsymbol{J}_{\mathrm{s}}$. To achieve this, we propose jet consistency loss $\mathcal{L}_{jet}$ to penalize the shape discrepancy between $\boldsymbol{J}_{\mathrm{s}}$and $\boldsymbol{J}_{\mathrm{G}}$, and $\mathcal{L}_{jet}$ is calculated through the points on the jet trajectory:

$$\mathcal{L}_{\mathrm{jet}}(\mathrm{G}) = \lambda_{\mathrm{jet}} \frac{1}{\mathrm{K}} \sum_{i=1}^{K} \left\| \boldsymbol{c}_s^i - \boldsymbol{c}_G^i \right\|^2, \tag{2}$$

where λ_{jet} is the discount factor, and $\boldsymbol{c}_{\mathrm{s}}$ and $\boldsymbol{c}_{\mathrm{G}}$ comprise K pixels on the trajectory of the simulated jet and the translated jet, respectively, the pixels are uniformly distributed along the jet trajectory. In detail, we first identify the top and bottom points

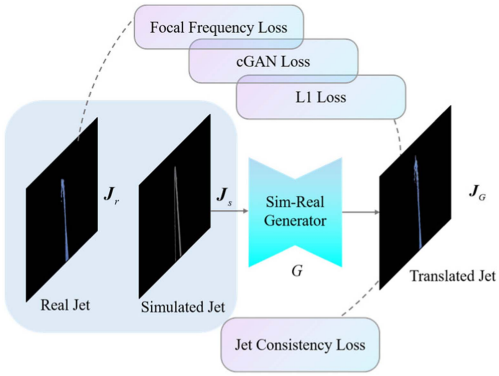

Fig. 4. Diagram of the loss of JetGAN. The additional jet consistency loss and FFL are incorporated into the network training.

of both the simulated and real jets (which indicate the minimum and maximum y-coordinates of the jet) in the jet masks. Then, for each individual jet, we divide the middle part of the jet, between the top and bottom points, into $\mathrm{K}-1$ bins. The average coordinate of the jet pixels within each bin is considered a point on the trajectory. By calculating the average coordinates in all bins and adding the top point of the jet, we obtain $\boldsymbol{c}_{\mathrm{s}}$ and $\boldsymbol{c}_{\mathrm{G}}$. In Fig. 3, we overlay a matched image of the simulated and real jets and use red dots to mark the positions of $\boldsymbol{c}_{\mathrm{s}}$ and $\boldsymbol{c}_{\mathrm{G}}$. The jet consistency loss $\mathcal{L}_{jet}$ can be regarded as aligning the overall shape of the jet. Furthermore, to ensure the similarity of local features between $\boldsymbol{J}_{\mathrm{G}}$ and $\boldsymbol{J}_{\mathrm{s}}$, the L1 distance loss in Pix2pix is adopted:

$$\mathcal{L}_{\mathrm{L1}}(\mathrm{G}) = \left\| \boldsymbol{J}_{\mathrm{r}} - \boldsymbol{J}_{\mathrm{G}} \right\|_1. \tag{3}$$

GANs are inevitably influenced by the background during training, potentially leading to the incorrect usage of some background features in sim-to-real translation, thus affecting the quality of the translated images. According to our observations, training GANs with jet images containing backgrounds results in jet images with inconsistent shapes before and after translation. We use masks to extract the jet regions from both simulated and real images to ensure that the jets translated by JetGAN are not affected by other objects. However, the large black regions in these images make the training unstable, causing GANs to generate predominantly black images. We address this issue in JetGAN by employing analysis in the frequency domain, simultaneously addressing the low-frequency preference [22], [23] of GAN. It is evident that black regions lack texture variation, significantly contributing to the low-frequency components of the image spectrum. Therefore, we introduce FFL to compute the loss of the frequency domain between $\boldsymbol{J}_{\mathrm{G}}$ and $\boldsymbol{J}_{\mathrm{s}}$, specifically penalizing losses stemming from high-frequency components. This directs the attention of the GAN toward image features of higher frequency. The FFL loss is defined as:

$$\mathcal{L}_{\mathrm{FFL}}(\mathrm{G}) = \lambda_{\mathrm{FFL}} \frac{1}{\mathrm{MN}} \sum_{\mathrm{u}=0}^{\mathrm{M}-1} \sum_{\mathrm{v}=0}^{\mathrm{N}-1} w(\mathrm{u},\mathrm{v}) \left| \boldsymbol{F}_{\mathrm{r}}(\mathrm{u},\mathrm{v}) - \boldsymbol{F}_{\mathrm{g}}(\mathrm{u},\mathrm{v}) \right|^2, \tag{4}$$

where $\boldsymbol{F}_{\mathrm{r}}$ and $\boldsymbol{F}_{\mathrm{g}}$ denote the images transformed into the frequency domain, $\boldsymbol{w}(\mathrm{u},\mathrm{v})$ denotes the weights at position(u,v) in the image of the frequency domain:

$$\boldsymbol{w}(\mathrm{u},\mathrm{v}) = \left| \boldsymbol{F}_{\mathrm{r}}(\mathrm{u},\mathrm{v}) - \boldsymbol{F}_{\mathrm{g}}(\mathrm{u},\mathrm{v}) \right|, \tag{5}$$

the weights of the low-frequency components near the origin are relatively lower compared to other parts. λ_{FFL} is a discount factor. The FFL loss serves to balance the foreground and background of the jet images, ensuring the stability of the training of the JetGAN, reducing artifacts and improving the quality of the generated jet images. Finally, we incorporate the conditional GAN loss:

$$\mathcal{L}_{\mathrm{cGAN}}(\mathrm{G},\mathrm{D}) = \log \mathrm{D}(\boldsymbol{J}_{\mathrm{s}}, \boldsymbol{J}_{\mathrm{r}}) + \log(1 - \mathrm{D}(\boldsymbol{J}_{\mathrm{s}}, \boldsymbol{J}_{\mathrm{G}})), \quad (6)$$

where D represents the discriminator. The losses of the JetGAN are shown in Fig. 4. Combining all the aforementioned losses, the complete JetGAN loss can be expressed as follows:

$$\mathcal{L}_{all} = \mathcal{L}_{cGAN} + \mathcal{L}_{L1} + \mathcal{L}_{jet} + \mathcal{L}_{FFL}. \quad (7)$$

IV. Deep Reinforcement Learning and FSA

By leveraging JetGAN to translate simulated jet images into the real domain, we narrow the gap between simulation and reality in the observation space. Subsequently, combining the JetGAN with deep reinforcement learning enables the training of the FSA. The training environment for FSA is a scene built in Unity3D. The jet image $\boldsymbol{J}_{\mathrm{s}}$ captured by the camera in this scene is translated into $\boldsymbol{J}_{\mathrm{G}}$ by the generator of JetGAN, which serves as the observation of the FSA. The FSA controls the jet by manipulating the pitch and yaw angles of the simulated water cannon. We adopt a discrete action space because the water cannon used in our firefighting robot is heavy and has significant resistance to movement, making precise control challenging. At each time step, the agent can choose to rotate the water cannon up, down, left, or right or to remain stationary. If the agent chooses to rotate the water cannon, it will turn a predetermined angle in the chosen direction.

Moreover, proper reward design is crucial for facilitating the effective learning of the AES. Intuitively, a hitting reward r_{hit} is needed, which follows the following rule:

$$r_{hit}(t) = \begin{cases} 1.0, & n_{sum} > T_n \\ 0.1, & n_t > 0 \\ 0, & otherwise \end{cases}, \quad (8)$$

where n_{sum} is the total number of jet particles that hit the flame target, n_t is the number of jet particles hit in the current step, and T_n is the threshold for the number of particles. Exceeding this threshold indicates the completion of the firefighting task. There is also a time penalty $r_{time} = -0.01$. The total return decreases with episode length. However, relying solely on these two rewards leads to the problem of a sparse reward. Before the jet reaches the flame target, the agent only receives penalties from the environment, which is detrimental to the training of the FSA and may lead to a negative policy. To solve this issue, we design a potential-based reward r_p for our scene, and r_p is defined in terms of the difference in distance from the jet to the flame target, expressed as follows:

$$r_p(t) = \lambda_p(d(t-1) - d(t)), \quad (9)$$

where $d(t)$ is the perpendicular distance from the center of the flame target to the center plane of the jet. The rotation angle of the water cannon for each step taken by the agent is fixed; thus, the variation of $|d(t-1) - d(t)|$ in the simulation is minimal. Therefore, we design the value of λ_p to make$|r_p|$ close to 0.1. The original sparse reward function becomes dense due to the presence of r_p. As the jet approaches the flame target, the FSA receives more rewards, facilitating effective exploration in the simulation. In summary, the reward $r(t)$ obtained by the agent at time step t is:

$$r(t) = r_{hit}(t) + r_{time}(t) + r_p(t). \quad (10)$$

We adopt the PPO algorithm for deep reinforcement learning because of its stability and convergence during training. The value of the discount factor γ is set to 0.99, and $\lambda_{GAE} = 0.95$. The ratio clip for PPO is set to 0.2. The network architecture of the agent is illustrated in Fig. 5.

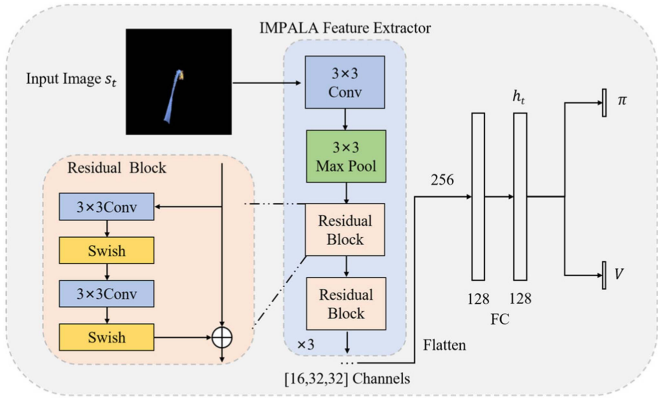

Fig. 5. Network architecture of the FSA consists of an IMPALA feature extractor, two fully connected layers, and two subnetworks: policy π and value V.

(a) (b)

Fig. 6. (a) Firefighting robot platform; (b) vision system.

V. Experiments

A. Experimental Setup

The platform of our firefighting robot and its vision system are shown in Fig. 6. Camera 1 is employed to capture images during the experiment, while the other sensors are not used. The jet is supplied by a fire hydrant, delivering a pressure of approximately 0.3 MPa and a flow rate of approximately 15 L/s. A 20-meter long fire hose connects the fire hydrant to the firefighting robot. The maximum range of the jet is approximately 10 meters when the pitch angle of the water cannon is 45°. The experimental site is located on a campus. Three dummy flame targets are used instead of a real flame for safety reasons. Each target, as shown in Fig. 7, comprises a metal plate printed with a flame image fixed on a retractable tripod. During the experiment, the tripod

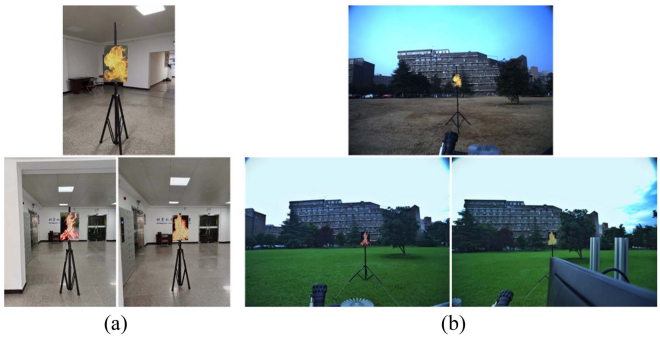

Fig. 7. Flame targets used in the actual experiments. (a) Dummy flame targets. (b) The targets in the camera image.

is anchored to the ground with ropes and stakes to withstand the impact of the jet.

