Adaptive Fog-Based Output Security for Augmented Reality

Surin Ahn
Department of Electrical Engineering
Princeton University
Princeton, NJ
slahn@princeton.edu

Maria Gorlatova
Department of Electrical Engineering
Princeton University
Princeton, NJ
mariaag@princeton.edu

Parinaz Naghizadeh
Department of Electrical and Computer Engineering
Purdue University
West Lafayette, IN
parinaz@purdue.edu

Mung Chiang
Department of Electrical and Computer Engineering
Purdue University
West Lafayette, IN
chiang@purdue.edu

Prateek Mittal
Department of Electrical Engineering
Princeton University
Princeton, NJ
pmittal@princeton.edu

ABSTRACT

Augmented reality (AR) technologies are rapidly being adopted across multiple sectors, but little work has been done to ensure the security of such systems against potentially harmful or distracting visual output produced by malicious or bug-ridden applications. Past research has proposed to incorporate manually specified policies into AR devices to constrain their visual output. However, these policies can be cumbersome to specify and implement, and may not generalize well to complex and unpredictable environmental conditions. We propose a method for generating adaptive policies to secure visual output in AR systems using deep reinforcement learning. This approach utilizes a local fog computing node, which runs training simulations to automatically learn an appropriate policy for filtering potentially malicious or distracting content produced by an application. Through empirical evaluations, we show that these policies are able to intelligently displace AR content to reduce obstruction of real-world objects, while maintaining a favorable user experience.

CCS CONCEPTS

• **Security and privacy** → **Systems security**; • **Human-centered computing** → **Mixed / augmented reality**; • **Networks** → *Middle boxes / network appliances*; • **Computing methodologies** → *Reinforcement learning*;

KEYWORDS

Augmented reality, visual output security, reinforcement learning, policy optimization, fog computing, edge computing.

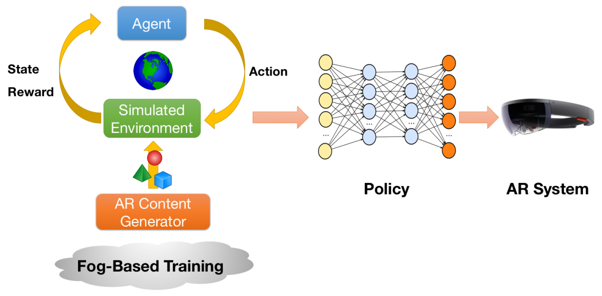


Figure 1: Pipeline for generating and deploying an output security policy for AR systems using deep reinforcement learning. Through simulations, an agent learns an optimal policy, represented as a neural network that maps observed states of the environment to actions, which can then be deployed in the display module of an AR device. At runtime, only the feedforward function of the neural network is used.

ACM Reference Format:
Surin Ahn, Maria Gorlatova, Parinaz Naghizadeh, Mung Chiang, and Prateek Mittal. 2018. Adaptive Fog-Based Output Security for Augmented Reality. In *VR/AR Network'18: ACM SIGCOMM 2018 Morning Workshop on Virtual Reality and Augmented Reality Network , August 24, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3229625.3229626

1 INTRODUCTION

Augmented reality (AR) has the potential to disrupt sectors as diverse as health-care, retail, education, industrial design, and the military [13]. Fully immersive AR devices are becoming more commercially available, with the recent introduction of AR-enhanced automotive windshields [8] and powerful head-mounted displays (HMDs) such as Microsoft's HoloLens [9] and Apple's recently revealed AR system [3]. Even smartphones can now be easily transformed into HMDs using low-cost peripherals like HoloKit [4].

