Personalized Augmented Reality Via Fog-based Imitation Learning

Surin Ahn
Department of Electrical Engineering
Stanford University
Stanford, CA
surinahn@stanford.edu

Maria Gorlatova
Department of Electrical and Computer Engineering
Duke University
Durham, NC
maria.gorlatova@duke.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

ABSTRACT

Augmented reality (AR) technologies are rapidly gaining momentum in society and are expected to play a critical role in the future of cities and transportation. In such dynamic settings with a heterogeneous population of AR users, it is important for holograms to be placed in the surrounding environment with regard to the users' preferences. However, the area of AR personalization remains largely unexplored. This paper proposes to use behavioral cloning, an algorithm for imitation learning, as a means of automatically generating policies that capture user preferences of hologram positioning. We argue in favor of employing the fog computing paradigm to minimize the volume of data sent to the cloud, and thereby preserve user privacy and increase both communication efficiency and learning efficiency. Through preliminary results obtained with a custom, Unity-based AR simulator, we demonstrate that user-specific policies can be learned quickly and accurately.

CCS CONCEPTS

• **Networks** → **Cyber-physical networks**; • **Human-centered computing** → **Mixed / augmented reality**;

KEYWORDS

Fog computing use cases, augmented reality, ML at the edge, privacy, behavioral cloning.

ACM Reference format:
Surin Ahn, Maria Gorlatova, Parinaz Naghizadeh, and Mung Chiang. 2019. Personalized Augmented Reality Via Fog-based Imitation Learning. In *Proceedings of Workshop on Fog Computing and the IoT, Montreal, QC, Canada, April 15–18, 2019 (IoT-Fog '19),* 5 pages.
https://doi.org/10.1145/3313150.3313219

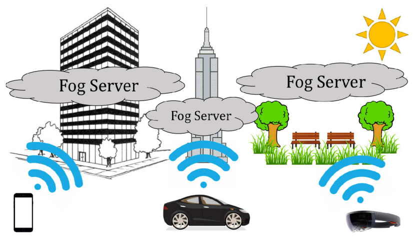


Figure 1: Geographically dispersed local servers (such as fog nodes) can help support interconnected cyber-physical systems, including AR devices, in a smart city setting.

1 INTRODUCTION

In augmented reality (AR), a layer of virtual content such as text, video, or holograms is superimposed onto a user's view of the real world [24]. AR is increasingly being adopted across numerous domains and is projected to generate billions of dollars in revenue within the next decade [21]. It is expected by many to become the next big computing platform, potentially as revolutionary as the personal computer and as ubiquitous as the smartphone. Not only is AR permeating the space of mobile devices [9], but it is also emerging in the form of fully immersive systems – whether as head-mounted displays (HMDs) [11, 16], or AR-enhanced windshields for vehicles [15]. A grand vision for AR is a smart city in which users can perceive real-time annotations of roads, buildings, transportation systems, and even people [1, 22, 23]; see Fig. 1.

Challenges in AR. Current and future AR deployments face a number of key challenges, including:

1) *Power and delay constraints.* The graphical demands of AR applications require significant power consumption. AR experiences are also very sensitive to delay, and the presence of lag in the processing of user interactions or virtual content could cause users to experience motion sickness, make navigation errors, or even get into accidents [7, 26]. Thus, enabling seamless and pervasive AR experiences will require a networking infrastructure, such as a fog or edge computing architecture, that supports ultra low-latency and resource-efficient data processing [14, 18, 27].

