You Can Enjoy Augmented Reality While Running Around: *An Edge-based Mobile AR System*

Haoxin Wang
hwang50@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

Jiang Xie
Linda.Xie@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

ABSTRACT

Edge computing is proposed to be a promising paradigm to bridge the gap between the stringent computation requirement of real-time mobile augmented reality (MAR) and the constrained computation capacity on our mobile devices. However, prior work on edge-assisted MAR may fail to achieve expected performance in multiple practical cases, e.g., irreparable network disruptions caused by wireless link instability and user-mobility which is a critical characteristic of popular MAR applications. In this paper, we design a new edge-based MAR system named *Explorer* that enables mobile users to acquire guaranteed MAR offloading performance even under network instability and frequent user-mobility. Additionally, analytical models are developed to provide timely estimation of the sources of object detection staleness. Furthermore, we implement the proposed *Explorer* in an end-to-end testbed.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Networks** → **Network mobility**.

KEYWORDS

Edge Computing, Mobile Augmented Reality, Mobility Management

ACM Reference Format:
Haoxin Wang and Jiang Xie. 2021. You Can Enjoy Augmented Reality While Running Around: *An Edge-based Mobile AR System* . In *The Sixth ACM/IEEE Symposium on Edge Computing (SEC '21), December 14–17, 2021, San Jose, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3453142.3491416

1 INTRODUCTION

Existing mobile augmented reality (MAR) applications show two critical characteristics, real-time and frequent-user-mobility. However, since the computation complexity of object detection algorithms are usually too high to run on our resource-constrained mobile devices, *how to perform object detection fast, stably, and accurately on mobile devices is a critical challenge.* Although, recently, several low-complexity object detection algorithms are proposed to be applied on mobile devices, e.g., Google MobileNet [2] and DeepMon [3], their performance in terms of the detection accuracy and computation latency are significantly worse than that of the state-of-the-art object detection algorithms.

(a) Correct object detection results (b) Detection results with staleness

Figure 1: Object detection staleness.

To overcome above challenges, edge computing is considered as a promising alternative. First, edge servers are provisioned with sufficient computation capacity to perform advanced object detection algorithms, which can significantly improve the detection accuracy and decrease the inference latency, compared with performing on mobile devices. Furthermore, due to the network proximity, edge computing is able to provide a wireless connection with low network latency between the server and the MAR user.

Motivations: However, in frequent-user-mobility MAR scenarios (i.e., in the context of continuous object detection), offloading must be applied with care due to the object detection staleness. We define staleness as a critical performance metric that captures whether detected objects are at the reported positions. An example is shown in Fig. 1, assume at time $t = 0$, an image frame is offloaded to an edge server for processing. When the object detection result is returned to the MAR device at $t = 600$ms, which is over 20 frames old, the scene captured by the user's camera may already change due to the user-mobility, and the located object is no longer at the reported position, as shown in Fig.1(b).

Object detection staleness might be caused by several reasons. First, the wireless link between MAR devices and edge servers is unstable and may incur a long end-to-end (E2E) latency during continuous image frame offloading and virtual content delivery due to (1) the time-varying and capacity-constrained wireless channels and (2) user-mobility-incurred throughput decline. Second, MAR applications that require users to move frequently, such as MAR cognitive assistance [1], may cause wireless connection disruptions, i.e., radio handoffs. Besides, compared to the cloud server, edge server's compute resource is finite. So that the edge server may be easily overloaded by numerous MAR service requests, which increases the inference latency. All of the aforementioned issues may result in object detection staleness.

Existing MAR offloading work mainly focused on decreasing the offloaded data size to reduce the E2E latency, e.g., decreasing the video frame resolution [6]. However, none of these work consider a highly possible case: *what if the current network connection no longer*

SEC '21, December 14–17, 2021, San Jose, CA, USA

ACM ISBN 978-1-4503-8390-5/21/12…$15.00
https://doi.org/10.1145/3453142.3491416

Table 1: Notation

Variable	Description
k_m^2	Frame resolution of the mth video frame (pixels^2)
γ	The number of bits carried by one pixel (bits)
LA_m	E2E latency of the mth video frame (s)
LA_m^{tr}	Wireless network latency of the mth video frame (s)
LA_m^{cp}	Computation latency of the mth video frame (s)
c_m	Computation complexity of the mth video frame (TFLOPS)
H	Channel gain
I	Interference power (watt)
σ^2	Background noise power (watt)
P	Transmission power (watt)
B	Network bandwidth (Hz)
f	Available computational resources on the server (TFLOPS)

can provide satisfactory offloading performance even though the video frames are compressed? This case may frequently happen when an MAR user moves due to the wireless link quality fluctuation and irreparable network disruptions.