B. Unity3D Scene

The simulated firefighting scene is built using the Unity3D engine. Its main components include the Obi emitter, the jet particle detector, the flame target, and the camera. The FSA consists of two components in the Unity3D scene: a camera and an Obi emitter. The camera serves as the visual sensor for the FSA, while the Obi emitter simulates the water cannon. Upon receiving an action in a specific direction, the emitter rotates by 2.5° toward the designated direction. At the same time, the number of steps required in the simulation does not differ significantly from reality by setting the distance between the emitter and the flame target. The flame target is constructed using a flame image. There is also a jet particle detector located at the flame target in the scene. The jet particle detector functions as a collision trigger, activating when jet particles intersect the collision box. It then counts the particles responsible for the collision, thus providing the number of particles hitting the flame. To ensure that the FSA actually learns the extinguishing task rather than merely guiding the jet to a fixed location, the position of the flame target generated in each episode is randomized. The randomization of the flame target is as follows:

$$p_{fire} = (x_0 + \tau x_m, y_0 + \tau y_m, z_0 + \tau z_m), \tag{11}$$

where τ is a random number selected from $[-1.0, 1.0]$, (x_0, y_0, z_0) is the initial position of the flame target, and (x_m, y_m, z_m) indicates the maximum range over which the flame target can vary. The initial angle of the jet emitter is randomly selected around the flame target in each episode. Additionally, the flame target in the simulated scene uses the same images as the dummy targets, with one of the three flame images randomly selected in each episode. Subsequently, the flame target is overlaid onto the translated jet image from JetGAN, as shown in Fig. 8, enabling the agent to simultaneously observe the jet and the flame target. The occlusion relationship between the generated jet and the flame image is the same as that between the jet and the flame in Unity3D, and this relationship can be easily obtained from simulation.

C. Sim-to-Real Image Translation

To construct the dataset required for JetGAN training, we conducted experiments in the real world and collected 3000 frames of jet images. These images are matched with simulated

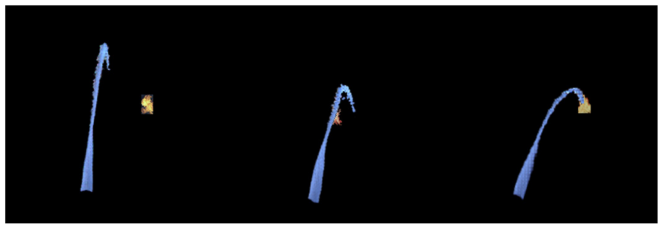

Fig. 8. Translated image of the simulated firefighting scene with the flame target added. (a) No occlusion. (b) Occlusion when missed. (c) Occlusion when hit.

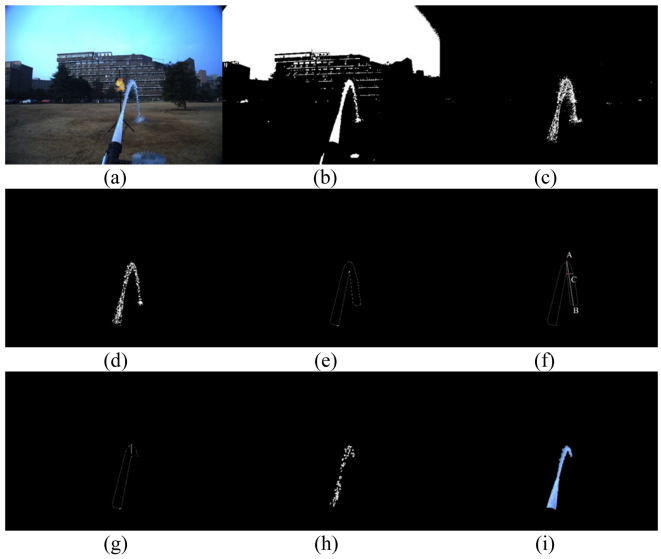

Fig. 9. Segmentation process of the real jet. The segmentation uses the color, motion, and shape characteristics of the jet.