Although AR has the potential to be vastly beneficial to society, the increased immersion and the blurring of lines between the virtual world and reality give rise to a number of security concerns. Malicious or bug-ridden applications could produce harmful output, such as virtual objects that obstruct crucial real-world information from the user. For example, an adversary could intentionally block important road signs from a driver's view, or produce output to distract the driver [6]. Sensory overload, caused by flashing visuals, shrill audio, or intense haptic feedback signals, could cause physiological damage to the user. The space of possible threats becomes amplified when networked AR devices can share mixed-reality experiences with each other or retrieve content from the cloud [7, 14]. Currently, commercial AR systems lack defenses to combat such adversarial output.

The field of AR security and privacy is still in its infancy, and current research has largely focused on *input privacy* (restricting the amount of sensor data accessible to third parties, e.g., [5, 12]), but less on *output security*. Lebeck et al. recently introduced an OS-level framework called Arya that constrains the visual output of AR applications based on context-specific policies that must be explicitly specified in a conditional fashion ("if AR objects violate condition x, then perform action y on the objects") using a set of policy primitives [6]. For example, in the case of AR-enhanced automotive windshields, virtual objects that obstruct pedestrians should be made transparent. Beyond Arya, there is a notable dearth of solutions for securing AR visual output.

Arya has laid the foundation for the field of AR visual output security and is a promising first step toward secure AR systems. We argue, however, that it is an intractable task to specify a set of rule-based policies for every conceivable AR scenario; the space of possible AR content, application behavior, and target environments is simply too large and diverse to tackle in a completely manual fashion. For example, consider an object displacement task in which we seek to move virtual objects to less obstructive positions in the environment. This is a nontrivial objective, as it is unclear how one might move the objects such that they simultaneously don't interfere with each other and don't obstruct real-world objects, which themselves may be moving (e.g., pedestrians or other vehicles). There is clearly a pressing need for intelligent solutions that enable immersive yet safe AR experiences.

In this paper, we build upon Arya by introducing a novel method that jointly leverages *reinforcement learning* (RL) and *fog computing* to automatically synthesize security policies for AR. Recent advances in deep RL [10, 15, 16] have enabled systems to learn how to perform complex tasks in a highly automated fashion. Furthermore, the most recent version of the HoloLens's custom multiprocessor, the Holographic Processing Unit, contains a dedicated artificial intelligence co-processor for flexibly implementing deep neural networks [11]. We design a "smart middlebox" system that employs state-of-the-art RL algorithms to filter potentially malicious or distracting elements from AR content before they reach the user's display. The filtering actions on virtual objects are learned offline through rigorous simulation-based training, whereby the RL agent interacts with an environment consisting of simulated, randomized real-world objects and holograms. The resulting neural network model can then be deployed on a physical AR system (see Figure 1). In practice, the "important real-world objects" can be extracted from spatial mappings of the environment using *object detectors* and *recognizers*, which are already standard components of AR systems. Our approach augments the security capabilities of existing Arya-like frameworks by producing complex policies that adapt to dynamic environments and that would otherwise be practically infeasible to implement by hand.

Simultaneously, we tackle the security, privacy, and latency issues that typically accompany cloud-based training of deep neural networks by instead performing training on a nearby fog computing node [1]. By virtue of its proximity to the end devices, the fog node can more easily incorporate contextual information about the environment [2] to produce more realistic training simulations. This work forms the basis for our broader vision of pushing intelligence toward the network edge to benefit Internet of Things (IoT) devices in a multitude of ways.

Our contributions. In this paper, we make the following specific contributions:

- We present an approach for automatically generating policies to secure AR visual output using deep RL, with simulation-based training performed on a local fog computing node. To the best of our knowledge, we are the first to explore this intersection of RL and fog computing in the context of AR visual output security.
- We define a novel obstruction metric that quantifies how well a given policy reduces the amount of malicious or distracting content displayed to the user.
- We implement a system prototype that demonstrates the ability of an RL-trained agent to learn complex object-displacement policies through trial-and-error in randomized simulations.
- We perform both a qualitative evaluation of the resulting RL-generated policies, as well as a quantitative analysis of policy convergence, frame rate, and obstruction. Finally, we present the results of a user study.