Contributions: In this paper, we propose an edge-based MAR system, *Explorer*, to enable fast, stable, and accurate object detection for MAR users with frequent-user-mobility. The contributions of this paper are summarized as follows,

(1) Proposing *Explorer* to mitigate the object detection staleness caused by the wireless link quality decline.
(2) Designing analytical models for estimating different sources of the object detection staleness in edge-based MAR systems.
(3) Implementing *Explorer* on an E2E system and conduct performance evaluations.

2 *EXPLORER*: COMPONENTS AND ALGORITHMS

At the heart of the proposed *Explorer* is that MAR clients offload object detection through WiFi at a higher priority and switch to cellular networks when their connected WiFi access point (AP)-attached edge servers fail to offer satisfactory MAR services. Thus, we consider all the AP-attached edge servers as *Dominating Edges* (DEs) and the cellular base station (BS)-attached edge server as the *Assistant Edges* (AEs). Switching the service between DEs and AEs requires sophisticated estimation of the performance decline sources and trigger mechanism.

We consider an edge-enabled network environment with one AE and N densely deployed DEs. Denote $\mathcal{N}$ as the set of DEs and a as the AE. We focus on a representative MAR user moving in the above mentioned network environment. Denote $\mathcal{M}$ as the set of generated video frames. Let L_m denote the location where video frame $m \in \mathcal{M}$ is generated. Due to the dense deployment of APs, multiple DEs can provide service to the MAR user for each video frame m at location L_m. And these DEs are denoted as $\mathcal{D}(L_m) \subseteq \mathcal{N}$. Meanwhile, the MAR user can obtain the service from AE a at any location L_m. Object detection of a particular video frame m is performed at either the user associated DE $n \in \mathcal{D}(L_m)$ or the AE a without being further offloaded to other DEs or a remote cloud. Denote $(a_{m,n}^w, a_{m,a}^c) \in \{(1,0),(0,1)\}$ as the edge association indicator which indicates the MAR user is served by DE n if $(a_{m,n}^w, a_{m,a}^c) = (1,0)$ and is served by AE a if $(a_{m,n}^w, a_{m,a}^c) = (0,1)$. As shown in Fig. 2, *Explorer* consists of four components: the offloading engine, edge computing resource tracker, tracking window controller, and data collector.

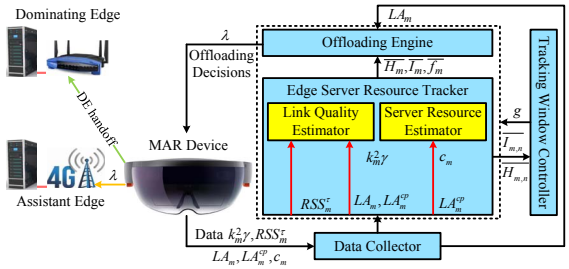

Figure 2: Overview of the *Explorer*.

2.1 Offloading Engine

The core component of our proposed *Explorer* is the offloading engine which determines when an MAR device should switch its video frame object detection from its associated DE to the AE. And based on the different sources of the link quality decline, the offloading engine provides the corresponding offloading scheme. Besides determining the timing of switching, the offloading engine can also quantify the number of frames that need to be offloaded to the AE based on the constraints of the LTE monetary cost and the battery drain. Therefore, our proposed offloading engine can not only dynamically mitigate the wireless link quality decline, but also guarantee a low energy consumption and monetary cost for the mobile device.

The offloading engine is executed when LA_m is not less than a threshold LA_{tg}. We use three types of trigger to determine the major source of the wireless quality decline.