images from Unity3D using our genetic algorithm, resulting in 3000 pairs of sim-real images. All images underwent jet segmentation. For simulated jets, we set the background in the Unity3D scene to black and directly applied binarization. For real jets, we followed the process shown in Fig. 9: after denoising with Gaussian filtering (a), we computed two masks using color thresholding (b) and three-frame differencing (c), and then combined the two masks, followed by applying erosion and dilation to obtain (d). Based on the combined mask, we divided it into left and right parts using the top point of jet, and calculated the convex hulls for each part (e). We call the convex hulls CH, which consists of the left convex hull CHL and the right convex hull CHR. We selected the convex hull on the side with a smaller y-axis span (in Fig. 9, this is CHR), drew a line from the highest point A to the lowest point B, and chose a point C at 25% of the distance from A along this line (f). Then, a horizontal line was drawn through C, and points with y-coordinates smaller than C were removed (g). We then used (d) within the processed CH obtained in (g). First, we removed smaller areas in (d), and the result is shown in (h). Finally, we applied region growing to obtain the complete jet, with the remaining regions in (h) as seeds and the region growing confined within the CH boundaries, resulting in the jet segmentation shown in (i).

We trained four different generators and conducted comparative experiments to validate the effectiveness of the JetGAN in jet image translation, as shown in Fig. 10. The four generators are

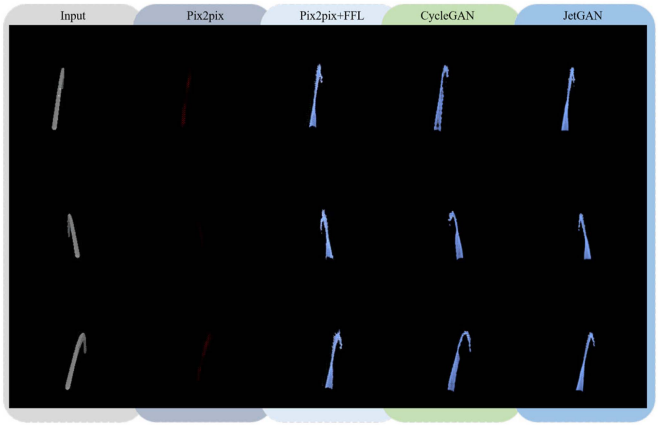

Fig. 10. Sim-to-real jet image translation by different methods. The first column shows the input simulated jet images.

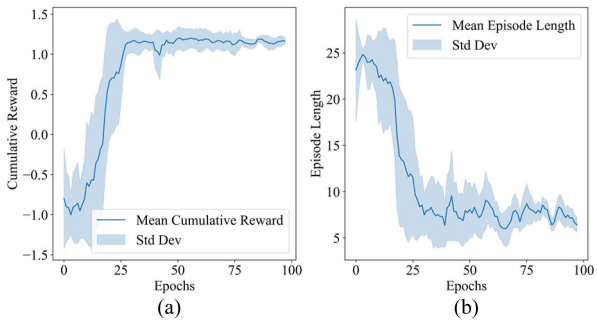

Fig. 11. Learning curves of FSA, including cumulative reward (a) and episode length (b).

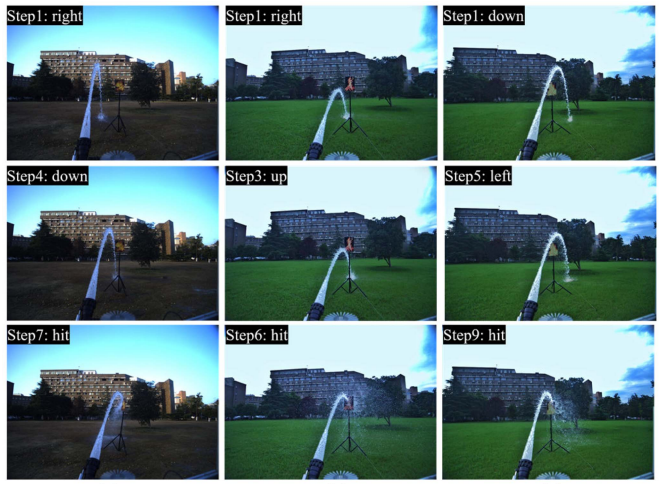

Fig. 12. FSA real-world experiments. The three columns of images, namely (a), (b), and (c), represent different rounds, with the current step number and the direction selected by FSA indicated at the top left corner.

