Output Security for Multi-user Augmented Reality using Federated Reinforcement Learning

Fengchao Wang*†, Yanwei Liu*¶, Jinxia Liu‡, Antonios Argyriou§, Liming Wang* and Zhen Xu*
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
‡Zhejiang Wanli University, Ningbo, China
§University of Thessaly, Volos, Greece

***Abstract*—With the rapid advancements in Augmented Reality, the number of AR users is gradually increasing and the multi-user AR ecosystem is on the rise. Currently, AR applications usually present results without limitations, which causes great latent danger to users, so it is necessary to apply strategies to ensure the safe output of AR. Due to the environmental diversities among the distributed users, the traditional approaches designed for single-user AR are not efficient for multi-user AR applications. Considering the characteristics of multi-user AR scenarios, we propose a multi-user AR output strategy model based on Federated Reinforcement Learning. With the device-fog-cloud hierarchical architecture, the proposed models are obtained first by Reinforcement Learning on users' devices, and are then hierarchically aggregated on the fog nodes and cloud server. We performed extensive AR simulations in Unity and obtained the results that show our method can avoid several security problems existent in multi-user AR applications.**

***Index Terms*—Multi-user augmented reality, output security, federated reinforcement learning, hierarchical aggregation**

I. Introduction

Augmented Reality (AR) is a promising technology that ingeniously integrates virtual information with the real world, which captures input from the user's surroundings and overlays output on the user's perception of the real world [1]. In recent years, rapid advancements in hardware allowed AR technology to make considerable progress and be used in quite a few areas, such as entertainment, education, health, and transportation. The newest tech trends show that the AR market will surpass $13360 million by 2024 [2]. Moreover, interest in exploring AR technology in diverse industries is increasing, and AR applications are on the way to flourish.

AR applications typically form a user group, and thereby constitute a multi-user AR ecosystem. These applications operate on top of 5G networks with edge intelligence, enabling the frequent data interaction between users with low latency [3]. Similar to multi-user Virtual Reality (VR) [4], there are also many application cases for multi-user AR. For example, the AR game Pokémon Go has a large number of game users who can simultaneously play online games with mutual interactions; AR collaborative office supports multiple users to collaborate and interact in different locations at the same time; Augmented vehicle reality enhances the vehicle perception of the visual information by sharing the visual information collected by multiple vehicles nearby [5].

Simultaneously with the increased use of AR applications, a series of security issues emerge in AR systems. One of the issues is the security of the AR content output. AR systems provide an immersive interactivity that enables virtual objects to coexist with real objects, but current AR applications usually present the results without any limitations. Consequently, malicious outputs created by attackers can easily not only cause click-hijacking [6], sensory overload [7], but also obstruct crucial real-world objects, such as pedestrians, vehicles, and so on, which all pose a great risk to users. Moreover, in a multi-user AR system, the risks users faced are from buggy or malicious apps, as well as other users, such as output conflict between AR contents shared by multiple users. The above-mentioned risks widely exist in all AR systems, especially in the immersive systems such as the AR windshield [8] and a head-mounted display (HMD). In such settings, the output directly affects the information that users can achieve and interact with others, so it is particularly important to ensure the security of AR output.

To solve the output security issues in AR, the computer security and privacy community have already taken steps toward reducing the conflict in AR output [9], [10], but it is not enough for the multi-user AR systems. Moreover, these approaches adopted manual settings in a rule-based output strategy with extremely huge workload, and cannot completely cover all the cases that may be encountered in an AR application. As for the security of multi-user AR, existing research work [11] primarily focused on the multi-user interaction. How to intelligently control the information output in multi-user AR system to ensure user security is still an open issue. Motivated by the naturally distributed architecture as well as the requirement for privacy preservation in user's data sharing in multi-user AR system, in this work we propose a federated reinforcement learning (FRL) framework to solve multi-user AR output strategy problems. The reinforcement learning (RL) model is trained on each AR device to automatically learn an output strategy that seeks for high-reward state-action, and the models from all devices are aggregated hierarchically on the fog node tier and the cloud server tier. In this way, we can take full advantage of FRL to learn the information from the diverse environments in multiple nodes, and establish a more

This work was supported in part by National Natural Science Foundation of China under Grants 61771469
¶Corresponding Author: Yanwei Liu (Email: liuyanwei@iie.ac.cn)

accurate and stable AR output strategy model. In comparison with the existing approaches of manually setting strategies as well as the single-user learning [9], our method is more adaptive, universal and scalable.

