Invited Paper: Edge-based Provisioning of Holographic Content for Contextual and Personalized Augmented Reality

Michael Glushakov*, Yunfan Zhang*, Yuqi Han†, Timothy James Scargill*, Guohao Lan*, Maria Gorlatova*
*Duke University, Durham, NC, †Tongji University, Shanghai, China
{mykhaylo.glushakov, yunfan.zhang, yuqi.han, timothyjames.scargill, guohao.lan, maria.gorlatova}@duke.edu

***Abstract*—Mobile augmented reality (AR) has been attracting considerable attention from industry and academia due to its potential to provide vibrant immersive experiences that seamlessly blend physical and virtual worlds. In this paper we focus on creating contextual and personalized AR experiences via edge-based on-demand provisioning of holographic content most appropriate for the conditions and/or most matching user interests. We present edge-based hologram provisioning and pre-provisioning frameworks we developed for Google ARCore and Magic Leap One AR experiences, and describe open challenges and research directions associated with this approach to holographic content storage and transfer. The code we have developed for this paper is available online.**

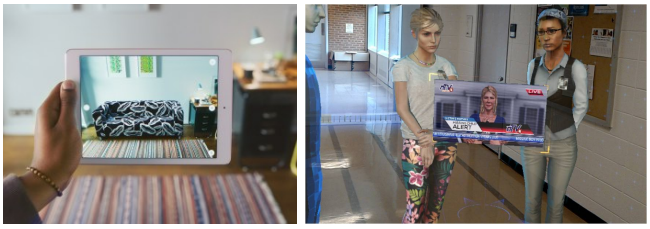
***Index Terms*—Mobile Augmented Reality, Edge Computing, Edge Caching, Google ARCore, Magic Leap.**

I. Introduction

Mobile augmented reality (AR), which integrates virtual objects with 3-D real environments in real time [1], has been in development since late 1960s [2], [3], [4], but has reached the consumers only recently, with the release of several commercial AR headsets [5], [6] and software development kits [7], [8]. Examples of modern AR experiences are shown in Figure 1: Figure 1(a) shows a tablet-based example of holographic furniture placed in a real room, and Figure 1(b) shows a head-mounted display-based holographic gaming experience generated in a corridor outside our lab. Deeply immersive, AR shows great potential in an array of areas including education [9], medicine [10], and retail [11]. In January 2020, the founder and CEO of Facebook Mark Zuckerberg said that he expects AR to "redefine our relationship with technology" [12], while Apple CEO Tim Cook said that "AR is the next big thing, and it will pervade our entire lives" [13].

Modern AR, however, has multiple limitations, including high energy consumption and difficulty in mapping dynamic scenes. Edge or fog computing, the distribution of computing resources closer to the end users, shows promise for overcoming current AR limitations and enabling additional AR capabilities [14]. *Multiple opportunities associated with edge-supported AR are not present in current, stand-alone or cloud-supported AR*: for instance, low-latency connections to computationally capable nodes, access to stationary persistent local sensors, and ability to create advanced adaptive experiences that leverage locally available data.

In this work we propose to use edge computing to deliver *contextual and personalized augmented reality experiences* via *edge-based hologram provisioning*. This work is motivated by the needs of AR applications we are currently developing to enhance the experience of visitors to Duke Lemur Center, which houses nearly 240 rare and endangered prosimian primates [15]. Visitors to the Lemur Center observe the animals in different outdoor and indoor enclosures. We will enhance the experience of the visitors by providing additional information about the lemurs via *personalized contextual holograms*. Possible AR contents include visualizations and vocalizations of specific lemur species according to the exhibit the user is located at, augmentations of other Madagascan flora and fauna to create a more realistic view of habitats, and other educational material such as interactive maps. Examples of AR experiences we are currently creating are shown in Figure 2.



(a) IKEA Place app [11] for placing virtual furniture in a physical room.