TABLE I
PERFORMANCE OF DIFFERENT FSAS

Agent	Success rates
Sim	1/30
Sim+DR	1/30
Sim+Real	6/30
JetGAN	19/30
JetGAN+DR	24/30

the original Pix2pix method, Pix2pix withFFL loss, CycleGAN, and the complete JetGAN incorporating jet consistency loss. All generators are trained on the same dataset, with training ended upon reaching 100 epochs. In Fig. 10, the leftmost column represents the input images. It can be observed that Pix2pix encounters difficulties during training, generating jets that are barely visible, attributed to the loss caused by the predominance of a black background in the image. Pix2pix+FFL addresses this issue by significantly improving training stability due to the loss of high-frequency image features. Additionally, the jets generated by CycleGAN exhibit some unusual shapes. However, the overlap area between the generated jets and the input jets is higher than that of Pix2pix+FFL. Finally, the shapes of the jets translated by the JetGAN closely match the trajectory of the simulated jet, indicating the effectiveness of the jet consistency loss.

D. Sim-to-Real Transfer of AES

Initially, the FSA learns AES using jet images translated by JetGAN in Unity3D. The return and episode length during training are shown in Fig. 11, ten runs are used to generate the curves, and the colored band represent standard deviation. The FSA requires no further training from real-world data after completing training in simulation. The learned AES can be directly transferred to actual firefighting robots.

We trained multiple FSAs to compare the effectiveness of different methods in the sim-to-real transfer of AES, as shown in Table I. In the Sim method, FSA is trained only in the simulation. Sim+DR uses domain randomization (DR) during training, incorporating random alterations in jet color and applying randomized wind forces to the jet, we also added random erase [24] to the simulated jet images. Sim+Real introduces behavioral cloning, using 30 real trajectories of manual firefighting. Specifically, an additional behavioral cloning loss is incorporated to provide a supervised signal while learning in the simulation. For JetGAN+DR, we added random erase to the jet images translated by JetGAN during training. Since untrained simulated jets can compromise the sim-to-real translation of jet images and further influence the agent performance, physical disturbances were not used in JetGAN+DR. The jet area is always extracted from camera images to eliminate the influence of the background when executing FSAs in the real world. We conducted automatic firefighting experiments on our firefighting robot platform using dummy flame targets, with the targets set approximately 2 m in height and 5 m away from the robot. Each method undergoes thirty tests, using ten preset initial angles for the water cannon for each dummy flame target. The time delta between steps is 2 s, the agent makes decisions based on the current state when the jet is fully stabilized. Additionally, a threshold is set due to the limited availability of firefighting water on site, one round is considered unsuccessful if the FSA fails to hit the target within 15 steps. This threshold balances difficulty with water usage.

From Table I, it can be observed that the Sim approach is impractical, as the FSA hit the flame target only once in all the experiments. Similarly, Sim+DR fails to bridge the gap between simulation and reality, indicating a need for more sophisticated

DR methods to further improve the robustness of the AES. The primary reason for the failure of Sim+Real is the limited availability of human expert experience, and acquiring such experience in large quantities is challenging. The main reason for failure in the experiments of the JetGAN lies in the imperfection of the jet segmentation algorithm, which introduces uncertainty into the observations of the FSA. However, JetGAN+DR enhances the robustness of the FSA, enabling it to effectively cope with the influence of randomness. The average number of steps for JetGAN+DR to complete the task is 9.8. Fig. 12 shows images captured by the camera at different time steps during the experiment. It is evident that the agent has indeed learned the characteristics of the firefighting task, with actions taken at each time step effectively reducing the distance between the jet and the flame target. It is noteworthy that using a higher-powered motor could allow for finer control of the water cannon angle, potentially enabling the agent with a continuous action space to hit smaller targets or perform tasks at greater distances.