To the best of our knowledge, this is the first work that combines reinforcement learning with federated learning to solve the security problem of AR output. Our contributions are summarized as follows:

1) We illustrate the output threats in multi-user interaction and design the corresponding output control strategies, which firstly solve the output security problem of the multi-user AR system.

2) Aiming for multi-user AR applications, we propose a hierarchical federated reinforcement learning framework for generating and aggregating the AR output strategy model for multiple users, coupling federated learning with reinforcement learning in a way that averages the distributed strategy gradients of multiple users' models.

3) To train the AR output control model more accurately with reinforcement learning, we design a set of novel reward functions that give full considerations to the output cases in co-located and non co-located multi-user AR applications.

The remainder of this paper is organized as follows. Section II reviews the related work. In Section III, we analyze the security issues in multi-user AR system. In Section IV, the proposed FRL system is explained in detail. Section V presents experimental results. Finally, Section VI concludes the paper.

II. Related Work

A. AR Output Security

The safety of AR systems has attracted the attention of researchers, and there has been plenty of research work on AR security. Some researchers laid their focus on taking measures to restrict access to AR applications [12], [13], [14], and intended to secure the privacy of users and bystanders. Focusing on AR output security, Roesner et al. [10] proposed a fine-grained output model for management at the AR object level. By transforming the AR objects to operating system primitives, the system can implement object-based constraints, dynamically manage and draw objects. According to this output model, researchers designed an AR platform ARYR, in which a series of output strategies of AR object granularity were preset. Before the AR object has been output, the platform will judge whether the AR object conforms to the preset rules and then adjusts the output. However, manual settings in a rule-based output strategy cannot completely cover all of the output cases in the AR applications, requiring thus extremely huge workload. For intelligently output policy settings, Ahn et al. [9] proposed a method using RL to generate adaptive strategies in AR systems. Using relevant information about the surrounding environment, the fog nodes run simulation training and apply RL algorithms to automatically learn an appropriate output strategy. Nevertheless, only a single user is considered in this method, which only learns limited environmental information owned by a single device.

All of the above methods did not take into account the output security of multi-user AR. For multi-user AR systems, Poretski et al. [15] investigated users' concerns about multi-user AR interactions by a user study, and then Ruth et al. [11] proposed a method to secure multi-user AR content sharing based on HoloLens. In contrast, the output problem of multi-user AR systems has not attracted sufficient attention until now. To cater to the multi-user AR application output scenario, we propose to use FRL to generate output control strategies, that fully exploit the enlarged information-perception space of multiple users, and handles the output case with multi-user interaction.

B. Reinforcement Learning

RL is a branch of machine learning, which is used to solve problems of agents in the process of interacting with the environment through learning strategies to maximize rewards or achieve specific goals [16]. RL systems usually consist of a dynamic environment and one or more agents that interact with the environment. Agents learn in a "trial and error" way, and select action decisions according to the current environmental state. Meanwhile, the environment changes accordingly under the influence of the agent's decision. The agent can obtain rewards and improvements based on its own decision-making and environmental changes. Actions plan to adapt to the environment, so as to maximize the expected reward results. RL is mainly used to learn to generate problem-solving strategies, that is, to generate corresponding actions based on the current state [17]. In more complex scene tasks, it is necessary to use the powerful perceptual ability of deep learning to automatically learn the abstract representation of the environment and use the representation as a basis for RL to generate decisions [18]. Therefore, Deep Reinforcement Learning (DRL) is proposed, which can realize direct control from original input to output through an end-to-end learning method.

C. Federated Learning

Federated Learning (FL) is one of the typical distributed machine learning methods that aim to establish a learning model over distributed data sets, while preventing data leakage [19]. Suppose there are N data owners, and hope to train a machine learning model on the entire data set. Traditional learning methods combine all data for centralized training to obtain the model. However, in FL, participants train local data at the devices, and those updated parameters are uploaded to the parameter servers after training. Parameters are aggregated by parameter servers to obtain the overall model parameters. After multiple iterations, the overall training model is finally obtained. For AR applications, the study in [20] proposed an FL algorithm based on mobile edge computing (MEC) to solve the problems of target detection and classification. To build policies of high quality over the distributed data sets, RL was utilized to be integrated with FL, with paying attention to user's privacy protection. Liu et al. [21] proposed a Lifelong Federated Reinforcement Learning framework under

autonomous navigation settings, using the scalable architecture and knowledge fusion algorithms. Liang et al. [22] proposed a Federated Transfer Reinforcement Learning for real-time knowledge extraction in autonomous driving.

