Wireless VR/AR with Edge/Cloud Computing

Xueshi Hou
Mobile Systems Design Lab, ECE Department
University of California, San Diego
x7hou@ucsd.edu

Yao Lu
Owlii Inc.
yao.lu@owlii.com

Sujit Dey
Mobile Systems Design Lab, ECE Department
University of California, San Diego
dey@ece.ucsd.edu

***Abstract*—Triggered by several head-mounted display (HMD) devices that have come to the market recently, such as Oculus Rift, HTC Vive, and Samsung Gear VR, significant interest has developed in virtual reality (VR) systems, experiences and applications. However, the current HMD devices are still very heavy and large, negatively affecting user experience. Moreover, current VR approaches perform rendering locally either on a mobile device tethered to an HMD, or on a computer/console tethered to the HMD. In this paper, we discuss how to enable a truly portable and mobile VR experience, with light weight VR glasses wirelessly connecting with edge/cloud computing devices that perform the rendering remotely. We investigate the challenges associated with enabling the new wireless VR approach with edge/cloud computing with different application scenarios that we implement. Specifically, we analyze the challenging bitrate and latency requirements to enable wireless VR, and investigate several possible solutions.**

***Index Terms*—Virtual Reality, wireless, edge-based, cloud-based.**

I. Introduction

Recent development of Virtual Reality (VR) and Augmented Reality (AR) systems have led to much interest in their adoption and use in several verticals, like education, enterprise, entertainment, manufacturing, media, medicine and transportation, and large expected addressable market size. In a recent report, Goldman Sachs predicted that VR and AR ecosystem would become an $80 billion market by 2025, roughly the size of the desktop PC market today [1]. According to the latest report from Citi bank, it is also forecasted that the market for VR technology could be a trillion-dollar industry by the year 2035 [2]. The VR and AR technologies are evolving much faster than many have anticipated. IDC expects VR/AR to reach mass adoption levels by 2021, when more than a billion people worldwide will regularly access apps, content, and data through the VR/AR platforms [3]. Furthermore, they also estimate that in 2017, 30% of consumer-facing Global 2000 companies will experiment with VR/AR as part of their marketing efforts.

Due to the immersive experiences promised, VR and AR will enable a new way of working, communicating and entertainment, and hence are predicted to be the next most promising choice of experiencing the Internet [4]. Currently, VR and AR systems are experiencing significant innovation and development. For instance, in recent years we have seen many well-designed HMD devices. There are mainly four types of VR HMD devices [5]: the first type is PC VR, which is tethered with PC, such as Oculus Rift [6], HTC Vive [7], etc.; the second type is Console VR, which is tethered with a game console, like PlayStation VR [9]; the third type is mobile VR, untethered with PC/console but with a smartphone inside, such as Samsung Gear VR [10], Google Daydream [11], etc.; the fourth type is wireless all-in-one HMD device, such as the unreleased Intel Alloy [12]. However, the current HMD devices including all four types mentioned above are still very heavy and large, negatively affecting user experience. Thus, the industry is striving to develop solutions to make HMD devices lighter, more comfortable and more portable.

For all four types of HMD devices mentioned above, they all perform rendering locally either on a mobile device tethered to an HMD device, or on a computer/console tethered to the HMD device. Consequently, today's VR/AR experience lacks portability (when using a heavy HMD device tethered to a mobile device), or mobility (when tethered to a computer). To enable lighter wireless (and hence portable and mobile) VR/AR experience, we propose possible solutions with edge/cloud computing. By performing the rendering on cloud or edge servers, we can make the VR/AR light weight as well as allow VR/AR experience from anywhere. However, the cloud/edge approach can also be challenging, facing bandwidth and latency issues. In this paper, we offer detailed analysis of its challenges and propose potential solutions. For simplicity but without loss of generality, while our examples and experiments are mainly carried out for VR in this paper, the discussions and techniques, in particular as described in Section IV.C, can apply to AR as well.

In this paper, we discuss how to enable light wireless VR/AR with edge/cloud computing. Section II presents the system overview and several application scenarios. In Section III, challenges from bitrate and latency requirements are analyzed in detail through experimental results. In Section IV, we investigate some possible solutions to address streaming problem under the architecture of cloud/edge-based solution. Finally, we draw our own conclusions in Section V.

II. System Overview & Application Scenarios

For teleporting and immersive experiences in VR, generally there will be animated scenes. Inside the scene, there will be animated avatars or reconstructed physical avatars. The difference between animated avatar and reconstructed physical avatar is that animated avatar only has one model with a fixed texture while reconstructed physical avatar normally has different models and different textures for each frame. As

discussed earlier, to enable portability and mobility of VR experiences, we propose rendering be done at either cloud servers or at edge server/device. There are multiple choices for both what to stream and how to stream. Note that by edge computing, we refer to both (a) remote edge computing servers located at the edge of the wireless network, like at radio access points or gateways of the mobile network core, and (b) "local edge" computing devices which may be available and used in the same physical space (like room) as an VR/AR user.