We begin by describing the threat model in Section 2, and provide an overview of policy optimization methods of RL in Section 3. In Section 4, we paint a detailed picture of our system design principles, which includes a justification for our use of RL, the agent's state space, action space, and reward function, and the definition of our obstruction metric. Next, we describe our experimental setup and empirical results in Section 5. Finally, we offer concluding remarks and plans for future work in Section 6.

2 THREAT MODEL

We consider the space of possible threats to output security associated with fully immersive AR systems such as HMDs and automotive windshields, which are potentially connected to the Internet or to each other. In particular, similar to [6], we seek to defend against AR applications that use virtual content to attempt a subset of the following:

- Obscuring real-world context.
- "Attacking" another AR application by obscuring or preventing the user from interacting with its content.
- Distracting or disrupting the user.

Note that the above threats can be posed by a general set of potential attackers, including an adversarial or bug-ridden application, another AR user that is connected to the current user over a network, or a cloud-based entity that controls the AR content sent to the device. In this work, we do not consider ways to prevent the fog node itself from being compromised; it is assumed that standard network security practices are employed for this purpose.

3 POLICY OPTIMIZATION

Reinforcement learning is the problem of learning to make optimal decisions through repeated interactions with an unknown, dynamic environment [17]. Formally, the environment is modeled as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \pi, R)$. An agent interacting with the environment observes its current state $s \in \mathcal{S}$ and takes an action $a \in \mathcal{A}$. The agent then receives a reward $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the state of the environment evolves to state s' according to the transition probability $\pi(s'|s, a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$.

The agent's decision making can be characterized by a policy π_θ, determining the action to take at any state. Here, θ is a vector that parameterizes the policy (e.g. the weights of a neural network for mapping states to actions). Such a policy can either be a deterministic function of the state s, so that $\pi_\theta(s) : \mathcal{S} \to \mathcal{A}$, or a stochastic function $\pi_\theta(a \mid s) : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which gives the probability of taking an action a, given that the current state is s.

The goal of the agent is to find an optimal policy π_θ that maximizes its utility, given by the expected cumulative reward under that policy. In policy optimization methods, the agent learns an optimal policy from the parameterized family of policies π_θ, by optimizing over all possible parameter vectors θ. This yields the optimization problem

$$\theta^* = \arg\max_\theta U(\theta) = \arg\max_\theta \mathbb{E}\Big[\sum_{t=1}^{H} R(s_t, a_t) \mid \pi_\theta\Big] \qquad (1)$$

where H denotes the time horizon, or the number of steps into the future that the agent is considering[1]. To solve (1), one can compute an estimator of the policy gradient $\nabla_\theta U(\theta)$, which can then be used to perform stochastic gradient ascent to update the policy parameters and maximize the expected accumulated reward. This approach leads to a class of reinforcement learning algorithms known as *policy gradient methods* [10, 15]. A recently developed suite of gradient methods known as Proximal Policy Optimization (PPO) algorithms [16] has been shown to be easier to implement, more general, and more data-efficient than previous policy gradient approaches. We use PPO as the learning algorithm in our experiments.

4 SYSTEM DESIGN

4.1 Why Reinforcement Learning?

Here, we outline the benefits of using reinforcement learning for AR visual output security, as opposed to other potential techniques. First, while rule-based heuristics are often easy to interpret and implement, they can be ill-suited to complex, dynamic environments. For this reason, RL techniques have been attracting attention across different research communities, including the field of networking [20]. A particular benefit of RL over heuristics is the ability to adjust policies simply by changing the reward function. A second potential technique is to use dynamic programming. However, while dynamic programming requires a complete probability distribution of possible state transitions – which may be infeasible to obtain – sample transitions suffice for training RL policies.