Different from the above approaches, we propose a novel FRL framework suitable for multi-user AR systems to generate the AR output strategy model, in which the particular multi-user characteristics are integrated with the algorithm pipeline of FL and RL.

III. Multi-user AR Output Security Analysis

Besides traditional software security issues, AR systems are vulnerable to attacks due to their particular operation of fusing the outputs of the real world and virtual objects. Especially in the immersive AR systems like HMD and AR-enabled windshield, what users can see is all supplied by AR systems. Once the AR systems have been attacked, the malicious or tampered output can easily reach users undetected. For example, when driving with an AR-enabled windshield, if the AR navigation has been attacked, its outputs can easily obstruct the other vehicles. In addition, attackers can mount clickjacking attacks that overlay transparent output on the normal output of the application, tricking users into unconsciously triggering links to malicious programs [6]. Furthermore, sensory overload, caused by flashing visuals, harsh sound, or intense haptic feedback signals, could cause physiological damage to users [7].

Apart from the threats motioned above, some specific output occlusions will occur inadvertently in multi-user systems, which may not be malicious but has a significant effect on multi-user interaction. One user, who uses an AR application to interact with others, may encounter output problems caused by both its own outputs and the others'. Usually, multi-user AR applications can be divided into two main categories: non co-located and co-located. In non co-located AR applications, the interaction among users is confined to their own user space, that is, the users can only send AR content and display it in their own output interface, without interfering with each other. For example, user A and user B use an AR message application at the same time, and user A sends a message to user B. After receiving the message, user B only needs to display it in his own output interface, without considering the display in user A's interface. In such cases, AR output from each user should be handled properly with only considering the threats coming from himself. While in co-located AR applications, such as community art [11], multiple co-located users share an output space, and the output of each user will be observed by other users in real time and so they affect each other. In such multi-user interaction, a user outputs an AR art on a specific position, followed by which if another user outputs an AR slogan or another AR art, then there will be possibly occlusion between AR outputs. This is because of different field of view (FoV) among users, so that from the perspective of user A, its output does not block that of user B, but it may form occlusion from the perspective of user B. To avoid this problem, the AR output of each user needs to take into account the perspectives of other users interacting with it.

IV. System Design

A. System Overview

In an AR system, as shown in Fig. 1, when people use AR devices interacting with the real world, the sensors first capture the environmental information and send them to the content process module. Then cooperating with the cloud, the content process module completes the information processing and receives the generated holograms. For meeting the low-latency requirement of AR, it is common to deploy fog nodes to assist this process. After that, the holograms would be output to users by the display module. In this work, we intend to add an output policy model between the display and the content process module.

Considering the multi-user AR scenario in the cloud-edge (fog) collaborative computing framework, there are a large number of users that use AR applications online simultaneously in a distributed way. Inspired by this, we design an FRL framework for multi-user AR output control, based on the Proximal Policy Optimization (PPO) algorithm and the federate learning concept. There are three tiers in the proposed architecture:

- **AR device**: Collecting the local environment information, an AR device can train and obtain a new output strategy model by the PPO algorithm. When the gradient information is received, the output model is updated.
- **Fog node**: With a specified aggregation period T_1, the fog nodes collect the gradient information from the online AR devices, and then carry out an average weighting of gradients, followed by returning them to the devices.
- **Cloud server**: Similar to fog node, the gradient information is collected from each fog node with the aggregation period T_2, and then the averaged gradient is obtained and returned to each fog node.

With the device-fog-cloud hierarchical architecture, exploring the training phase on AR devices can minimize the privacy leakage risk, and the fog nodes and cloud can cover large amount of devices and collect more information. Meanwhile, adopting the fog nodes close to devices can reduce the latency and accelerate the training.