Case 1: user-mobility-incurred case. (Channel gain trigger H_{tg}) In wireless networks, channel gain is a critical metric to estimate the wireless link quality between two communication nodes. It may vary with time, due to fading and node mobility. The value of channel gain is an applicable metric to estimate the level of user-mobility-incurred latency. In our designed offloading engine, when $\overline{H_{m,n}} \le H_{tg}$, the MAR device is triggered to switch its object detection tasks to the AE to maintain a small staleness. Meanwhile, the MAR user will re-associate with a nearby DE $i \in \mathcal{D}(L_m)$ for a better DE service, where we define this process as a *DE handoff*.

Case 2: temporary link quality decline case. (Radio interference trigger I_{tg}) Radio interference is another important metric to estimate the wireless link quality. We classify radio interference into two categories: *inter-DE* and *non-802.11 interference*. (i) Any traffic of the nearby DEs on the same channel or adjacent channels is defined as the inter-DE interference, where it may vary with time due to the user mobility. (ii) DEs are operated in the 2.4 or 5 GHz shared ISM band. The 2.4 GHz band is shared with other non-802.11 devices, which could lead to a significant amount of interference. Therefore, the increase of the DE's radio interference may be permanent (inter-DE interference) or temporary (non-802.11 interference). In our designed offloading engine, when $\overline{I_{m,n}} \ge I_{tg}$, the MAR user is triggered for a switching but no DE handoffs.

Case 3: temporary server resources exhausted case. (Available DE computational resource trigger f_{tg}) The inference latency may be significantly impacted by the computation latency because the available computational resources $f_{m,n}$ at the associated DE n is limited. Similar to the non-802.11 interference, we design that when $\overline{f_{m,n}} \le f_{tg}$, the MAR user will be triggered for a switching but no DE handoffs.

After determining the timing of switching, we next quantify the number of frames that need to be offloaded to the AE, which is denoted as λ. As stated above, there are two different switching scenarios: (i) switching with DE handoffs and (ii) switching without DE handoffs. For scenario (i), the MAR user keeps offloading video frames to the AE during a DE handoff. Since the latency of a DE handoff is long, around 3s [7], the MAR user suffers a long period without DE services during the DE handoff process. For scenario (ii), we build a model to calculate the number of frames that need to be offloaded to the AE, based on the constraints of the LTE monetary cost and the battery drain. The lower and upper bound of λ, (i.e., λ_{min} and λ_{max}) are calculated by,

$$P_c \frac{\lambda_{min}\overline{k^2\gamma}}{R(\overline{H_{z,a}}, \overline{I_{z,a}})} + 2e_{sw} \leq P_w \frac{\lambda_{min}\overline{k^2\gamma}}{R(\overline{H_{m,n}}, \overline{I_{m,n}})}, (z, m \in \mathcal{M}) \quad (1)$$

$$q\lambda_{max}\overline{k^2\gamma} \leq Q, \quad (2)$$

where **(1)** says that the battery drain of offloading after switching to the AE cannot exceed that of maintaining the link with the DE, and **(2)** says that the LTE monetary cost cannot exceed the maximum target Q (\$); P_w and P_c are the transmit power of the WiFi and cellular interface of MAR devices, respectively; $\overline{k^2\gamma}$ is the average data size of the video frames that have already been offloaded; $\overline{H_{z,a}}$, $\overline{I_{z,a}}$ and $\overline{H_{m,n}}$, $\overline{I_{m,n}}$ are the latest recorded estimation results of AE a and DE n, respectively, extracted from the edge computing resource tracker (describe in Section 2.2); e_{sw} is the energy consumption of conducting a switching; and q is the LTE monetary cost (\$/bit). **If the calculated** $\lambda_{min} > 0$, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$ are the current estimated wireless link quality of the associated DE n, and they are less accurate for representing the future link quality if the MAR user moves fast and frequently. In order to make the calculated λ more accurate, we choose λ based on the value of the tracking window size g which is extracted from the tracking window controller (described in Section 2.3),

$$\lambda = \lambda(g), \lambda \in [\lambda_{min}, \lambda_{max}]. \quad (3)$$

For example, when g is small indicating that the MAR user is moving fast and frequently and $R(\overline{H_{m,n}}, \overline{I_{m,n}})$ might change fast, we choose a relatively small value of λ; and vice versa. **If the calculated** $\lambda_{min} \leq 0$, the offloading engine reduces the transmit latency via decreasing the resolution of the next offloaded frame or executing existing computation-based solutions [5, 6]. The details of our designed offloading engine are given in Algorithm 1.

