Volumetric Media Streaming for Augmented Reality

Jounsup Park[1], Philip A. Chou[2], Jenq-Neng Hwang[1]
[1]Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA
Email : {jsup517, hwang}@uw.edu
[2]8i Labs, Inc., Bellevue, WA, USA
Email : phil@8i.com

***Abstract* – Volumetric media, popularly known as holograms, need to be delivered to users using both on-demand and live streaming, for new augmented reality (AR) and virtual reality (VR) experiences. As in video streaming, hologram streaming must support network adaptivity and fast startup, but must also moderate large bandwidths, multiple simultaneously streaming objects, and frequent user interaction, which requires low delay. In this paper, we introduce the first system designed specifically for streaming volumetric media. The system reduces bandwidth by introducing 3D tiles, and culling them or reducing their level of detail depending on their relation to the user's view frustum and distance to the user. To allocate bits among different tiles across multiple objects, we introduce a simple greedy yet provably optimal algorithm for rate-utility optimization, whose utility measures is based not only on the underlying quality of the representation, but on the level of detail relative to the user's viewpoint and device resolution. Simulation results show that the proposed algorithm provides superior quality compared to existing video-streaming approaches adapted to hologram streaming, in terms of utility and user experience over variable, throughput-constrained networks.**

Keywords – Hologram, voxelized point cloud, client buffer management, DASH, tile, free-viewpoint video

I. INTRODUCTION

From the first release of commercial video on demand (VoD) streaming systems over the Internet in 1997 until the present, streaming video on demand has grown exponentially along several dimensions. The first systems streamed 176x144 (QCIF) video at 15 frames per second (fps), over 56 Kbps modems, compressed to 40 Kbps. Today, 1920x1080 (Full HD) video at 30 fps is regularly streamed to broadband users at 20 Mbps, thanks to a doubling of the last mile bandwidth to the consumers every 2.5 years, and a doubling of the compression quality every 10 years, or faster. During the same period, the Internet backbone capacity has increased 40% annually to about 300 Tbps, not including the proprietary networks of content providers such as Amazon, Facebook, Google, and Microsoft **Error! Reference source not found.**. Cache-based content distribution networks (CDNs), originally developed for web traffic, were leveraged by basing streaming video protocols on HTTP, and making video servers look like stateless web servers, both for on-demand and live streaming [2]. The combination of these developments has positioned over-the-top (OTT) streaming VoD to become the dominant means for consuming video, in any form, worldwide. The upcoming deployment of 5G will cement this position by commanding mobile platforms as well [3].

While OTT live and on-demand streaming video are on their way to becoming the dominant way of delivering video, new forms of immersive media have recently been born, offering experiences well beyond ordinary video. Such new forms of immersive media include *spherical video* for virtual reality (VR), and *volumetric media*, popularly known as *holograms*, for augmented reality (AR) as well as VR. These new media will also be streamed live and on demand [4].

Just like streaming ordinary video, streaming spherical video must be *network-adaptive*, that is, adaptive to network channel variability [5][6][7][8]. For this aspect of streaming spherical video, the same network-adaptive techniques used for streaming ordinary video can be used: buffering [9], monitoring and predicting the state of the network [10][11][12], and selecting representations at the appropriate bit rates to stream in order to maximize the quality perceived by the users [13].

Streaming volumetric media, or holograms, for AR or VR must handle even higher levels of user interaction than streaming spherical video for VR [14]. Since holograms support full 6DOF, or free-viewpoint, not only may a user turn his/her head to change the view direction, but may also navigate freely among a multitude of holograms, changing both view direction and position in space, in the process potentially changing the proximity to the various holograms over a wide range of distances, and changing the direction from which he/she sees the holograms.