In contrast to supervised learning, RL does not require a large amount of labeled training data to learn the desired behavior. Rather, the agent uses trial-and-error to automatically learn a policy that strives for high-reward state-action pairs and avoids high-penalty ones. This is particularly useful in environments that change frequently: instead of having to generate an entirely new training set for each possible scenario, we need only provide the agent with environmental details (which can be automated with sensors) and an appropriate reward function. Another distinguishing feature of RL is its closed-loop nature: an RL model captures the potential consequences of an agent's actions over extended time horizons. This is useful for AR visual security since the placement of a hologram could potentially interfere with other holograms or moving real objects in the future.

[1]Note that infinite horizons are also possible, in which case the agent seeks to maximize its discounted cumulative reward $\sum_{t=1}^{\infty} \delta^t R(s_t, a_t)$.

4.2 The Role of Fog Computing

A world of ubiquitous AR will require a network infrastructure that supports low-latency, personalized computing. Our vision is that fog servers would provide local computing resources to different geographical regions. A fog node, which would already possess relevant spatial information about its surrounding environment, could run training simulations offline to generate context-specific security policies, potentially with the aid of cloud servers. An AR device that enters a new region could then connect to the local fog node and download a policy, and also help to improve the policy through online learning. Security agreements would likely need to be established between AR system developers, application developers, and fog server administrators. The potential complexities of such relations are beyond the scope of this paper.

4.3 State Space

In this paper, we assume a fully observable MDP in which the agent has complete knowledge of the state $s_t \in \mathcal{S}$ to make an informed decision about its next action. We define s_t to consist of the locations of all holograms and real-world objects, and the size of their bounding boxes, at time t. 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. Thus, if there are N virtual objects and M real-world objects in the environment, then the state space is a subset of $\mathbb{R}^{5(M+N)}$.

In reality, the sensors used to acquire the state of the environment are likely to suffer from noise, in which case the more appropriate RL model is a *partially observable MDP* (POMDP). Our AR simulator, described in Section 5, is equipped to simulate noisy observations. We leave POMDPs to be explored in future work.

4.4 Action Space

Each of the agent's actions can be encoded as either an integer (for discrete actions) or a real number (for continuous actions). We endow the agent with the ability to change the location (within the physical bounds of the simulated environment) and transparency of each virtual object. Therefore, if there are N virtual objects in the environment, the agent's action space has size $4N$. We choose to manually implement the transparency action as a simple thresholding rule (if an object is within x distance units of the user, or within y distance units of a real-world object, make it transparent). Thus, the agent's action space has size $3N$, and the additional transparency action is implicit. In this sense, our approach yields a hybrid between adaptive and manually specified policies.

4.5 Reward Function

The reward quantifies how well the agent is accomplishing its task of reducing the distraction or obstruction levels of AR content presented to the user, and is computed as a function of the state of the environment and the agent's actions taken in the previous time step. Arguably, the reward function is the most important factor in determining the policy. The choice is context-specific and is guided by the user's or system designer's goals. In our experiments, we consider one possible reward function, which we define below. However, many different potential reward functions exist, each likely to result in different agent behaviors.

Let $f : C \rightarrow C'$ be the bijective function that maps original holograms to their policy-modified counterparts, where C is the set of original holograms, and C' is the set of policy-modified holograms. Also, let $\mathcal{R}$ denote the set of real-world objects. We define the reward at time t to be

$$R_t = \sum_{\substack{O' \in C': \\ R \in \mathcal{R}}} \Big(\alpha \cdot Dist_{XY}(R, O') - \beta \cdot Diff_Z(R, O')\Big) - \gamma \sum_{\substack{O \in C, O' \in C': \\ f(O) = O'}} Dist(O, O') \tag{2}$$