VI. Conclusion

This letter proposes an AES for firefighting robots based on sim-to-real transfer, achieving automatic fire extinguishing through GANs and deep reinforcement learning. Initially, a genetic algorithm is employed to produce a paired sim-real jet image dataset, and JetGAN is trained with jet consistency loss and FFL to translate simulated jet images into the real domain. Subsequently, a FSA is trained in the Unity3D scene using the translated images. Finally, the trained FSA is implemented on an actual firefighting robot, and firefighting experiments are conducted in a real environment using dummy flame targets, validating the effectiveness of the AES. In future work, we plan to conduct robustness experiments to test the potential of JEA, and apply AES to real flame targets and conduct experiments in more diverse real-world scenarios.

References

[1] M. Birouk and N. Lekic, "Liquid jet breakup in quiescent atmosphere: A review," *Atomization Sprays*, vol. 19, no. 6, pp. 501–528, 2009.

[2] J. G. McNeil and B. Y. Lattimer, "Autonomous fire suppression system for use in high and low visibility environments by visual servoing," *Fire Technol.*, vol. 52, no. 5, pp. 1343–1368, Feb. 2016.

[3] J. G. McNeil and B. Y. Lattimer, "Robotic fire suppression through autonomous feedback control," *Fire Technol.*, vol. 53, no. 3, pp. 1171–1199, Sep. 2017.

[4] J. H. Kim, J. W. Starr, and B. Y. Lattimer, "Firefighting robot stereo infrared vision and radar sensor fusion for imaging through smoke," *Fire Technol.*, vol. 51, no. 4, pp. 823–845, 2015.

[5] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[7] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

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

[9] E. J. Lee, W. S. Ruy, and J. Seo, "Application of reinforcement learning to fire suppression system of an autonomous ship in irregular waves," *Int. J. Nav. Archit. Ocean Eng.*, vol. 12, pp. 910–917, 2020.

[10] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3444–3459, Jul. 2023.

[11] F. Muratore, M. Gienger, and J. Peters, "Assessing transferability from simulation to reality for reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1172–1183, Apr. 2021.

[12] Y. Chebotar et al., "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *Proc. Int. IEEE Conf. Robot. Automat.*, 2019, pp. 8973–8979.

[13] G. Tiboni, K. Arndt, and V. Kyrki, "DROPO: Sim-to-real transfer with offline domain randomization," *Robot. Auton. Syst.*, vol. 166, 2023, Art. no. 104432.

[14] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.

[15] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 10920–10926.

[16] Y. Lin, J. Lloyd, A. Church, and N. F. Lepora, "Tactile gym 2.0: Sim-to-real deep reinforcement learning for comparing low-cost high-resolution robot touch," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10754–10761, Oct. 2022.

[17] P. M. Scheikl et al., "Sim-to-real transfer for visual reinforcement learning of deformable object manipulation for robot-assisted surgery," *IEEE Robot. Automat. Lett.*, vol. 8, no. 2, pp. 560–567, Feb. 2023.

[18] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2017, pp. 5967–5976.

[19] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.

[20] T. Park, A. A. Efros, R. Zhang, and J. Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 319–345.

[21] J. Han, M. Shoeiby, L. Petersson, and M. A. Armin, "Dual contrastive learning for unsupervised image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2021, pp. 746–755.

[22] L. Jiang, B. Dai, W. Wu, and C. C. Loy, "Focal frequency loss for image reconstruction and synthesis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 13919–13929.

[23] X. Chen, C. Xu, X. Yang, and D. Tao, "Attention-gan for object transfiguration in wild images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 164–180.

[24] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Int.*, 2020, pp. 13001–13008.