In this paper, we address the problems of streaming holograms in AR/VR, using novel approaches to accommodate the higher level of user interactivity. We extend the notion of 2D tiles [15][16] used in spherical video to 3D tiles for holograms, so that different tiles may have different resolutions depending on where the user is positioned and looking relative to each tile, thus saving bandwidth by focusing on what the user is looking at. While the concept of 3D tiles may seem straightforward, there are several complicating issues. First, many of the 3D tiles, as regions of space, are empty over a significant part of the time [17][18]. Thus, even if the user viewpoint is not changing, the set of tiles occupied by content and visible to the user is content-dependent and changing. Especially considering that there can be many more tiles in 3D than in 2D, this requires an efficient way to index and address the tiles for holograms in AR/VR, while not needed for

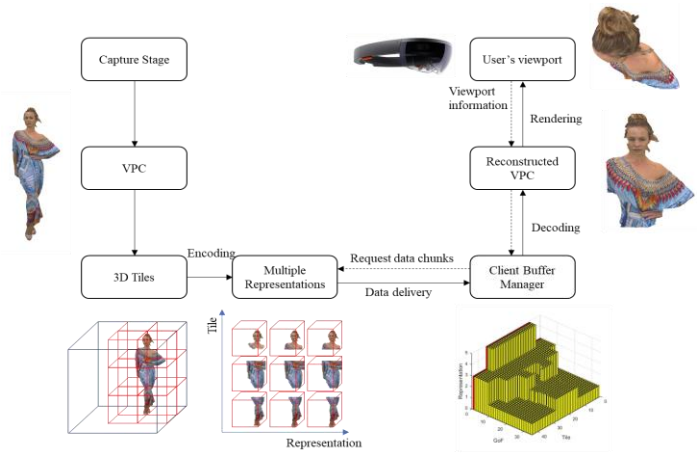


Fig 1. Volumetric Media Streaming System

spherical video in VR. Second, in 3D the tiles may be occluded or far away. When streaming spherical video, the distance from the user to the scene is not variable. However, when streaming holograms, the scale of a tile in the user's view, and hence the utility of its decoded representation, can change dramatically as an inverse function of the user's distance to the tile, and may also be limited by the resolution of the rendering device. We develop a model of the utility of the tile representations based not only on its bitrate and whether the tile is visible, but also on the distance of the tile from the user. Finally, we develop a greedy yet optimal algorithm for maximizing the utility of the tiles subject to a rate constraint and couch it within the client buffer manager of the streaming client, to request an optimal set of tiles at every transmission opportunity. The algorithm naturally handles bitrate allocation between multiple holograms streaming simultaneously, as well as their playback at different speeds. We believe our work is the first to address streaming of volumetric media or holograms for AR or VR.

An overview of our system for streaming volumetric media is shown in Fig. 1. On the left, a web server stores the volumetric media objects, or holograms, for streaming. Each object is represented as a sequence of voxelized point cloud (VPC) [19] frames, the frames are grouped into groups of frames (GOFs), and the sequence of GOFs is divided temporally into segments. Each segment is independently compressed to a small set of representations, each representation at a different bitrate. The bitrates are constant across all segments. Each GOF is divided spatially into tiles, which are independently coded. A media presentation description (MPD), or manifest, accompanies each object to describe the collection of representations, while a segment index accompanies each segment to index the collection of tiles within the segment. On the right, the client buffer manager (CBM) in an HTTP client downloads the manifest, downloads the segment index for each segment in the window, estimates the throughput, estimates the user position, calculates the optimal collection of 3D tiles that will maximize utility for a given rate constraint, requests the tiles, downloads them into the buffer, advances the window, releases the tiles that fall out of the window for rendering, decoding, and presentation by the application, and repeats.

Experimental results show that the proposed volumetric media streaming systems offer the higher quality than alternative designs, because our utility maximization algorithm can select the best bit rates of the 3D tiles.

This paper is organized as follows. Sections II and IIIII detail our utility functions and rate-utility optimization algorithm. Section IIV presents experimental results, followed by the conclusion in Section V.

II. UTILTY MEASURES

The utility of a tile depends on how much useful information it brings to the user. This depends in turn on the relationship of the tile to the viewpoint of the user. If a tile is outside the viewpoint of the user, then it has no utility. If it is far away, then its utility may be low because it covers a small area. On the other hand, as the tile moves quite close, then its utility may saturate because its spatial resolution is intrinsically finite. And of course a tile coded with a higher bitrate will generally have a higher spatial resolution or higher PSNR or both. Unfortunately, the utility of a tile in the far future is uncertain, because the user's viewpoint in the future is uncertain. Hence in principle one can evaluate only the *expected* utility of a future tile. To complicate matters, frequently there are multiple points of view, certainly in stereoscopic systems, but possibly also in multi-user systems.