B. State and Action Space

In RL of each agent, the environment is usually modeled as a Markov decision process (MDP), which is denoted as a four tuple (S, A, P, R), where

(1) S is a set of finite states, and $s_t \in S$ is the state of the agent at time t;
(2) A is a set of finite actions, and $a_t \in A$ denotes the actions taken by the agent at time t;
(3) $P : S \times A \times S \to [0, 1]$ is the probability distribution function of state transition. $s_{t+1} \sim P(s_t, a_t)$ denotes the probability of agent executing action a_t in state s_t and transferring to the next state s_{t+1};

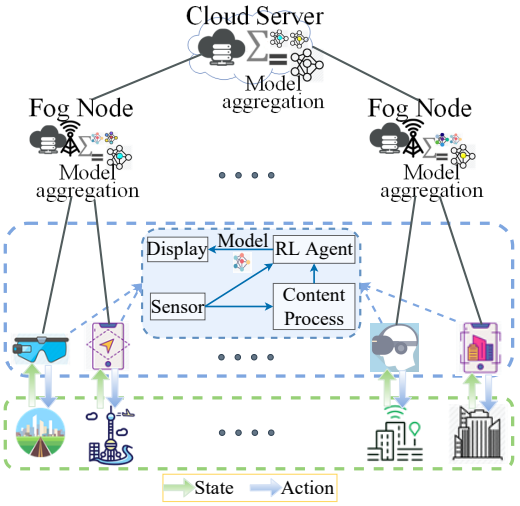

Fig. 1. The system architecture.

(4) $R : S \times A$ is the reward function. $r_t \sim R(s_t, a_t)$ represents the immediate reward value obtained by the agent executing action a_t in state s_t.

In this work, the state space consists of the locations of all holograms and real-world objects, the size of their bounding boxes, and the transparency of the holograms. Besides those, it also contains the AR output FoV. For multi-user AR, the locations of users are also included in the state space to estimate the views of other users. Each location is given by a three-dimensional real vector, and each bounding box is represented by its width and height when projected onto the user's screen. Action space contains the adjustment of the holograms' 3D position, transparency, and bounding box. Agents can take either continuous or discrete action to adjust the output of holograms to maximize the reward value.

In RL for each device, the PPO algorithm [23] is used to learn the policy. Specifically, the objective function is defined as:

$$L^{\text{PG}}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right] \tag{1}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t, and $\widehat{E}[\cdot]$ is the empirical average over a finite batch of samples. In the PPO algorithm, the agent obtains the expectation of samples gathered from an old policy $\pi_{\theta_{\text{old}}}$ under the new policy π_θ by the importance sampling. The objective function of the conservative policy iteration is:

$$L^{\text{CPI}}\left(\theta\right) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[l_t(\theta) \hat{A}_t \right] \tag{2}$$

However, the maximization of L^{CPI} would lead to an excessively large policy update without a constraint. To address this deficiency, the PPO objective function is given by:

$$L^{\text{CLIP}}\left(\theta\right) = \hat{\mathbb{E}}_t \left[\min \left(l_t(\theta) \hat{A}_t, \text{clip} \left(l_t(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{3}$$

where $\epsilon = 0.2$, and $\text{clip}(l_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$ modifies the surrogate objective by clipping the probability ratio, removing the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$. Then the parameters of π_θ are updated with the gradient ∇L^{CLIP} as follows:

$$\theta = \theta - \eta_\theta \nabla L^{\text{CLIP}}\left(\theta\right) \tag{4}$$

where η_θ is the learning rate of optimization.

C. Reward Function

Considering the obstruction, flicker, and click hijacking in AR output, we design the following reward function. Some key notations are summarized in Table I.

TABLE I
PARAMETERS AND NOTATIONS

Notation	Definition
h, H	Original object and the set of holograms
w, W	Object and the set of real-world
h', H'	Policy-modified objects and the set of holograms
α, β, γ	Importance of requirements
τ	Transparency of holograms
μ_h	Ratio of the bounding box
τ_0, μ_0	Thresholds of transparency and bounding box ratio
B_h, B_{ini}	Current and initial bounding box
o, O	Object and the set of User's own holograms
u, U	Object and the set of other users' holograms
m, M	Original objects and the set of shared holograms
m', M'	Policy-modified objects and the set of shared holograms
i, I	User and the set of other users
V, V_i	Transformation matrix from world space to camera space
λ	Trigger of multi-view reward