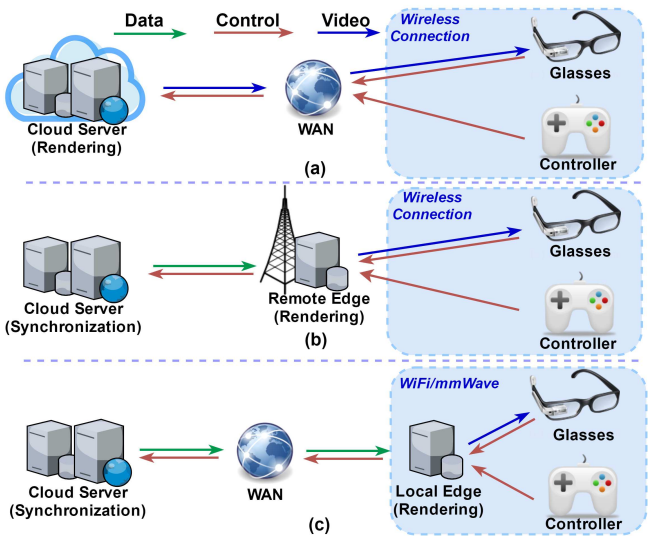

Fig. 1. Three possible approaches to realize the wireless VR/AR with cloud/edge-based implementation: (a) rendering on the cloud server; (b) rendering on the remote edge server; (c) rendering on the local edge device.

As for what to stream, there are three types of content:

T1. Render the field of view (FOV) (current view) remotely and stream the corresponding video to the user's glasses;

T2. Render multiple views remotely, stitch to a 360-degree video, and stream the video to the user's glasses;

T3. Compress model as well as texture, stream them to the user's local edge device at first, then render at the local edge device and stream the video to the user's glasses.

In terms of how to stream, Figure 1 shows three possible approaches as follows:

A1. Stream the video directly from cloud server.

A2. Stream the video from remote edge server.

A3. Stream the video from local edge device.

It should be noted that content T1 and T2 can be used in streaming solutions both A1 and A2 while content T3 can only be used in streaming solution A3. In the following sections, we will provide some examples to show that using streaming solution A1 with content T1 is impossible due to the ultra-low latency requirement, which leaves us the only option of streaming content T2. However, 360-degree video can only provide 3DoF (3 Degrees of Freedom) experience which may limit the extent of immersive experience. For streaming solution A2, both content T1 and T2 may work, but the latency may still be large and experience may not be good enough. For streaming solution A3 with content T3, it offers 6DoF (6 Degrees of Freedom) so the experience will be good, however, the compression is challenging especially for streaming physical avatar models and textures.

We next discuss the advantages and disadvantages of the three rendering approaches. Table I presents a comparison of different rendering locations (cloud, remote edge server, local edge device). The primary advantage of using cloud servers for performing rendering is that it allows users to experience VR/AR when mobile, or from any place (while not having to wear heavy HMD attached to a mobile device, like Samsung Gear). Mobility can also be enabled by rendering on remote edge servers located at the mobile network core (gateways) or radio access network (base station/access point), but will require to support migration of VR applications between edge servers depending on the movement of the users. For example, if the remote edge server is located at a base station of a cellular network, and the user moves from the associated cell to the neighboring cell, then the VR application will have to be migrated to the edge server associated with the new cell. Such migration support needs to manage any additional delay that may be associated with the VR migration, which may be very challenging due to the ultra-low latency requirements of VR/AR. In contrast, rendering on local edge device located in the same physical space as the user (same room, or same building), will afford either no mobility, or very limited mobility.

However, the cloud and remote edge options, while providing for user mobility and ease of use from any location, have costs in terms of network bandwidth and delay associated with them. Streaming of VR content rendered in the cloud will need backhaul bandwidth, as well as either cellular bandwidth if the user is outdoor or maybe unlicensed WiFi if the user is indoor, leading to not only bandwidth cost but also network delay possibly due to congestion. Streaming from remote edge will consume cellular bandwidth and maybe associated with some delay associated with congestion on the wireless link, while streaming from local edge device can be performed purely using unlicensed WiFi network and is not expected to have additional network delay. As we will see in the next section, VR streaming bitrate needed may be very high, leading us to develop techniques to reduce the bandwidth needed to make cloud/edge-based VR feasible. However, proposed techniques like MUE (multi-user encoding) described in Section IV to reduce VR bandwidth requirements, is easier to apply to cloud-based VR than when rendering is done on the edge. The latter will be explained later in Section IV.