We model the (expected) utility of a tile to be a function of the bandwidth of the representation selected for the tile, weighted both by the number of distinguishable voxels in the tile and by the probability that the tile will be visible (i.e., in the user's frustum and facing the user) when it is decoded and rendered.

To be more specific, we change notations from the previous section. For tile k, with representation n_k, and viewpoints $v \in \mathcal{V}$, the utility $U_k(n_k)$ of the tile is given by

$$U_k(n_k) = u(B_{n_k}) \times \max_{v \in V} \{LOD(n_k, v) \times P_k(v)\}, \quad (1)$$

where $u(B)$ is a function indicating the utility per distinguishable voxel of the tile as a function of the bandwidth of the representation, $LOD_k(n_k, v)$ is the number of distinguishable voxels of the tile if the tile is visible from the current viewpoint v, and $P_k(v)$ is the probability that the tile would be visible when rendered, if the current viewpoint is v. The latter two factors depend on the spatial resolution of the representation, the size of the tile, the point of view relative to the tile, and the resolution of the viewing device. However, the only parameter that the CBM can control to maximize the utility is n_k. All three factors are described in detail next.

A. Utility per distinguishable voxel

We model the utility of the tile per distinguishable voxel as an increasing function of the bandwidth of the tile's encoding representation, as a proxy for the quality of the representation. Though the utility of quality to any given user is hard to quantify, in general it should be monotonically increasing with the bitrate,

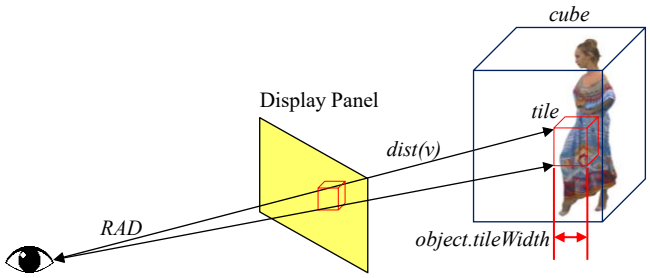
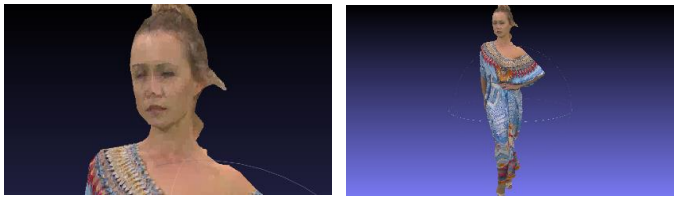


Fig 2. The approximate field of view across the tile in radians (RAD)



(a) VPR<PPR (b) VPR>PPR

Fig 3. VPR and PPR

should flatten out at high bitrates, and should be zero when nothing is transmitted. Thus, we model the utility as an affine function of the logarithm,

$$u(B) = \begin{cases} \alpha \log \beta B, & B > 0 \\ 0, & B = 0 \end{cases}, \quad (2)$$

where α and β are normalization coefficients that bring u into the range $[0,1]$ for all bandwidths B_m of the representations $m = 1, \ldots, M$. These coefficients are then kept constant for the duration of playback. Many Mean Opinion Score (MOS) tests frequently show such a logarithmic pattern [20].

B. Distinguishable voxels in a tile

We model the number of distinguishable voxels in a tile as the number of distinguishable voxels in the square area roughly covered by the tile, that is, the square of the number of degrees of view linearly across the tile times the number of distinguishable voxels per degree of view linearly across the tile. In turn, the number of degrees of view linearly across the tile is approximately the width of the tile divided by the distance of the tile from the viewpoint. Furthermore, the number of distinguishable voxels per degree of view linearly across the tile is the minimum of the number of voxels per degree of view linearly across the tile and the number of pixels per degree of view across the display. This information is computed based on the user's view pose and frustum, and the display resolution, passed to the CBM.