For the occlusion problem, the 3D position of the AR output is adjusted to avoid the collision between the AR output object and real-world objects. This part of the reward is defined as:

$$R_1 = \sum_{h \in H, w \in W} \left\{ \begin{array}{l} ReLU\left(w_z - h_z\right) \times \\ \left[Dist_{XY}\left(w, h'\right) - \alpha Dist_{XY}\left(h, h'\right) \right] \end{array} \right\} \tag{5}$$

where

$$Dist_{XY}\left(w, h'\right) = |w_x - {h'}_x| + |w_y - {h'}_y| \tag{6}$$

$$Dist_{XY}\left(h, h'\right) = |h_x - {h'}_x| + |h_y - {h'}_y| \tag{7}$$

and ReLU function is defined as:

$$ReLU(x) = \left\{ \begin{array}{ll} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{array} \right. \tag{8}$$

and the XY-plane is the projection plane of AR display. In this plane, by rewarding for increased distance $Dist_{XY}\left(w, h'\right)$, the above reward function ensures that the new policy-modified outputs will not obstruct real-world objects. In addition, by penalizing large $Dist_{XY}\left(h, h'\right)$, the position change constraint is introduced to make the AR output change as minimum as possible to avoid occlusion. Without this constraint, the strategy model may be inclined to adjust the AR output position as far as possible, pursuing a higher reward value. But in a practical application, this adjustment will affect the normal application of AR. Although there is a high reward value, it is not a correct output strategy. Finally, we add a constraint of the Z-plane which indicates the depth information of the hologram. If the hologram is in front of the real object, the output should be adjusted according to the above reward, and vice versa.

In order to place restrictions on the transparent output, we use the reverse ReLU function to limit the AR output whose transparency is lower than τ_0 that is empirically set

to 0.8. When the output transparency is higher than 0.8, the transparency has no contribution to the value of reward function. When the output transparency is below 0.8, with the decrease of transparency, it will have a reverse effect on the value of reward function. This part of reward is defined as:

$$R_2 = -\sum_{h \in H} \beta ReLU\left(\tau_0 - \tau_h\right) \tag{9}$$

Similarly, we limit the abrupt change of hologram size to avoid sensory overload caused by flashing visuals, using the reverse ReLU function. This part of reward is given by:

$$R_3 = -\sum_{h \in H} \gamma ReLU\left(\mu_0 - \mu_h\right) \tag{10}$$

where

$$\mu_h = \left| \frac{B_h - B_{h\_ini}}{B_{h\_ini}} \right| \tag{11}$$

The value of μ_h ranges from 0 to $+\infty$. If μ_h exceeds the μ_0 that it is empirically set to 0.5, the value of reward function will have a negative effect.

For non co-located multi-user AR applications, the occlusion between different users' outputs is a critical issue. To avoid being disrupted by malicious users, we ensure the normal output of user's own holograms as much as possible by adjusting the others' output. Similar to R_1, we define the reward as:

$$R_4 = \sum_{o \in O, u \in U} \left[Dist_{XY}\left(o, u'\right) - \alpha Dist_{XY}\left(u, u'\right)\right] \tag{12}$$

where the $Dist_{XY}\left(o, u\right)$ and $Dist_{XY}\left(u, u'\right)$ are similar to Eqs. (6) and (7). Utilizing this reward function, we hope that the output of other users can be adjusted as small as possible to avoid occlusions.

Apart from several parts of the reward mentioned above, the output of co-located multi-user AR applications should also consider the different perspectives among users. Due to the different FoV, the same real-world objects in the views of the involved users may be different. Hence, in multi-user interaction, when showing something to others, the user should not only ensure the holograms output properly in his own view, but also in the views of the other users. To achieve this goal, the agent needs to collect multi-user information and computes the 2D to 3D and then 3D to 2D transformations in both the camera space and the world space. This part of reward is defined as:

$$\begin{aligned} R_5 &= \lambda \sum_{i \in I} \sum_{m \in M, w \in W} \left\{\mathrm{ReLU}\left(T_{ci}(w_{z'}) - T_{oi}(m_{z'})\right) \times \Delta P_o\right\} \\ \text{with } \Delta P_o &= \Delta P_o(w, m) - \alpha \Delta P_o(m, m') \\ \Delta P_o(w, m) &= Dist_{X_i' Y_i'}\left(T_{ci}(w), T_{oi}(m')\right) \\ \Delta P_o(m, m') &= Dist_{X_i' Y_i'}\left(T_{oi}(m), T_{oi}(m')\right) \end{aligned} \tag{13}$$

where

$$T_{ci}(w) = V_i \times w \tag{14}$$

$$T_{oi}(m) = V^{-1} \times m \times V_i \tag{15}$$

$$T_{oi}(m') = V^{-1} \times m' \times V_i \tag{16}$$

The $X_i'Y_i'$-plane is the projection plane of i-th user's display, and the calculation of $Dist_{X_i'Y_i'}\left(T_{oi}(m), T_{oi}\left(s'\right)\right)$ and $Dist_{X_i'Y_i'}\left(T_{ci}(w), T_{oi}\left(s'\right)\right)$ are similar to Eqs (6) and (7). $T_{ci}(w)$ is the coordinate transformation of real-world object from world space to camera space, and $T_{oi}(m)$ is the coordinate transformation of shared hologram from one camera space to another camera space. Owing to the pertinence of this part of reward function, parameter λ is only triggered when the agent receives multi-user information.