Algorithm 1: Offloading engine

Input: LA_m, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$, $\overline{f_{m,n}}$, $\overline{H_{z,a}}$, $\overline{I_{z,a}}$, LA_{tg}, H_{tg}, I_{tg}, f_{tg}
Output: g and λ.

```
if Tracking Window Controller triggered = true then
    if LA_m ≥ LA_tg then
        j ← 0; /* Case 1. */
        if H_{m,n} ≤ H_tg then
            DE handoffs start← true; (a^w_{m,n}, a^c_{m,a}) ← (0, 1);
            while True do
                j ← j + 1; λ ← j; /* Offload frames to AE a */
                if DE handoffs complete = true then
                    /* Switch to DE i */
                    (a^w_{m+λ,i}, a^c_{m+λ,a}) ← (1, 0), i ∈ D(L_m);
                    break;
        /* Case 2 and 3. */
        else if H_{m,n} > H_tg && (I_{m,n} ≥ I_tg || f_{m,n} ≤ f_tg) then
            Calculate λ_min, λ_max, λ via (1), (2), (3);
            if λ_min > 0 then
                (a^w_{m,n}, a^c_{m,a}) ← (0, 1); /* Switch to AE a */
                while True do
                    j ← j + 1; /* Offload frames to AE a */
                    if j = λ then
                        (a^w_{m+λ,n}, a^c_{m+λ,a}) ← (1, 0); /* Back to DE */
                        break;
            else if λ_min ≤ 0 then
                k^2_{m+1} ← k_min × k_min; /* Decrease the frame
                    resolution or other existing solutions. */
        g ← 1; /* Reset the tracking window size. */
    return g, λ
```

2.2 Edge Computing Resource Tracker

The functions of the edge computing resource tracker are estimating the wireless link quality, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$ and $\overline{H_{z,a}}$, $\overline{I_{z,a}}$, and the available computational resource, $\overline{f_{m,n}}$, using the information provided by the data collector. There are two components in it, link quality tracker and server resource tracker.

2.2.1 link Quality Tracker. This component is responsible for estimating the wireless link quality leveraging RSS, user-side data, and network-side data recorded by the data collector. Two metrics are used to represent the wireless link quality: (i) channel gain H^{τ}_m and (ii) interference I^{τ}_m. $\overline{H_m}$ and $\overline{I_m}$ are defined as the average channel gain and interference of the mth video frame, respectively, where

$$\overline{H_m} = 10^{\frac{\sum_{i=1}^{n} \hat{RSS}^i_m}{10 \cdot n} - 3} \cdot (a^w_{m,n} P_w + a^c_{m,a} P_c)^{-1}. \quad (4)$$

Based on Shannon's Theorem we have,

$$\overline{I_m} = \frac{(a^w_{m,n} P_w + a^c_{m,a} P_c) \cdot \overline{H_m}}{k^2 \gamma} - (a^w_{m,n}\sigma^2_w + a^c_{m,a}\sigma^2_c), \quad (5)$$

where k^2_m and γ are extracted from the data collector; and $LA^{tr}_m = LA_m - LA^{cp}_m$. In addition, we assume that the background noise of DE, σ^2_w, and AE, σ^2_c, are constant.

2.2.2 Server Resource Tracker. This component is responsible for estimating the available computational resources of the edge server. $\overline{f_m}$ is calculated as: $\overline{f_m} = \frac{c_m}{LA^{cp}_m}$, where LA^{cp}_m is extracted from the data collector and c_m is calculated via the computation latency model (described in Section 3.2).

2.3 Tracking Window Controller

Since sometimes an MAR user may stay at a location for a while and the wireless link quality may not change much, executing the resource tracker and offloading engine frequently will drain the battery and consume lots of computational resources on the MAR device. To address this issue, we design a tracking window controller to dynamically adjust the frequency of executing the edge computing resource tracker (i.e., tracking window size) and the offloading engine.

