Received November 8, 2020, accepted November 21, 2020, date of publication December 4, 2020,
date of current version December 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042616

Practical Methods for Efficient Resource Utilization in Augmented Reality Services

GEORGE KOUTITAS[1], (Member, IEEE), SHASHWAT VYAS[1], CHAITANYA VYAS[1], SHIVESH SINGH JADON[1], AND IORDANIS KOUTSOPOULOS[2], (Senior Member, IEEE)
[1]Ingram School of Engineering, Texas State University, San Marcos, TX 78666, USA
[2]Department of Informatics, Athens University of Economics and Business, GR 104 34 Athens, Greece

Corresponding author: George Koutitas (george.koutitas@txstate.edu)

The work of Iordanis Koutsopoulos was supported by the EU CHIST-ERA Project "LeadingEdge" (call 2018).

ABSTRACT This work presents a novel approach that adopts content caching techniques towards reducing computation and communication costs of Augmented Reality (AR) services. The application scenario under investigation assumes an environment of static objects, each one associated to a holographic content. The goal is to devise practical low-overhead methods so as to reduce the amount of resources above that are needed for the most resource-demanding AR process, namely object recognition. The proposed method is based on caching images using a combination of metrics to rank them such as: (i) an object popularity index which favours objects that are most probable to be requested for recognition, (ii) the percentage of times when the object label has been encountered in the past, (iii) the probability that an image is similar enough with already encountered past images with the same label. The aforementioned image caching method drastically reduces database searches and returns the matched object that satisfies the needs of object recognition. We also devise a binary decision operator that initiates the object recognition process only upon comparison of spatial data of the AR device with the targeted object. The resulting performance is measured using a client-server architecture and components such as Wireshark, Unity Profiler, and Python. For our proposed architecture we deploy an edge server to satisfy the demands of the AR service. Results indicate that the proposed methods can significantly reduce both the computational resources and the induced network traffic, thus improving user experience.

INDEX TERMS Augmented reality services, object caching, edge computing, network offloading, resource allocation.

I. INTRODUCTION

Augmented Reality (AR) and Virtual Reality (VR) are soon expected to create new opportunities that will drastically change various sectors of modern economy. Typical applications of VR are the online gaming and the education industries [1]. AR technologies allow users to maintain visual connection to the physical world, while holographic content is overlaid onto the physical image. AR is usually referred to as Mixed Reality (MR) when the hologram can interact with the physical environment through the occlusion mechanism [2]. AR is expected to open new frontiers in smart-city services, retail shopping, operational teams' training, and customer engagement and experience because it enables a mixed experience for users through a combination of physical and holographic spaces [3], [4].

AR services require high bandwidth and processing power, primarily due to the object detection and object recognition processes that are employed in the holographic visualization on the physical space. The recent study [5] posits that in 5G networks, the bandwidth needs for AR and VR visual content streaming will range between 5-25Mbps for 3D image models, data visualization and telepresence. For 360-degree High Dynamic Range (HDR) video, bandwidth requirements are in the range of 10-50Mbps, while for 6 Degrees of Freedom (6DoF) and Point Cloud streaming the corresponding range is 0.2-1Gbps [5]. Another important challenge to address is service latency that determines User Experience (UX) in interactive and continuous AR services [6].

In AR services, *object recognition* is the process that accounts for the largest portion of communication and

The associate editor coordinating the review of this manuscript and approving it for publication was Yanjiao Chen.

computation resources, since the system continuously needs to recognize objects from images that fall within the Field of View (FoV) of the user headset, in real time [7]. Several tasks are involved along this process, such as image database search, frame capture and transmission, feature extraction, object classification, object recognition, template matching, object tracking and annotation [7]. There exist two different approaches of existing Software Development Kits (SDKs) regarding the location where the tasks above take place. In the first one, the tasks are executed only at the client side (i.e. at the AR headset) in an effort to reduce latency and network traffic. In the second approach, the majority of tasks are processed entirely at a cloud server, thus having the client at continuous offloading mode. Clearly, the first scenario increases the processing needs at the client side and drains its battery faster, whereas the second case incurs additional transfer delays.

An edge computing architecture may provide the right balance between these extreme scenarios. Continuous AR applications need to process captured frames at a rate of about 30 frames per second (fps), thus leading to a target latency of about 33msecs for acceptable user experience. There exist several ways to reduce latency, such as through eliminating unnecessary transmissions to the cloud or by using powerful CPUs or GPUs, but most of them rely on additional, expensive computational resources.

This article presents two novel and practical approaches that can help reduce computation and communication costs involved in AR services by introducing, implementing and evaluating a low-cost edge computing architecture specifically suited for AR scenarios. The main novel aspects of our method are summarized as follows:

- In an effort to reduce *computation* costs, we propose an edge cache-based architecture that uses object caching. The proposed method is based on *caching images* using a combination of metrics such as: (*i*) an object popularity index which prioritizes objects that are most probable to be requested for recognition, *(ii)* the percentage of times when the object label has been encountered in the past, (*iii*) the probability that an image is similar enough with already encountered past images with the same label.
- In an effort to reduce *communication* costs, we define a binary decision operator as a filter mechanism that initiates object recognition only when necessary, namely when the user location and field of view cover the location of the target object.
- In order to demonstrate our approach in a practical application use case, we develop a measurement system based on an interactive game in Unity that mimics user movements in a grocery store with 1,000 items. We integrate a client-server architecture and capture real data using WireShark, Unity Profiler and Python libraries.

The paper is organized as follows. In Section II, we present an overview of related work. In Section III, we present the system model and proovide a description of the application scenario. In Section IV, we present our proposed solution that consists of a novel image caching method and a spatial filtering technique to perform object recognition judiciously. In Section V we describe the measurement setup and in Section VI we show the result of simulations and experiments. Finally, section VII concludes our work.

II. RELATED WORK

Resource allocation and optimization are critical in edge computing and have received a lot of attention in the literature. A first challenge is to locate the edge nodes to perform the computations in the presence of a continuous stream of arising computation requests, or queries. The authors in [8] abstract the problem above and use the backpressure principle to solve the problem of computation operation placement and traffic routing to intended destinations so as to maximize throughput in terms of rate of computation query satisfaction. A utility-aware contract-based resource allocation algorithm for edge computing, called *Zenith*, is presented in [9]. Contracts are established between service providers and infrastructure providers that guarantee a resource allocation regime with low latency. An instance of a resource allocation problem for energy efficiency in AR applications is studied in [10], while leveraging the collaboration opportunities in order to save communication and computation resources. Further networking challenges pertaining to AR applications are studied in [7], [11], where it is verified that indeed, the most challenging processes in terms of resource consumption are object detection and recognition. Opportunities and research challenges that big data brings to AR are presented in [12].

Edge computing mechanisms such as caching for AR applications have been studied in the literature e.g [13], where a technique is presented that improves the streaming of a bulky 360° video whose major part comprises scene information never accessed by the user. Other scenarios of using caching jointly with traffic routing or content recommendations are presented in [14]–[16]. However, these do not address AR specifics with the exception of [18], where the authors formulate the problem of computation and bandwidth resource allocation in a static scenario with the goal to increase accuracy in context classification in AR applications.