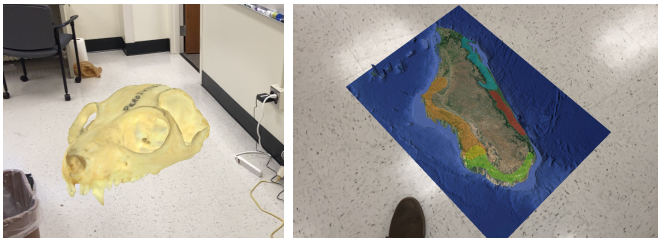
(b) A virtual scene placed into a real corridor outside our lab.

Fig. 1. Mobile augmented reality (AR) examples, generated with: (a) a tablet, and (b) a Microsoft HoloLens head-mounted display.

In our envisioned applications, personalization could be made based on both environmental context (e.g., weather, time of day) and user profile or preferences (e.g., current location, previously viewed content, age, native language). For example, we can imagine that researchers will be presented by default with more scientific content such as lemur skeletal structures and taxonomic classification, while children will be guided and informed via an animated helper. Depending on the weather, that character will be behaving differently, finding shade under trees for example. Holograms and audio of other fauna will be dependent on time of day, to represent the species active at that time. First-time visitors may be presented with more introductory and navigational information. As users proceed through the setting, their preferences regarding hologram content (i.e., what they previously engaged with) and position should be learned to provide them with an optimal experience going forward.

To this end, we design and develop an edge computing-

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.



(a) (b)

Fig. 2. Augmented reality applications we are currently creating to enhance the experience of visitors to Duke Lemur Center [15]: (a) a 3D model of a lemur skull, and (b) a virtual map of Madagascar, highlighting habitats of different lemur species.

based architecture that supports personalized and contextual AR. In our architecture, holograms are stored on the edge server, rather than on a mobile device, and are transmitted to the mobile devices when necessary. In Sections III and IV we present a *mobile phone-based* Google ARCore [7] and a *headset-based* Magic Leap One [6] case studies for our architecture. The Google ARCore case study focuses on comparing edge-based and cloud-based on-demand hologram provisioning, and on exploring the most suitable file formats for Google ARCore architectures. The Magic Leap case study demonstrates an approach that involves Unity [16], one of the most popular platforms for creating 3D experiences. We assume that it is more common for headset-based experiences, that are more immersive and allow users to better see the details of the holograms, to require larger holograms, which can result in substantial latency when provisioned directly on user's request. Correspondingly, in addition to considering on-demand hologram provisioning, we also develop an approach for *proactive hologram transmissions*, where holograms are transmitted to the users ahead of when they are needed. In Section V we discuss challenges and directions associated with the on-demand and proactive transmissions of edge-based holographic content.

In summary, our contributions are as follows:

- We propose an architecture for personalizing AR experiences by using edge servers to transmit appropriate holographic content to the users.
- We demonstrate the key components of the proposed architecture in Google ARCore and Magic Leap One case studies. Our case studies highlight advantages of the proposed edge-based architecture. The case studies also suggest the best file format to use for Google ARCore on-demand hologram transmissions, and showcase a Magic Leap geolocation-based proactive hologram transmission approach.
- Towards fully realizing our proposed architecture, we identify and discuss several challenges and research directions associated with edge-based on-demand and proactive hologram transmissions.

The Google ARCore case study implementation code is available at [17]. The Magic Leap case study implementation code is available at [18].

II. Background and Related Work

Context-aware and personalized AR: Multiple context-aware and personalized AR applications have been proposed. For instance, AR experiences can be adapted to user profiles, gaze, and physiological states of the user [19]. Researchers that develop context-aware AR applications generally focus on advanced AR application capabilities and user experiences, rather than on mechanisms for obtaining appropriate holographic content: in most prototypes, the content is simply pre-stored in the AR app itself. While some contextual AR adaptations do not require loading new content (e.g., changing text labels; scaling, rotating, or moving around [20], [21] holograms; changing the color or transparency [22] of a hologram; animating holograms), many envisioned context-aware applications benefit from displaying different holograms under different environmental and user conditions.