We use the variation of the wireless link quality to dynamically determine the tracking window size, since the variation of the available computational resources of the edge server is unpredictable (e.g., the number of MAR users who will connect to the server

is unpredictable) and does not have a direct correlation with the user-mobility. In addition, we only execute the tracking window controller when the MAR user is offloading its video frames to the DE. We define $\overrightarrow{\Phi_{m,n}} = (H_{m,n}, I_{m,n})$ and $\overrightarrow{\Phi_{j,n}} = (H_{j,n}, I_{j,n})$ as the wireless link state vector of the MAR user connected with DE n at location L_m and L_j, respectively. $\Phi'_{m,n} = \frac{\|\overrightarrow{\Phi_{m,n}} - \overrightarrow{\Phi_{j,n}}\|}{g}$ and $\Phi''_{m,n} = \Phi'_{m,n} - \Phi'_{j,n}$ are defined for describing the wireless link quality variation rate and the variation trend, respectively, where $m - j = g$ and g is the tracking window size with an initial value 1.

The details of dynamically adjusting the tracking window size g is described as follows. First, the tracking window controller checks if (i) the mth video frame is offloaded to DE n; (ii) $m - j$ is equal to the current window size g. The tracking window controller then extracts the channel gain $H_{m,n}$ and the interference $I_{m,n}$ from the data collector if the mth video frame satisfies the above two requirements. The tracking window size g is increased by 1 if the calculated wireless link quality variation rate $\Phi'_{m,n}$ is not larger than a preset rate threshold δ, which indicates that the current wireless link quality does not vary much. Increasing the window size will decrease the tracking frequency and thus, save the MAR device's battery and computational resource. Otherwise, we keep the window size if the current link variation trend $\Phi''_{m,n} \leq 0$ (i.e., the current link variation rate is decreasing) or we reduce g according to a preset regression function $\theta(g)$ if $\Phi''_{m,n} > 0$ (i.e., the current link variation rate is increasing).

2.4 Data Collector

The Data Collector is responsible for recording the MAR user-side, network-side, and server-side data, to facilitate the other three components. The user-side data includes k_m^2 and γ. The network-side data includes (i) the continuous RSS of the wireless link within each video frame transmitting period $RSS_m^\tau, \tau \in [0, LA_m^{tr}]$ (e.g., from the moment that the MAR user starts transmitting the mth video frame to the moment of finishing the transmission); (ii) the inference latency LA_m. Server-side data includes LA_m^{cp} and c_m.

3 *EXPLORER*: ANALYTICAL MODELS

In this section, to implement four components of *Explorer* and evaluate the object detection staleness, we design analytical models for analyzing the edge-based MAR system. The inference latency of the mth video frame can be defined as $LA_m = LA_m^{tr} + LA_m^{cp}$. Since the data size of the detection results is usually small, we do not include the latency caused by returning object detection results.

3.1 Image Offloading Latency Model

The image offloading latency is determined by the user's video frame resolutions and wireless channel quality. We assume that the AR video generated by the MAR user is preprocessed into video frames with the resolution of $k_m \times k_m$ pixels. Thus, the data size of the mth video frame is calculated as $k_m^2\gamma$ bits. In addition, Shannon's Theorem is used to model the wireless uplink channel quality. Denote $H_{m,n}^\tau$ and $H_{m,a}^\tau$ as the channel gain when the MAR user transmits the mth video frame to DE $n \in \mathcal{D}(L_m)$ and AE a at time $\tau \in [0, LA_m^{tr}]$, respectively. The maximum achievable uplink transmission rate for transmitting the mth video frame at τ is given by $r_m^\tau = (a_{m,n}^w B_w + a_{m,a}^c B_c) \log_2 \left(1 + a_{m,n}^w \frac{P_w H_{m,n}^\tau}{\sigma_w^2 + I_{m,n}^\tau} + a_{m,a}^c \frac{P_c H_{m,a}^\tau}{\sigma_c^2 + I_{m,a}^\tau}\right)$ where B_w and B_c are the channel bandwidth of WiFi and cellular networks, respectively; σ_w^2 and σ_c^2 are the background noise power, and $I_{m,n}^\tau$ and $I_{m,a}^\tau$ are the interference power at DE n and AE a while transmitting the mth video frame at τ, respectively. Therefore, the wireless network latency experienced by the mth video frame is modeled as

$$LA_m^{tr} = \frac{k_m^2 \gamma}{R_m}, \tag{6}$$

where $R_m = \frac{\int_0^{LA_m^{tr}} r_m d\tau}{LA_m^{tr}}$, which is the average wireless uplink data rate for transmitting the mth video frame. Moreover, the energy consumption for transmitting the mth video frame can be modeled as $E_m^{tr} = (a_{m,n}^w P_w + a_{m,a}^c P_c) LA_m^{tr}$.

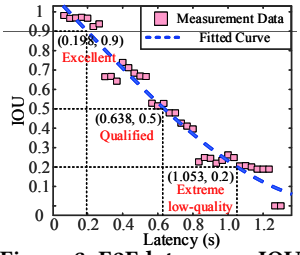

Figure 3: E2E latency vs IOU.

3.2 Computation Latency Model

The computation latency is closely related to the computation complexity of a user's task and available computational resources at the user associated edge server. Let $f_{m,n} \in (0, F_n]$ and $f_{m,a} \in (0, F_a]$ be the available computational resources at DE n and AE a, respectively, where F_n and F_a are the computation capacity of DE n and AE a, respectively. We assume that $f_{m,n}$ and $f_{m,a}$ do not change during the whole computation processing of the mth video frame. Therefore, the computation latency for the mth video frame is

$$LA_m^{cp} = \frac{c_m}{a_{m,n}^w f_{m,n} + a_{m,a}^c f_{m,a}}. \tag{7}$$

In addition, computation complexity c_m is closely related to the video frame resolution k_m^2. Since we focus on the wireless link level performance rather than the edge server system level performance in this paper, we consider an existing computation complexity model [6] described as $c_m = \psi(k_m^2) = 7 \times 10^{-10} k_m^3 + 0.083$, where $\psi(k_m^2)$ is convex with respect to the mth video frame resolution k_m^2. Although this model is built by implementing a special object recognition framework, YOLOv3 [4], our work is applicable to any other computation complexity model.

3.3 Object Detection Staleness Model

To evaluate the staleness, we use the Intersection over Union (IOU) metric, which is defined as,

$$IOU = \frac{area|O \cap G|}{area|O \cup G|}, \tag{8}$$

where O and G are the bounding boxes of the detected object and the ground truth, respectively. The value of IOU highly depends on the E2E latency. A larger E2E latency usually results in a lower IOU, which denotes a higher staleness. Therefore, we model the IOU as a function of the E2E latency LA_m. To do so, we implement an object recognition framework, YOLOv3, on a Nvidia Jetson TX2. Fig. 3 shows that the IOU decreases when the E2E latency becomes larger. Such a relationship between the IOU and LA_m can be characterized by a convex function, e.g., the measurement data can be fitted by a convex function,

$$IOU_m(LA_m) = 0.1323 \times LA_m^3 - 0.02898 \times LA_m^2 - 0.9623 \times LA_m + 1.091, \tag{9}$$

with the root mean square error (RMSE) of 0.05.

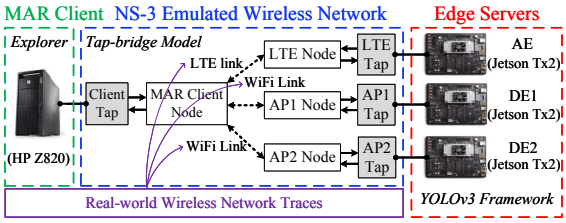

Figure 4: The edge-based MAR testbed.

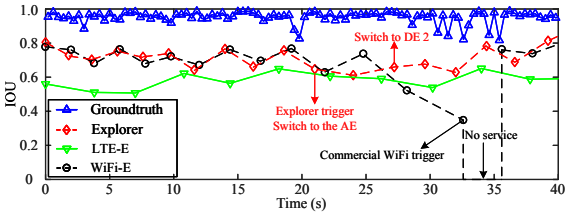

Figure 5: Sampled measurement IOU.

Table 2: Experimental results

	Average IOU	$IOU \leq 0.5$	$IOU \leq 0.2$
LTE-E	0.58	7.6%	0%
WiFi-E	0.62	17.5%	6.8%
Explorer	0.71	0%	0%
Groundtruth	0.94	0%	0%

4 IMPLEMENTATION AND EVALUATION

In this section, we implement the edge-based MAR system on a testbed, shown in Fig. 4, which consists of three components: the MAR client with *Explorer*, edge servers, and emulated network. We conduct experiments based on the implementation to validate the performance of *Explorer*.

4.1 Edge-based MAR System Implementation

MAR client with *Explorer*: The MAR client is developed in a HP Z820 workstation. It sends real-world captured video frames to an edge server and overlays the received information on its corresponding objects. The frame resolution of the video is 640×480. Four functional modules are implemented in the MAR client. The first one is a data collector that records the information, including the frame resolution, E2E latency, and transmission latency of each offloaded frame. The second one is an edge computing resource tracker that estimates the wireless channel quality using the recorded E2E latency and transmission latency. The third module contains a tracking window controller and an offloading engine, which responsible for determining handoff strategies based on Algorithm 1. The fourth module is a data communication module which streams the video frames to an assigned server selected by the offloading engine and receiving the object detection results.

Edge servers: There are three edge servers in our testbed, one AE and two DEs. These three edge servers are implemented on three NVIDIA Jetson TX development kits. They are developed to process the offloaded video frames and send the detection results back to the MAR user. Two major modules are implemented on each server. The first one is the service connection module which establishes a socket connection with the MAR client and returns detection results to the MAR client. The second one is the object detection module, YOLOv3 [4], performs the object detection.

Emulated network: The wireless network connecting the edge servers and the MAR client are emulated using NS-3. The MAR client and three edge servers are connected to their corresponding tap nodes in the emulated network via the tap-bridge model of NS-3. To emulate the variations of the wireless link quality when the MAR user is moving, we record 10 real-world wireless network traces in our campus building. The measured data includes the E2E latency and throughput variations while a user is moving. NS-3 takes the traces as input and dynamically varies the quality of the emulated wireless network while transmitting video frames.

4.2 Performance Evaluation

Fig. 5 illustrates the IOU comparison between our proposed Explorer and three baselines, where (i) *LTE-only solution:* frames are only offloaded to the BS-attached server (LTE-E); (ii) *WiFi-only solution:* frames are only offloaded to AP-attached servers (WiFi-E); (iii) *groundtruth:* all frames are directly computed on edge server (i.e., no data transmission). The sampling period is every 6 frames. Handoff trigger in IEEE 802.11 commercial WiFi products is to count the number of continuously missed beacons, or when the RSSI is below a certain threshold [7]. However, as shown in Fig. 5, this triggering mechanism is not sensitive to the performance of MAR and makes the IOU significant low during the handoff triggering, whereas the triggering mechanism in *Explorer* maintains a higher IOU. Table 2 shows the experimental results of the average IOU summarized from the experiments. *Explorer* achieves significant improvement on IOU as compared to LTE-E and WiFi-E. Furthermore, *Explorer* guarantees IOU > 0.5 when the MAR user is moving.

5 CONCLUSION

In this paper, we proposed *Explorer* to mitigate the object detection staleness caused by the wireless link quality variation, especially the user-mobility-incurred wireless network quality decline. To the best of our knowledge, this is the first study of reducing the MAR E2E latency from the communication perspective. In addition, *Explorer* is complementary to all existing computation-based MAR offloading solutions to further reduce the MAR E2E latency. The performance of *Explorer* is validated by experimental evaluations.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation (NSF) under Grant No. 1718666, 1731675, 1910667, 1910891, and 2025284.

REFERENCES

[1] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2014. Towards wearable cognitive assistance. In *Proc. ACM SenSys.*

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[3] Loc N Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. Deepmon: Mobile GPU-based deep learning framework for continuous vision applications. In *Proc. ACM Mobisys.* 82–95.

[4] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. *arXiv* (2018).

[5] Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han. 2020. Energy drain of the object detection processing pipeline for mobile devices: Analysis and implications. *IEEE Transactions on Green Communications and Networking* 5 (2020), 41–60.

[6] Haoxin Wang and Jiang Xie. 2020. User preference based energy-aware mobile AR system with edge computing. In *Proc. IEEE INFOCOM.* 1379–1388.

[7] Haoxin Wang, Jiang Xie, and Xingya Liu. 2018. Rethinking Mobile Devices' Energy Efficiency in WLAN Management. In *Proc. IEEE SECON.*