2) *Security and privacy.* As AR systems collect and analyze continuous streams of fine-grained sensor data, including video, audio,

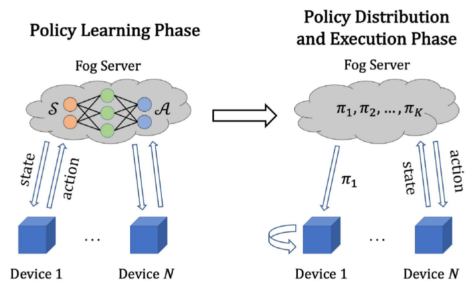


Figure 2: System diagram illustrating how a local server first trains several policies using state-action pairs from nearby devices, then either distributes these policies to powerful devices or executes them on behalf of resource-poor devices.

and user interactions with the environment, there is a pressing need to regulate what can be seen and manipulated by third parties such as cloud-based platforms [20]. From a security standpoint, we also need to ensure that holograms are presented to the user in a safe, non-distracting, and unobstructive way [2, 12, 13]. These safety mechanisms will become essential as AR-equipped vehicles and glasses turn into mainstream cyber-physical systems.

3) *Personalization.* By virtue of their complex arrays of sensors, actuators, and information processing capabilities, AR systems are in many ways more intimately connected to their users than any devices before. While a substantial amount of work has been done on context-aware AR [10], to the best of our knowledge, there is no existing literature on *personalizing hologram placements based on user preferences.* We expect this kind of personalization to be necessary to accommodate a heterogeneous population of AR users, especially those using HMDs. The users may differ in attributes such as height, eyesight, or hand dominance, all of which may influence their preferences for where holograms should be placed.

Our contributions. This paper advances the personalization dimension of AR systems by developing a method to automatically learn user preferences of hologram placements using *behavioral cloning*, an autonomous imitation learning technique,[1] which has been employed successfully in several different scenarios where a human expert's demonstrations can serve as valuable training data (e.g., autonomous vehicles [5]). We argue that the first two challenges (power/delay constraints and security/privacy) can be addressed by leveraging the availability of local computing resources, i.e., using an architecture consistent with *fog computing* [6]. By keeping the data processing closer to the devices and further away from the cloud, such architectures improve network latency, increase control over who gets to see the information, and reduce the chance of successful eavesdropping by an adversary [8]. Our preliminary results indicate that behavioral cloning holds promise for the future of AR personalization.[2]

2 PERSONALIZED HOLOGRAM PLACEMENT

We begin with a brief overview of imitation learning. We then formalize the AR personalization problem and present our behavioral cloning approach, which is illustrated in Fig. 2.

[1]See our demo of AR behavioral cloning at https://youtu.be/L8yF2bz-ymI.
[2]An extended version of this paper can be found at https://tinyurl.com/y5c96prd.

2.1 Imitation Learning Overview

In many contexts related to robotics and artificial intelligence, an autonomous agent is tasked with learning a desired behavior by observing an "expert" perform the task. This is the main idea of *imitation learning* (also known as *apprenticeship learning* or *learning from demonstration* [3]). More formally: Given an execution trace (i.e., a set of demonstrations) from an expert, can an agent learn to imitate the expert's policy? *Behavioral cloning* [4] is an approach to imitation learning in which the policy is estimated directly from a trace of the expert's policy $\pi^* : \mathcal{S} \to \mathcal{A}$ that maps states to actions, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. This trace is viewed as the "training set," and is represented as a sequence of state-action pairs $\{(s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots\}$. Any supervised learning method of choice, e.g., neural networks or SVMs, can be applied to fit a model to the data.

2.2 Problem Formalization

We consider a scenario in which N users in close proximity to one another are employing AR devices, which may be mobile phones, HMDs, heads-up displays, or others. At each time step t, a nearby server (such as a fog server) presents user i with a random, simulated environmental state denoted by $s_t^i \in \mathcal{S}$, which may consist of the locations and rotations of real-world and virtual objects, the sizes of their bounding boxes, and features of the application content (e.g., category). Additionally, the local server may ask the user for some specific attributes, such as height or eyesight, which can be incorporated into the state. In general, the server can generate these simulated states based on environmental context, by incorporating statistics about the real-world environment (e.g., locations, sizes, and frequencies of real-world objects based on sensors or video feeds). These statistics can help the local server create more realistic, context-specific simulations, which can then improve the efficiency of training and effectiveness of the resulting policy.