Edge computing and edge caching: It is well-established that edge computing can help AR [14]. Multiple lines of work have examined the use of edge computing to aid AR object recognition [23], [24], [25], video rendering and encoding [26], and simultaneous localization and mapping [27]. The idea of edge-based content caching has also been proposed; research on it includes examinations of edge-based data repositories [28], edge computing-specific wireless data transmission techniques [29], and caching policy optimizations [30]. To the best of our knowledge, we are the first to develop and examine edge-based holographic content loading frameworks for common AR platforms.

On-demand loading of holographic content: Runtime loading of content (3D models of characters and objects, texture files, sound effects) is a known technique in computer game development [31]. For instance, in a popular Unity game development engine [16], runtime loading of content can be achieved through the creation of "Asset Bundles" [32]. Dynamic content loading can be used to reduce the size of initial game installation file and to provide rarely accessed features only when they are required. On-demand loading of holographic content is used in browser-based mobile AR ("Web AR" [33]), which has recently been under active development. In this paper we explore providing runtime-loadable holographic content via edge servers, and discuss associated challenges, constraints, and research directions specific to AR and edge computing.

III. Google ARCore Case Study

In this section, we demonstrate the proposed architecture in a Google ARCore platform case study. We focus on comparing edge-based and cloud-based on-demand hologram provisioning, and on exploring the most suitable file formats for Google ARCore hologram provisioning architectures. The code we developed for this case study is available at [17].

A. Architecture

We created a Google ARCore-based architecture where an edge server decides which 3D models to return to a mobile phone based on pre-set contextual information, such as the

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.

weather, time of day, user's native language, or history of prior user interest in specific holograms. To enable this functionality, we are relying on Google ARCore runtime 3D model loading capability for different file formats, which has been a part of ARCore starting at version 1.5 released in September 2018.[1] In our prototype implementation, to display holograms on the screen of a mobile device, AR application use the Augmented Images functionality of ARCore: specifically, the holographic models are displayed when a pre-selected image is recognized in the environment [35].

Creating edge-based contextual AR applications: A content creator, such as an artist, a museum curator, or an educational director of a wildlife center, can create an account and upload images, which will be used as *anchors* in the Augmented Images ARCore experience, to the server. The content creator can then choose an appropriate *model determination script* that looks at contextual parameters such as the weather, time of day, or language of the user to determine which 3D model to return when the anchor image is recognized in the environment. Each model determination script has a set number of possible conditions, for which separate 3D models need to be provided. The model determination script used in our experiments determines the weather and the time of day at the location of the server, and uses that information to choose between 4 conditions: "Clear-Light", "Clear-Dark", "Rain-Light", and "Rain-Dark". The script can be easily modified to consider other contextual parameters instead. We note that *this architecture readily allows for the development of AR experiences by content creators with no coding knowledge*, since changing the 3D content provided to the users only requires replacing images that are used to display holograms, or replacing the holographic models themselves.

AR app initialization: A mobile phone user can use the unique clientId assigned to each content creator to first load a JSON array with each entry containing the URL to the image file and the ID of the 3D model that the API refers to as "RenderableId". The AR application then downloads all of the image files whose URLs are contained in the JSON array via HTTP GET method.

On-demand hologram transmissions: Once all images are loaded, the application launches the AR activity which uses the ARCore Augmented Images API to track the downloaded images. When one of the images is recognized, the application sends an HTTP GET request with the RenderableId matching that image. The server then calls the model determination script to determine which condition is currently met, and returns the URL of the corresponding 3D model. The client application loads the model and places it over the image.

B. Performance evaluation

Our performance evaluation centered on assessing hologram transfer latencies for edge-based and cloud-based holographic content storage options. We also compared the performance of hologram transfer techniques for two different file types, specifically the GLB and the SFB 3D model file formats, which both contain the 3D model and the corresponding textures. The GLB is a binary form of the GL Transmission Format (glTF), developed to be an interoperable format with a minimal file size. The Sceneform Binary asset (SFB) is an Android-native 3D model format. For the same 3D model, an SFB file can potentially be rendered faster than the GLB counterpart, which needs to undergo extra processing.