where

$$Dist_{XY}(R, O') = |R_x - O'_x| + |R_y - O'_y| \tag{3}$$

$$Diff_Z(R, O') = R_z - O'_z \tag{4}$$

$$Dist(O, O') = ||\text{Pos}(O) - \text{Pos}(O')||_2 \tag{5}$$

By rewarding for increased XY-plane distance $Dist_{XY}(R, O')$, the above choice of reward function ensures that the new policy-modified objects O' will not obstruct real-world objects. Additionally, by penalizing large Z-plane differences $Diff_Z(R, O')$, we discourage the agent from placing the O' directly in front of real-world objects. Finally, by penalizing large $Dist(O, O')$, the reward function encourages the O' to be close to the original holograms they represent. The weights α, β, γ represent the importance of each of the above requirements. We set $\alpha = 2.0, \beta = 1.0$, and $\gamma = 1.5$ in our experiments.

Alternatively, we can define a more goal-oriented reward function that penalizes the agent whenever a hologram intersects either another holograms or a real object, from the user's perspective. To encourage utility preservation, the agent can also be rewarded for keeping the holograms visible to the user. We plan to compare different reward functions in future work.

4.6 Obstruction Metric

In previous work, the "goodness" of proposed policies was evaluated on solely a qualitative basis. This motivates the need for quantitative metrics that capture the degree to which new policies present improvements to visual output security. To this end, we introduce the *obstruction* metric, which is the percentage of the user's display obstructed by holograms in a single video frame. For a given set of holograms C in the environment, where each object $O \in C$ is a set of pixels, the obstruction induced by C is defined as

$$Obs(C) = \frac{\text{Area of Display Obstructed by } \left(\bigcup_{O \in C} O\right)}{\text{Total Area of Display}}. \tag{6}$$

This metric benefits from a clear operational interpretation: a value of, e.g., 0.4, indicates that 40% of the user's display is obstructed by holograms under the given policy. Note that the contribution of each object to $Obs(C)$ should be weighted by its transparency, since transparent objects do not obstruct the display. Moreover, when objects intersect spatially, the maximally opaque object should dominate over the others. Finally, objects that are themselves obstructed by virtual or real-world objects (and are therefore not visible) should not contribute to $Obs(C)$.

Table 1: Policy optimization hyperparameters in this work.

Parameter	Value
Discount factor	0.99
GAE parameter	0.95
Policy ratio threshold	0.2
Entropy regularization	0.001
Time horizon	2048
Hidden layer size	64
Batch size	64
Number of epochs	5

5 EXPERIMENTS

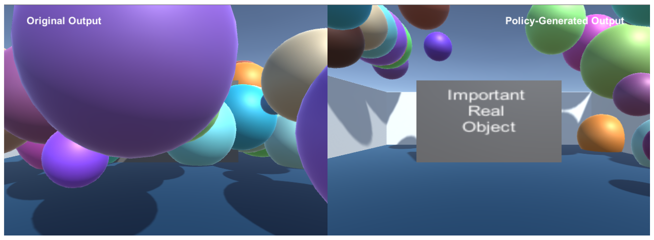
5.1 AR Simulator

Drawing inspiration from the experimental design of [6], we built an *AR simulator* in the Unity game engine [18] that allows us to create and customize virtual scenes representing real-world environments. In this work, we developed a simulated world of dimensions 20x10x20 meters with a large object representing a real-world sign. Within this simulator, we also built an *AR content generator* that allows us to simulate an arbitrary number of holograms. In the current version, a set of holograms of varying shapes, sizes and colors are first spawned in random locations within the simulated world, and then launched on randomized spatial trajectories until the simulation is stopped.

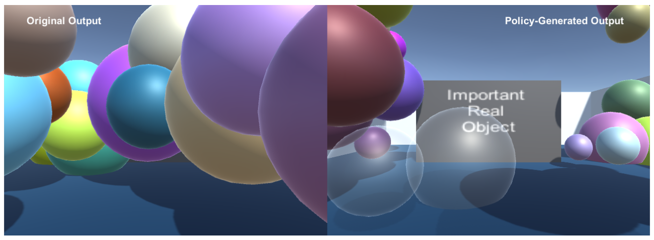
Reinforcement learning functionality was incorporated into our simulations using the Unity Machine Learning Agents SDK [2] and TensorFlow [19]. Communication between Unity and Python occurrs over an open socket. The simulations were run on a workstation (fog node) with an Intel Core i7-7700K CPU @ 4.20 GHz, 16 GB RAM, and an NVIDIA GeForce GTX 1070 graphics card with 1920 CUDA cores.