In [17] the authors present a caching approach for video delivery using the concept of distributed helpers, while the work [19] presents an image prefetching and edge caching method. Finally, the work [20] uses the graphics processing unit (GPU) and frame caching on mobile devices to execute the large DNNs required for continuous video processing. Another use of caching for image processing is presented in [21], where reusable image regions are identified by exploiting the input video's internal structure. However, this method does not include any means to prioritize images.

This work proposes a practical low-cost method based on object caching for continuous AR applications, and a spatial filtering algorithm for non-continuous AR applications that both help reduce computational and communication

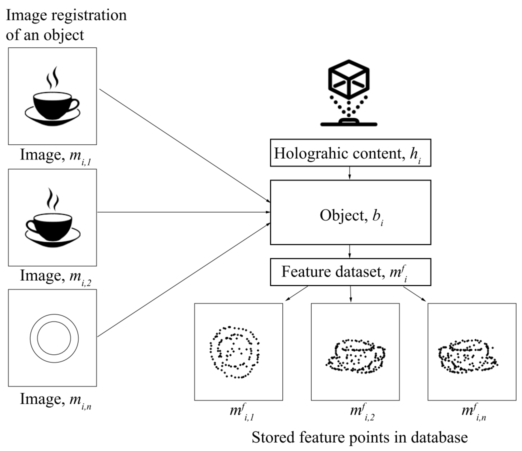


FIGURE 1. The objects in an AR application: holographic association and feature-point database. There are several images associated with an object, and each image involves a feature dataset. These images are stored in a cache, together with their accompanying holographic and extracted features.

costs. The work [19] has some conceptual similarities to our approach since it also employs image caching in a so called "recognition cache" on the device. The selection of the images to cache is performed by exploiting spatiotemporal locality. Our work complements their approach by enhancing the metrics based on which the images are cached.

In continuous AR cases, holographic content needs to be always overlaid on the physically captured image, as the user moves. On the other hand, in non-continuous AR cases there is a pre-registration phase that reduces the amount of times when real- time object detection is needed. The AR setting under investigation concerns cases where the object position is static; thus visual attention can be used to guide the caching algorithm. However, our approach can also work in a dynamic scenario with object detection requests arising continually.

III. SYSTEM MODEL

A. DESCRIPTION OF THE APPLICATION SCENARIO

An AR service provider offers holographic content to a set U of users within a specific geographic region Λ. Within the area, there exist objects with identifiers $b_i \in Q, i \in \{1, 2, \ldots N\}$, where Q is the set of N objects. Each object is associated to a unique holographic content $h_i \in H, i \in \{1, 2, \ldots, H$ with $H = N_.$ An example scenario for the application of such holographic content is in retail shops. In order to deliver the AR service to customers, the AR provider needs to perform offline two registration processes at the initialization phase: a) *object registration,* and b) *hologram association*, presented in Fig. 1. During object registration, a number of "training" images, are related to an object b_i. The index n in the figure indicates that more than one images may in fact be associated to a registered object. Images are processed, and their corresponding feature points are extracted using methods such as ORB, KAZE, SURF [22]. Feature points are mathematical representations of key regions of an image and may include corners, edges or even parts of the image.

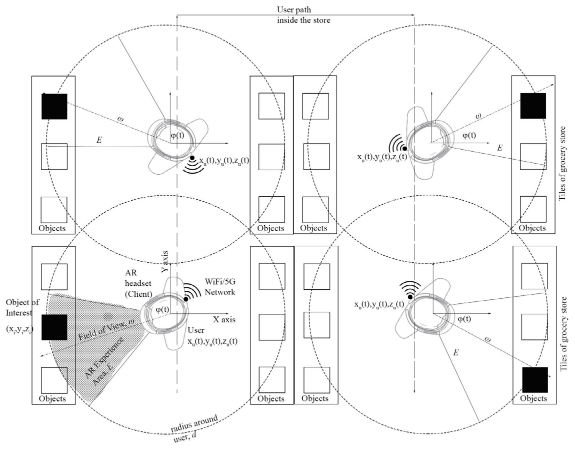


FIGURE 2. Top-view description of our scenario: a user with an AR headset and app moves within an area with static objects.

A top view of the area of interest Λ is shown in Fig. 2. This area is a grocery store and is one of the most challenging AR environments. Users walk across the grocery store corridors with their smartphone, and they request information about different grocery products at different times by pointing the smartphone camera to the product. Several users may simultaneously request data visualization, and they coexist in a scene with a large density of static objects. We assume a two-dimensional (2-D) setting here, however the concept can be extended to a 3-D one.

Using an AR mobile app, the user can walk around and observe holographic content overlaid on top of the physical object. Each holographic content is associated to its object with 2-D coordinates $\mathrm{x_i}$, $\mathrm{y_i}$. The user position is mapped to 2-D coordinates ($\mathrm{x_u}$, $\mathrm{y_u}$). With respect to user location, the object polar coordinates are:

$$r = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2} \tag{1a}$$

$$\varphi = tan^{-1}\left(\frac{y_i - y_u}{x_i - x_u}\right) \tag{1b}$$

The AR device has a specific Field of View (FoV) with angle ω. For example, existing AR headset technologies have a FoV in the range $30 \leq \omega \leq 60$. For User Experience (UX) purposes, the holographic content that is visualized is the one associated to the set of objects $D \subset Q$ that fall inside an AR Experience Area, denoted by **E**, which is a sector of ω degrees angle, i.e. a portion of a hypothetical circle (or a sphere) of radius d around the user position that includes an area $E = \pi d^2 \cdot \omega / 360$, as indicated in Fig. 2.

B. DISAGGREGATION OF AR PROCESSES

As the user navigates along her path, there are various tasks that need to be executed to deliver the AR service to her. An example is shown in Fig. 3. The first task is frame processing. Frames are images in the FoV E_u of user u. Upon capture, these frames are reduced in size, typically to 480 x 270 pixels, they are converted to gray scale, and

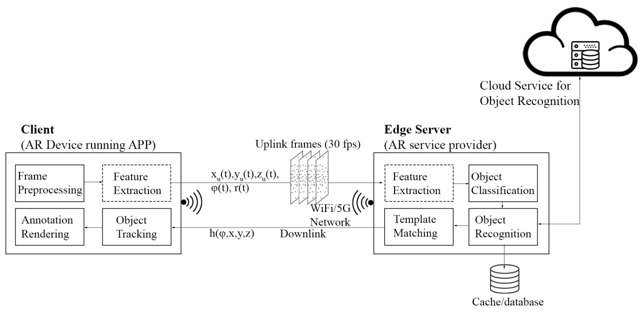


FIGURE 3. Overview of the procedures executed at the client and server side for an AR application.

they are encoded into a *jpeg* file. Frame processing takes about 12% of the total amount of computational resources needed for the AR service [7]. Frames are then transmitted to the server, usually at a frame rate of $\lambda = 30$ fps. The next step is *feature extraction* for each frame. This is usually performed with algorithms such as ORB [22]. Feature extraction identifies important and unique features of each image that are used for image comparison. Feature extraction represents approximately 21% of the computation load [7].