TABLE I
3D MODELS USED IN GOOGLE ARCORE CASE STUDY

Model Name	File Type	Size
Duck	GLB	120 KB
	SFB	238 KB
Cybertruck	GLB	321 KB
	SFB	646 KB
Tower	GLB	19.1 MB
	SFB	19.4 MB

TABLE II
GOOGLE ARCORE CASE STUDY FILE TRANSFER TIMES

3D model	Phone	File Type	Avg. Latency (ms)	
			Edge	Cloud
Duck	Nokia 7.1	GLB	27.6	70.16
		SFB	65.5	108.8
	Pixel 3	GLB	19.7	107.1
		SFB	34.3	149.9
Cybertruck	Nokia 7.1	GLB	65.5	100.4
		SFB	102.7	206.5
	Pixel 3	GLB	40.7	121.1
		SFB	114.6	142.8
Tower	Nokia 7.1	GLB	2,239.0	2,795.4
		SFB	2,087.5	2,393.4
	Pixel 3	GLB	1,488.7	1,218.3
		SFB	1,814.8	1,485.3

Our experiments were conducted for three different 3D models of a range of sizes, with two different mobile phones, a mid-range one (Nokia 7.1) and a higher-end one (Google Pixel 3). As a cloud server, we used a cloud platform-as-a-service (PaaS) Heroku platform [36]. As an edge server, we used a server connected to our local network, with a 10 ms RTT and 50 Mbps link to the local network. There was a light to modest amount of background traffic in our local network during the experiments.

The 3D models we used are listed in Table I. As can be seen in this table, for lightweight 3D models, a GLB file is smaller than the SFB file. Specifically, for the smallest "Duck" model, the SFB/GLB file size difference is 1.98x, and for the slightly larger "Cybertruck" model, the SFB/GLB file size difference is 2x. For larger models, GLB files can be only slightly smaller than SFB files. In our experiments, for the largest "Tower" model, the GLB file is 0.3 MB smaller, which translates to the percentage difference of 1.5%.

Table II shows the file transfer latencies we observed when transferring the 3D models from edge and cloud servers, for different 3D models, mobile devices, and file types. Table III shows the latencies we observed when loading and displaying a hologram after it was received by the mobile device. Each of the experiments presented in Tables II and III has been repeated 10 or more times. As expected, we observe that smaller models ("Duck", "Cybertruck") can be transferred to

[1]Our current architecture supports runtime loading of static 3D models only. The architecture can be extended to support the loading of animated models as well, which has been possible on Google ARCore starting at version 1.7 [34].

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.

TABLE III
FILE STORAGE LOAD AND PROCESSING TIMES

3D model	Phone	File Type	Avg. Latency (ms)
Duck	Nokia 7.1	GLB	171.0
		SFB	40.8
	Pixel 3	GLB	94.2
		SFB	39.1
Cybertruck	Nokia 7.1	GLB	370.1
		SFB	66.0
	Pixel 3	GLB	203.7
		SFB	55.6
Tower	Nokia 7.1	GLB	2,425.3
		SFB	605.6
	Pixel 3	GLB	1,079.6
		SFB	344.2

the mobile device faster from the edge server than from the cloud server. On average, cloud hologram transfer times are slower by 67 ms, with the minimum difference of 28 ms for the "Cybertruck" model transferred to a Pixel 3 phone in an SFB format, and the maximum difference of 115 ms for the "Duck" model transferred to a Pixel 3 phone in an SFB format. For the large "Tower" model, we observed that edge hologram transfer latencies were in some cases higher than cloud transfer latencies, which we attribute to considerable bandwidth variations on our internal on-campus network that we used to access the edge server (bandwidth variations would affect large file transfers more than small file transfers). Given that latencies above 100 ms are perceptible to the users [37], we believe that making smaller holograms available via edge, rather than cloud, servers, will improve user experience with on-demand hologram provisioning architectures.