Finally, considering all the above cases, the overall reward function is defined as:

$$R = R_1 + R_2 + R_3 + R_4 + R_5 \tag{17}$$

D. Model Aggregation

Tow-tier Fog-Cloud hierarchical aggregation is used to generate the federated model, where the cloud collects the information of multi-user environments to improve the robustness of the model and the fog nodes share the computation burdens of the cloud to reduce the network transmission delay. The whole process of model aggregation is shown in Algorithm 1. Initialized with the distributed model, online agents, interacting with the environment, train and update the model π_θ using RL (Lines 1-6). Lines 8-11 describe the model aggregation of agents at the fog nodes. Fog nodes collect the gradient of agents $g_{\theta_i}(k)$ and return the averaged gradient $g_\theta^f(k)$ to the agents (Lines 13-15). Similarly, the model aggregation and gradient $g_\theta(k)$ return at cloud is presented in Lines 17-22.

Algorithm 1 FRL training

```
1: Initialized all agents with distributed model parameters
2: for t = 1,2,...,T do
3:    for each agent i = 1,2,...,N in parallel do
4:       Run the RL model π_θ for s_t and do action a_t
5:       Get r_{t+1}, s_{t+1} from the environment
6:       Compute the gradient g_θ and update π_θ
7:    end for
8:    if t|T_1 = 0 then
9:       for each fog-node f = 1,2,...,F in parallel do
10:         Receive the gradient from agent_i, i ∈ C^f
11:         g_θ^f(k) ← average(g_{θ_1}(k), g_{θ_2}(k)...g_{θ_N}(k))
12:      end for
13:      for each agent in parallel do
14:         g_{θ_i}(k) ← g_θ^f(k)
15:      end for
16:   end if
17:   if t|T_1T_2 = 0 then
18:      g_θ(k) ← average(g_θ^1(k), g_θ^2(k)...g_θ^F(k))
19:      for each agent in parallel do
20:         g_{θ_i}(k) ← g_θ(k)
21:      end for
22:   end if
23: end for
```

V. Experiments and Results

A. Experimental Design

In the experiment, we use Unity 3D to simulate AR environment, creating some AR scenarios such as AR-enhanced automotive windshields, AR navigation [8] and AR Community Art [11]. Using the Unity Machine Learning Agents SDK and TensorFlow, we built a FRL network incorporating into our simulations in Python. The communication between Unity and Python is implemented over open socket. During the training, the agent samples data through interaction with the given simulated environment and optimizes its objective function using Adam [24] optimization algorithm. Hyperparameters used in the experiments are shown in Table II.

TABLE II
Hyperparameters used in FRL

Hyperparameters	Value	Hyperparameters	Value
Batch Size	10	Buffer Size	100
Learning Rate	0.0003	Hidden Units	128
Num Layers	3	Max Steps	500000

To evaluate the performance of the proposed method (FRL-based), we carried out extensive experiments, comparing the AR simulation results of our FRL-based control model with those of the state-of-the-art approaches without output control and with the single user RL (RL-based) output control. For the simulation, the single user RL output control model is similar to that proposed in [9] except the reward function. In addition, we also compared the results of the single-view and the multi-view output policy.

B. Performance Evaluation

Following the environment simulation, we trained different output policy models. In Fig. 2, we offer two representative scenes: the left scene is the view of AR windshield, and the right one is the AR navigation interface. Both of the scenes in Fig. 2 are the initial output. In the left scene, the AR advertisement (AD) shields the car on the road, and the AR navigation information shields the trash in the right scene. Using the RL output policy model trained with its own environment, as shown in Fig. 3, the agent outputs AR content avoiding obstructing the real-world object.

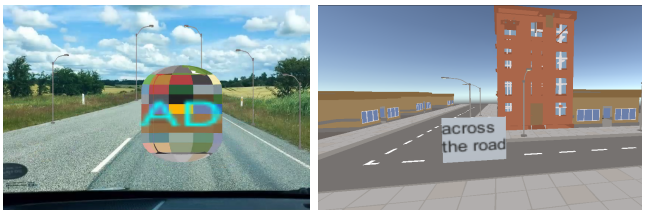

Fig. 2. The AR output results without output policy model.