User i then sends back its chosen action $a_t^i \in \mathcal{A}$, which encodes how the user displaced the holograms from their original positions. Thus, at each time step, the local server collects N state-action pairs $(s_t^i, a_t^i),\ i = 1, \ldots, N$. Over some period of time T, the local server collects T samples per user, forming an $N \times T$ training set:

$$\begin{bmatrix} (s_0^1, a_0^1) & (s_1^1, a_1^1) & (s_2^1, a_2^1) & \cdots & (s_T^1, a_T^1) \\ (s_0^2, a_0^2) & (s_1^2, a_1^2) & (s_2^2, a_2^2) & \cdots & (s_T^2, a_T^2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ (s_0^N, a_0^N) & (s_1^N, a_1^N) & (s_2^N, a_2^N) & \cdots & (s_T^N, a_T^N) \end{bmatrix} \tag{1}$$

Here, each column of the training set gives the state-action pairs of all users at a particular time step, and each row gives the state-action pairs of a specific user during the entire duration of training. In practice, the value of N may range from one to hundreds, depending on the number of users near the local server. The number of demonstrations per user, T, on the other hand, would depend on the complexity of the environment/state space. From the training data in (1), the local server wishes to generate a set of policies that closely mimic the N users' demonstrations.

2.3 Learning Policies with Behavioral Cloning

In the *policy learning phase*, the local server can view each row of the training set as a distinct trace $d_i,\ i = 1, \ldots, N$. Then, for each

Algorithm 1 Learning Personalized Hologram Placements

```
1:  procedure LearnHolograms
2:  Input: Number of users $N$, horizon $T$
3:  Output: Policies $\pi_1, \pi_2, \ldots, \pi_K$, where $K \leq N$
4:  Initialize: $t \leftarrow 0$
5:  While $t < T$ :
6:      Send out simulated states $s_t^i$ to users $i = 1, \ldots, N$
7:      Collect actions $a_t^i$ from users
8:      $t \leftarrow t + 1$
9:  Generate $K$ different policies from user demonstrations $(s_t^i, a_t^i)$
    using behavioral cloning
10: Return $\pi_1, \pi_2, \ldots, \pi_K$
```

user, the local server estimates a placement policy π_i based solely on d_i, for a total of N different policies. For more efficient training, the local server could also identify similar users and merge their training data to produce a joint policy. The local server would then have K different training sets $d_1, d_2, \ldots, d_K$, $K \leq N$, which lead to K different policies $\pi_1, \pi_2, \ldots, \pi_K$. However, this approach would require us to define a notion of "similarity" that can be efficiently estimated by the local server. Here, we employ a multi-layer neural network to learn each of the N policies, where the input and output layers correspond to states and actions, respectively. The learning method is summarized in Algorithm 1. Note that the local server can re-run this algorithm whenever new users enter the vicinity.

2.4 Distributing and Executing Policies

After the policies are learned, they are cached in the local server and can be distributed to nearby users with powerful AR devices that are capable of running policies (e.g., Microsoft's HoloLens [17]). Powerful devices can simply download and run policies themselves, thereby reducing reliance on (and communication costs from) the local server. In the policy execution phase, the AR device must compute its own state based on what it observes *in real time*. From the device's raw video stream, it must extract the relevant features of the state using real-time image processing techniques such as the popular YOLO object detection algorithm [19] or the HoloLens's spatial mapping functionality.

If resource constraints prevent an AR device from running the policy itself (as with, e.g., smartphones or low-power HMDs), then the local server can compute the policy output based on the state sent from the device. As a consequence, the device must communicate much more frequently with the local server to send states and receive corresponding actions over the network. As with powerful AR devices, image processing techniques will be required to compute the state. However, for especially resource-poor devices, raw images will have to be offloaded to the local server or another edge node to be processed first. Edge-based offloading of deep learning for video applications has been explored in [18] and related works.

3 EVALUATION

Using an AR simulator that we implemented, we present preliminary results demonstrating that behavioral cloning can be used to generate a policy which accurately captures a user's hologram placement preferences.