We also observed that, while the smaller size of GLB files reduces file transfer times, the extra time spent converting the file to be Sceneform-compatible at runtime is greater than the time saved on transferring the file. For example, while our smallest "Duck" 3D model on a Nokia 7.1 mobile phone is transferred from an edge server 38 ms faster when stored in the GLB format (see Table II), it loads 130 ms faster when stored in the SFB format (see Table III). Our largest "Tower" 3D model on a Pixel 3 phone is transferred 326 ms faster when stored in the GLB format, but loads 735 ms faster when stored in the SFB format. This indicates that *the extra processing time required for the GLB files outweighs the time saved on loading by using a smaller file format*, independent of the model size or the capabilities of the phone running the AR application. Thus, for runtime edge-based Google ARCore hologram provisioning, it is more efficient to convert holographic content to larger Android-native SFB files ahead of time than to store the models on the edge in smaller GLB files.

IV. MAGIC LEAP ONE CASE STUDY

In this section, we describe a Magic Leap One case study that examined *on-demand* hologram transmissions and location-aware *proactive* hologram transmissions. The code we developed for this case study is available at [18]

TABLE IV
3D MODELS USED IN MAGIC LEAP ONE CASE STUDY

Model Name	Triangle Count	Size	
		W/out Compression	W. Compression
Stanford Bunny	5K	7.4 MB	1.4 MB
Lucy	33K	10.6 MB	1.7 MB
Asian Dragon	87K	79 MB	7.7 MB

A. On-Demand Hologram Transmissions

We experimented with runtime hologram loading with Magic Leap One [6], one of the prominent commercial AR headsets in today's market. To match what developers would commonly use when creating headset-based AR experiences, we conducted our experiments on the Unity [16] APIs offered by the Magic Leap One platform. The 3D models that we used in the experiments were designed to match the common model qualities in modern AR gaming experiences. The three 3D models we selected from the Stanford 3D Scanning Repository [38] are summarized in Table IV. The models were in .obj format and had triangle counts of 5K, 33K, and 87K to emulate low, medium, and high holographic model quality. The textures used for the models had 2K by 2K resolution and were encoded in PNG format. We used the AssetBundle [32] feature built into Unity to compress 3D models and export them into a Unity-proprietary binary format for runtime hologram loading. The compression algorithm we used was LZMA. During compression, we did not modify the 3D meshes or the textures, so the quality of the models remained unchanged. We then stored the exported 3D models on an edge server with eight-hop distance, 10 ms RTT, and 50 Mbps bandwidth to our local network. The Magic Leap was connected to our local network through 5GHz WiFi. There was a light to modest amount of background traffic in our local network during the experiments.

We considered two types of hologram placement strategies in our experiments. The first type of hologram was placed at a fixed location and was programmed to appear at a specific time. In other words, the placement of the holograms was predetermined and independent from user interactions. We designed this type of strategy to simulate static holograms such as traffic signs and billboards. The other type of holograms were triggered by user activity: these holograms would appear once the user encountered a specific object in the environment, for example, a specific picture. This type of holograms simulates personalized and context-based holograms. In both cases, the 3D models were transmitted from the server to the device using standard HTTP protocol. We then performed runtime loading for both types of models, and observed its impact on the user experience. A summary of the model loading times we observed is presented in Table V.

For our 3D models with triangle counts of 5K, 33K, and 87K and a texture resolution of 2K by 2K, our application, on average, took 1,702, 1,715, and 3,325 ms respectively for the models to finish downloading. It would, on average, take another 267, 372, and 175 ms to decompress and parse the models before the holograms were presented to the user. We observed that the actual hologram transfer speed achieved

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.