The experimental pipeline can be summarized as follows: (1) A simulated environment is created in Unity, and the number of holograms to simulate is specified in the content generator. (2) An agent is trained within this environment using PPO implemented with TensorFlow. Hyperparameters that were used in our experiments are provided in Table 1. (3) The resulting TensorFlow graph model is embedded into Unity using TensorFlowSharp[3] to allow the newly trained agent to make decisions in the simulated environment.

[2]Currently in its open beta version: https://github.com/Unity-Technologies/ml-agents
[3]https://github.com/migueldeicaza/TensorFlowSharp



(a) 30 objects



(b) 40 objects

Figure 2: Examples of visual outputs before and after RL-generated policies are applied. (a) In the original output (left), most of the user's field of view is obstructed by holograms. In the policy-generated output (right), the holograms are intelligently displaced to make the sign visible while still keeping most of the virtual objects within the user's view. (b) Even when the holograms are accidentally placed too close to the user or in front of real-world objects, the transparency policy ensures that important environmental details are not obstructed.

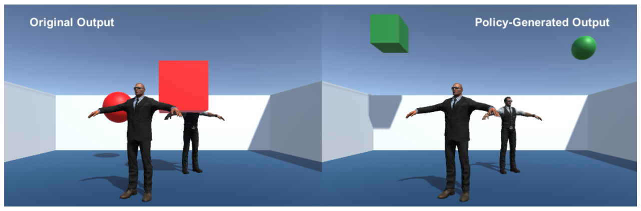


Figure 3: *Left:* The holograms interfere with the user's view of the pedestrians. *Right:* The policy reconfigures the holograms to not obstruct the pedestrians while remaining visible to the user.

5.2 Qualitative Evaluation

Following the system design outlined in Section 4 and the previously described pipeline, we trained an intelligent agent to regulate the placement of randomly generated holograms to prevent obstruction of the simulated real-world object. We offer two representative images, given in Figure 2a and Figure 2b, comparing our policy-generated visual output to the original, unfiltered output.

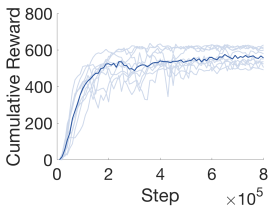
We also considered a more realistic scenario in which there are two pedestrians walking in front of the user, and two applications are simultaneously attempting to display holograms in the environment. As seen in Figure 3, when the applications attempt to interfere with the user's view of the pedestrians, the policy moves the holograms such that they are sufficiently far from the pedestrians, yet still fully visible to the user.

5.3 Policy Convergence

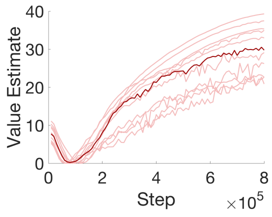
Figure 4a shows the agent's cumulative reward as a function of the number of training steps that have elapsed, in a simulation of three holograms. Over a successful training session, the cumulative reward should increase and converge to a final value, which indicates that the agent has identified a (possibly locally) optimal policy. We indeed observe this behavior in the displayed curve, which quickly increases in the beginning before plateauing to a stable value. Figure 4b shows the value estimate, i.e., the amount of future reward that the agent anticipates. As expected, the value estimate increases over the duration of training.