New applications and experiences that can be potentially enabled by VR/AR can be categorized to gaming, entertainment, simulation/training, shopping and collaborating applications. In the entertainment category, new applications like virtual museum/gallery/theater/tourism services can enable new experiences; in the simulation category, users can do training, designing and exercises with high interaction using VR/AR; considering shopping VR applications, they enable online shopping or real estate services; and when it comes to collaborating type applications, including virtual meet-

TABLE I
COMPARISON OF DIFFERENT RENDERING LOCATIONS (CLOUD, REMOTE EDGE SERVER, LOCAL EDGE DEVICE)

	Cloud	**Remote Edge** *(Mobile Network Core, Radio Access Node)*	**Local Edge** *(Same physical space as user)*
Allows Light Weight Glasses	Yes	Yes	Yes
Allows Mobility	Yes	Yes, with edge migration	No
Network Bandwidth Consumed	Backhaul, Cellular (outdoor), WiFi (indoor)	Cellular	WiFi, Bluetooth
Network Delay	High	Medium	Low
Potential to apply proposed solutions (like MUE)	Most	Partial	NA

ing/hospital/classroom, they can help people work together and collaborate remotely. For our proposed cloud/edge-based approach to be successful, we need to understand challenges for these emerging categories of applications, including ensuring high user experience, dealing with cloud service cost, achieving data protection, satisfying ultra-low response time requirement as well as high bandwidth requirement.

III. CHALLENGES

In this section, we look at two classes of applications, entertainment/simulation/collaboration applications which are characterized by virtual spaces in which multiple virtual users interact, train and collaborate, and gaming applications which are typically characterized by higher levels of activity and speed. We develop cloud/edge streaming based representative virtual space (virtual classroom) and VR gaming applications, and experimentally determine the challenges to enable such cloud/edge-based VR applications.

Enabling edge/cloud-based wireless virtual spaces can be really challenging. It has been shown earlier that cloud/edge-based video rendering and streaming (for applications like cloud-based mobile gaming) requires high cloud and network bandwidth as well as low response time, satisfying which can be challenging considering dynamic variability of available wireless channel conditions and bandwidth [13] [14]. Furthermore, use of HMD for VR experiences makes the requirements much steeper. Since distance between user eyes and HMD screen is very low, user experience is much more sensitive to video artifacts, which requires much higher video quality, and hence higher video bitrate. Moreover, head motion significantly increases latency sensitivity, causing much higher frame rate and bitrate needed.

To better understand the requirements and challenges of enabling virtual space experiences using remote rendering (at the cloud or edge), we conducted the following experiments with a virtual classroom application we developed using Unity [15], which is presented in Figure 2(a)(c). The virtual classroom is rendered remotely on a server, and the rendered video is encoded using H.264/AVC encoder at 1080p resolution and GOP of 30. Subjects experience the virtual classroom on a PC monitor, as well as with Oculus Rift DK2 as VR HMD. Table II shows the video bitrate needed and acceptable round-trip response time for a good user experience of the virtual

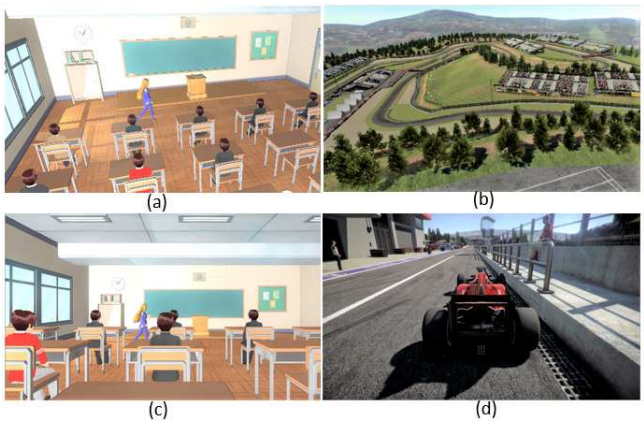

Fig. 2. Two applications used in our experiments: (a)(c) show a virtual classroom application while (b)(d) demonstrate a racing game application.

classroom application, depending on whether PC or Oculus is used for viewing, and whether the user (Oculus) has head motion. The results validate that when using HMD, very high data rate and very low latency are required for an acceptably high user experience. Note that the bitrates shown in Table II are for a single 1080p video stream (corresponding to a single user in the virtual space) – the bitrate needed will significantly increase if 4K or higher resolution is used, and the total requirements will significantly increase depending on the number of virtual users in the space.

Similarly, we also did the following experiments with a racing game application using Unity [15], which is demonstrated in Figure 2(b)(d). The differences between these two applications are mainly that in general virtual classroom can be more static than the racing game. The content change in virtual classroom without head motion generally lies in the movement of teacher and occasional movement of students, while for racing game the view will be continuously and significantly changed since the car is almost always moving. Table III shows the video bitrate needed and acceptable round-trip response time for a good user experience of the racing game application with different types of display devices and motion scenarios (i.e., whether PC or Oculus is used, and whether the user has head motion). The results for racing game also validate that when using HMD, high data rate and low latency are required for an acceptably high user experience. Comparing Table II and Table III, we can observe that data rate needed for

TABLE II
BITRATE AND LATENCY REQUIREMENTS FOR A VIRTUAL CLASSROOM APPLICATION, UNDER DIFFERENT TYPES OF DISPLAY DEVICES AND MOTION SCENARIOS.

Display Device	*Head Motion*	*Framerate & QP*	*Bitrate (1080p)*	*Acceptable Total Latency*
PC Monitor	–	45fps, QP=20	5.8Mbps	100∼200ms
Oculus	–	45fps, QP=15	10.9Mbps	28ms
Oculus	√	75fps, QP=15	28.2Mbps	22ms

TABLE III
BITRATE AND LATENCY REQUIREMENTS FOR A RACING GAME APPLICATION, UNDER DIFFERENT TYPES OF DISPLAY DEVICES AND MOTION SCENARIOS.

Display Device	*Head Motion*	*Framerate & QP*	*Bitrate (1080p)*	*Acceptable Total Latency*
PC Monitor	–	45fps, QP=20	16.6Mbps	<100ms
Oculus	–	45fps, QP=15	33.9Mbps	28ms
Oculus	√	75fps, QP=15	39.7Mbps	22ms

racing game is higher than virtual classroom since generally the former can have more content motion.

To address the dual challenge of requiring very high bitrate and very low latency for remote server based VR applications (i.e. virtual space and gaming applications), we propose novel solutions consisting of key innovations in streaming VR rendered videos and rendering models. In the following, we describe more details of our proposed techniques and also present other possible solutions, which can satisfy the ultra-low latency requirement and address the bitrate/bandwidth challenges.

IV. POSSIBLE SOLUTIONS FOR CLOUD/EDGE-BASED WIRELESS VR

In this section, we investigate some possible solutions to address the high bit rate and ultra-low latency requirements of the cloud/edge-based approach to enabling wireless VR/AR. The possible solutions are discussed for streaming the different types of content introduced in Section III: *FOV* video (T1), *360-degree* video (T2), and models for *6DoF* video (T3) respectively in Sections IV-A, B and C.

A. Streaming FOV Video

As is introduced above, this subsection describes possible solutions to address the bandwidth/latency challenges associated with FOV video, that is, rendering the FOV of the user, encoding and streaming it. In the following, we will investigate how to reduce the sum of bitrates of the cloud-rendered video streams of multiple users in cloud/edge-based applications, while preserving high perceived video quality.

Several existing approaches can be potentially used to reduce bitrate of cloud rendered video streams, including asymmetric video encoding [16] [17] and asymmetric video rendering [18] [19], by encoding and/or rendering the left and right views of a user with different quality. However, since these techniques may potentially compromise video quality, they cannot be used for HMDs which are extremely sensitive to video artifacts. Other methods such as [20] have considered

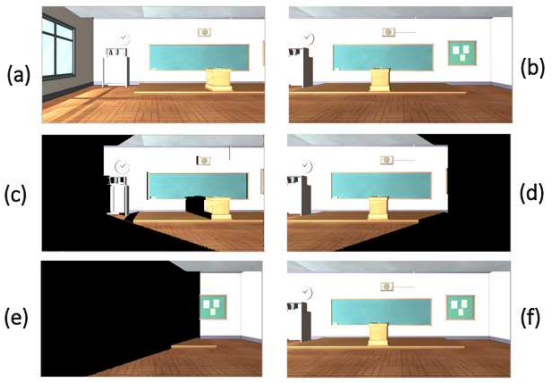

Fig. 3. Illustration of common view and residual view extraction and synthesis (a) primary view A; (b) secondary view B; (c) common pixels from view A; (d) common pixels from view B; (e) residual view in B; (f) generated view B from common view and residual view.

multiple players gaming situations, taking advantage of peer-to-peer sharing between multiple players in the same game scene. But their analysis of correlation of different user views is limited since they proposed the correlation model only for third-person games, where players watch the entire game scene in a bird-view. Therefore, such methods cannot apply to VR applications like virtual spaces and VR gaming.

In a virtual space, we can expect multiple participants to have common views. Figure 3 presents the views in a virtual classroom application, which we have developed using Unity [15]. Figures 3(a) and 3(b) illustrate the rendered views of two students A and B, and Figures 3(c) and 3(d) show the large common views they share. The above observation motivates us to propose a new encoding and transmission approach which can significantly reduce the video bit rate that needs to be transmitted from the edge/cloud server to the user devices.

For a set of users participating in a current virtual space session, we define the view of the user which shares the most common pixels with other users as the primary view (and corresponding primary user), and the other views as secondary

views (and corresponding secondary users) [21]. For each secondary view, we can calculate its residual view as the difference with the primary view, by extracting the common view from the secondary view. We term the above process of extracting common view and determining residual views multi-user encoding (MUE) [21]. Figure 3 shows an example scenario from the virtual classroom application. Figure 3(a) and 3(b) represent a primary view and secondary view, and Figures 3(c) and 3(d) show the common view with respect to the primary and secondary views respectively. Figure 3(e) shows the residual view for the secondary view B.

Subsequently, we propose a hybrid-casting approach to significantly reduce the bitrate needed to transmit the multiple user views, without compromising view quality. Instead of unicasting the rendered video of each user, we can multicast the primary view from the edge node to all the participating users and unicast each residual view to the corresponding secondary user. In the example shown in Figure 3, the primary view shown in Figure 3(a) will be multicast to both users A and B, and the residual view in Figure 3(e) will be unicast to user B, instead of unicasting the full views of A and B (Figures 3(a) and 3(b)). Finally, on the user devices, all users will receive the primary view. The secondary users will also receive their residual views. The primary users will decode and display the video directly while the secondary users will decode both primary view and residual view, transform the primary view to common view and combine common view with residual view to get his secondary view. Figure 3(f) shows user B synthesizing its secondary view from the transmitted common view and residual view as explained above.

Depending on the number of users in the virtual space, their positions and view angles, a single primary view may lead to some secondary views having little or no common views, and thus large residual views. Hence, we propose to partition the users into one or more groups, with each group having a primary view and zero or more secondary views, such that the total video bitrate of the primary and residual views that need to be transmitted from the cloud/edge node to the user devices is minimized.

In [21], we develop a prototype of our proposed multi-user encoding and hybrid-casting approach in MATLAB. We implement a grouping algorithm that considers explicitly every possible grouping choice of regarding each view in a group as primary view. Applying the technique to virtual classrooms (using Unity [15] and Oculus Rift SDK [6]) of different sizes and different number of participants, we see up to 48.4% reduction in number of pixels needed for the views of all the students, compared to the conventional approach of transmitting all the views as individual unicast streams [21].

In the future, theoretical formulations and efficient real-time algorithms for multi-user encoding have to be developed, which can be applicable to different types of edge/cloud-based VR applications. Heuristic grouping algorithms will need to be developed with real-time performance, since optimal grouping done for the experimental results presented here has exponential time complexity, taking more than an hour of CPU time (using 3.57GHz Intel Core i7 processor, 32G memory). When edge computing is used, the grouping algorithms need to be aware of not only the virtual locations and views of the users in the virtual space so that users in a group share large amount of common pixels, but also the physical location and association of the users to the edge servers such that users in a group are associated with the same edge server. Also, to demonstrate the full potential for the hybrid-casting approach, multi-cast capabilities of different wireless access networks will need to be considered, and efficient hybrid-casting approaches for the primary and secondary views will need to be developed and validated.

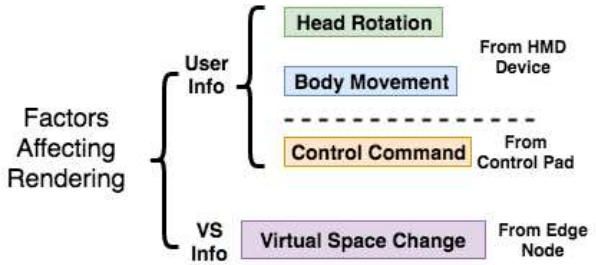

Fig. 4. Factors determining when and what rendering needs to be performed in VR.

B. Streaming 360-degree Video

In this subsection, we present a possible solution of streaming 360-degree video to address the latency challenge. Specifically, we describe our proposed approach to reduce head rotation latency, consisting of 360-degree video rendering, multi-user encoding and hybrid-casting. As is shown in Figure 4, there are four factors that will determine when and what new view needs to be rendered for a user in a virtual space, including head rotation, body movement, and control commands coming from the user, and virtual space changes registered by the edge/cloud server as a result of changes by other users (including their body movements, and/or changing position of objects through control commands). Moreover, in some VR applications that are related to real physical scene (i.e., virtual hospital), changes in the virtual space can also be caused by updates of the 3D reconstruction model resulting from any changes in that physical space (i.e., a surgery in a real hospital). While the acceptable response time to changes due to control command by the user and/or any virtual space change is 100ms–200ms, (like 100ms requirement for first-person shooting games [22]), a user will expect much lower latency of 20ms–30ms with her head rotation or body movement, as has been experimentally shown earlier (Tables II and III). Any new user information, like head rotation or control command, will be transmitted from their devices to the edge/cloud server.

As opposed to the VR approach described in the previous section, where the FOV is rendered, encoded and streamed only after new head rotation or body movement of the user is tracked and transmitted to the cloud/edge, in the case of 360-degree video approach, the 360-degree video is rendered in a regular interval, or with any changes in the user control

Fig. 5. 360-degree Panorama of virtual classroom.

TABLE IV
RESULTS SHOWING SIGNIFICANT COMMON PANORAMA VIEWS AND SAVINGS FOR MULTIPLE VIRTUAL CLASSROOM USERS.

Secondary View	$View_1$	$View_2$	$View_3$	$View_4$	$View_5$
Common Pixel Ratio	0.57	0.63	1	0.63	0.58
Pixel Saving	48.2%				

information or virtual space change (not in response to user head movement), and streamed and cached at the user device. When the user subsequently performs head rotation, the new head position is tracked and used to select the appropriate video from the cached 360-degree video in the user device, and displayed on the user VR glasses, thus eliminating the delays associated with FOV rendering and streaming. The head-rotation latency will be reduced to head tracking delay (less than 4ms with current technologies) and HMD display delay (less than 11ms with current displays), and hence meeting the ultra-low latency requirement of 20ms–30ms.

However, while this approach can significantly reduce head rotation latency, it can also significantly increase computation at the edge/cloud server (for rendering and stitching multiple camera views) as well as the bit rate needed to transmit 360-degree video for each user associated with the application session, which is about 4 times more bitrate than for 90-degree field of view [23]). We propose to investigate the use of multi-user encoding technique to significantly reduce the bitrate needed to transmit the 360-degree videos to the users, as well as explore its use to reduce the rendering cost by rendering just the primary view and residual views of secondary users, instead of 360-degree views of all users.

To understand the potential of multi-user encoding for 360-degree rendering, we have developed a prototype of rendering 360-degree panorama views of our virtual classroom application. Figure 5 shows the panorama view for a single user in the virtual classroom. Table IV shows results with five students in the same row with the middle one chosen as the primary view, calculating the percentage of common pixels with primary view between their panoramas and obtaining the pixel saving when applying our MUE approach [21]. $View_1$–$View_5$ are the student views in the row from left to right. The preliminary results show that we can achieve a high pixel saving (48.2%) using MUE technique compared with streaming all the five panorama views, which can be promising in making the 360-degree approach feasible in terms of bandwidth needed for the networks connecting the cloud or edge node to its users.

In the future, the above approach needs to be developed more comprehensively, including exploring different number of cameras to create the 360-degree videos to see if bitrates of the 360-degree videos can be further reduced, developing real-time algorithms for MUE encoding, and experimenting with more number of users and different applications. Also, more research and development is needed to study how to stitch the 360-degree views and create the 360-degree videos more efficiently.

C. Streaming 6DoF Content

For applications such as teleporting, people from remote locations could be teleported to the same virtual place in VR or superimposed at a physical place in AR. In this case, people need to move around and interact with others. Thus, 6DoF content can greatly improve the overall immersive experience, compared to even 360-degree video that can provide 3DoF (rotation of the head). In order to provide 6DoF experience, both full 3D geometric information and surface color information need to be generated and provided. In the cloud/edge architecture, these geometric information and color information need to be compressed and streamed. Improving compression efficiency is the most common way to reduce bitrate without sacrificing quality. However, for the solution of the compression framework has not been formulated well enough, leaving tremendous opportunities in this area. In the following, we investigate the current status of the technical solutions and propose future trends.

For gaming applications, animated avatars need to be compressed and streamed. Those models need to be sent only once and stored at local edge server, thus it is not very challenging in this case. For teleporting applications, physical avatars need to be compressed and streamed. Normally both the model and texture need to be reconstructed and updated for each frame, resulting in huge amount of data to be streamed. It should be noted that although theoretically one model can be used to represent the physical avatar, it has been proved to be less vivid and may create uncanny valley [31].

For the representation of physical avatars, currently, there are mainly two types of format, namely point cloud and mesh. Different formats have completely different compression frameworks. Figure 6(a) shows an example of mesh [24] and Figure 6(b) demonstrates an example of point cloud [25]. With the mesh format, the surface is represented by a lot of

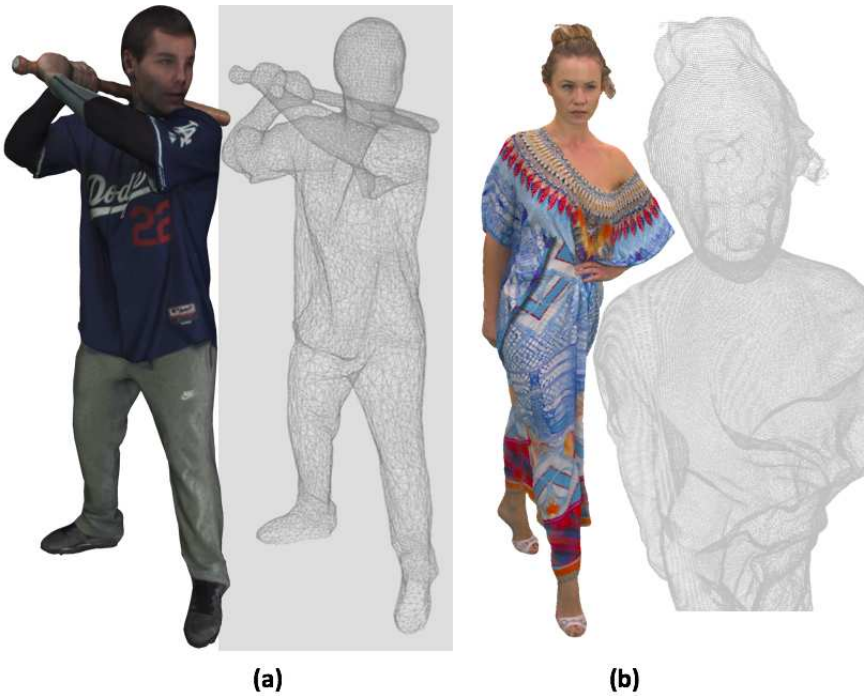

Fig. 6. (a) an example of mesh, (b) an example of point cloud.

triangles. The mesh will have texture maps and each triangle will have a UV coordinate mapping to one pixel in the texture map. With the point cloud format, the model is represented by a lot of points in the 3D space with a color associated with each point.

TABLE V
PERFORMANCE OF DIFFERENT COMPRESSION STRATEGIES

#	*Format*	*Compression Method*	*Bitrate*
1	Mesh	S. Escolano et al. [26]	2Gbps
2	Point Cloud	R. Schnabel et al. [27]	450Mbps
3	Point Cloud	R. Queiroz et al. [28]	150Mbps
4	Point Cloud	E. Pavez et al. [29]	45Mbps
5	Mesh	A. Collet et al. [30]	15Mbps
6	Mesh	Owlii et al. [24]	9Mbps

The compression of static mesh has been studied for a long time while point cloud compression is a new area. Both dynamic mesh compression and point cloud compression have attracted much research interest recently. Table V summarizes the performance of some published solutions. The performance is based on the mesh with about 20K–40K triangles and texture map with 1Kx1K resolution or point cloud with 200K points. For method 1 from S. Escolano et al. [26], both the mesh and texture is barely coded. For method 2 from R. Schnabel et al. [27], it does intra-frame only for geometry and it is based on an octree data structure. At the same time, color space is chosen as 8:1:1 YUV and is uncoded. For method 3 from R. Queiroz et al. [28], the geometry coding is the same as method 2 while the color is coded using RAHT. For method 4 from E. Pavez et al. [29], both intra-frame coding and inter-frame coding is used for coding both geometry and color. For method 5 from A. Collet et al. [30], it uses non-rigid registration to estimate the correspondence between meshes and further make the connectivity to be consistent within a group of frames. The length of a group is normally 2030. Then they use simple motion estimation with Golomb coding for mesh and standard H.264/AVC codec for texture map. Method 6 from Owlii [24] further improves method 5 by 1) using a better non-rigid registration technique to increase the length for each group, 2) using a TFAN [32] based technique for intra-frame mesh compression and 3) using FAMC [33] codec for inter-frame mesh compression.

In terms of future research directions, for dynamic mesh compression, further performance improvement could be achieved by replacing H.264/AVC codec with H.265/HEVC for texture and designing better non-rigid registration technique to increase the group length. For point cloud compression, better techniques need to be developed to do inter-frame prediction and color information may be compressed more efficiently. In addition to improving coding efficiency, adaptation techniques could also be developed for streaming both mesh and point cloud.

V. CONCLUSION

In this paper, we discuss how to enable a lighter wireless VR/AR experience with edge/cloud computing. By shifting extensive rendering to the cloud/edge/local server, we can make the VR/AR glasses to be light weighted. We look at several application scenarios and give a system overview. The challenges from bitrate and latency requirements are analyzed in detail with experimental results. Then we investigate and propose several possible solutions to enable cloud/edge-based

wireless VR/AR, which can better satisfy the ultra-low latency requirement or address the bandwidth challenge.

Acknowledgement

This work was supported in part by the Center for Wireless Communications at UC San Diego.

References

[1] H. Bellini, "The Real Deal with Virtual and Augmented Reality," [Online]. Available: http://www.goldmansachs.com/our-thinking/pages/virtual-and-augmented-reality.html

[2] L. Graham, "Citi eyes a trillion-dollar industry in virtual reality technology," [Online]. Available: http://www.cnbc.com/2016/10/14/citi-eyes-a-trillion-dollar-industry-in-virtual-reality-technology.html

[3] "IDC Sees the Dawn of the DX Economy and the Rise of the Digital-Native Enterprise," [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS41888916

[4] K. Weida, "IS VIRTUAL REALITY THE FUTURE OF THE INTERNET?" [Online]. Available: https://www.highspeedinternet.com/resources/virtual-reality-future-internet

[5] "Capitalizing on ViewersH́unger for Virtual and Augmented Reality White Paper," [Online]. Available: http://www.conexta .co.uk/content63331

[6] [Online]. Available: https://www3.oculus.com/en-us/dk2/

[7] [Online]. Available: https://www.htcvive.com/

[8] [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us/

[9] [Online]. Available: https://www.playstation.com/en-us/explore/playstation-vr/

[10] [Online]. Available: http://www.samsung.com/us/explore/gear-vr/

[11] [Online]. Available: https://vr.google.com/daydream/

[12] https://newsroom.intel.com/chip-shots/intel-unveils-project-alloy/

[13] S. Wang and S. Dey, "Adaptive Mobile Cloud Computing to Enable Rich Mobile Multimedia Applications," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 870–883, June 2013.

[14] S. Dey, Y. Liu, S. Wang, and Y. Lu, "Addressing Response Time of Cloud-based Mobile Applications," in *Proc. of the first ACM International Workshop on Mobile Cloud Computing and Networking (MobileCloud13)*, Bangalore, India, July 29–August 1, 2013, pp. 3–10.

[15] [Online]. Available: https://unity3d.com/

[16] G. Saygili, C. G. Cihat and A. M. Tekalp, "Evaluation of asymmetric stereo video coding and rate scaling for adaptive 3D video streaming," *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 593–601, June 2011.

[17] S. Valizadeh, A. Maryam and N. Panos, "Bitrate reduction in asymmetric stereoscopic video with low-pass filtered slices," in *Proc. of 2012 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, Jan. 13–Jan. 16, 2012, pp. 170–171.

[18] Y. Lu, Y. Liu, and S. Dey, "Enhancing Cloud Mobile 3D display gaming user experience by asymmetric graphics rendering," in *Intl. Conf. on Computing, Networking and Communications (ICNC)*, pp. 368–374. 2014.

[19] Y. Lu, Y. Liu, and S. Dey, "Cloud mobile 3D display gaming user experience modeling and optimization by asymmetric graphics rendering," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 517–532, 2015.

[20] W. Cai and V. C. Leung, "Multiplayer cloud gaming system with cooperative video sharing," in *Cloud Computing Technology and Science (CloudCom), IEEE 4th Intl. Conf.*, Taipei, Dec. 3–6, 2012, pp. 640–645.

[21] X. Hou, Y. Lu and S. Dey, "A Novel Hyper-cast Approach to Enable Cloud-based Virtual Classroom," in *Proc. of IEEE International Symposium on Multimedia (ISM16)*, San Jose, Dec. 2016, pp. 533–536.

[22] S. Wang and S. Dey, "Modeling and characterizing user experience in a cloud server based mobile gaming approach," in *IEEE GLOBECOM 2009*, Hawaii, Nov. 30–Dec. 4, 2009., pp. 1–7,

[23] [Online]. Available: http://www.360rize.com/2015/02/4k-vr-360-video-what-is-it-and-how-can-i-produce-it/

[24] [Online]. Available: http://owlii.com/

[25] [Online]. Available: https://8i.com/

[26] S. Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, et al, "Holoportation: Virtual 3D Teleportation in Real-time," in *Proc. of the 29th Annual Symp. on User Interface Software and Technology*, Oct. 2016, pp. 741–754.

[27] R. Schnabel and R. Klein, "Octree-based Point-Cloud Compression," *Spbg*, pp. 111–120, July 2006.

[28] R. Queiroz and P. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[29] E. Pavez, P. Chou, R. Queiroz and A. Ortega, "Dynamic Polygon Cloud Compression," *arXiv preprint* arXiv:1610.00402, Oct. 2016.

[30] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, et al, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (TOG)*, vol. 34, issue 4, pp. 69, Aug. 2015.

[31] [Online]. Available: https://en.wikipedia.org/wiki/Uncanny_valley

[32] K. Mamou, T. Zaharia, and F. Prłteux, "TFAN: A low complexity 3D mesh compression algorithm," *Computer Animation and Virtual Worlds*, vol. 20, issue 2-3, pp. 343–354, June 2009.

[33] K. Mamou, T. Zaharia, and F. Prłteux, "FAMC: The MPEG-4 standard for animated mesh compression," in *IEEE Intl. Conf. on Image Processing (ICIP)*, San Diego, CA, Oct. 12-Oct. 15, 2008, pp. 2676–2679.