TABLE V
MODEL LOADING TIMES ON MAGIC LEAP ONE

Model Name	Transfer Time	Decompression Time
Stanford Bunny	1,702 ms	267 ms
Lucy	1,715 ms	372 ms
Asian Dragon	3,325 ms	175 ms

with the Magic Leap One was slower compared with the 50 Mbps that was achievable through other devices (e.g., when using a laptop), which suggests some deficiency in Magic Leap's network stack design, such as its TCP congestion control algorithm. *Our subjective user experience suggests that the latency of hologram runtime loading is more perceptible on activity-triggered context-based holograms compared with fixed holograms*. It would usually take a few seconds for users to familiarize themselves with a new physical environment before starting to look for holograms, which would naturally conceal the late appearance of fixed holograms. However, for context-based holograms, the user would be expecting the holograms right away, and the latency caused by runtime loading would be more readily perceptible.

B. Proactive Hologram Transmissions

To make hologram transfer times imperceptible to the users, we experimented with *proactive hologram transmissions* based on AR device geolocations. Although Magic Leap One supports tracking the user location with a relative 3D coordinate system, it does not provide the absolute device coordinate in the physical world. It also lacks the API to recognize the surrounding physical space, such as a specific building or office space. The lack of these capabilities imposes a challenge on location-aware proactive loading of 3D models, since it is difficult to gain awareness of the precise location of the user. Therefore, we designed our proactive hologram transmission approach based on *third party WiFi sniffing*. We placed a Raspberry Pi in the room where the holographic 3D models were intended to be displayed. Once the Raspberry Pi detected the presence of the Magic Leap by sniffing its MAC address, it issued a notification to the edge server; the edge server would then push the corresponding 3D models to the Magic Leap One through HTTP long-polling. This way, we can preload 3D models onto Magic Leap One before the models are needed, thus hiding the hologram transfer latency from the users. Since all hologram transfer latencies in our case study exceed 1.5 s (see Table V), which is readily noticeable by the users, proactive hologram transfer has the potential to significantly improve user experience, especially for dynamic context-based holograms.

V. CHALLENGES AND DIRECTIONS

In this section, we discuss challenges associated with on-demand and proactive transmissions for edge-based holographic content provisioning.

A. On-demand Transmissions

In context-aware AR applications, edge server transmits holographic 3D models to mobile devices on demand. To ensure seamless user-hologram interactions, a low transmission latency is usually required. In practice, the latency is affected by a number of parameters, i.e., the background network traffic load, the number of provisioning demands received by the edge server, and the wireless connection between the mobile device and the edge server. These parameters are not always ideal and are not always under control. Inevitably, users will be subjected to high transmission latencies when the network is congested or when the server is in high demand.

Scalable hologram transmissions: One research direction is to design dynamic hologram transmission policies that adapt the hologram level of detail to user perception. For mobile AR applications, a hologram can be rendered at a smaller scale when it is displayed at a farther distance from the user, where its details are less perceptible to the users. In addition, in personalized AR applications, users may potentially pay more attention to the holograms that are personalized to them than to the others. Thus, based on the rendering distance and user's preferences, we can selectively omit unnecessary details during 3D model transmission. One solution is to create and store holographic models with different resolutions and scales. For instance, we can adopt the scalable video coding (SVC) [39] principles to encode the original hologram into scales with different levels of detail. During the transmission, the server can choose which hologram variant to transmit based on AR application needs, user preferences, and other user, system, and network parameters.

B. Proactive Transmissions

An edge server may also transmit holograms in advance, as we have demonstrated, for example, in our Magic Leap case study in Section IV-B. By transmitting holograms proactively and caching them on user devices, we can achieve instantaneous hologram rendering with no hologram transmission delay. This approach can also help optimize network bandwidth utilization over time and avoid potential network congestion. However, identifying the right holograms to transmit and cache requires more study.

On-device hologram caching: Due to the limited on-device storage, efficient content caching and eviction policies are required in our architecture. The mobile device needs to make a caching or eviction decision whenever it receives a new hologram. For contextual and personalized AR applications, there are a number of factors to be considered in the policy design, including user preferences, the number of times that the user has previously accessed the hologram, the size of the hologram, and the distance between the user and the hologram.