However, in another scene where the car and trash appear at the same time, using the RL output policy model that is trained with the environment that only has a car cannot be effective. As Fig. 4 shows, in the left picture with RL output policy model, the AR message output shields the trash, which affects normal driving. While in the right one with FRL output policy model, its output can avoid obstructing both the car and

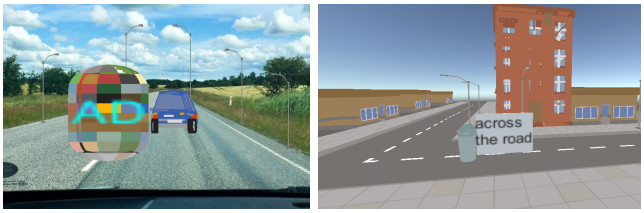

Fig. 3. The AR output results with RL output policy model.

the trash. The results in Fig. 4 obviously show the superiority of the FRL-based model over the RL-based model.

Fig. 4. The same scene with RL-based and FRL-based output policy models respectively.

Fig. 5 shows the loss curves of RL and FRL training. The loss of FRL is a little lower than that of RL on average, and is more stable. This is because FRL utilizes more environmental information to learn the policy, and compared with RL, it is less likely to experience drift affected by a small number of iterations of a single agent. Overall, the losses of FRL and RL both present a downward trend and finally approach an extremely low value.

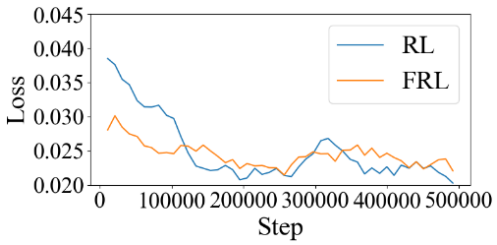

Fig. 5. The loss curves of FRL and RL.

We ran the experiments ten times and averaged the results of cumulative rewards for the proposed FRL-based approach. As shown in Fig. 6, the green area is the range of reward values for multiple experiments, and the blue line is the averaged reward value over the ten iterations of experiments. Overall, the reward is on the rise, and tends to be stable after 200k steps. Although different agents in different environments may have differences on reward components with the proposed reward function, the total reward can also continue to rise and converge to an upper bound.

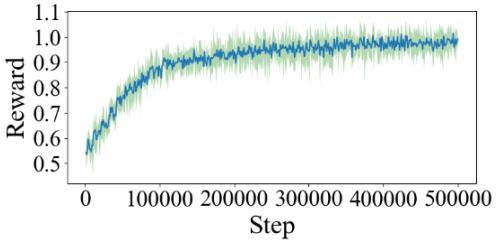

Fig. 6. The reward of FRL training.

Aiming at the output of the co-located multi-user AR application, the proposed method also trained output policy model

that fuses multi-user perspective information. The results are shown in Fig. 7 and Fig. 8. We simulate a scene where two users are in the community art application at the intersection, with one user showing the balloon to the other. In the view of balloon owner (the left in Fig. 7), the balloon does not obstruct the real-world trash, but it do obstruct the trash in the view of the other user (the right in Fig. 7). So the other user may hit the trash when going straight. This example indicates the hidden danger in multi-user AR interaction. As shown in Fig. 7, the single-view output policy model cannot solve this problem for the lack of multi-view information. In the contrast, the multi-view output policy model obtained by the proposed method considered the perspective of all interactive users to output the holograms in a proper location, as illustrated in Fig. 8.

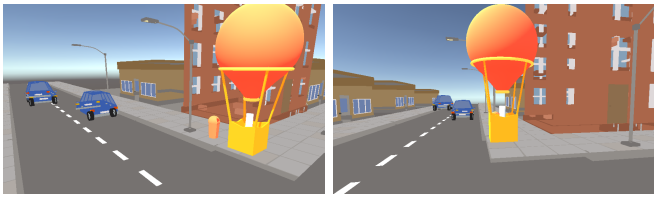

Fig. 7. The AR output of different user-view with single-view output policy model.

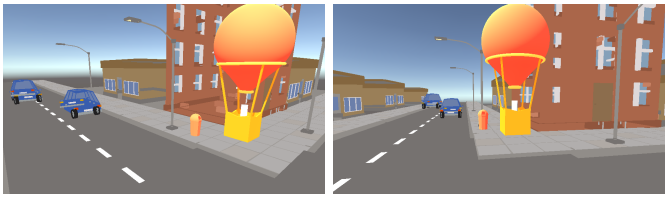

Fig. 8. The AR output of different user-view with multi-view output policy model.

VI. Conclusion

In this work, we proposed a FRL-based method for generating AR output strategy model suitable for multi-user AR applications. Via the hierarchical aggregation of models, the proposed method can effectively learn the information of multi-user AR scenarios and improve the robustness of the adaptive output strategy model. Currently, most of the existing AR security technologies only consider the visual output, while AR systems do not only have the visual output, but also auditory, tactile and other types of non-visual output. Our future research work will consider the security problems caused by non-visual output, as well as the conflict between multiple types of outputs.

References

[1] J. Carmigniani and B. Furht, "Augmented reality: An overview," in *Handbook of Augmented Reality*, 2011.

[2] "Headworn ar revenue forecast, 2019-2024," 2020. [Online]. Available: https://artillry.co/artillry-intelligence/headworn-ar-revenue-forecast-2019-2024/

[3] Y. Li and W. Gao, "Muvr: Supporting multi-user mobile virtual reality with resource constrained edge cloud," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018.

[4] X. Liu, C. Vlachou, F. Qian, C. Wang, and K. Kim, "Firefly: Untethered multi-user vr for commodity mobile devices," in *USENIX ATC 2020*, Boston, MA, USA, 2020.

[5] H. Qiu, F. Ahmad, F. Bai, M. Gruteser, and R. Govindan, "Avr: Augmented vehicular reality," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys2018)*, New York, USA, 2018, pp. 81–95.

[6] F. Roesner, T. Kohno, and D. Molnar, "Security and privacy for augmented reality systems," *ACM Commun.*, vol. 57, no. 4, pp. 88–96, Apr. 2014.

[7] S. Baldassi, T. Kohno, F. Roesner, and M. Tian, "Challenges and new directions in augmented reality, computer security, and neuroscience – part 1: Risks to sensation and perception," *arXiv:1806.10557*, Jun. 2018.

[8] R. Haeuslschmid, B. Pfleging, and F. Alt, "A design space to support the development of windshield applications for the car," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2016, pp. 5076–5091.

[9] S. Ahn, M. Gorlatova, P. Naghizadeh, M. Chiang, and P. Mittal, "Adaptive fog-based output security for augmented reality," in *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network*, New York, NY, USA, 2018, pp. 1–6.

[10] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner, "Securing augmented reality output," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 320–337.

[11] K. Ruth, T. Kohno, and F. Roesner, "Secure multi-user content sharing for augmented reality applications," in *USENIX Security Symposium*, 2019.

[12] S. Jana, A. Narayanan, and V. Shmatikov, "A scanner darkly: Protecting user privacy from perceptual applications," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 349–363.

[13] S. Jana, D. Molnar, A. Moshchuk, A. Dunn, B. Livshits, H. J. Wang, and E. Ofek, "Enabling fine-grained permissions for augmented reality applications with recognizers." USA: USENIX Association, 2013, pp. 415–430.

[14] J. Jensen, J. Hu, A. Rahmati, and R. LiKamWa, "Protecting visual information in augmented reality from malicious application developers," New York, NY, USA, 2019, pp. 23–28.

[15] L. Poretski, J. Lanir, and O. Arazy, "Normative tensions in shared augmented reality," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, Nov. 2018.

[16] R. S. Sutton and A. G. Barto, in *Reinforcement Learning: An Introduction(2 ed.)*. Cambridge, USA: MIT Press, 2017.

[17] H. Wang, N. Liu, Y. Zhang, D. Feng, F. Huang, D. Li, and Y. Zhang, "Deep reinforcement learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 21, 2020.

[18] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Fort Lauderdale, FL, USA: PMLR, 20-22 Apr. 2017, pp. 1273–1282.

[20] D. Chen, L. Xie, B. Kim, L. Wang, C. Hong, L. Wan, and Z. Han, "Federated learning based mobile edge computing for augmented reality applications," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 767–773.

[21] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: A learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[22] X. Liang, Y. Liu, T. Chen, M. Liu, and Q. Yang, "Federated transfer reinforcement learning for autonomous driving," *arXiv:1910.06001*, Oct. 2019.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, Jul. 2017.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.