5.4 Frame Rate

Figure 5a compares the frame rate, expressed in frames per second (FPS), of our AR simulation with and without an RL-generated policy as the number of virtual objects in the environment is varied. The frame rate was obtained using the real-time statistics viewer built into Unity. For a small number of objects, activating the policy reduces the simulation frame rate by roughly 10-12 FPS compared to the no-policy case. Interestingly, as the number of objects increases, the no-policy and with-policy curves appear to converge. Even with 50 holograms in the scene, the resulting frame rate is well within the range necessary for a seamless user experience.

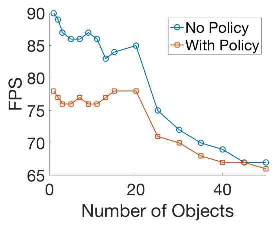


(a) Cumulative reward

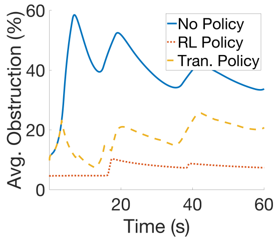


(b) Value estimate

Figure 4: Plots of different reinforcement learning metrics over the duration of 10 training sessions for a simulation with 3 holograms. Curves in a pale color represent individual training instances, while curves in a dark color represent the median over all instances. (a) The agent's cumulative reward quickly converges to a maximum value. (b) The value estimate increases over time, indicating that the agent becomes increasingly optimistic of its future reward.



(a) Frame rate



(b) Average obstruction

Figure 5: (a) Frames rate of our AR simulation in the no-policy and with-policy cases, as a function of the number of holograms rendered in the environment. (b) A moving cumulative average of obstruction values for no-policy, RL policy, and transparency policy.

5.5 Obstruction Metric

We collected obstruction values (using the metric from Section 4.6) over a period of 60 seconds when running no policy, an RL-generated policy, and a transparency policy (one of the main policies proposed in [6]) that makes objects transparent when they exceed a distance threshold from the user or real-world objects. Figure 5b plots the moving cumulative average of obstruction values for these three cases. We see that in the original (no-policy) output, there is both a high average obstruction (final average: 34%) and high variability in obstruction. The transparency policy improves upon this, with an average obstruction of 21%, by making holograms transparent when they would otherwise completely block the user's view. Finally, the RL policy offers a significant reduction to 7%, by both incorporating the transparency policy and intelligently displacing objects to less obstructive configurations.

An obvious limitation of using this metric in isolation is that a policy can achieve an obstruction value of 0% by either making holograms transparent all the time, or deleting them. In future work we will establish different *utility metrics* that quantify the usefulness of AR content preserved by a given policy.

5.6 User Study

We conducted a user study in which 9 participants were asked to compare 6 pairs of Arya-generated images and RL-generated images, each with anywhere from 2 to 40 holograms overlaid on a simulated real-world environment. Using a Likert scale from 1 (strongly disagree) to 5 (strongly agree), participants were first asked to specify the degree to which they felt that applications displayed in the RL-generated image were less obstructive than those in the Arya-generated image. The average response was found to be 4.52 (between "agree" and "strongly agree"). For the same pairs of images, participants were also asked to rate how strongly they felt that the applications in the RL-generated images were more useful than the applications in the Arya-generated image. The average response in this case was 4.44. Moreover, all 9 users stated that they would prefer the RL-generated images in an AR experience. These results suggest that RL-based object displacement policies improve upon the existing Arya policies with respect to both security and utility.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a set of new approaches that lay at the intersection of augmented reality, security, and machine learning. We presented a method for automatically generating policies to secure visual output in AR systems, using reinforcement learning and fog computing. A local fog node runs AR simulations to learn an appropriate policy for filtering potentially malicious or distracting content produced by an application, guided by a reward function specified by the user or system designer. We also defined a new obstruction metric to quantify the performance of a visual output policy. We showed that our RL-generated policies demonstrate great promise for intelligently displacing AR content to keep users' displays distraction- and obstruction-free, while causing relatively little degradation in frame rate.