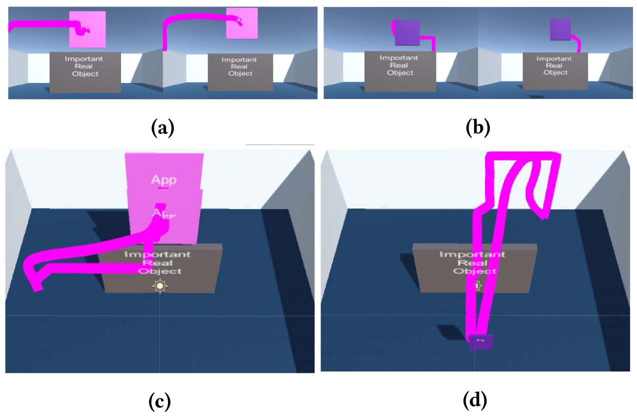


Figure 3: Two examples of behavioral cloning. In (a) and (b), the left screen shows the user-controlled hologram, while the right screen shows the agent-controlled hologram. The bright pink trail is the hologram's trajectory. Figs. (c) and (d) show the birds-eye view of (a) and (b), respectively.

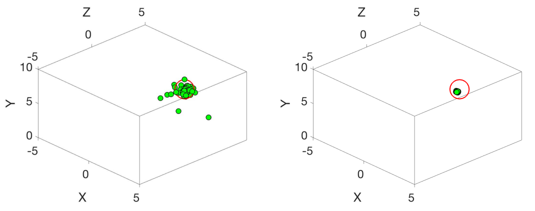
3.1 Evaluation Setup

Using the popular Unity game engine, we built an AR simulator with C# that randomly generates holograms (represented as colored tiles to resemble applications) in a virtual HMD-like experience. A large sign reading "Important Real Object" is placed in the center of the environment, both to serve as part of the state and to remind the viewer that anything other than the holograms constitutes the "simulated real world." On top of the AR simulator, we implemented the behavioral cloning algorithm with a multi-layer neural network using the Unity Machine Learning Agents Toolkit [25]. There are two key components to behavioral cloning in this implementation:

- *Teacher agent*: The agent controlled by the user, who provides a state-action pair at every video frame, which is then broadcast to the student agent, i.e., local server. The full trace of state-action pairs serves as the training set.
- *Student agent*: The completely autonomous agent that observes and tries to learn from the teacher agent's (user's) demonstrations. The student agent updates its policy as new demonstrations are provided.

At the beginning of each trial in the learning phase, a hologram of random size and color is spawned in a random location in the simulated environment and presented to the user as though he/she is looking through an HMD. The user uses the keyboard to reposition the hologram as desired, and can simultaneously view the agent's progress due to the split-screen feature of our simulator. From these demonstrations, the agent estimates a policy (represented as a neural network) that best estimates the user's behavior. After 500 frames, the environment is reset and a new trial begins.

We employed a standard neural network architecture for behavioral cloning, in which an input layer takes a feature vector representing the state of the environment, feeds this through four hidden layers, and produces at the output layer an action vector indicating how the holograms should be displaced from their current positions. In each of the following experiments, we ran 50 full training sessions, which is equivalent to training 50 independent and identical agents in the same environment.



(a) 1 demo **(b) 40 demos**

Figure 4: Content-independent hologram destinations at the beginning and end of training. Green circles represent the hologram destinations chosen by the agent; the red circle represents the target destination.

3.2 Content-Independent Hologram Placement

We first show that the student agent is able to learn the user's placement preference of a single hologram, independent of the application's content. That is, the agent's goal is simply to mimic the way in which the user moves around a hologram from arbitrary initial positions in the environment. The state of the environment at every video frame is given by the coordinates of the hologram and real-world object, and the sizes of their bounding boxes. Including features of real-world objects in the state is important because users may have different placement preferences depending on how real objects are configured. The action vector indicates how much the user moves the hologram along the x, y, and z directions.

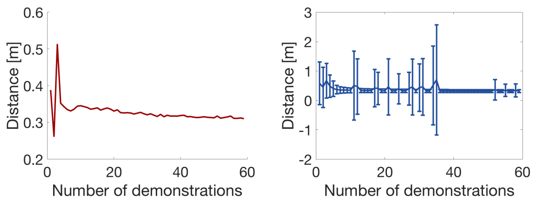
Qualitative comparisons between the user and agent hologram placements are given in Fig. 3. Observe that the agent is not only able to learn the user's preferred final position of the holograms, but also the *physical trajectory* that is taken. The full training session was $\approx$ 7 minutes; but the agent's actions began to closely match those of the user in much less time. Furthermore, the agent learns that the user prefers smaller holograms to be placed closer to them, to maintain a similar level of visibility as a larger hologram.

We collected data on 50 training sessions, each with 60 demonstrations from a deterministic (noiseless) teacher agent. At the end of each demonstration, we recorded the final destinations of both the teacher agent's and student agent's holograms. Fig. 4 shows these destinations for all 50 training sessions, at the beginning and end of training. As more user demonstrations from the teacher are provided, the student agent's accuracy and precision improve. The metric we use to assess the performance of the student agent is the distance between its hologram and the teacher agent's hologram at the end of each trial. Fig. 5 shows the median and mean (with standard error) of this metric over the duration of training. Overall, the agent's actions converge quickly to the target, but with high variability near the beginning of training. A couple of outliers around the 30-40 demonstration mark result in high standard error.

We say *convergence* has occurred when the agent moves the hologram to within a specified radius threshold (in meters) from the target, three consecutive times. Under a threshold of 0.45 meters, we found that convergence occurs in under 10 demonstrations more than 80% of the time. Looser thresholds lead to faster convergence.

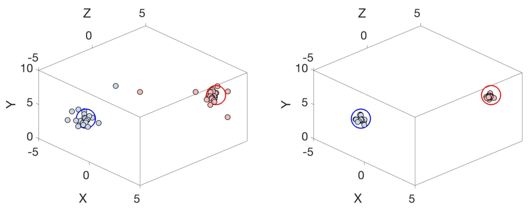
3.3 Content-Dependent Hologram Placement

We next test the student agent's learning ability when features of the hologram content are included in the state. The motivation is that



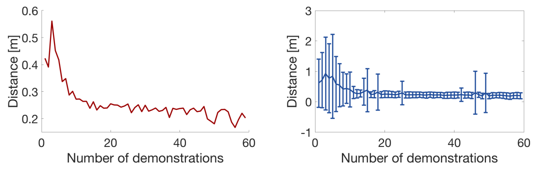
(a) Median **(b) Mean**

Figure 5: Median/mean error, content-independent case.



(a) 4 demos **(b) 13 demos**

Figure 6: Content-dependent hologram destinations. Each color corresponds to one application category.



(a) Median **(b) Mean**

Figure 7: Median/mean error, content-dependent case.

users may prefer having different types of applications (e.g., email or navigation) in different positions in the perceived environment. In our AR simulator, we use the hologram's color to represent its application category. This color is then encoded as an integer in the environmental state. Our experiment uses two different categories, represented by red and blue, and we apply a deterministic teacher function that moves holograms in a straight line to one of two separate target destinations (depending on the hologram's color).

Fig. 6 shows the student agent-driven destinations of the blue and red holograms at two different points in the training process. Early on, the agent yields a few misclassifications. However, even as early as 13 demos, full accuracy is achieved. Policy convergence is shown in Fig. 7. Under a distance threshold of 0.35 meters, convergence occurred in under 15 demos more than 80% of the time.

4 CONCLUSIONS AND FUTURE WORK

We proposed a fog-based behavioral cloning method for the emerging area of AR personalization, and demonstrated the promise of this approach via simulations. The use of edge servers for policy learning and distribution offers many benefits such as privacy preservation, faster training time, and low-cost communication, all of which will be essential in future fog-supported cyber-physical systems. Our future work includes evaluating the approach in more complex and realistic environments, and experimentally deploying it in a fog computing testbed with physical AR devices.

ACKNOWLEDGMENTS

This work was supported in part by the Comcast Innovation Fund Research Grant, NSF CSR-1812797 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 the reviewers for their helpful feedback.

REFERENCES

[1] 2011. Black Mirror. (2011). http://www.imdb.com/title/tt2085059/.
[2] Surin Ahn, Maria Gorlatova, Parinaz Naghizadeh, Mung Chiang, and Prateek Mittal. 2018. Adaptive Fog-based Output Security for Augmented Reality. In *Proc. ACM SIGCOMM VR/AR Network Workshop*.
[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57, 5 (2009), 469–483.
[4] Michael Bain and Claude Sommut. 1999. A framework for behavioural cloning. *Machine intelligence* 15, 15 (1999), 103.
[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv:1604.07316* (2016).
[6] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proc. MCC'12*.
[7] T. Braud, F. H. Bijarbooneh, D. Chatzopoulos, and P. Hui. 2017. Future Networking Challenges: The Case of Mobile Augmented Reality. In *Proc. IEEE ICDCS'17*.
[8] 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.
[9] Google. 2017. ARCore. (2017). https://developers.google.com/ar/
[10] Jens Grubert, Tobias Langlotz, Stefanie Zollmann, and Holger Regenbrecht. 2017. Towards pervasive augmented reality: Context-awareness in augmented reality. *IEEE Transactions on Visualization and Computer Graphics* 23, 6 (2017), 1706–1724.
[11] 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
[12] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner. 2018. Arya: Operating System Support for Securely Augmenting Reality. In *Proc. IEEE Symposium on Security and Privacy*.
[13] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner. 2018. Towards Security and Privacy for Multi-User Augmented Reality: Foundations with End Users. In *Proc. IEEE Symposium on Security and Privacy*.
[14] Qiang Liu and Tao Han. 2018. DARE: Dynamic Adaptive Mobile Augmented Reality with Edge Computing. In *Proc. IEEE ICNP'18*.
[15] Melanie May. 2015. Augmented reality in the car industry. (August 2015). https://www.linkedin.com/pulse/augmented-reality-car-industry-melanie-may/
[16] Microsoft. 2016. HoloLens. (2016). https://www.microsoft.com/en-us/hololens
[17] Microsoft. 2017. Second version of HoloLens HPU will incorporate AI coprocessor for implementing DNNs. (2017). https://www.microsoft.com/en-us/research/blog/second-version-hololens-hpu-will-incorporate-ai-coprocessor-implementing-dnns/
[18] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen. 2018. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In *Proc. IEEE INFOCOM'18*.
[19] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640* (2015).
[20] Franziska Roesner, Tadayoshi Kohno, and David Molnar. 2014. Security and privacy for augmented reality systems. *Commun. ACM* 57, 4 (2014), 88–96.
[21] 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
[22] Luis Sanchez, Luis Muñoz, Jose Antonio Galache, Pablo Sotres, Juan R Santana, Veronica Gutierrez, Rajiv Ramdhany, Alex Gluhak, Srdjan Krco, Evangelos Theodoridis, et al. 2014. SmartSantander: IoT experimentation over a smart city testbed. *Computer Networks* 61 (2014), 217–238.
[23] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. 2011. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In *The future Internet assembly*. Springer, 431–446.
[24] Dieter Schmalstieg and Tobias Höllerer. 2016. *Augmented Reality: Principles and Practice*. Addison-Wesley, Boston, MA.
[25] Unity. 2018. Unity ML-Agents Toolkit. (2018). https://github.com/Unity-Technologies/ml-agents
[26] Cedric Westphal. 2017. Challenges in Networking to Support Augmented Reality and Virtual Reality. (March 2017). https://www.ietf.org/proceedings/98/slides/slides-98-icnrg-challenges-in-networking-to-support-augmented-reality-and-virtual-reality-cedric-westphal-00.pdf
[27] Wenxiao Zhang, Bo Han, and Pan Hui. 2018. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proc. ACM Multimedia Conference'18*.