Location-aware hologram deployments: The likelihood of a user's need for a particular hologram is highly dependant on user's location. Thus, together with the personalized information of the user, the details of users' location, orientation, and direction of motion should also be leveraged by the edge server in making proactive hologram transfer decisions. We can readily obtain coarse-grained estimates of user locations by using either motion sensor-based [40] or WiFi fingerprint-based methods [41]. However, additional research is required

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.

to develop efficient methods that accurately predict not only the overall user position, but also AR device orientation and user gaze direction.

VI. Conclusions

In this paper we proposed and developed edge computing-based architectures for delivering contextual and personalized mobile augmented reality (AR) experiences via on-demand and proactive provisioning of holograms from edge servers. Our case studies, conducted with Google ARCore and Magic Leap One AR platforms, suggested the best file format to use for Android-based on-demand hologram transmissions, and showcased a Magic Leap geolocation-based proactive hologram transmission method. We identify and discuss open challenges and future research directions associated with the proposed edge-based on-demand and proactive hologram transmissions. The code we developed for our case studies is available online.

Acknowledgments

This work was supported in part by the Lord Foundation of North Carolina and by NSF awards CSR-1903136 and CNS-1908051. The models used in the Google ARCore case study are from the Khronos Group [42] and CGTader [43]. The models used in the Magic Leap case study are from the Stanford 3D Scanning Repository [38].

References

[1] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.
[2] I. E. Sutherland, "The ultimate display," *Multimedia: From Wagner to virtual reality*, pp. 506–508, 1965.
[3] D. Schmalstieg and T. Hollerer, *Augmented Reality: Principles and Practice*. Addison-Wesley, 2015.
[4] W. Barfield and T. Caudell, "Military applications of wearable computers and augmented reality," in *Fundamentals of wearable computers and augmented reality*. CRC Press, 2001, pp. 639–662.
[5] Microsoft Inc., "Hololens," 2020, https://www.microsoft.com/en-us/hololens.
[6] Magic Leap, "Magic Leap One," 2020, https://www.magicleap.com/.
[7] Google Inc., "Google ARCore," https://developers.google.com/ar/, 2020.
[8] Apple Inc., "Apple ARKit," https://developer.apple.com/documentation/arkit, 2020.
[9] M. Billinghurst and A. Duenser, "Augmented reality in the classroom," *Computer*, vol. 45, no. 7, pp. 56–63, July 2012.
[10] L. Chen, T. W. Day, W. Tang, and N. W. John, "Recent developments and future challenges in medical mixed reality," in *IEEE ISMAR*, 2017.
[11] A. Pardes, "Ikea's new app flaunts what you'll love most about AR," *Wired*, Sept. 2017, https://www.wired.com/story/ikea-place-ar-kit-augmented-reality/.
[12] G. Sloane, "Mark Zuckerberg puts on his augmented reality goggles to look into the 2020s," *AdAge*, Jan. 2020, https://adage.com/article/digital/mark-zuckerberg-puts-his-augmented-reality-goggles-look-2020s/2226326.
[13] E. Burke, "Tim Cook: 'AR will pervade our entire lives'," *SiliconRepublic*, Jan. 2020, https://www.siliconrepublic.com/machines/tim-cook-ar-war-ducks-healthtech.
[14] J. Chakareski, "NSF visioning workshop on networked virtual and augmented reality communications: The future VR/AR network – towards virtual human/object teleportation," Apr. 2018, http://strugaorg.ipower.com/Jakov/NSF_NetworkedVRAR_Workshop/.
[15] Duke University, "Duke Lemur Center," https://lemur.duke.edu/, 2020.
[16] Unity Technologies, "Unity," 2020, https://unity3d.com.
[17] M. Glushakov, "Git repository: Google ARCore on-demand hologram transmissions," 2020, https://github.com/michaelglu/SmartEdgePaper.
[18] Y. Zhang, "Git repository: Magic Leap on-demand and proactive hologram transmissions," 2020, https://github.com/YunfanZhang42/SmartEdgeMagicLeap.
[19] J. Grubert, T. Langlotz, S. Zollmann, and H. Regenbrecht, "Towards pervasive augmented reality: Context-awareness in augmented reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 6, pp. 1706–1724, 2017.
[20] S. Ahn, M. Gorlatova, P. Naghizadeh, M. Chiang, and P. Mittal, "Adaptive fog-based output security for augmented reality," in *ACM SIGCOMM VR/AR Network Workshop*, 2018.
[21] J. DeChicchis, S. Ahn, and M. Gorlatova, "Demo: Adaptive augmented reality visual output security using reinforcement learning trained policies," in *ACM SenSys*, 2019.
[22] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner, "Securing augmented reality output," in *IEEE S&P*, 2017.
[23] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *IEEE INFOCOM*, 2018.
[24] Z. Liu, G. Lan, J. Stojkovic, Y. Zhang, C. Joe-Wong, and M. Gorlatova, "CollabAR: Edge-assisted collaborative image recognition for augmented reality," in *ACM/IEEE IPSN*, 2020.
[25] L. Liu, H. Li, and M. Gruteser, "Edge assisted real-time object detection for mobile augmented reality," in *ACM MobiCom*, 2019.
[26] L. Zhang, A. Sun, R. Shea, J. Liu, and M. Zhang, "Rendering multi-party mobile augmented reality from edge," in *ACM NOSSDAV*, 2019.
[27] X. Ran, C. Slocum, M. Gorlatova, and J. Chen, "ShareAR: Communication-efficient multi-user mobile augmented reality," in *ACM HotNets*, 2019.
[28] I. Psaras, O. Ascigil, S. Rene, G. Pavlou, A. Afanasyev, and L. Zhang, "Mobile data repositories at the edge," in *USENIX HotEdge*, 2018.
[29] L. T. Tan, R. Q. Hu, and L. Hanzo, "Heterogeneous networks relying on full-duplex relays and mobility-aware probabilistic caching," *IEEE Transactions on Communications*, vol. 67, no. 7, pp. 5037–5052, July 2019.
[30] X. Li, X. Wang, P. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1768–1785, Aug 2018.
[31] S. Friston, C. Fan, J. Doboš, T. Scully, and A. Steed, "3DRepo4Unity: Dynamic loading of version controlled 3D assets into the Unity game engine," in *ACM Web3D*, 2017.
[32] Unity, "Unity manual: Loading resources at runtime," 2019, https://docs.unity3d.com/Manual/LoadingResourcesatRuntime.html.
[33] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, and J. Chen, "Web AR: A promising future for mobile augmented reality—state of the art, challenges, and insights," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 651–666, April 2019.
[34] Google, "Sceneform animation," 2020, https://developers.google.com/ar/develop/java/sceneform/animation.
[35] Google, "Recognize and augment images," 2020, https://developers.google.com/ar/develop/c/augmented-images.
[36] Heroku, "Heroku platform," 2020, https://www.heroku.com/.
[37] R. B. Miller, "Response time in man-computer conversational transactions," in *ACM Fall Joint Computer Conference*, 1968.
[38] Stanford Computer Graphics Laboratory, "Stanford 3D scanning repository," 2014, http://graphics.stanford.edu/data/3Dscanrep/.
[39] H. Schwarz and M. Wien, "The scalable video coding extension of the h. 264/avc standard," *IEEE Signal Processing Magazine*, vol. 25, no. 2, p. 135, 2008.
[40] S. Han, M. Kim, B. Lee, and S. Kang, "Directional handoff using geomagnetic sensor in indoor WLANs," in *IEEE PerCom*, 2012.
[41] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen, "A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned," in *ACM/IEEE IPSN*, 2015.
[42] "Khronos group," 2020, https://www.khronos.org/.
[43] "CGTrader," 2020, https://www.cgtrader.com/.

Authorized licensed use limited to: Duke University. Downloaded on September 18,2020 at 16:25:35 UTC from IEEE Xplore. Restrictions apply.