Next, *object classification* is performed through hypothesis formation. This is usually done through pre-trained general-purpose Neural Networks (NN) classifiers. Regardless of the type of NN, the output is always a vector of likelihoods $\underline{L} = (L_l : l \in Q)$, where l is the label of the object. Note than an image may in general contain more than one objects of label l.

The next step is *object recognition* which is at the core of AR services. Depending on the NN type, object classification and recognition may be performed simultaneously or not. An object needs to be recognized so that an appropriate hologram is associated with the object and projected to the user. Due to the need for sequential and extensive database search, this task is computationally heavy and time-consuming, and its execution is usually performed at a central server (cloud) or preferably at an edge server. In order to recognize an object, the feature points of a frame need to be compared to those of registered images. Let s_{ij} be a measure of feature similarity for of a pair of images $(i\ ,j)$, where i is the processed image in question and j denotes an image in the database. If the entire database is parsed, the outcome of object recognition is the image in the database with the highest similarity to the processed frame.

Object recognition accounts for approximately 33% of the overall computational demands of an AR service. Once an object is recognized, *template matching* verifies the result of object recognition and calculates the poses of targets in the 6DoF space. Template matching corresponds to about 3.3% of the overall latency of the AR service [7]. The last tasks of AR service are *object tracking* and *annotation rendering*, and they are performed at the AR device. Object tracking accounts for 5% of the overall latency, and it tracks the feature points of an object so that the holographic content can follow it. For static AR services as the ones we consider in this work, this is not needed. Finally, *annotation rendering* updates the 6DoF pose of an object and calculates a 3D pose of the annotation. This is accounts for 7% of the overall latency. According to [7], the latency for network data transfer corresponds to about 18% of total latency.

C. COMPUTATIONAL DEMANDS

We focus here on the most demanding task, namely object recognition. Assuming that a transmitted frame at each time is an image of size q bits and that the average data transmission rate over the wireless channel is c bps, the average transmission time for an outbound image is:

$$T_{up} = \frac{q}{c}. \tag{2}$$

In the downlink (inbound), the serv er responds on the same communication channel with the holographic content h of p bits. Assuming symmetric uplink and downlink data transmission rates, the average transmission time for inbound holographic content corresponding to an image is:

$$T_{down} = \frac{p}{c}. \tag{3}$$

An important factor that affects total delay is feature matching. Assuming that the comparison of the feature points of a frame to those of a registered image is a task of workload w flops, and there are a total of K images present in the cache, the total processing time of all images in the database, with a processing speed cs flops/sec is:

$$T_P = \frac{w.K}{cs} \tag{4}$$

Assuming a propagation delay T_{prop} between client and server, the total average roundtrip time delay per user per frame is approximated as:

$$T_{AR} \approx T_{up} + T_{down} + T_{prop} + T_P \tag{5}$$

For continuous AR services, it is recommended that $T_{AR} < 33$ msecs. The workload w is substantial and it can be reduced by minimizing the number of database searches related to feature matching. This is precisely the topic of this work.

As the number of users U within the area of interest Λ increases, important resource allocation challenges need to be addressed. The expected total in- and outbound data traffic exchanged between a client and server per user request in an area with U users for each frame is:

$$D = \sum_{u \in U} (q_u + p_u) \tag{6}$$

where q_u, p_u are the image and holographic content size for user u. In that area, the server computational requirements are

$$C = \sum\nolimits_{u \in U} w_u, \tag{7}$$

assuming that a user makes one request for object recognition. Note that w_u is the workload for feature matching per user.

Algorithm 1 Caching Algorithm Based on Visual Attention

```
1. Object Registration in the database
- For each object $i \in N$, create feature points using ORB algorithm,
- register feature points in database
- assign holographic content to each group of feature points
- store position $x_i, y_i$ of objects
2. Computation of object significance probability and of metric (8)
- For each object $i \in N$ calculate significance probability
  $p_i$ as follows:
  - Count the number of users $k_i$ in whose FoV the object
    $i$ is located (the object is in the FoV of a user if the
    user-object distance satisfies $r \leq \delta_d \& (\varphi - \frac{\omega}{2}) \leq \varphi$
    $\leq (\varphi + \frac{\omega}{2})$ - assume $\delta_d$=5m, $\omega$=$30^0$, a typical FoV).
    $p_i = k_i / \sum_i k_i$ (normalized, so that $p_i \in [0, 1]$).
    - Compute metric $g_i$ as in (8).
    END
3. Caching
- Cluster objects into groups closely located to each other
- Sort objects of each group based on
  $g_i$: $[b] \rightarrow sort(g_j, 'descend')$
- Place sorted objects in the cache until cache capacity is reached
4. Object Recognition
- Choose sensitivity parameter $\theta^{ratio}$ (e.g. $\theta^{ratio} = 0.6$) and
  feature matching threshold $\theta$ (e.g. $\theta = 200$ matched features)
- For each incoming frame $i$, compute similarity $s_{ij}$ with each cached
  image $j$ (BRUTE FORCE matching algorithm)
- Declare perfect match at image $\min\{j : s_{ij} \geq \theta\}$ and return
  corresponding object as recognized object
ELSE IF no match is found
- Send image to cloud image database.
```

IV. PROPOSED ALGORITHM

In order to avoid large client-server propagation delays, we assume that the AR service provider is willing to execute the service locally at the edge. Following the notation of Fig. 1, for each user position (x_{u}(t), y_{u}(t)) at time t and a view angle $\varphi_u(t)$ of the user, the method needs to identify the holographic content $h(x,y,\varphi,t)$ that will be delivered to each user $u \in U$. To reduce the total roundtrip delay T_{AR}, we propose two methods.

The first one uses edge caching in a novel manner in order to reduce the computation cost of object recognition. This method can be implemented for continuous AR use cases. The second algorithm defines a binary decision operator which can be thought of as a spatial filter that initiates object recognition only when the user is near a predefined object of interest. This method helps to reduce communication costs.