To be specific, let the width of the tile in the real world be the width of the tile in voxels (object.tileWidth) times its cube-to-world scale (object.cubeToObjectScale), and let the field of view of the tile at unit distance be approximated by its real-world width. Let position of the tile in the real world be its (x, y, z) position in voxels (as determined from its Morton code [21]) time its cube-to-object translation (object.cubeToObject Translation), and let the distance to the position of the tile from viewpoint v be $dist(v)$. Then the approximate field of view across the tile in radians (Fig. 2) is

$$RAD_k(v) = \frac{objec.tileWidth \times object.cubeToObjectScale}{dist(v)}. \quad (3)$$

Next, let the number of voxels in the tile per radian (VPR) be the width in voxels of the bounding cube in representation n ($object.representation[n].width$), divided by the width of the bounding cube in the real world ($object.maxWidth * object.cubeToObjectScale$), times the distance to the tile:

$$VPR_k(n, v) = \frac{object.representation[n].width \times dist(v)}{object.\max Width \times object.cubeToObjectScale}. \quad (4)$$

Finally, let the number of pixels per radian (PPR) across the display device be the number of pixels across the display (display.horzPixels) divided by the field of view of the frustum,

$$PPR_k(v) = \frac{display.horzPixels}{veiw(v).frustum.horzFOV} \quad (5)$$

Then the minimum between $VPR_k(n, v)$ and $PPR_k(v)$ is the number of distinguishable voxels per degree across the tile, and

$$LOD_k(n, v) = [RAD_k(v) \times \min\{VPR_k(n, v), PPR_k(v)\}]^2 \quad (6)$$

is the number of distinguishable voxels in the square area roughly covered by the tile. Fig 3(a) shows the user's view when the number of the voxels per radian is smaller than the number of the pixels per radian. Fig. 3(b) shows the user's view when the number of the pixel per radian is smaller than the number of the voxels per radian.

C. Probability that a tile will be visible

The last part of the utility model is the probability $P_k(v)$ that if the current viewpoint is v then tile k will be visible by the time the tile emerges from the trailing edge of the window and is displayed to the user.

The uncertainty of whether tile k will be visible by the time it emerges from the window is due, of course, to the uncertainty of the user's behavior in the interim. If the viewpoint of the user could be accurately predicted, then $P_k(v)$ could be set close to 0 or 1. If it were close to 0, then the utility of the tile would be close to 0, and no bits would need to be wasted transmitting the tile. The bits could be used instead to improve the quality of tiles for which $P_k(v)$ is close to 1.

Thus, for AR streaming, user prediction is an important problem, just as network prediction is important for all streaming. User adaptivity and network adaptivity are analogous. The problem of user prediction is therefore a new area for research in AR streaming.

In this paper, however, we keep the user prediction model very simple. Specifically, we predict that if a tile k has media time τ_k, it will be visible to the user when it emerges from the

window if its position is visible to the user in the current view v at time t, with prediction error probability 0.1 if the tile is early in the window (close to the trailing edge $W_{trail}(t)$), increasing linearly to 0.4 if the tile is late in the window (close to the leading edge $W_{lead}(t) = W_{trail}(t) + \Delta W(t)$). That is,

$$P_k(v) = \begin{cases} 1 - P_k^{err}(v), & \text{if } k \text{ currently visible from } v \\ P_k^{err}(v), & \text{otherwise} \end{cases}, \quad (7)$$

where

$$P_k^{err}(v) = 0.1 + 0.3\min\{1, (\tau_k - W_{lead}(t)) / \Delta W(t)\}, \quad (8)$$

This is a simple model of increasing uncertainty of what the user will be looking at further away in time. Undoubtedly the model can be easily improved using machine learning.

III. Utility Maximization

In this section, we present an algorithm for utility maximization that is greedy yet provably optimal.

$$U(\mathcal{M}) = \sum_{k=1}^{K} U_k(m_k), \quad (9)$$

subject to

$$R(\mathcal{M}) = \sum_{k=1}^{K} b_k(m_k) \le R_i,$$

where $\mathcal{M} = \{m_1, \dots, m_K\}$ are the representations for all tiles $1, \dots, K$ in the window, and $b_k(m)$ is the number of bits that would be required to get representation m for tile k. If $m = n_k$, the representation for tile k that is already in the buffer, then $b_k(n_k) = 0$, because it takes no additional bits to get representation n_k for tile k. If there is no representation for tile k yet in the buffer, then $n_k = 0$ and still $b_k(0) = 0$. By convention, $U_k(0) = 0$.