This work opens up numerous exciting avenues for future research. In particular, we are currently working towards:

- Designing a method for automatically encoding user visual preferences into reward functions, and comparing the effect of different functions on the resulting policies.
- Deploying and testing our approach on physical AR devices.
- Formalizing a multi-agent reinforcement learning problem for collaborative multi-user AR scenarios.
- Considering ways in which other forms of AR signals, such as audio and haptic feedback, can potentially be regulated using policy optimization.

ACKNOWLEDGMENTS

This work was supported in part by the Department of Electrical Engineering and the School of Engineering and Applied Science at Princeton University, the Comcast Innovation Fund Research Grant, and Defense Advanced Research Projects Agency (DARPA), under contract No. HR001117C0052 and No. HR001117C0048. The opinions, findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency. We thank Dr. Andrew S. Lan for useful discussions, the participants of our user study for their time, and the reviewers for helpful feedback.

REFERENCES

[1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *ACM MCC'12*.

[2] Mung Chiang and Tao Zhang. 2016. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things J.* 3, 6 (December 2016), 854–864.

[3] Mark Gurman. 2017. Apple is ramping up work on AR headset to succeed iPhone. (November 2017). https://www.bloomberg.com/news/articles/2017-11-08/apple-is-said-to-ramp-up-work-on-augmented-reality-headset

[4] HoloKit. 2017. HoloKit. (2017). https://holokit.io

[5] Suman Jana, David Molnar, Alexander Moshchuk, Alan Dunn, Benjamin Livshits, Helen J. Wang, and Eyal Ofek. 2013. Enabling fine-grained permissions for augmented reality applications with recognizers. In *USENIX Security*.

[6] Kiron Lebeck, Franziska Roesner, and Tadayoshi Kohno. 2017. Securing augmented reality output. In *IEEE Symposium on Security and Privacy*.

[7] Kiron Lebeck, Kimberly Ruth, Tadayoshi Kohno, and Franziska Roesner. 2018. Towards Security and Privacy for Multi-User Augmented Reality: Foundations with End Users. In *IEEE Symposium on Security and Privacy*.

[8] Melanie May. 2015. Augmented reality in the car industry. (August 2015). https://www.linkedin.com/pulse/augmented-reality-car-industry-melanie-may/

[9] Microsoft. 2016. HoloLens. (2016). https://www.microsoft.com/en-us/hololens

[10] Volodymyr Mnih, Adrià Puigdomènech Badia, Alex Graves Mehdi Mirza, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *arXiv:1602.01783 [cs.LG]*.

[11] Marc Pollefeys. 2017. Second version of HoloLens HPU will incorporate AI coprocessor for implementing DNNs. (July 2017). https://www.microsoft.com/en-us/research/blog/second-version-hololens-hpu-will-incorporate-ai-coprocessor-implementing-dnns/

[12] Franziska Roesner, Tadayoshi Kohno, and David Molnar. 2013. Security and privacy for augmented reality systems. In *Comm. of the ACM*.

[13] Goldman Sachs. 2016. Virutal & Augmented Reality. (January 2016). http://www.goldmansachs.com/our-thinking/pages/technology-driving-innovation-folder/virtual-and-augmented-reality/report.pdf

[14] Dieter Schmalstieg and Tobias Höllerer. 2016. *Augmented Reality: Principles and Practice*. Addison-Wesley, Boston, MA.

[15] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust region policy optimization. In *arXiv:1502.05477 [cs.LG]*.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. In *arXiv:1707.06347 [cs.LG]*.

[17] Richard Sutton and Andrew Barto. 2017. *Reinforcement Learning: An Introduction* (2 ed.). The MIT Press, Cambridge, MA.

[18] Unity Technologies. 2018. Unity. (2018). https://unity3d.com

[19] TensorFlow. 2018. TensorFlow. (2018). https://www.tensorflow.org/

[20] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. 2017. Machine Learning for Networking: Workflow, Advances and Opportunities. In *IEEE Network*.