A. IMAGE CACHING ALGORITHM

A first key idea is that we would like to keep in the cache images of items that are popular in terms of user request

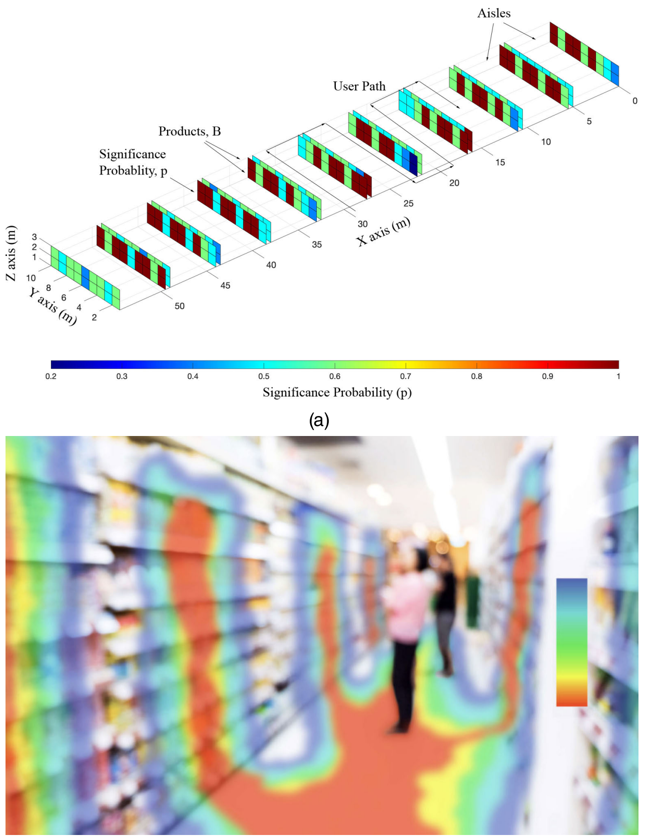


FIGURE 4. a) 3D view of a grid layout scenario with the significance probability of various products in aisles. Red color indicates high significance. b) visual attention map visualization [24].

probability. Popularity is related to the frequency of holographic content request for a specific object, which is directly associated to the visual attention of users on specific objects of the scene (Fig. 4). At the same time, cached images should also guarantee high probability for correct object recognition. The algorithm parses through the set of cached images according to a priority index. This process is shown in Fig. 5. If a match is not found in the local cache, the object recognition task is sent to the cloud, and the entire database lookup is performed for image matching.

1) SIGNIFICANCE OF AN OBJECT

Assuming sequential database search, images with higher popularity should be stored first in the cache. Image popularity is computed as follows. For a set of candidate objects $S \subset N$, consider a Markov chain depicted as a graph where nodes are objects, and a link (i, j) between nodes denotes transitions of user requests from object i to j.

The stationary distribution of the Markov chain is denoted by vector $\boldsymbol{p} = (p_i : i \in S)$ and captures the "significance" of objects. That is, it models the visual attention of users [23] in terms of an estimate of the long-term probability that a user requests an object. Usually, information about visual attention can be obtained through eye-tracking technologies. There are various metrics to be considered such as: (a) the *time to first*

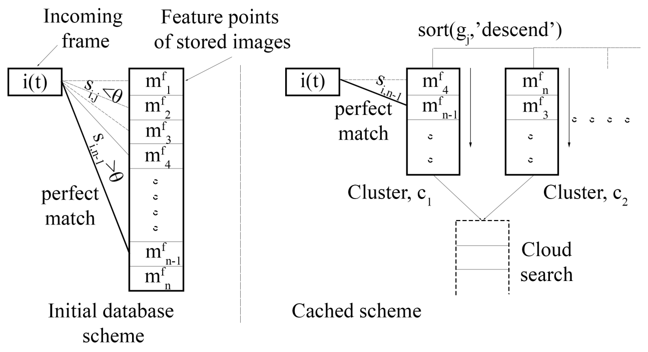


FIGURE 5. Comparison of the traditional sequential feature matching with the proposed cache-based approach. Instead of performing a sequential full search of the whole database and stop at the first image that exceeds a similarity threshold with the incoming image frame (on the left), our approach searches smaller image clusters (on the right) and thus has more chances to find a match faster.

fixation that is equal to the time it took the user to focus on a specific product, (b) the *fixation count* that models the number of times users looked at a specific product, c) *first fixation duration* that models how long a user spent gazing at an object for the first time, and d) the *average fixation duration* that models how long a user spent gazing at an object on average [24]. In this work, we assume that the significance is just the fixation count that models the number of times a specific object was observed by users based on their position, $x_u(t)$, $y_u(t)$ and headset direction $\varphi_u(t)$. The significance above can be normalized by dividing over significances of all objects in that location so that it is bounded in [0,1] and be mapped to a probability value.

An example is depicted in Fig. 4, while the computation of significance is detailed in *Algorithm 1*. The structure of the scenario is assumed to be a grid layout [23]. For the purpose of our study, a significance value was computed based on Monte Carlo simulations due to unavailability of an eye-tracking method and relevant data. Repeated random walks of users on a user path were used to obtain numerical results of $\boldsymbol{p}$. Each user was assumed to walk on the path, as shown on Fig. 4, with an initial φ_0. At each time t, the value φ(t) is computed according to $\varphi(t) = \varphi_0 \pm \delta_\varphi$, where δ_φ is a statistic derived assuming that the Normal distribution is followed, i.e. $\varphi(t) \leftarrow \mathbf{N}(\pm 90, 20)$, where $\mathbf{N}$ denotes a *Normal* distribution with $\pm 90^o$ mean and standard deviation 20^o. The Monte Carlo numerical values were computed from 1,000 randomly generated user walks in the store.

2) OBJECT IMAGE CACHING

Requests for object recognition arrive sequentially from users with frame rate λ requests/sec. The problem is to determine the subset of images of the database to include in the local cache, and the order in which they are cached, so as to expedite object recognition. We assume that an input image is sequentially checked for matching with each image in the cache. Thus, the order in which registered images are cached is important for the overall performance. Let F, $|F| \geq |N|$ represent the entire set of images in the backend database which was created initially during registration. Recall that an object is in general associated with several images. The cache needs to choose a subset C_l of images for all objects b_i, $i \in N$.

3) FEATURE EXTRACTION AND FEATURE MATCHING

Two important steps of the algorithm are feature extraction and feature matching. For feature extraction, we implemented the Oriented FAST and Rotated BRIEF (ORB) algorithms in *OpenCV*, which are address feature detection, similarly to SIFT and SURF algorithms [22], [25]. The ORB algorithm uses FAST to find keypoints in the image, and then it applies *Harris* corner detection to find the top M points.

Another important mechanism is feature matching. For the purpose of our study, we implement the Brute-Force (BF) Matcher feature matching algorithm in *OpenCV* [25]. BF Matcher takes the descriptor of one feature in the first image and compares it to all features of the other images using a distance metric, e.g. Hamming or Euclidean distance. The algorithm returns the total number of matched features. Assuming that two images i and j have feature datasets $m_i^f = (u_1, u_2 \ldots, u_n)$ and $m_j^f = (v_1, v_2 \ldots, v_n)$, the number of matched features is $s_{ij} = \#\{k : u_k = v_k, k = 1, 2, \ldots, n\}$. Clearly, among all registered images, the one that has the largest number of matched features with those of the input frame or that has a number of matched features that is larger than a threshold, is considered as the matched image to the input frame. Namely, a match between the feature datasets of the two images occurs when $s_{ij} \geq \theta$, where θ is the matching threshold (for our purpose, we set $\theta = 200$ matched features). An example of the value of s_{ij} is depicted in Fig. 6. An input frame was compared to 300 different cached images. The perfect match had value $s_{ij} = 1,917$ or $\log(s_{ij}) \sim 3.29$; namely 1,917 features were matched between the two images.

An important practical parameter that concerns image comparison at the level of individual feature points needs to be calibrated in the ORB and BF Matcher algorithms. This parameter is the matching threshold, $\theta^{ratio} \in [0, 1]$. A smaller threshold θ^{ratio} implies a stricter and more sensitive comparison between images at the level of individual feature points. Hence, a smaller threshold θ^{ratio} leads to fewer but more accurate matches between examined pairs of images. On the other hand, if we increase θ^{ratio} towards 1, the comparison between feature points is less strict, and more feature points will be matched. Thus, we have "easier" image matches but with larger uncertainty, and this makes object detection more challenging. This is because of the nature of the search algorithm for matched images in the cache.

Consider an incoming image frame. During the sequential comparison with cached images, the first encountered image in the cache for which $s_{ij} \geq \theta$ is declared as a match, and the algorithm stops. This image may be any image from the set of possible matched ones. Because the set of possible matched images is larger for larger θ^{ratio}, this means the chances to find a good match are fewer if θ^{ratio} is larger. The situation for two indicative values of threshold, $\theta^{ratio} = 0.6$ and $\theta^{ratio} = 0.8$ is depicted in Fig. 6.

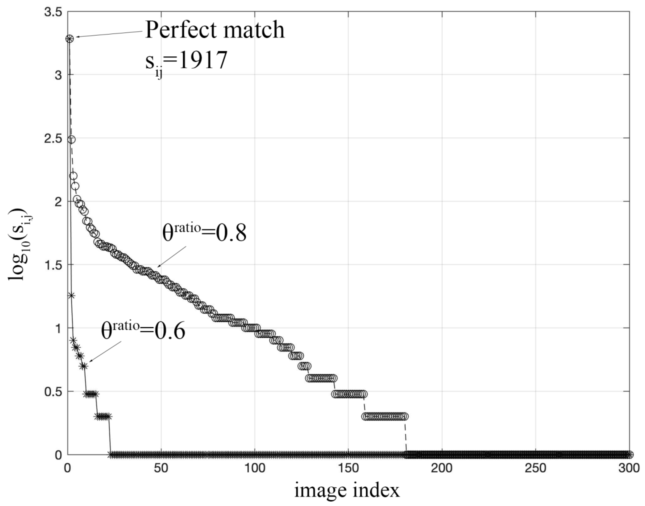


FIGURE 6. **Similarity metric s_{ij} based on ORB. The logarithmic scale presents feature similarity between an input frame and all images in the cache in descending order. All values smaller than 0 were set to 0.**

4) PERFORMANCE METRIC

Performance is measured in terms of delay and the probability of mislabeling the image; the latter corresponds to associating the image with an incorrect object. The cache should have appropriate images so as to reduce delays and keep probabilities of mislabeling low. If K images are in the cache, the maximum delay for parsing the cache ismuch smaller than the traditional exhaustive search in the entire database. The idea behind image caching is as follows: the ranking of an image in the cache should be high if: (*i*) the object is popularm namely it is requested by users with high probability; (*ii*) its label is encountered often; and (*iii*) it is very probable that an image is very similar with already encountered past images with the same label.

For each image i in the cache, we compute an empirical distribution of similarity to past images that have been classified with the same label. If we define as l the image label, we can compute the empirical probability $z_i(l)$ that the similarity of cached image i to past images that are labelled as l, is greater than θ. The idea is that cached images should be similar enough to past images with the same label so as to facilitate accurate and fast object recognition. Since a cached image i has an already defined label L_i in the set of labels $\{1, \ldots, N_0\}$, we can compute for each image i the metric:

$$g_i = p_i \cdot z_i(L_i) \cdot \psi(L_i) \tag{8}$$

where p_i is the significance (popularity) of object i, and $\psi(L_i)$ is the empirical probability of having encountered label L_i in the past, which can be computed from historical data. Images are cached in decreasing order of the metric g_i.

B. SPATIAL FILTERING METHOD

The caching method above can be applied in continuous AR applications in which the user holds the AR device and expects to see the overlaid holographic content on the physical object in real time. In such scenarios, object detection is executed continuously in the backend to provide appropriate hologram association. Our objective was to reduce computational needs of object recognition through image caching.

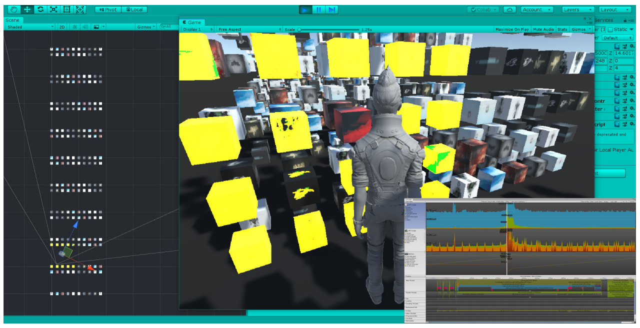


FIGURE 7. **Emulator of a user in the grocery store. User was able to navigate the avatar. The left section presents the position of avatar in store. The diagram on the bottom right is a screenshot of the Unity Profiler. A simplified demo can be found in [26].**

In another use case, the user tries to reach a specific targeted object through using a navigation AR tool. Following the notation of Fig. 2, object recognition should be initiated only when the target object is inside the AR Experience area E_u of user u. A binary filter operator is used to check whether this condition holds so as to decide whether to enable the object recognition engine or not, according to the rule:

$$Q_{u,i}(r,\varphi) = \begin{cases} 1, & r \leq \delta_d \textit{ and } (\varphi - \frac{\omega}{2}) \leq \varphi \leq (\varphi + \frac{\omega}{2}) \\ 0, & \textit{otherwise} \end{cases} \tag{9}$$

With reference to the architecture of Fig. 3, this filter operator is implemented at the client side to decide on whether or not to initiate the processes from feature extraction and beyond.

V. MEASUREMENT AND IMPLEMENTATION SETUPS

A. MEASUREMENT SETUP

A measurement system over a client-server architecture was implemented using the platforms of Unity, Python, and OpenCV. The client and the server were connected through Transmission Control Protocol (TCP) and Internet Protocol (IP) using socket programming. Sockets were created based on the IP address and port numbers of the endpoints.

1) CLIENT INFORMATION

The client was a laptop device that was running an emulator of human movement within the area of interest. The emulator is a game in which a human avatar can walk inside a grid layout area of static objects. This is presented in Fig. 7, while the script flow of the implementation is presented in Fig. 8. Within the client, the following processes occur: OpenCV was used to capture images on the field of view and location of the user. The *hashcode* of images was then created and converted to *base64* string. This frame was sent to the server

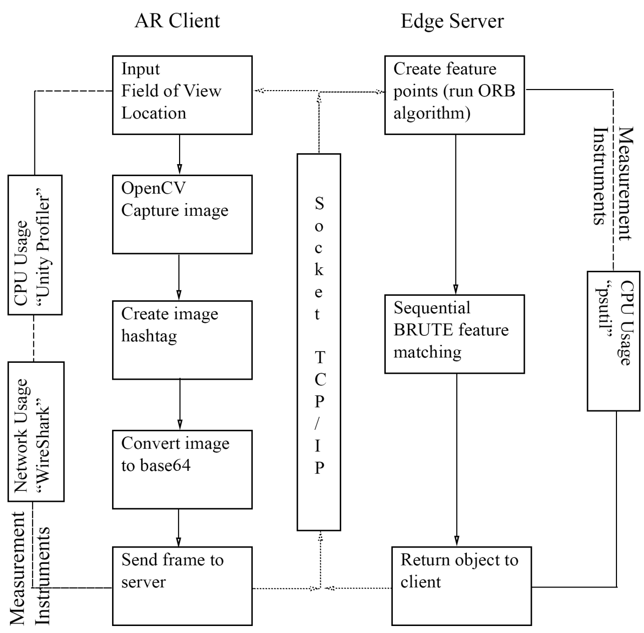


FIGURE 8. Script flow for the implementation of the measurement procedure.

using the socket. To capture the network in-flow to the client and out-flow data from the client, the client was integrated to a third-party application called *Wireshark*. The sent and received packets were captured according to the IP address of the client and the server. The I/O Graph was used to record traffic per second. CPU measurements were captured with the *Unity Profiler* which is a tool used to get performance data for an AR application.

2) SERVER INFORMATION

The server was an edge device with the specifications presented in Table 1. The server received the frame from the client and by using the ORB algorithm, it calculated the feature points. The features points of each frame were then compared in a sequential manner to the features of cached images in the cache, and feature matching was performed with the BRUTE FORCE algorithm.

The best-fit image on the database was returned to the client through the Socket. All processes were performed in the form of a *Python* script and *OpenCV* libraries and algorithms. In *RaspberryPi* we used a *Python* script, named *"psutil"* library to capture CPU usage; this is a cross-platform library for retrieving information on running processes and system utilization.

B. IMPLEMENTATION CHALLENGES

The proposed approaches have some challenges associated with their implementation. For best performance, the caching algorithm requires information about the user and object locations. This is important in order to create the clusters, as indicated in Fig. 5. The clusters are groups of images that exist in nearby distance, and they are used to further reduce the time consumed in the sequential search process. There

TABLE 1. Edge server specifications.

Field	Description
Name	Raspberry Pi 3
SoC	Broadcom BCM 2837
CPU	4x ARM Cortex-A53, 1.2GHz
GPU	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900MHz)
Network	10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage	microSD
GPIO	40-pin header, populated
Ports	HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

are various ways to obtain knowledge about user location: one is through an indoor localization system; another one is to have the 3D scene available in the AR application and to have spatial mapping and Simultaneous Localization and Mapping (SLAM) as possible mechanisms to find the user location.

The locations of objects are usually known since store owners always keep a log of product position and quantities. If this information is not available, product locations can be found with specialized cameras and SLAM algorithms deployed on the scene. An alternative approach is to rely on participatory sensing from users. Assuming that a training period is available, during which the AR apps of users feed the system with frames of object images that can be classified in the backend database, the position of objects can then be computed. In that case, the AR application provider is only required to register objects in the database.

VI. RESULTS

A user may navigate in the aisles and she looks at products through a headset rotation. We modeled 1,000 different objects, each represented by a different image. Each object has a specific location in space and a significance probability.

A. IMAGE CACHING ALGORITHM

The results for the caching algorithm are presented in Fig. 9, Fig. 10 and Fig. 11. Measurements concern the edge server sincethis is the entity that executes object recognition. For our experimental investigation, we measured and compared the performance of the three cases below.

Case 1 assumes no caching. This means that, as the user moves in space, the uplink frame is compared to all registered objects in the database. When a match is found, the process ends. This case is expected to have the highest computational costs and is used as a reference for comparison.

Case 2 models the scenario in which the user location is given as an input to the cache database search procedure, and the sequential search is limited to those items among the set of cached ones with distances smaller than the threshold δ_d. Case 2 is a heuristic version of a caching that is used to help quantify potential savings. Finally, *Case 3* is the proposed

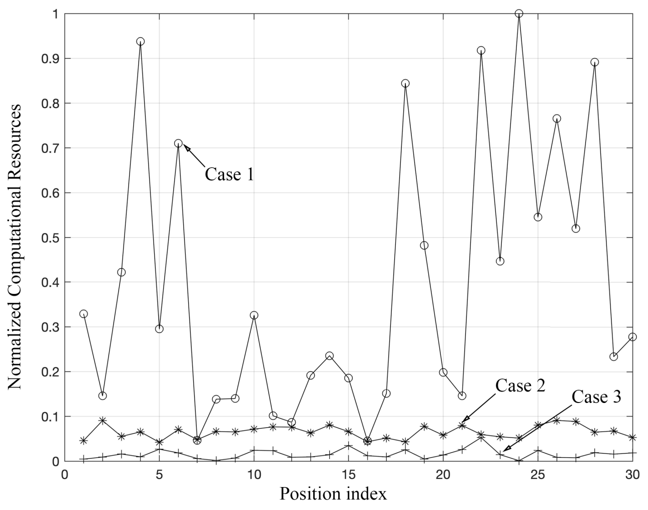


FIGURE 9. Comparison of computation resources for the three use cases.

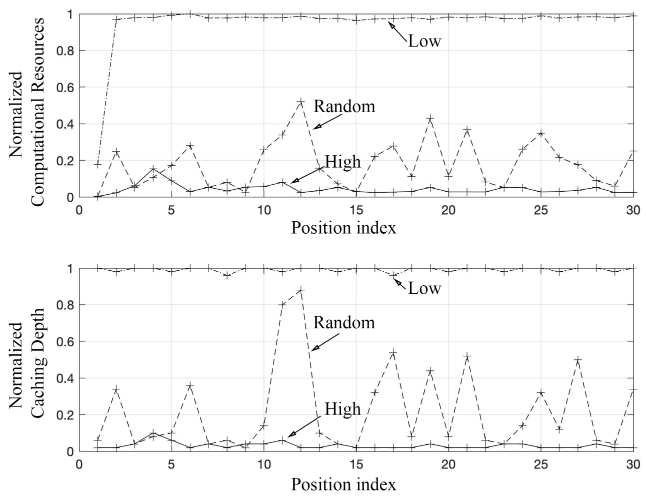


FIGURE 10. Computation resources and caching depth for Case 3 and for 3 different user profiles.

caching method, where objects in the cache database are sorted and cached according to the metric defined in (8).

The results are presented as normalized values compared to the worst-case scenario of Case 1. In Fig. 9, it is observed that for Case 1, the required computational resources, modeled by the CPU and execution time, are much larger than those compared to Cases 2 and 3. The reduction of the amount of computational resources for Case 3 is substantial, and this is expected since ranking objects according to the significance probability and the other metrics provides the local cache with accurate results, without the need for in- depth search in the database. Case 3 showed an improvement of 96% compared to Case 1 and 17% compared to Case 2 according to the mean values. There are occasions when Case 1 required almost the same amount of computation resources with those in the caching approach. This is because it is still possible that feature matching occurs early in the search. Of course, such a likelihood is low.

Different user viewing behavior profiles: Fig 10 presents the simulation results only for Case 3 and for three different user profiles. The *High* user profile models a user that was moving in space and observing objects with angles that follow the pattern of the modeled visual attention. The *Low* user profile models a user that looks at a direction opposite to the one of the modeled visual attention, and it was used as a reference point since it is the worst-case scenario. Finally, the *Random* profile assumed random user observations $\varphi(t)$.

The purpose of this measurement campaign is to observe the effect of user observation profile on the cache search depth and the needed computational resources. The cache search depth is the normalized minimum index of the local cache that created an accurate result. This is computed in normalized fashion as $depth = j/|C|$ and takes values between 0 and 1, where $|C|$ is the size of the image set in the cache and j is the index of the first element in the cache database where feature matching occurs, meaning that $s_{ij}>\theta$. It is observed that for the *High* user profile, the caching index is very small, namely close to 0. This means that the sequential search of the cache immediately returned a correct match after few parsed objects. This is expected because the user sends in the uplink a frame image that is similar to the one cached in the database, due to the applied popularity index. On the other hand, the *Low* user profile requires the largest amount of resources, since the uplink frame is not easy to match. For that reason, the cache search depth and computation resources are very large and close to 1, implying that feature matches occurred mostly towards the last elements of the database. The *Random* profile yields results that fall in between the two extreme cases above, as expected. The average cache search depth of the High user was equal to 3% of the database, whereas for the Random it was 23.1%, and for the Low user profile, it was 98.2%. The higher the cache search depth, the largest the probability that a frame needs to be sent to cloud for object recognition.

Finally, Fig. 11 presents a comparison of the total execution time versus number of users that require access to the AR service. It is shown that there exists an almost linear increase of the execution time as expected, but in Case 3 the total delay was reduced.

B. SPATIAL FILTERING METHOD

This section describes the client-side measurement results for the proposed spatial filtering algorithm. We focus on the client side, since for non-continuous AR applications, the client is responsible for finding the appropriate time to initiate object recognition, based on *a priori* information such as the targeted object and user current location.

Three cases are again considered: *Case A* assumes no spatial filtering so that the client always sends frames to the server, similarly to the continuous AR case. Following the notation (9), *Case 2* assumes that the client will only start the object recognition process ($Q = 1$) only when the distance is smaller than threshold δ_d. Finally, *Case 3* is a full implementation of (9) and $Q = 1$ when distance is larger than δ_d and $\varphi\text{-}\omega/2<\varphi<\varphi + \omega/2$. Results are presented in Fig. 12. The user is assumed to walk in an aisle, and

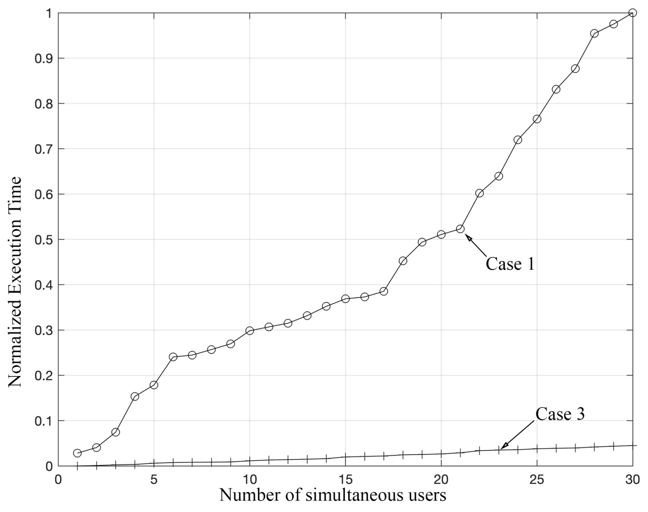


FIGURE 11. Execution time vs. active requests for the AR service.

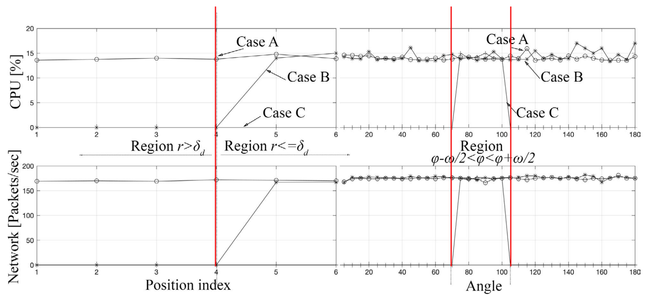


FIGURE 12. CPU utilization and total network traffic (in and out) for the client. Simulations concern the spatial filtering algorithm.

at position index 4 the user enters the region $r<\delta_d$. The network traffic and CPU utilization increased for Case B, as expected. Up to this point, the average CPU and the number of transmitted/received packets of Case 1 was 13.9% and 170 packets/sec respectively, whereas for Case 2 they were equal to 4.9% and 55.9packets/sec respectively. Clearly, for Case 1 it was 0% and 0 packets/sec, since the object recognition process was not triggered. At position index 6, the user is assumed to be in front of the targeted object and rotates from 0^o to 180^o degrees. The location of the targeted object was at 90^o. It can be seen that for Case 3 the network traffic and the CPU utilization of the client increased only when the field of view and the orientation of the user covers the object location. The average CPU utilization and number of transmitted/received packets for Case 1 was 14.2% and 174.1packets/sec respectively, whereas for Case 2 these are equal to 14.9% and 174.2 packets/sec. For Case 1, the average CPU and packets were 2.4% and 19.1 packets/sec. Cases 1 and 2 give similar results since in both cases object recognition is executed all the time because the user is within the region $r<\delta_d$.

VII. CONCLUSION

With the forthcoming massive wave of commercialization of Augmented Reality (AR) devices at retail level and the development of AR services, edge computing will be key towards providing the holy grail of low-latency user experience. Towards this goal, applications should be designed so as to optimize computational and communication costs. In this work we started from a practical use case scenario, that of a grocery store, where users move with their smartphones and continuously generate object recognition requests as they seek holographic information about grocery products. We presented an image caching algorithm that helps an edge server of an AR service provider reduce computational load by taking into account the user visual attention to objects. This was modeled through an object significance metric that captures expected user visual attention, and it guides the population of the local cache and the sorting of its elements (objects) to achieve an accurate match. Besides object popularity, cache parsing delay and label accuracy where also taken into account in ranking objects to cache. It was experimentally shown that the proposed image caching approach can reduce the server computational load by a factor larger than 90%. Furthermore, the network traffic from the AR client to the server was reduced through a filter-like method that initiates object recognition only when needed, namely when the object location falls within the client FoV. Finally, a proof-of-concept validation was presented. As a future step to this research, we plan to make a real deployment using "real-life" evaluation on a handheld mobile device in a real small-scale grocery store.

REFERENCES

[1] Y. Yuan, "Paving the road for virtual and augmented reality [standards]," *IEEE Consum. Electron. Mag.*, vol. 7, no. 1, pp. 117–128, Jan. 2018.

[2] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[3] G. Koutitas, S. Smith, G. Lawrence, and K. Noble, "Smart responders for smart cities: A VR/AR training approach for next generation first responders," in *Smart Cities in Application*. Cham, Switzerland: Springer, 2020.

[4] G. Koutitas, J. Jabez, C. Grohman, C. Radhakrishna, V. Siddaraju, and S. Jadon, "XReality research lab—Augmented reality meets Internet of Things," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 1–2.

[5] Q. Report, *Augmented and Virtual Reality: The First Wave of 5G Killer Apps*. New York, NY, USA: ABIResearch, 2017.

[6] M. Rank, Z. Shi, and S. Hirche, "Perception of delay in haptic telepresence systems," *Presence, Teleoperators Virtual Environ.*, vol. 19, no. 5, pp. 389–399, Oct. 2010.

[7] W. Zhang, B. Han, and P. Hui, "On the networking challenges of mobile augmented reality," in *Proc. Workshop Virtual Reality Augmented Reality Netw. - VR/AR Netw.*, 2017, pp. 24–29.

[8] A. Destounis, G. S. Paschos, and I. Koutsopoulos, "Streaming big data meets backpressure in distributed network computation," in *Proc. IEEE INFOCOM 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[9] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, "Zenith: Utility-aware resource allocation for edge computing," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jun. 2017, pp. 47–54.

[10] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[11] H. Feng, J. Llorca, A. M. Tulino, and A. F. Molisch, "On the delivery of augmented information services over wireless computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7.

[12] C. Bermejo, Z. Huang, T. Braud, and P. Hui, "When augmented reality meets big data," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jun. 2017, pp. 169–174.

[13] J. Chakareski, "VR/AR immersive communication: Caching, edge computing, and transmission trade-offs," in *Proc. VR/AR Netw. Workshop Virtual Reality Augmented Reality Netw.*, 2017, pp. 36–41.

[14] S. Ioannidis and E. Yeh, "Jointly Optimal Routing and Caching for Arbitrary Network Topologies," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 6, pp. 1258–1275, 2018.

[15] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Hold'em caching: Proactive retention-aware caching with multi-path routing for wireless edge networks," in *Proc. 18th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2017.

[16] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Caching-aware recommendations: Nudging user preferences towards better caching performance," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[17] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1107–1115.

[18] L. E. Chatzieleftheriou, G. Iosifidis, I. Koutsopoulos, and D. Leith, "Towards resource-efficient wireless edge analytics for mobile augmented reality applications," in *Proc. 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[19] U. Drolia, K. Guo, and P. Narasimhan, "Precog: Prefetching for image recognition applications at the edge," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[20] L. N. Huynh, Y. Lee, and R. K. Balan, "DeepMon: Mobile GPU-based deep learning framework for continuous vision applications," in *Proc. 15th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2017, pp. 82–95.

[21] M. Xu, M. Zhu, Y. Liu, F. X. Lin, and X. Liu, "DeepCache: Principled cache for mobile deep vision," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw. - MobiCom*, 2018, pp. 129–144.

[22] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *Proc. Int. Conf. Comput., Math. Eng. Technol. (iCoMET)*, Mar. 2018, pp. 1–10.

[23] I. Stulec, K. Petljak, and A. Kukor, "The role of store layout and visual merchandising in food retailing," *Eur. J. Econ. Bus. Stud.*, vol. 4, p. 331, Jan. 2016.

[24] Eyeware. *Predictive Retail Analytics: How to Use Data to Understand Shopper Behavior and Grow Sales*. Blog Post. Accessed: 2019. [Online]. Available: https://eyeware.tech/blog/how-to-use-retail-analytics/.

[25] *OpenCV Documentation*. Accessed: Nov. 1, 2020. [Online]. Available: www.opencv.org

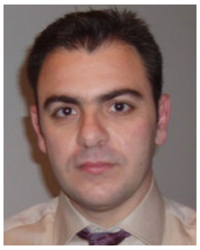
[26] *DEMO Concept Video*. Accessed: Nov. 1, 2020. [Online]. Available: https://youtu.be/e3rlXmHNUio and [Online]. Available: https://xreality.wp.txstate.edu/demos/

GEORGE KOUTITAS (Member, IEEE) received the B.Sc. degree in physics from the Aristotle University of Thessaloniki, Greece, and the M.Sc. (Hons.) and Ph.D. degrees (EPSRC Scholarship) from the University of Surrey, U.K. He is currently an Academician and an Entrepreneur in electrical and computer engineering. He is also an Assistant Professor with the Ingram School of Engineering, Texas State University, and the Director of the XReality Research Laboratory. He has published more than 58 scientific articles that have received more than 1200 citations. He has founded two startup companies in Austin, TX, in the areas of customer engagement and virtual/augmented reality. Through his startup journey, he created digital platforms and helped large corporations implement digital strategies in outdated sectors, such as electric utilities and medical response. His main research interests include wireless communications, augmented and virtual reality, and the Internet of Things. During his studies, he received the "Nokia Prize" for the best overall performance.

SHASHWAT VYAS is currently pursuing the bachelor's degree in computer science with Texas State University, TX, USA. He has been an Active Software Developer with specialization in augmented reality at the XReality Research Laboratory, Texas State University. He has been a part of interdisciplinary research teams working for projects in the areas of health, the Internet of Things, networking, and industry 4.0.

CHAITANYA VYAS is currently pursuing the Bachelor of Science degree in computer science with Texas State University, TX, USA. He is currently working as an Augmented Reality Developer with the X-Reality Laboratory, Texas State University. He has been involved as an Active Researcher in multidisciplinary projects that involve industry 4.0, medical response, and the Internet of Things.

SHIVESH SINGH JADON is currently pursuing the bachelor's degree in computer science with Texas State University, TX, USA. He has been an Active Researcher with the XReality Research Laboratory, Texas State University. He has worked in interdisciplinary projects that cover digital twin, the Internet of Things, and first responders. His research interests include extended reality, human–computer interaction, and visualization.



IORDANIS KOUTSOPOULOS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Greece, in 1997, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park (UMCP), MD, USA, in 1999 and 2002, respectively. He was an Assistant Professor with the Athens University of Economics and Business (AUEB), Athens, Greece, from 2013 to 2016, where he has been an Associate Professor with the Department of Informatics, since 2016. He was a Lecturer and an Assistant Professor with the Department of Electrical and Computer Engineering, University of Thessaly, from 2005 to 2010 and 2010 to 2013, respectively. His research interests include control and optimization and on applications of machine learning, with application areas, such as mobile crowdsensing, wireless networks, social networks, recommender systems, smart energy grid, and cloud computing systems. He received the Fulbright Scholarship from 1997 to 2003, the Marie Curie Fellowship from 2005 to 2007, the Single-Investigator European Research Council (ERC) Competition Runner-Up Award for the project RECITAL: Resource Management for Self-Coordinated Autonomic Wireless Networks from 2012 to 2015, and three Best Paper Awards for research on online advertising, network economics, and network experimentation.