Because we can restrict our search for the optimal $\mathcal{M}$ to the upper convex hull $\hat{\mathcal{S}}$ of the set of points $\mathcal{S} = \{(R(\mathcal{M}), U(\mathcal{M}))\}$ in the rate-utility plane, we can instead solve the easier problem of maximizing the Lagrangian,

$$U(\mathcal{M}) - \lambda R(\mathcal{M}) = \sum_{k=1}^{K} [U_k(m_k) - \lambda b_k(m_k)], \quad (10)$$

for some $\lambda > 0$. Moreover,

$$\max_{\mathcal{M}} U(\mathcal{M}) - \lambda R(\mathcal{M}) = \max_{\{m_1,\dots,m_K\}} \sum_{k=1}^{K} [U_k(m_k) - \lambda b_k(m_k)]$$
$$= \sum_{k=1}^{K} \max_{m} [U_k(m) - \lambda b_k(m)], \quad (11)$$

so the maximization problem can be solved independently for each tile. For each λ, the solution

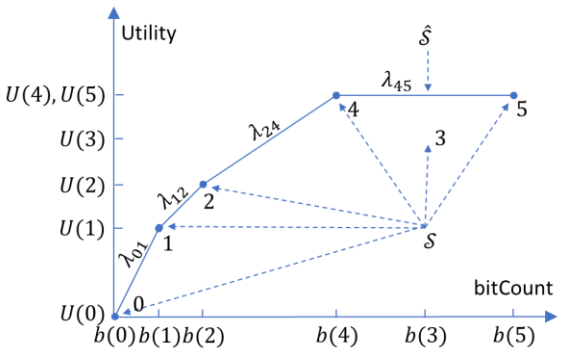


Fig 4. Rate-utility points for a tile

$$m_k(\lambda) = \underset{m}{\text{argmax}}[U_k(m) - \lambda b_k(m)] \quad (12)$$

for tile k lies on the upper convex hull $\hat{\mathcal{S}}_k$ of the set of points $\mathcal{S}_k = \{(b_k(m), U_k(m))\}$ in the rate-utility plane[1], and the points on the vertices of the convex hull are swept out in order of increasing $b_k(m)$ as λ decreases from infinity to zero.

As an example, Figure 4 shows (with the tile index k suppressed) a set of six rate-utility points $\mathcal{S} = \{(b(m), U(m)) : m = 0, \dots, 5\}$, with index 0 corresponding to the null representation and indices 1-5 corresponding to five representations for the tile's object in order of increasing *object.bandwidth[m]*. Points 0, 1, 2, 4, and 5 lie on the upper convex hull $\hat{\mathcal{S}}$ in order of increasing $b(m)$. Let λ_{01}, λ_{12}, λ_{24}, and λ_{45} be the slopes of the line segments between these points on $\hat{\mathcal{S}}$. Then the optimal representation for this tile for any given λ is

$$m(\lambda) = \begin{cases} 0 & \lambda_{01} < \lambda < \infty \\ 1 & \lambda_{12} < \lambda \le \lambda_{01} \\ 2 & \lambda_{24} < \lambda \le \lambda_{12} \\ 4 & \lambda_{45} < \lambda \le \lambda_{24} \\ 5 & 0 \le \lambda \le \lambda_{45} \end{cases}. \quad (13)$$

For this tile, λ_{01} is a threshold for λ above which no representation is requested. The maximum of such threshold across all tiles is a threshold for λ above which no representations are requested for any tiles. As λ decreases from this threshold, $b_k(m_k(\lambda))$ increases for every tile k, and hence $R(\mathcal{M})$ also increases. Thus λ can be decreased step-by-step until the constraint $R(\mathcal{M}) \le R_i$ would be violated. For this value of λ, the representation $m_k(\lambda)$ for tile k can be requested from the server if $m_k(\lambda) > 0$. This is our basic rate-utility optimization algorithm.

One modification to the basic algorithm is crucial to be able to update, at a request of opportunity t_i, the representation of a tile that remains in the window and already has a representation from request opportunity t_{i-1}, for example, if the tile suddenly increases in utility because the user has turned to look at it. In the modification, the representation of tile k from previous request opportunities is saved in variable n_k, and $b_k(n_k)$ is

[1] Not only does the convex hull $\hat{\mathcal{S}}$ outperforms $\mathcal{S}$ in the sense that for any point $(R, U) \in \mathcal{S}$, there exists a dominating point $(\hat{R}, \hat{U}) \in \hat{\mathcal{S}}$ such that $\hat{R} \le R$ and $\hat{U} \ge U$, but also every point on $\hat{\mathcal{S}}$ can be achieved with timesharing or randomization between points in $\mathcal{S}$.

set to 0, the rationale being that to obtain representation n_k again at the current request opportunity would take 0 bits. The utility $U_k(n_k)$ is left unchanged. With this modification, the initial point on the upper convex hull for tile k is $(0, U_k(n_k))$, rather than $(0,0)$. This raises the initial point of the upper convex hull, thus flattening the convex hull, making it difficult to reach other representations along the upper convex hull unless λ is allowed to become large (e.g., if the estimated bit budget R_i suddenly becomes large) or unless some other representation suddenly increases in utility (e.g., if the user turns her head to look at the tile). The full algorithm is shown in Table 1.

TABLE 1.
RATE-UTILITY MAXIMIZATION ALGORITHM

Set $R_{current} = 0$.
For each tile k:
  Set $m_k = n_k$ (the existing representation)
  If $b_k(n_k) = \max_m b_k(m)$ then set
    $\lambda_k^* = 0$ and $m_k^* = n_k$.
  Else set
    $$\lambda_k^* = \max_{m:b_k(m)>b_k(n_k)} \frac{U(m)-U(n_k)}{b_k(m)-b_k(n_k)},$$
    $$m_k^* = \operatorname{argmax}_{m:b_k(m)>b_k(n_k)} \frac{U(m)-U(n_k)}{b_k(m)-b_k(n_k)}.$$
while $R_{current} < R_i$
  Let $k = \max_k \lambda_k^*$ be the tile with the greatest λ_k^*.
  If $\lambda_k^* \le 0$ then break.
  Update $R_{current} = R_{current} + b_k(m_k^*) - b_k(n_k)$.
  Set $n_k = m_k^*$.
  If $b_k(n_k) = \max_m b_k(m)$ then set
    $\lambda_k^* = 0$ and $m_k^* = n_k$.
  Else set
    $$\lambda_k^* = \max_{m:b_k(m)>b_k(n_k)} \frac{U(m)-U(n_k)}{b_k(m)-b_k(n_k)},$$
    $$m_k^* = \operatorname{argmax}_{m:b_k(m)>b_k(n_k)} \frac{U(m)-U(n_k)}{b_k(m)-b_k(n_k)}.$$
end while

IV. Experimental Results

We investigate the performance of the proposed method when users are moving or controlling the viewpoint in AR applications. Deterministically generated camera paths are used to model user behavior. The camera (Fig. 5) moves around an object to see the object. This is for modeling the rotation of an object in a client's display. As the camera moves close to the object, smaller number of tiles is visible and those tiles should have higher quality.

To see the advantages of using the proposed algorithm for tile-based streaming, the average representations in user's view are measured with different tile sizes. Fig. 6(a) and Fig. 6(b) show the estimated throughput for the stable network and the variable network respectively. Fig. 7(a) shows the simulation result with a stable network condition, and Fig. 7(b) shows the simulation result with a variable network. Depth indicates the number of division of cube. With smaller size of tiles, we can achieve better utility in both cases, since we can allocate more bits to only visible tiles.

Simulation results show that the algorithm chooses higher bitrates for visible tiles. There is network variation and

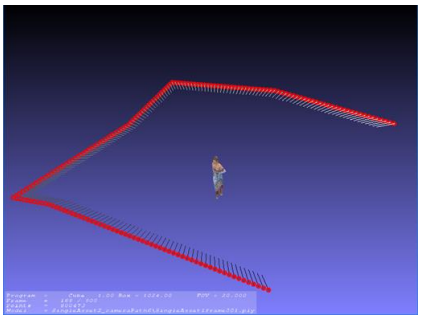
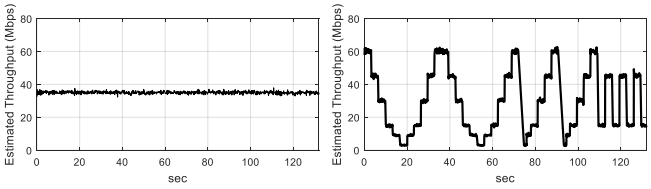
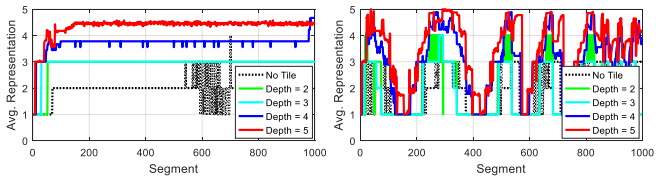


Fig 5. Camera Path



(a) Stable Network (b) Variable Network

Fig 6. Network Variations



(a) Stable Network (b) Variable Network

Fig 7. Avg. Representations on the user's view

sometimes the network throughput gets very bad, but window-based algorithm can tolerate the bad network condition using existing data on the buffer and requesting lower bitrate to save a bandwidth.

Another advantage of the proposed algorithm is that it can be used for multiple objects without changing the algorithm. It loads five objects and other parts are the same as the single object simulator. Fig. 9. shows the location of objects and the camera path (camera is always looking at yellow triangle). Fig. 10. shows the utility values ((a) Asset1, (b) Asset2, (c) Asset3, (d) Asset4, and (e) Asset5) of each object while user's view point is moving. Since the multiple objects compete for limited resource, the algorithm allocates more bits to visible tiles of visible objects. The utility will be counted only when they are visible. Since Asset1 is not in the view while the camera is moving, its utility value is always zero. Asset3 is located in the center and always in the camera's view. Therefore, it has better utility than other assets. Asset2, Asset3, and Asset5 have utility value only when they are in the camera's view. This kind of AR streaming systems always need to be ready for sudden changes of user behaviors, therefore, the proposed algorithm also requests non-visible tiles with lower quality to ensure that there are some data exist in one of user's view. This is how Asset2, Asset3, and Asset5 can have high utility when they suddenly come into the camera's view.

V. Conclusion

This is the first paper to deal with on-demand and/or live streaming of volumetric media for VR/AR applications. The

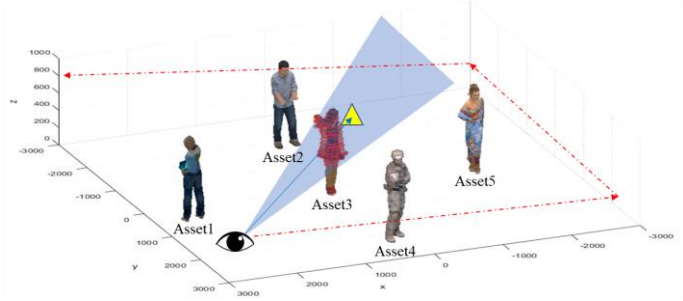


Fig 8. Multiple assets in the user's view

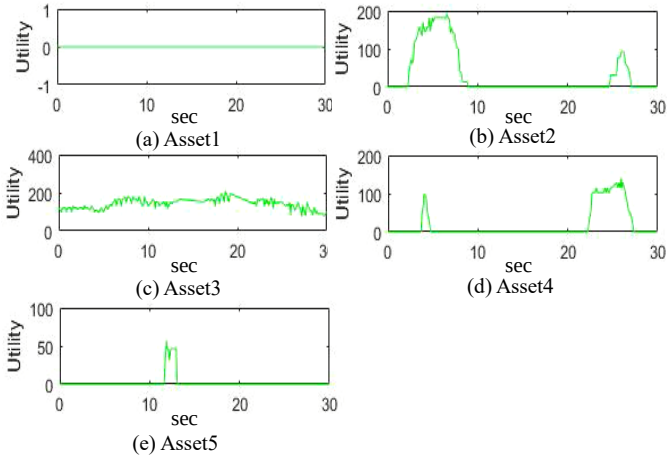


Fig 9. Utilities of Multiple Assets

main new aspect of streaming volumetric content for VR/AR applications, compared to streaming video (or even spherical video), is the high amount of user interactivity. This requires the streaming to be not only network-adaptive but also user-adaptive. Thus, new approaches are needed to minimize both network bandwidth and perceived latency due to user interactions. