IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 11, 2022, accepted May 8, 2022, date of publication May 11, 2022, date of current version May 16, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3174351

Low Latency Streaming for Path-Walking VR Systems

WON-KI SEO AND CHAE EUN RHEE, (Senior Member, IEEE)
Department of Information and Communication Engineering, Inha University, Incheon 22212, South Korea

Corresponding author: Chae Eun Rhee (chae.rhee@inha.ac.kr)

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-0-02052, and in part by the Research and Development Program of MOTIE/KEIT (developing low power deep learning HW IP design technology for image processing in CMOS image sensors) under Grant 20010582.

ABSTRACT Highly immersive content in the form of extended reality (XR) is attracting attention as an alternative to conventional video services such as YouTube and Facebook. Many galleries and museums already offer online virtual reality (VR) tours where users are free to choose the spot they want to move to, beyond merely looking around. Although the ease of implementation, this key-spot hopping is still far from giving the real feeling of walking. Meanwhile, in recent volumetric or light-field-based studies, view rendering that supports free and continuous viewpoint movements has been attempted. With online services in mind, however, the high data volume and computational complexity are a big obstacle to practical applications. Path-walking VR, the target system of this paper, can be a good compromise, where the viewer can enjoy the virtual space while walking along the route. The interactive path-walking VR service is entry-level immersive video, but streaming over the network is still challenging. One of the main problems to be tackled is that the movement patterns of viewers need to be reflected in the streaming strategy to improve the quality of experience. Unlike unidirectional video, the movement of the viewer determines which images and how many images should be transmitted. This paper proposes schemes to reduce streaming delays by reflecting the viewer's movement characteristics. It is differentiated from existing studies for omnidirectional video in that the proposed schemes control not only image quality but also view update rate. The first is a caching strategy which takes advantage of the geometrical locality of the virtual space that the viewer will soon reach a position close to the current position. This not only reduces the communication delay from the server, but also decreases the burden of server-side request handling. The second scheme uses the relationship between the viewer's speed and the field of vision. The image quality is adjusted according to the viewer's speed and head direction. Experimental results show that the proposed schemes achieve stable viewer's experience by considering walking characteristics in virtual space. It is expected that the results of this paper will provide insight to those who design interactive streaming systems for immersive media applications.

INDEX TERMS Immersive media, interactive virtual reality, video streaming, omni-directional video, low-latency, caching.

I. INTRODUCTION

Over the past decade, platforms that provide video services such as YouTube and Facebook have grown significantly. Their main content involves the simple watching of videos with little or no user interaction, such as live streaming and video-on-demand (VOD) services. Recently, trends with regard to the consumption of multimedia have become highly diversified. There is an increasing demand to create content and share experiences with others. To do this, user-experience-centered virtual reality (VR), augmented reality (AR), and mixed reality (MR) are positioned as new alternatives.

Emerging captured-image-based VR services allow users to walk inside virtual spaces, beyond merely looking around. Virtual tour applications, such as Google Arts & Culture [1] and the National Gallery [2], are prime examples. As shown in Figure 1 (a), telepresence is provided to remote viewers,

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

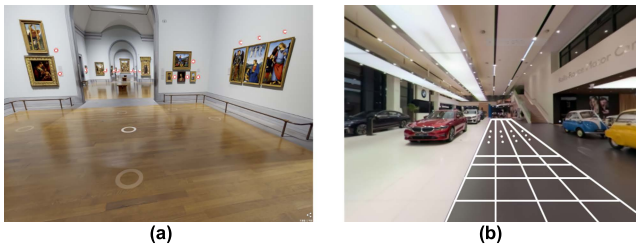

FIGURE 1. Comparison of (a) spot-based VR and (b) Path-walking VR.

allowing them to enjoy 360-degree views while moving to various spots inside the virtual space. The advantage of such a spot-based VR tour is the ease of implementation. With only a few shooting images, general users can easily build a virtual tour service using a commercial tool such as Google VR tour creator [3]. However, key-spot hopping is insufficient to provide the experience of walking. In recent volumetric or light-field-based studies, view rendering that supports free and continuous viewpoint movements has been attempted. With online services in mind, the high data volume and computational complexity are obstacles to commercialization. Currently, 4G technology barely supports entry-level VR applications. B5G/6G is known to support the specification of ultra-low latency of 1ms or sub-ms. However, considering available network performance, multi-user environment, and the demand for high-quality video services, it is difficult to realize a true free viewpoint streaming right away. The path-walking VR can be an intermediate step. Given a set of images taken while moving a 360-degree camera, viewers can walk freely on the camera path. This can be referred to as path-walking VR, and an example is shown in Figure 1 (b). It is the easiest and most practical approach to realize experience-oriented virtual space based on existing video streaming platforms.

In an interactive VR streaming service targeting multiple users, the end-to-end (E2E) latency felt by the user must be very short. Under this constraint, path-walking VR has the following challenges. Unlike spot-hopping, path-walking is continuous. Thus, view data must be streamed seamlessly accordingly. Also, unlike passive video watching, the moving pattern or speed of the viewer is dynamics and unpredictable. To tackle these problems, this paper proposes two schemes. The first is geometric distance-based caching (GDC). When walking along a continuous path, view images of nearby locations will be needed soon. Also, users can linger in a neighborhood, revisiting the same places. Reflecting these characteristics, a geometric locality-based cache is placed on the client side. Cache hit reduces the request handling burden on the server side and indirectly contributes to the reduction of E2E latency. Second, the exploring speed aware streaming (ESS) scheme uses the relationship between the viewer's speed and the field of vision [4]. When viewers run quickly, they only look forward without looking around thoroughly. However, view updating must be done frequently in proportion to the exploration speed. The ESS takes advantage of the trade-off between update rate and the quality of area outside of interest. It reduces latency by lowering the quality of the out-of-interest area in the 360-degree image when the user moves fast and enables high view update rate.

The remainder of this paper is organized as follows. Section II reviews related research works. Section III briefly introduces the overall proposed system. The proposed schemes are described in detail in Section IV. The experimental results are presented in Section V. The paper is concluded in Section VI.

II. RELATED WORK

There have been many studies for low-latency VR streaming. Most techniques start from the transmission of omnidirectional videos. It utilizes the fact that the field-of-view (FoV) of device such as a head-mounted display (HMD) is limited and only partial views are transmitted, not 360-degree full views [5]–[9]. Tile-based viewport-dependent techniques are most widely used. A full view is divided into multiple tiles. The tile set belonging to the FoV is transmitted at a high-quality level, whereas the rest of the set utilizes low quality transmission, reducing the amount of data and maintaining the quality of experience (QoE) of the user. Instead of changing the combination of high- and low-quality tile sets every frame, a segment, which is a set of frames, can be used as a transmission unit [10]–[16]. [17] considers the characteristics of contents. They take advantage of the fact that it is difficult for the viewer to clearly perceive a distant or fast-moving scene. Based on the relative speed and depth of objects in the scene, tiles are grouped and transmitted at high or low bitrates.

Several studies suggest prefetching methods through FoV prediction, where a FoV prediction model is based on the user's head trace record. In addition, various prediction methods such as linear regression [11], [18], long short-term memory (LSTM) model [14], [19], and a Gaussian probabilistic model [20], have been attempted. Other experimental results [21] show that if a separate prediction model is used for each user, higher prediction accuracy can be achieved compared to the one-size-fits-all method. In many cases, the user's FoV shows a similar pattern. Accordingly, researchers have attempted to reduce server requests by introducing a cache. The work in [22] reduces duplicate requests by grouping users so that they do not become competitors when users in close physical proximity request the same data. In [23], a FoV-aware caching policy is proposed, where the relationship between tile requests and the corresponding quality levels are learned through maximum likelihood estimation, after which the tile with the lowest probability is removed from the cache. Various other research works focusing on placing the cache on an edge server are also in progress [24]–[26].

Recent studies deal with immersive video streaming, which supports a high level of degree-of-freedom (DoF) by which users can feel motion parallax according to changes in the head position as well as the head rotation. Most

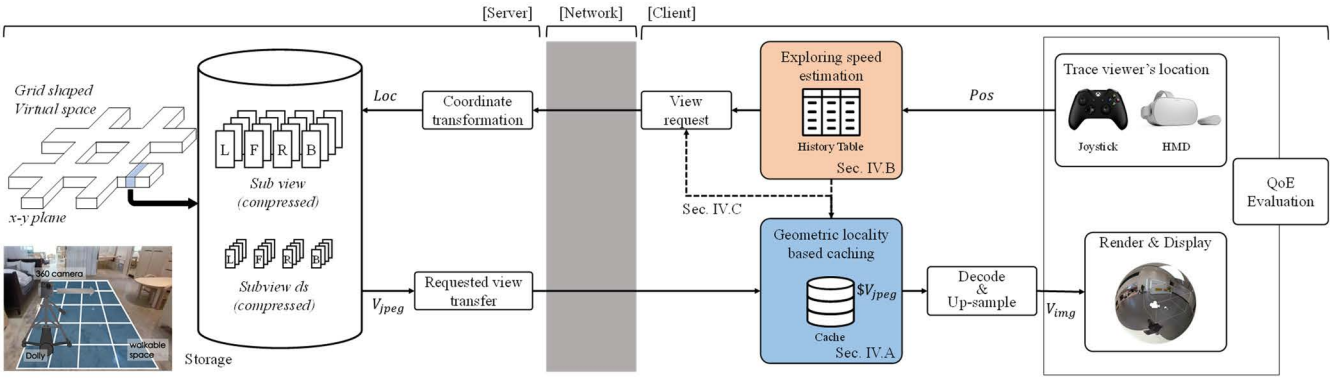

FIGURE 2. Overview of the path-walking VR streaming environment.

of these volumetric videos are based on point cloud and light field technologies. A FoV-based tiling technique, often used in omnidirectional video, is typically applied. However, as high DoF requires a significantly large increase in data, various new methods have been proposed. One line of research [27]–[30], [32] proposes a method to transmit the point cloud robustly and reliably according to the network conditions. Two other studies [28], [33] use progressive compression, where a low-quality point cloud is basically transmitted, with additional data transmitted when the network conditions are fine, enabling a high-quality point cloud. The multi-view system of [31] down-samples the surrounding views of the reference view to reduce the bitrate. Tile-based transmission has also been attempted by dividing the point cloud into spatial and temporal three-dimensional (3D) tiles [34], whereas the point cloud has also been divided into a 3D mesh for transmission [35]. Other work [29] enables adaptive streaming by proposing DASH-PC, which fits DASH to a point cloud. To reduce the transmission latency further, many studies have attempted to increase the prediction accuracy, such as by FoV predictions [37], viewport predictions [28], [38] and latency compensation [35]. [36] adjusts the bitrate by estimating the network condition with an improved bitrate estimator called EVeREst. However, the high DoF makes accurate prediction more difficult than in the low-DoF case, such as omnidirectional videos. Another option is to interpolate the position and rotation information acquired from Microsoft HoloLens by means of the spherical linear interpolation of rotations (SLERP) and to apply it to the AutoReg prediction model [39]. A protocol for applying the conventional dynamic adaptive streaming of two-dimensional (2D) video to a light-field-based system has also been devised, showing good subjective results [40]. Other studies [41], [42] reduce the E2E latency by putting the light-field data necessary for rendering in the cache on the GPU on the client side.

Thus far, existing research on high-DoF video streaming is at the level of extending omnidirectional video streaming techniques. The point here is to control the image quality by distinguishing viewport and non-viewport areas. In 3-DoF+ streaming applications, viewpoint change should be additionally supported. However, there were few studies to dynamically control the view update rate in consideration of the video scene or user movement characteristics. The schemes proposed in this paper are the first attempts to control both image quality and update rate by actively reflecting user movement and attention in a 3-DoF+ environment.

III. SYSTEM OVERVIEW

The virtual space assumed in this paper is constructed in the following way. For scalability and processing convenience, the 360-degree camera shoots densely moving along the grid on the floor. One side of the unit square forming the grid is 0.6m and it consists of 120 images. The location of a camera and the corresponding view are saved on the server. In order to satisfy smooth communication in an interactive streaming scenario, not only the short transmission time due to the low bitrate but also the encoding/decoding time are important. In general, video coding has better compression efficiency than image coding and a hardware-accelerated codec using GPUs effectively accelerates complex video coding process. However, it is observed that the advantage of this GPU-based video codec is valid for single and long video services. Note that, the transfer time between the host and device memory and the time for memory allocation/deallocation process are included in the decoding time. The proposed system handles multiple and short pseudo-sequences. In this case, an additional time overhead due to the movement to the GPU or memory allocation is added for every pseudo-sequence, and thus, the decoding time increases rapidly. This consideration leads to the decision to use the image codec on the CPU rather than the video codec on the GPU. The proposed system uses JPEG.

Figure 2 shows the path-walking VR streaming environment assumed in this paper. A full-view, 4K (4096 × 2048, 24bit) 360-degree image photographed along the grid path is vertically divided into four sections to create a *sub-view*. Each *sub-view* is stored in the server's storage along with the location information on the x-y plane of the virtual space. The *sub-view-ds*, down-sampled from each *sub-view*, is also

saved. The black arrow represents the basic data path. The client transmits the *Pos* to the server. The *Pos*, which is obtained from a joystick, HMD or a controller, contains the position of the x-y plane and the head direction of the viewer. The server calculates the virtual 3D world coordinate *Loc* from the real-world coordinate *Pos* and streams the corresponding 360-degree image V_{jpeg} to the client in an on-the-fly manner through the network. The client decodes V_{jpeg} into V_{img}. This is displayed to a 360-degree sphere.

In an interactive streaming system, it is known that the E2E latency has a budget of approximately 16 to 20ms [43]. This latency, T_{e2e}, can be broken down as follows.

$$T_{e2e} = T_{comm} + T_{render} + T_{misc} \tag{1}$$

T_{comm} is the communication time and T_{render} is the time required to decode the *sub-view* and display it on the sphere. When receiving *sub-view-ds*, an up-sampling operation is also required. The other miscellaneous processing time is called T_{misc}. T_{render} and T_{comm}, which account for a large proportion of T_{e2e}, should be reduced for real-time streaming. IEEE 802.11n networks have an average bandwidth of about 30 Mbps. Assuming that the 4K JPEG image size is 0.45MB, the time budget for T_{e2e} is consumed only by T_{comm}. Even if the bandwidth or latency is greatly improved in a B5G/6G environment, T_{comm} can be still long due to fluctuations of the effective bandwidth or the request handling burden of the server in a multi-user environment. Therefore, it is necessary to reduce both the request frequency and the amount of data transmission. The decoding time of 4K JPEG images in T_{render} is also difficult to ignore and largely depends on the computing capacity.

In Figure 2, the proposed schemes are shown in blue and orange. The caching scheme marked in blue takes advantage of the fact that the probability of view reuse is closely related to the viewer's current position. This contributes to a decrease in the request frequency and will be explained in detail in Section IV.A. The scheme marked in orange uses a history table containing the viewer's recent movement records. Through this, the exploring speed (*ES*) is estimated to reduce the amount of data and, consequently, to shorten T_{e2e} without deteriorating the quality of the viewer experience. This will be explained in detail in Section IV.B. The dotted line in Figure 2 shows the information sharing between two schemes, with the corresponding integration presented in Section IV.C.

The user's QoE can be monitored by adding a QoE evaluation module to the client side. In many video streaming systems, visual quality and latency are the primary metrics of QoE. Inspired by previous studies, the proposed system defines the immersive score to reflect the two factors of quality and latency. Immersive score is calculated as the product of quality rate and latency rate, and each has a value in the range of 0 to 1. Unlike one-way streaming, in interactive streaming like the proposed system, it is the user who decides which view to display. The viewport of the 360-degree image is determined according to the user's head direction. The expected quality of experience is set to 1 according to the user's movement state, and differences between expectations and actuals are scored. When a user sees a view with the expected quality, the quality rate = 1. Also, the degree of attention or view update rate for the views changes according to the user's moving speed. If the view is updated in time according to the user's movement speed, the latency rate = 1. To sum up, if the client receives the image with the expected quality within the maximum allowable delay time, it can be said that the QoE is not degraded. In Section V.C, the result of measuring the immersive score is presented through a simple simulation assuming a multi-user environment.

IV. PROPOSED SCHEMES

A. GEOMETRIC DISTANCE BASED CACHING

In cache designs for path-walking VR systems, the order of image access is closely related to the acquisition location of the image. The closer the geometric distance to the current viewpoint is, the higher the probability of visiting becomes. Conversely, the chances of visiting a distant location sooner or later are low. In addition, there is ultimately a limit on the geometric distance that can be accessed.

In Figure 3, the viewer's moving route is displayed on the x-y plane. The circle represents a walking step and the view is updated in each circle. The view image at the dotted circle position does not exist in the client's cache as of now. Given that all circles through which the arrow passes in Figure 3 (a) are visited initially, cold start misses occur in the cache. The missing view is requested to the server. The view sent from the server is stored in the cache. Assuming that the cache size is 15, the cache becomes full in (1, 3). From (0, 3) in Figure 3 (b), cache replacement occurs. When a commonly used cache replacement policy such as the first-in-first-out (FIFO), least recently used (LRU), and least frequently used (LFU) type is adopted, views of (0, 0), (1, 0) and (2, 0) are evicted in that order. In Figure 3 (c), when a cache miss occurs at a revisited location such as (0, 0), it is expressed as a circle with an X. As shown in Figure 3 (c), cache misses will occur from (0, 0) to (5, 0), visited previously but subsequently evicted.

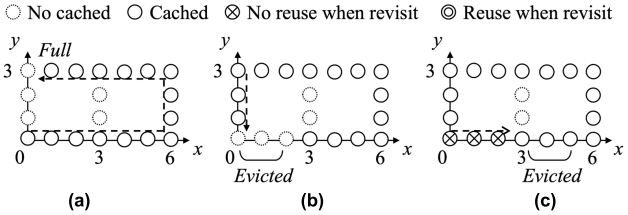

FIGURE 3. Snapshots of cache replacement according to viewer's moving route (a) first 6 views (b) second 6 views (c) third 6 views and (d) fourth 6 views with FIFO, LRU and LFU policy.

In Figure 4, the cache replacement policy considering geometric locality is applied. Figure 4 (a) shows the same moving route presented in Figure 3. When visiting (0, 3), (0, 2) and (0, 1) in sequence, the view of the farthest location

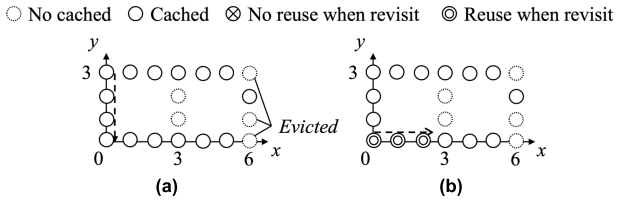

FIGURE 4. Snapshots of cache replacement according to viewer's moving route (a) third 6 views and (b) fourth 6 views with the proposed GDC policy.

```
Input :  V : cached viewpoints, N : Number of candidate viewpoi
         c : a current viewpoint
Output : t : a victim for replacement
begin
Step 1.
1.    for all viewpoints v ∈ V do
2.        calculate dist(v, c) Euclidean distance between v and c
3.        push dist(v, c) to dist_list
4.    sort dist_list by descending
5.    select N-elements from sorted dist_list
Step 2.
6.    for all N-elements n do
7.        calculate path(n, c) shortest path(n, c) between n and c
8.        push path(n, c) to path_list
9.    t = a viewpoint of Max(path_list)
10.   return t
end
```

FIGURE 5. The process of the selection a victim of replacement with GDC policy.

is replaced. Here, the geometric distance is calculated using the Euclidean distance. When there are multiple locations with an identical geometric distance, LRU is applied. As a result, views of (6, 0), (6, 1) and (6, 3) are evicted in that order. In Figure 4 (b), cache hits are observed for a while due to revisiting. Those are represented by circles with an O. Specifically, this type of cache hit is valuable in a virtual tour system as viewers often hover looking around the virtual space. The proposed GDC replacement policy is shown as a pseudo code in Figure 5. V denotes all viewpoint sets in the cache, whereas c is the current viewpoint input. N is the number of victim candidates and is a user-defined variable. The Euclidean distance between c and all elements of V is calculated. Among them, N candidates having a large distance value are selected. The distance of the shortest path between N candidates and the current viewpoint c is calculated. This is done because even if the Euclidean distance is small, the moving distance, which is the distance the viewer actually moves, can be large. Among the N candidates, the location with the largest shortest path is selected as a victim to be evicted.

B. EXPLORING SPEED AWARE STREAMING

The proposed ESS utilizes the relationship between the viewer's exploring speed and the view area on which viewer can concentrate, i.e., the field of vision. When the viewer moves slowly, there will be time to walk and look around. Naturally, the field of vision increases. On the other hand, the purpose of a fast-moving viewer is not to look, but to move, and it is difficult to care about surroundings other than the heading direction. Meanwhile, if the viewer moves fast, the view-update rate should be high as well. In the proposed ESS, every time the view image is updated, the corresponding viewer's motion acceleration is recorded in the history table. The current ES is estimated based on the acceleration history. In (2), $\text{tuple}_{\text{acc}}$ (t) is the acceleration value captured by the motion sensor during the t_{th} view update. Window size (WS) refers to the number of reference views used for the acceleration calculation. WS is determined by the sampling rate of the motion sensor. For example, if the sampling rate is 60Hz, WS is set to 60. ES is the average value of $\text{tuple}_{\text{acc}}$ (t) recorded during the past WS. If the history size (HS), which is the number of views recorded in the history table, is smaller than WS, then WS is set to HS. The ES calculation is very simple. Thus, it has little effect on T_{e2e}. In this paper, the ES level (ESL) is divided into three, as in (3). When the viewer's moving speed ranges from 0 to 1, th_{ES1} and th_{ES2} are set to 1/3 and 2/3 of the maximum speed, respectively. The maximum speed of the viewer can be preset according to the theme of the virtual space. For example, the system may set the maximum speed to be low in an art gallery and high in the theme park.

$$ES\,(t) = \frac{\sum_{i=t-WS}^{WS} tuple_{acc}\,(i)}{WS} \quad (ifHS < WS, WS = HS) \tag{2}$$

$$ESlevel = \begin{cases} 0 : Slow & (0 \le ES < th_{ES1}) \\ 1 : Fast & (th_{ES1} \le ES < th_{ES2}) \\ 2 : SuperFast & (th_{ES2} \le ES) \end{cases} \tag{3}$$

When receiving a view request from a client, the server adjusts the quality of the *sub-view* according to the ESL. In Figure 6 (a), the full view consists of four *sub-views*, and the center is assumed to be front. When the head direction of the viewer equals front, the view quality that varies according to the ESL is shown in Figures 6 (b), (c) and (d). When

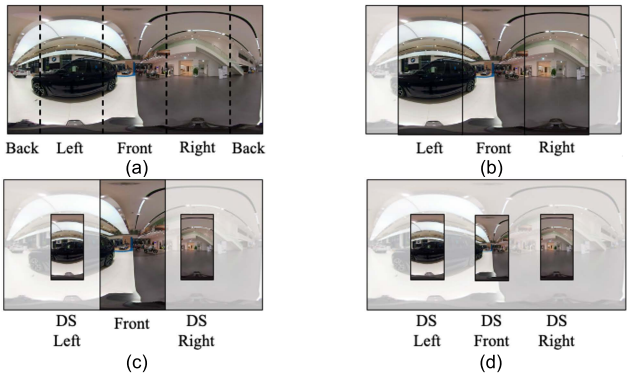

FIGURE 6. Change of viewer's field of vision on a view image (a) four sub-views (b) When ESL= slow (c) When ESL= fast (d) When ESL= super-fast.

ESL=slow in Figure 6 (b), *sub-views* in the left and right directions are sent together with the front. On the other hand, when ESL=fast or super-fast, viewers will not have time to look around. Therefore, *sub-view-ds* is used. When the network bandwidth is insufficient, this reduces T_{comm}, and the JPEG decoding time in T_{render} also decreases. The area that has not been transmitted is filled with black on the client side. Because it covers 270 degrees, which is sufficiently larger than the 135-degree human FoV, the probability that the viewer accidentally sees the area filled with black is low. Hereafter, the partial views of Figures. 6 (b), (c) and (d) are denoted by *3sv*, *1sv-2dsv* and *3dsv*, respectively. The full view of Figure 6 (a) is can also be expressed as *4sv*.

C. MODIFIED GEOMETRIC DISTANCE CACHING CONSIDERING THE EXPLORING SPEED

If GDC and ESS are simply combined, a cache hit is determined only when the viewpoint, head direction and ESL all match. Thus, the miss rate significantly increases compared to the GDC-only case. To avoid this, the criterion for a cache hit is modified in the GDC-ESS integrated system. This is shown as a pseudo code in Figure 7. VP_{req}, HD and $VQ_{req}(H)$ refer to the viewpoint of the requested view, the head direction, and the quality (down-sampled or not) of the *sub-view* in HD, respectively, whereas $VP_{\$}$ and $VQ_{\$}(HD)$ represent the cached view information. For HD, four directions are presented with from 0 to 3. Here, -1 represents counterclockwise rotation such as front to left, whereas $+1$ represents clockwise rotation such as right to back. For VQ, *sub-view-ds* and *sub-view* are indicated as 1 and 2, respectively, and if not in the cache, it is set as 0. First, the views that match the viewpoint of the requested view are searched. If there is no matching viewpoint, it is a miss. When a view with the same viewpoint is found, the case where the difference between $VQ_{\$}$ and VQ_{req} is greater than 0 is counted for three *sub-views* centered on HD. When count=3, it is a full hit, whereas it is a miss when count=0. Otherwise, it is defined as a partial hit. With a partial hit, decoding and rendering start anyway with the view stored in the cache. At the same time, a client requests the necessary *sub-view* from the server and supplements it later. By introducing partial hit, cached view images are flexibly used.

```
Input :   VP_req : a requested viewpoint, HD : a current head direction,
          VQ_req(HD) : a requested sub-view quality of HD,
Output :  stat : cache status
Begin
1.    count ← 0
2.    if VP_req == VP_$
3.        for k=-1 to 1
4.            Rotated_HD= HD+4 (mod 4)
5.            if (VQ_req(Rotated_HD) – VQ_$(Rotated_HD)) >=0
6.                count = count +1
7.        if count == 3
8.            stat ← full hit
9.        else if count < 3 && count >0
10.           stat ← partial hit
11.       else
12.           stat ← miss
13.   else
14.       stat ←miss
      return stat
end
```

FIGURE 7. The process of the decision of modified cache hit condition.

Figure 8 shows the reduced T_{e2e} due to the partial hits. It is assumed that the view with ESL=slow && HD=front is required at the current viewpoint. The cache in any case has a view with the same viewpoint, but the condition of this view is ESL=slow && HD=right. On the client side, the cache table search time and the ESL calculating, decoding, and rendering times are shown in blue, yellow, purple, and gray, respectively. Orange denotes the network communication time in the B5G/6G environment. The time required to prepare the requested view on the server side is denoted in green. The cache miss situation is shown in Figure 8 (a). The client requests *3sv* with ESL=slow to the server. In the experimental environment of this paper as described in Section V, T_{e2e} at this time is 32ms. In the case of the partial hit in Figure 8 (b), the *sub-views* in the front and right directions are decoded first, whereas the *sub-view* in the left direction not in the cache is requested to the server. The number of requests in a partial hit is reduced to one-third compared to a miss. Also, the communication time is hidden by the decoding time. Therefore, T_{e2e} decreases from 32ms to 21ms.

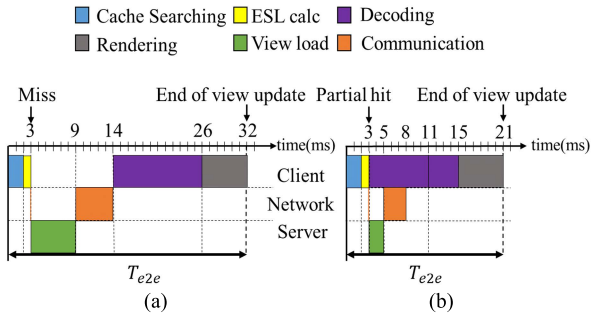

FIGURE 8. Comparison of view update timeline (a) before applying partial hit condition and (b) after applying partial hit condition.

In the GDC-ESS integrated system, the replacement policy also must be modified. In the GDC-only system discussed in Section IV.A, the view that is geometrically distant is evicted through the Euclidean distance and the shortest path search is performed locally. When ESS is applied together, the number of *sub-views* and their VQs of the victim candidates are also considered. For example, it is assumed that two victim candidates v_1 and v_2 are determined according to the conditions of the GDC-only system. v_1 and v_2 were previously visited and cached with ESL=slow && HD=front. Subsequently, v_1 was revisited with ESL =slow && HD=back, and at this point all *sub-views* in four directions were stored in the cache. v_1 with *sub-views* in four directions will be more likely to be a cache hit in the future compared to v_2 where the *sub-views* in three directions are stored. In addition, the view quality VQ is considered. The cache hit probability is higher when the VQs of the *sub-views* in three directions are all full samples than

when the Qs of the *sub-views* in four directions are down-sampled. In practical interactive streaming systems, server and client latency may be greater than expected for various reasons. In this case, the increase in T_{e2e} can be prevented by applying frame skipping.

V. EXPERIMENTAL RESULTS

A. EXPERIMENT ENVIRONMENT

For the experiments, four walking paths are assumed to consider the various exploration scenarios of the viewer. Figure 9 show the top view of a virtual space. The line represents the path. The length of path corresponds to the number of images and it is proportional to the moving distance. Specifically, the solid line illustrates the user's walking trace which is denoted by $s \rightarrow t$ hereafter. s and t are starting and ending points, respectively. The head direction is identical to the viewer's movement direction. Green, orange and red colors denote the ESL=slow, fast and super-fast, respectively. The path denoted as simple_cycle and worst_cycle in Figure 9 (a) and (b) are such that (s→t) x 2, where the path from s to t is visited twice. Among the four walking scenarios, worst_cycle has the longest path. Thus, it takes a long time to go back to the viewpoint already visited before, which is an example of the worst case in terms of caching. maze_trace and museum_trace of Figures 9 (c) and (d) are created based on the actual movement of the user in the virtual space. maze_trace of Figure 9 (c) has many direction changes. Also, if users encounter a dead end ($i_3 \rightarrow u_3$), they must return in the opposite direction($u_3 \rightarrow i_3 \rightarrow i_4$). There are three exhibition rooms in museum_trace of Figure 9 (d). It is assumed that the user takes a tour twice counterclockwise along the walls of each room.

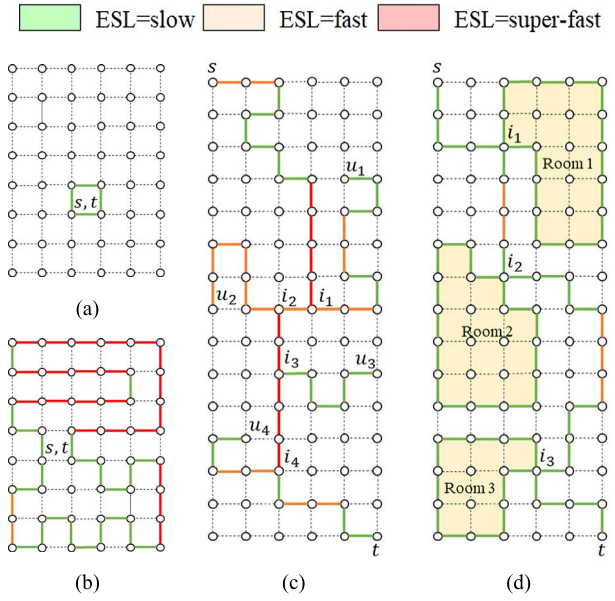

FIGURE 9. Walking scenarios in virtual space (a) *simple_cycle* (b) *worst_cycle* (c) *maze_trace* (d) *museum_trace*.

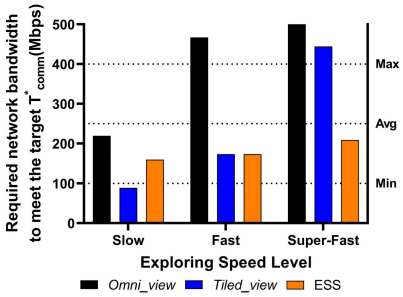

FIGURE 10. Estimated T_{e2e} comparison based on required network bandwidth.

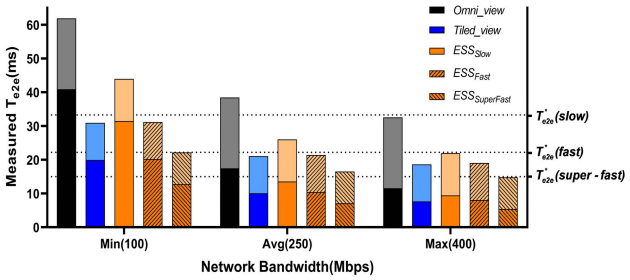

FIGURE 11. Measured T_{e2e} comparison.

In the experiments shown in Figures 10 and 11, the server and the client run on a single computer to minimize various variables that may occur in the real network delivery process and to see the maximum possibility of each approach in an ideal network environment. Figures 12 and 13 show the cache hit rate of the client and the number of requests to the server. It is a value determined by the walking scenario and request policy, not by the network situation. Thus, the server and client are run on a single computer as well. In Figures 14 and 16, each server and client communicate through the network. The specifications of computing platform used in the experiment are as follows: an Intel Core i7-7700K, 48GB of RAM, a Crucial MX300 CT525MX300SSD1 525GB solid-state drive (SSD), and a NVIDIA GTX 1060 card with 3GB of memory. In Figure 14, the server and client run on independent laptops and communicate through a wired network assuming an ideal wireless network environment. Its bandwidth is controlled using a traffic control application [44]. In the experiment shown in Figure 16, the performance is measured on an actual 5G network. The specifications of the server are as follows: an Intel Core i5-8300H, 32GB of RAM and a NVIDIA GTX 1050 graphics card, whereas the specifications of the client are as follows: of Intel Core i7-9750H, 16GB of RAM, a NVIDIA GTX 1660Ti graphics card. For JPEG decoding, the libjpeg-turbo library [45] is used. Up-sampling is implemented with a simple nearest-neighbor interpolation algorithm. In the system implementation that utilizes the schemes proposed in Section IV, the chunk of views, not a single view, is used as a processing unit in consideration of the spatial

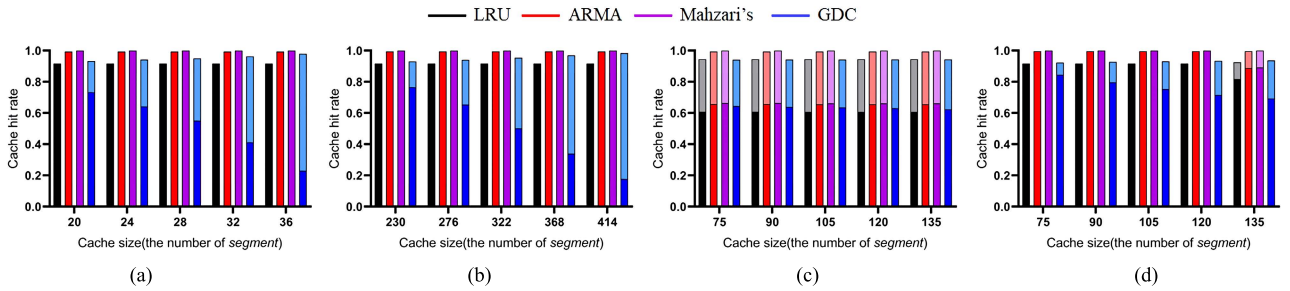

FIGURE 12. Comparing the reduction of cache penalty for client as the cache hit rate for four walking scenarios (a) ***simple_cycle*** (b) ***worst_cycle*** (c) ***maze_trace*** (d) ***museum_trace***.

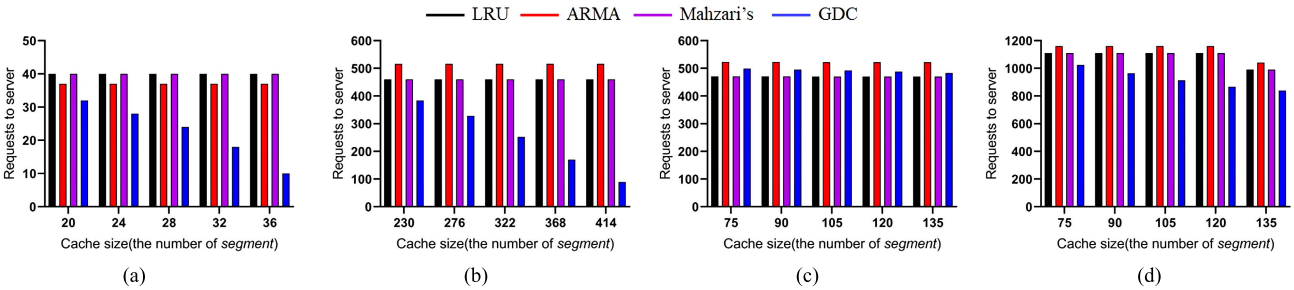

FIGURE 13. Comparing the reduction of cache penalty for server as the number of server requests for four walking scenarios (a) ***simple_cycle*** (b) ***worst_cycle*** (c) ***maze_trace*** (d) ***museum_trace***.

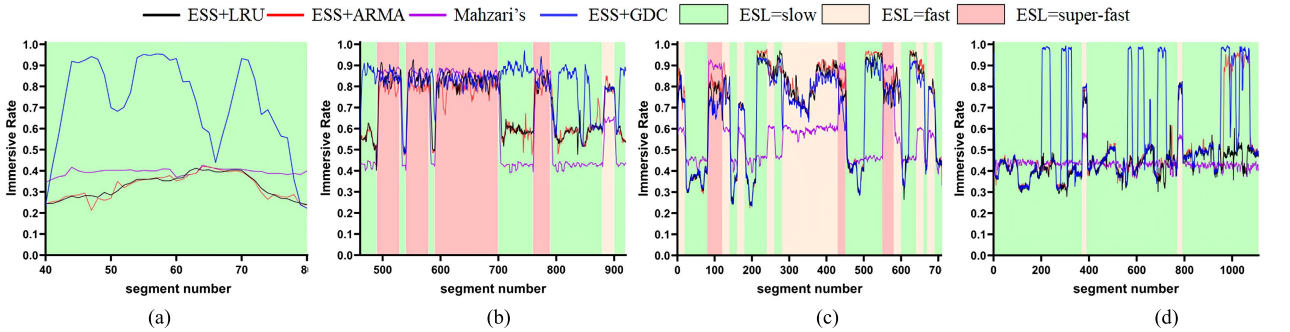

FIGURE 14. Immersive rate depending on four schemes, which are ESS+LRU, ESS+ARMA, Mahzari's and ESS+GDC, for four walking scenarios (a) ***simple_cycle*** (b) ***worst_cycle*** (c) ***maze_trace*** (d) ***museum_trace***.

locality of the viewer's movement. In the experiment, one side of each square that forms a grid-shaped path consists of 120 images in 4K resolution and this is divided into multiple *segments*. Hereinafter, the *segment* is the unit of transmission and caching. It is assumed that there are no changes in the walking direction or speed within the *segment* unit. For GDC, the view prefetch technique is not used. The thresholds of ''slow'', ''fast'' and ''very fast'' movements are determined as follows. A device such as a treadmill that can receive a walking or running motion as an input is rare. Therefore, the maximum speed that can be reached by a general controller such as a joystick is set to ''very fast'', and 1/3 and 2/3 of the ''very fast'' are set to ''slow'' and ''fast'', respectively.

B. PERFORMANCE EVALUATION OF ESS

In Figure 10, Omni_view transmits and renders the full-view covering 360 degrees. This is a simple and basic scheme, but the data transmission amounts are large. In Tiled_view, the amount of transmitted data is reduced by considering the head direction [5]–[11]. However, the viewer's speed is not taken into account. The proposed ESS considers both viewer's speed and field of vision. Tiled_view in this experiment is set identically to *1sv-2dsv* of ESS, where full-view is divided into four tiles (=*sub-views*) and one *sub-view* and two *sub-view-ds* are transmitted. The proposed ESS flexibly adjusts the amount of data transmitted in three types: *3sv*, *1sv-2dsv*, and *3dsv*, reflecting both head direction and speed. For the viewer's QoE, it is assumed that the target T^*_{e2e} should not exceed 33, 22 and 15 ms when ESL=slow, fast and super-fast, respectively. In ESL=slow, a typical update rate of 30 frames-per-second (=33 ms) was set as the baseline. When the user is in ESL=fast or super-fast, it is set to be 1.5 and 2 times the update rate of ESL=slow. This is based on the information that the average walking speed of a person is 3 to 4.5 km/h and

the running speed is about 8 to 12 km/h. Focusing on communication, each ESL has the target communication delay T^*_{comm}, which is determined by subtracting T_{render} and T_{misc}, from the target T^*_{e2e} of each ESL. Compared to T^*_{comm}, T_{misc} and T_{render} are relatively constant. Based on empirical data, T_{misc} is set to 1ms, whereas T_{render} is set to 20ms, 11.5ms, 10ms and 8.4ms on average when full-view (=*4sv*), *3sv*, *1sv-2dsv* and *3dsv*, respectively. When considering image transmission in JPEG format, the size of full-view (=*4sv*), *3sv*, *1sv-2dsv* and *3dsv* are approximately set to be 480KB, 360KB, 200KB and 120KB, respectively. Given the viewer's ESL, the required network bandwidths of three schemes can be calculated as follows.

$$Required\ bandwidth = data\ size\ (Mbit) \times \frac{1000}{T^*_{comm}(ms)} \quad (4)$$

If the network environment is sufficient enough to meet the required network bandwidth and the transmission delay does not exceed T^*_{comm}, images can be streamed without QoE degradation. In Figure 10, the horizontal axis represents ESL, whereas the left vertical axis represents the required network bandwidth to meet the target T^*_{comm}. For reference, the minimum (100 Mbps), average (250 Mbps) and maximum (400Mbps) network bandwidth supported in the actual environment of the 5G sub6 band [46] are indicated by the three dotted lines in the figure. The performance of the three schemes is analyzed based on the viewer's movement and the required network environment. When the ESL is fast, the view update rate should be high. At this time, the required network bandwidth will increase accordingly. Omni_view, indicated by the black bars, is able to satisfy the target T^*_{comm} only when ESL=slow under the avg-or-above network environment. In Tiled_view, indicated by blue bars, images are smoothly streamed when ESL=slow or when ESL=fast under the avg-or-above network environment. However, when ESL=super-fast, the required network bandwidth inevitably increases as the viewer moves faster. It is because that Omni_view and Tiled_view do not have a solution capable of achieving the target T^*_{comm} that fits the viewer's motion characteristics. On the other hand, ESS, indicated by the orange bars, responds flexibly to the viewer's various speeds given the avg-or-above network environment. From this result, it is expected that the proposed ESS will be able to meet the target T^*_{comm} for all ESL conditions in a 5G sub6 bands bandwidth environment with an average or higher. Although not presented in the experiment, the ESS can also adaptively handle the min network environment through additional down-sampling.

In Figure 11, T_{e2e} taken for streaming in a directly wired network environment is measured and compared with the estimated performance in Figure 10. The horizontal axis represents the given network bandwidth. As in Figure 10, min, avg and max denote the representative network bandwidth supported in the actual environment of the 5G sub6 band [46]. For the experiment, the bandwidth is controlled by the aforementioned traffic control application. The vertical axis represents the measured T_{e2e}. For reference, the target T^*_{e2e} of each ESL is indicated by the three dotted lines in the figure. Omni_view, Tiled_view and the proposed ESS are illustrated in black, blue and orange, respectively. For each bar, the dark color denotes T_{comm}, whereas the light color denotes the sum of T_{render} and T_{misc}. In the min bandwidth, as expected in Figure 10, only Tiled_view barely satisfies the target $T^*_{e2e}(slow)$. When the network bandwidth is increased to the avg level, the measured T_{e2e} values of Omni_view and $ESS_{SuperFast}$ slightly exceed the estimation in Figure 10. At the max bandwidth, the measured T_{e2e} is nearly identical to the estimated performance in Figure 10. This experiment shows that the proposed ESS adapts more flexibly to various network environments while also satisfying the target T_{e2e} compared to Omni_view and Tiled_view.

C. PERFORMANCE EVALUATION OF GDC

In this subsection, the cache performance is analyzed when the proposed GDC is applied in addition to ESS. In a cache based interactive streaming system, there are two types of miss penalties: the view update delay on the client side and the request handling burden on the server side. On the client side, the cache hit rate contributes to the short response time. A prefetching request with a high cache hit rate will be preferred over an on-demand request in terms of latency. However, from the server's point of view, the client's cache hit rate has nothing to do with the server's request handling load. What matters is the number of requests. If the prediction accuracy is 100%, only the necessary data will be requested to the server in prefetching exactly like on-demand. However, if the prediction fails, an additional request is made for the data needed immediately, which increases the redundant request overhead for the server. Therefore, despite the high cache hit rate which is good for quick response time for clients, the server's workload can be high due to prediction failure. If the path is complicated and difficult to predict, the request overhead that the server has to deal with increases further. To reduce this, data reuse on the client side plays an important role, mostly depending on the cache replacement policy.

To show that the proposed GDC is suitable for streaming systems for multiple users, it is compared with dead reckoning with an autoregressive moving average (ARMA) filter replacement strategy [47], least-recently-used (LRU) policies and Mahzari's approach [23]. LRU evicts the oldest viewpoint based on the viewer's visit time. ARMA predicts the view position based on history, with unlikely viewpoints replaced according to the LRU policy. ARMA predicts the future viewpoint based on the past 10 viewpoints. In the study by Mahzari, FoV-aware edge caching is applied to adaptive 360-degree video streaming. This approach shows very high FoV prediction accuracy rates with the view quality adjusted according to the network bandwidth. In the experiment in this study, Mahzari's prediction accuracy is set to 100% assuming the best situation. However, due to its adaptiveness to the network bandwidth, a down-sampled low-quality view must be

transmitted under a 100Mbps condition. Mahzari's approach also uses the LRU caching policy, similar to ARMA.

Figure 12 shows the cache hit rate depending on the cache size on the client side. Cache updating is conducted in *segment* units to utilize the concept of spatial locality. Four walking scenarios are tested. Bars in black, red, purple and blue represent the LRU, ARMA, Mahzari's and GDC schemes, respectively. In each bar, the light color denotes the ratio of hits by data reuse out of all cache hits. The basic LRU has a low hit rate, and there is no data reuse at all or little data reuse, as shown in Figures 12 (a), (b) and (d). The data reuse in Figure 12 (c) is due to the partial hit when returning from u_k to i_k. ARMA shown in red, has a high hit rate regardless of the cache size. In ARMA which utilizes the history-based prediction, a cache miss occurs only when the viewer changes direction, i.e., turns a corner in the four walking scenarios. However, like LRU, there may be no or little data reuse except Figure 12 (c). Mahzari's method with 100% prediction accuracy always shows a cache hit rate of 1. However, due to the LRU-based cache replacement policy, it has a weakness in terms of data reuse rate, like LRU and ARMA. On the other hand, the proposed GDC schemes reflects the geometric characteristics of path walking with cycles, thereby increasing the data reuse rate with the cache size in all walking scenarios. In particular, in Figures 12 (c) and (d) created from the user's actual movement record, GDC shows a similar or higher data reuse rate compared to that of the other three schemes even with a small cache size. The analysis of the results in Figure 12 is as follows. When the cache size is not large enough and the user only goes forward without looking back along the cycle path, conventional cache replacement techniques that do not consider geometry characteristics will acts as FIFOs. All viewpoints that are visited before one-cycle are replaced and data cannot be reused in a new cycle. In the case of returning in the opposite direction on the same path, transmission data is reduced by requesting only a part of the sub-view to the server. In terms of the overall cache hit rate, ARMA and Mahzari's approaches are superior to the proposed GDC. However, as shown in Figure 11, the proposed system sufficiently satisfies the target T^*_{e2e} at the average network bandwidth due to ESS, reducing the cache miss penalty on the client side and thereby lowering the impact of the cache hit rate on the streaming performance.

The real strength of GDC lies in how it reduces the server burden by reducing the number of requests through a high data reuse rate. In particular, in a multi-user environment, the frequent view update requests affect the response time of the server, consequently lowering the viewer's QoE. Figure 13 shows the number of server requests in the environment identical to that in Figure 12. In contrast to the high hit rate in Figure 12, the LRU, Mahzari's and ARMA schemes show far more requests than GDC, especially in the walking paths in Figures 13(a), (b) and (c), which have cycles. These schemes disregard the geometry-based data reuse and mainly depend on spatial locality via prefetching. In blue GDC, the number of requests decreases as the cache hit rate of Figure 12 increases. In addition, in walking scenarios with cycles, the number of requests can be drastically reduced as the cache size increases. For a clearer evaluation of the performance of the cache replacement policy and how it affects the QoE, an immersive rate is defined considering both the view quality and T_{e2e}. The immersive rate, ranging from 0 to 1, is calculated by multiplying the latency rate and the quality rate, as shown in (5).

$$\text{Immersive rate} = \text{quality rate} \times \text{latency rate} \tag{5}$$

Here, the latency rate is the normalized difference between the target T^*_{e2e} and actual T_{e2e} under the given ESL. The target T^*_{e2e} is the latency to be satisfied for each ESL, illustrated by the three dotted lines shown in Figure 11. In the real environment, the change in response time of the server may vary depending on various factors. There was a difficulty in setting up an experimental environment that accurately reflects the effect of response time. To model the change in response time according to the number of requests, the following conditions are used in the experiment. In the server side, a multi-user server environment is assumed. If the request throughput exceeds a certain level, the response time may increase rapidly. This server has a normal response time of 2-4ms, but the peak response time can be increased by about 10 times, up to 29-31ms. A client experiences peak delay every N request due to the occasional increase in request throughput in a multi-user server environment. In other words, N denotes the period of occurrence of peak delay. When data is reused in the client, the load on the server is reduced, increasing to N = N + a, and the client will experience the peak delay less frequently. On the other side, prediction failure from the client increases the amount of data requested to the server, reducing to N = N - b, and the client will experience the peak delay more often. For the experiment, the initial N is about 10, whereas a and b are set to 5.

The quality rate represents the ratio of the actual *sub-view* quality to the expected *sub-view* quality for the four directions under the given ESL. The quality rate calculation follows the rules below. The quality of *sub-view* and *sub-view-ds* is represented by 2 and 1, respectively, and different weights are given to four *sub-views* according to the viewer's head direction. The viewer's head direction and the two adjacent directions have weights of 0.5 and 0.25, respectively, whereas the back direction has a weight of 0. For example, in the current scheme, head direction=front and ESL=fast, whereas the cached view of the same viewpoint has head direction=right and ESL=fast. The quality rates of the left, front, and right directions are 0/1, 1/2, and 2/1, respectively. Because the back direction is not visible, it is not considered. The quality rate of each *sub-view* has a maximum value of 1. Therefore, the sum of the weighted quality rates in the left, front and right directions is $0.25 \times 0 + 0.5 \times 0.25 + 0.25 \times 1 = 0.5$.

In Figure 14, the network bandwidth is set to 100Mbps using a traffic control application [44]. The green, orange, and red areas in the graph correspondingly represent ESL=slow,

fast and super-fast, which follows the walking scenarios in Figure 9. The horizontal axis represents the *segment* number, whereas the vertical axis represents the immersive rate of each *segment.* The immersive rate is averaged using a sliding window algorithm with a window size of 4. In Figures 14 (a) to (d), four walking scenarios are tested. The cache size was set to 384 in simple_cycle and 4416 in worst_cycle, whereas it was 1440 in other two scenarios. The *segment* size is set to 12. Four schemes are compared. LRU, GDC, ARMA combined with ESS and Mahzari's approach are denoted by black, blue, red and purple, respectively. As already confirmed in Figure 11, the proposed ESS greatly reduces the effect of a cache miss penalty on the network bandwidth above the average level. Moreover, as shown in Figure 13 and Figures 14, the proposed GDC can more effectively reduce the server request burden compared to other replacement policies, in this case LRU, ARMA and Mahzari's method. The experimental results show that ESS+GDC achieves a stable immersive rate in four different walking scenarios.

The moving pattern in Figure 14 (a) is simple and ESL is always slow. In terms of the quality rate, three schemes combined with ESS have a value of 1 because the expected and the actual view quality levels are identical. However, Mahzari's approach, which adjusts the quality level via the network bandwidth rather than the viewer's speed, has a quality rate of 0.5 due to the narrow bandwidth of 100Mbps. Meanwhile, in terms of the latency rate, during the first one-cycle, T_{e2e} of Mahzari's method is smallest compared to the other schemes because it transmits a low-quality view. Therefore, with a high probability, its latency rate is 1. However, after the first one-cycle which brings about a cold start miss, the proposed ESS+GDC method has a high data reuse rate, as shown in Figures 12 and 13. When the *segment* is reused, T_{comm} of T_{e2e} disappears and the appearance of the peak response time is consequently reduced. This makes the latency rate high. On the other hand, in ESS+LRU and ESS+ARMA, there is no increase in the latency rate even after the first one-cycle because they have no data reuse. From this analysis, the advantage of the proposed scheme can be explained as follows. Under a situation in which the viewer slowly looks around the space, in the first one-cycle, all schemes have a similar immersive rate. However, past that point, the proposed ESS+GDC is able to provide a high level of QoE while showing a higher immersive rate compared to the other schemes.

Figure 14 (b) shows the immersive rate in the worst case, in which the moving pattern is complex and the revisit period, one-cycle, is very long. After the first one-cycle, the proposed ESS+GDC has a high reuse rate of *segments*. Accordingly, it achieves a higher immersive rate than the other schemes. Figures 14 (c) and (d) have small cache size and complicated moving pattern. Figure 14 (c) shows a variety of ESLs, with data reuse observed in all schemes unlike simple_cycle in Figure 14 (a). In terms of the quality rate, all three schemes combined with ESS have a value of 1. Mahzari's approach has

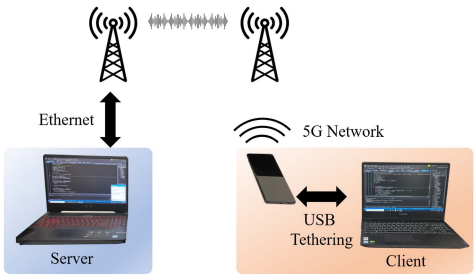

FIGURE 15. Commercial 5G Network experiment setup.

values of 0.5, 0.66 and 1 respectively when ESL=slow, fast and super-fast. When comparing the latency rate, all schemes show a similar T_{e2e} when ESL=super-fast. In the other ESL cases, Mahzari's approach has the smallest T_{e2e}. Therefore, this method achieves better immersive rate than the others when in the absence data reuse and when ESL=super-fast. Meanwhile, in ESS+ARMA, the server request burden is greatly increased when the prediction fails at the corner of the walking path; thus, the immersive rate in this case fluctuates somewhat. ESS+LRU shows an immersive rate similar to that of ESS+GDC in Figure 14 (c), where partial hit occurs. On the contrary, ESS+GDC shows stable and high immersive rates in Figure 14 (a), (b) and (d). Therefore, ESS+GDC achieves a similar or high immersive rate to the existing caching policy in the real-trace-based scenarios.

D. SYSTEM DEMONSTRATION ON THE 5G NETWORK

Figure 15 illustrates our experimental setup for evaluating the proposed system in a commercial 5G network, which uses a 5G new radio (NR) with a 3.5 GHz carrier frequency (FR1). A Samsung Galaxy S21 Ultra smartphone is used to connect to the 5G network. The server and client are devised using two laptops. The server is connected to an external network with an Ethernet cable, and the downlink throughputs are 300Mbps, 450Mbps and 600Mbps in the minimum, average and maximum cases, respectively. The client is connected to the 5G smartphone with a USB cable and communicates with the server through USB tethering. The Client's downlink throughput is 130Mbps, 200Mbps, and 400Mbps in the minimum, average, and maximum cases, respectively. The round-trip-time (RTT) between the client and server was measured by ping and it was approximately 46ms. T_{e2e} of the following experimental results includes this RTT.

The measured T_{e2e} in the environment of Figure 15 is presented in Figure 16. The horizontal axis represents the transmitted *segment*, whereas the vertical axis represents the average T_{e2e} of each *segment* measured through three experimental trials. The four walking scenarios in Figure 9 are used. Unlike Figure 14, in Figure 16, sudden high latency is observed. This occurs because the 5G network changes to a 4G network when the connection is unstable. Although it was difficult to obtain an error-free result in the real environment, it was confirmed that the trend and patten of the measured

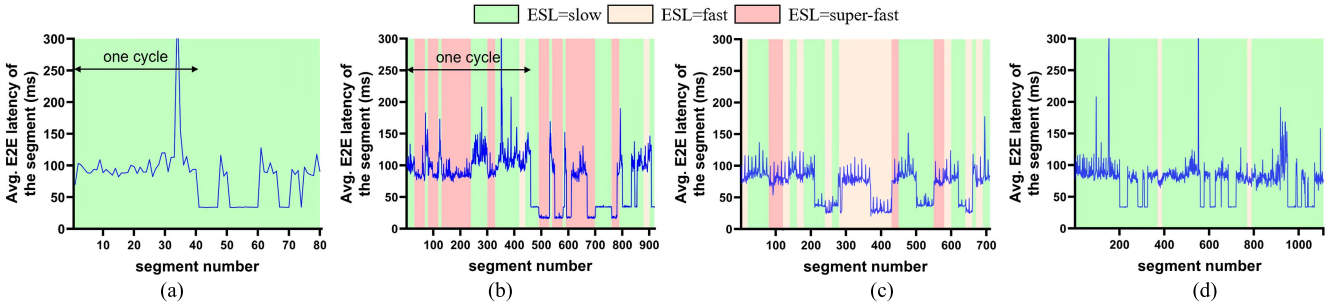

FIGURE 16. E2E latency per segment for four walking scenarios (a) ***simple_cycle*** (b) ***worst_cycle*** (c) ***maze_trace*** (d) ***museum_trace*** in real 5G network.

results were generally consistent with the expected results. In Figures 16 (a) and (b), there is no view to reuse before first one-cycle of walking is completed. Given that there is no cached view, all necessary views must be requested from the server. Frequent requests increase the probability of a stall due to the RTT. For the slow, fast and super-fast schemes, T_{e2e} is approximately 100ms, 90ms and 80ms, respectively. From the second cycle, cached views are not requested from the server. In Figure 16 (b), most required views are partial hits. Here, only one or two *sub-views* are requested from the server, with less than three *sub-views* requested when a cache miss. This leads to latency close to the target T^*_{comm} required by ESL, even under the non-ideal environment of a commercial 5G network. In Figure 16 (c), data reuse occurs because there are three exhibition rooms with two cycles. The path-walking VR tour system here guarantees low latency streaming to provide a highly immersive experience to viewers in a network environment with a bandwidth of 250Mbps or more as shown in Figure 16. Therefore, this experiment confirms that the proposed technology can be smoothly applied to commercial products if the VR service becomes a full-fledged service in the 5G NR FR2 network environment.

VI. CONCLUSION

This paper proposes the low latency view streaming for a path-walking VR system. The main contributions are as follows: i) the amount of transmitted data is adjusted according to the user's moving speed and ii) geometry-based caching is adopted to reduce the view-retransmitting overhead and the server's request burden. iii) Two schemes are harmoniously integrated by modifying the cache hit condition. The proposed schemes show stable viewer immersive score relative to other schemes that do not consider the concept of geometrical locality. Path-walking VR systems with the proposed schemes are practical and have high potential for commercialization. In addition, the approach here can easily be combined with previously published latency techniques [22], [24]–[26] for mobile networks. It is expected that the results of this paper will provide insight to those who design a B5G/6G systems for immersive media applications.

REFERENCES

[1] *Google Arts & Culture*. Accessed: Mar. 29, 2021. [Online]. Available: https://artsandculture.google.com/

[2] *The National Gallery, Google Virtual Tour*. Accessed: Mar. 29, 2021. [Online]. Available: https://www.nationalgallery.org.U.K./visiting/virtual-tours/google-virtual-tour

[3] *Create a Virtual Tour*. Accessed: Mar. 29, 2021. [Online]. Available: https://arvr.google.com/tourcreator/

[4] B. Marsh, B. Lautner, L. Fournier, J. Klang, P. Anelli, W.-E. Kang, U. Brumec, and K. Cota, *The Role of Road Engineering in Combatting Driver Distraction and Fatigue Road Safety Risks*. Accessed: Oct. 11, 2021. [Online]. Available: https://www.doria.fi/handle/10024/133674

[5] J. Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," in *Proc. ACM Int. Conf. Multimedia Syst. (MMSys)*, May 2016, pp. 1–3.

[6] C. Zhang, Q. He, J. Liu, and Z. Wang, "Exploring viewer gazing patterns for touch-based mobile gamecasting," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2333–2344, Oct. 2017.

[7] J. Song, F. Yang, W. Zhang, W. Zou, Y. Fan, and P. Di, "A fast FoV-switching DASH system based on tiling mechanism for practical omni-directional video services," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2366–2381, Sep. 2020.

[8] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, Apr. 2011.

[9] C. Concolato, J. L. Feuvre, F. Denoual, F. Mazé, E. Nassor, N. Ouedraogo, and J. Taquet, "Adaptive streaming of HEVC tiled videos using MPEG-DASH," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 8, pp. 1981–1992, Aug. 2018.

[10] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An HTTP/2-based adaptive streaming framework for 360° virtual reality videos," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 306–314.

[11] F. Qian and B. Han, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2018, pp. 99–114.

[12] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1252–1260.

[13] X. Feng, V. Swaminathan, and S. Wei, "Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 3, no. 2, pp. 1–22, Jun. 2019.

[14] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proc. 27th Workshop Netw. Oper. Syst. Support Digit. Audio Video.*, Jun. 2017, pp. 67–72.

[15] M. Hu, J. Chen, D. Wu, Y. Zhou, Y. Wang, and H.-N. Dai, "TVG-streaming: Learning user behaviors for QoE-optimized 360-degree video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 4107–4120, Oct. 2020.

[16] Z. Xu, X. Zhang, K. Zhang, and Z. Guo, "Probabilistic viewport adaptive streaming for 360-degree videos," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.

[17] W. Quan, Y. Pan, B. Xiang, and L. Zhang, "Reinforcement learning driven adaptive VR streaming with optical flow based QoE," in *Proc. IEEE 18th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2020, pp. 231–236.

[18] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *Proc. Special Interest Group Data Commun.*, Aug. 2019, pp. 394–407.
[19] C.-L. Fan, S.-C. Yen, C.-Y. Huang, and C.-H. Hsu, "Optimizing fixation prediction using recurrent neural networks for 360° video streaming in head-mounted virtual reality," *IEEE Trans. Multimedia*, vol. 22, no. 3, pp. 744–759, Mar. 2019.
[20] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 315–323.
[21] J. Chen, X. Luo, M. Hu, D. Wu, and Y. Zhou, "Sparkle: User-aware viewport prediction in 360-degree video streaming," *IEEE Trans. Multimedia*, vol. 23, pp. 3853–3866, 2021.
[22] X. Zhang, X. Hu, L. Zhong, S. Shirmohammadi, and L. Zhang, "Cooperative tile-based 360° panoramic streaming in heterogeneous networks using scalable video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 1, pp. 217–231, Jan. 2018.
[23] A. Mahzari, A. T. Nasrabadi, A. Samiei, and R. Prakash, "FoV-aware edge caching for adaptive 360° video streaming," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 173–181.
[24] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360° videos," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jul. 2019, pp. 171–180.
[25] P. Maniotis and E. Bourtsoulatze, "Tile-based joint caching and delivery of 360° videos in heterogeneous networks," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2382–2395, Sep. 2020.
[26] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7573–7586, Nov. 2019.
[27] J. Park, P. A. Chou, and J.-N. Hwang, "Rate-utility optimized streaming of volumetric media for augmented reality," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 149–162, Mar. 2019.
[28] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan, "Toward practical, volumetric video streaming on commodity smartphones," in *Proc. 20th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2019, pp. 135–140.
[29] M. Hosseini and C. Timmerer, "Dynamic adaptive point cloud streaming," in *Proc. 23rd Packet Video Workshop*, Jun. 2018, pp. 25–30.
[30] R. Diniz, "Volumetric video capture, object reconstruction and telecommunication methods for mixed reality experiences," Ph.D. dissertation, Dept. CS., Brasília Univ., Brasília, Brazil, 2018.
[31] J. Jeong, D. Jang, J.-W. Son, and E.-S. Ryu, "Bitrate efficient 3DoF+ 360 video view synthesis for immersive VR video streaming," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2018, pp. 581–586.
[32] M. Zampoglou, K. Kapetanakis, A. Stamoulias, A. G. Malamos, and S. Panagiotakis, "Adaptive streaming of complex web 3D scenes based on the MPEG-DASH standard," *Multimedia Tools Appl.*, vol. 77, no. 1, pp. 125–148, Jan. 2018.
[33] G. Lavoué, L. Chevalier, and F. Dupont, "Streaming compressed 3D data on the web using Javascript and WebGL," in *Proc. 18th Int. Conf. 3D Web Technol. (Web3D)*, Jun. 2013, pp. 19–27.
[34] J. Park, P. A. Chou, and J.-N. Hwang, "Volumetric media streaming for augmented reality," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[35] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, and V. Tankovich, "Holoportation: Virtual 3D teleportation in real-time," in *Proc. 29th Annu. Symp. User Interface Softw. Technol.*, Oct. 2016, pp. 741–754.
[36] M. Liubogoshchev, E. Korneev, and E. Khorov, "EVeREst: Bitrate adaptation for cloud VR," *Electronics*, vol. 10, no. 6, p. 678, Mar. 2021.
[37] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "Multi-path multi-tier 360-degree video streaming in 5G networks," in *Proc. 9th ACM Multimedia Syst. Conf.*, Jun. 2018, pp. 162–173.
[38] L. He, W. Zhu, K. Zhang, and Y. Xu, "View-dependent streaming of dynamic point cloud over hybrid networks," in *Proc. 19th Pacific Rim Conf. Multimedia*, Sep. 2018, pp. 50–58.
[39] S. Gül, D. Podborski, T. Buchholz, T. Schierl, and C. Hellge, "Low-latency cloud-based, volumetric video streaming using head motion prediction," in *Proc. 30th ACM Workshop Netw. Oper. Syst. Support Digit. Audio Video*, Jun. 2020, pp. 27–33.
[40] P. A. Kara, A. Cserkaszky, M. G. Martini, A. Barsi, L. Bokor, and T. Balogh, "Evaluation of the concept of dynamic adaptive streaming of light field video," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 407–421, Jun. 2018.
[41] C. Birklbauer, S. Opelt, and O. Bimber, "Rendering gigaray light fields," *Comput. Graph. Forum*, vol. 32, no. 2, pp. 469–478, May 2013.
[42] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, and P. E. Debevec, "A system for acquiring, processing, and rendering panoramic light field stills for virtual reality," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, Nov. 2018.
[43] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. Silva, "VR is on the edge: How to deliver 360 videos in mobile networks," in *Proc. Workshop Virtual Reality Augmented Reality Netw.*, Aug. 2017, pp. 30–35.
[44] *NetLimiter 4*. Accessed: Nov. 7, 2020. [Online]. Available: https://www.netlimiter.com/
[45] *LibJPEG-Turbo*. Accessed: Nov. 5, 2020. [Online]. Available: https://libjpeg-turbo.org/
[46] *We Ran 5G Speed Tests on Verizon, AT&T, EE and More: Here's What we Found*. Accessed: Nov. 11, 2020. [Online]. Available: https://www.cnet.com/features/we-ran-5g-speed-tests-on-verizon-att-ee-and-more-heres-what-we-found/
[47] P. Ramanathan, M. Kalman, and B. Girod, "Rate-distortion optimized interactive light field streaming," *IEEE Trans. Multimedia*, vol. 9, no. 4, pp. 813–825, Jun. 2007.

WON-KI SEO received B.S. degree in computer engineering from Inha University, Incheon, South Korea, in 2019, where he is currently pursuing the master's degree with the Department of Information and Communication Engineering. His research interests include low latency streaming and light field processing.

CHAE EUN RHEE (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2000, 2002, and 2011, respectively. From 2002 to 2005, she was an Engineer at the Digital TV Development Group, Samsung Electronics Company Ltd., Suwon City, South Korea, where she was involved in bus architecture and MPEG decoder development. In 2013, she joined the Department of Information and Communication Engineering, Inha University, South Korea, where she is currently working as a Professor. Her research interests include algorithm and architecture design of video coding for HEVC and H.264/AVC, configurable video coding for real time systems, and the next generation virtual reality systems.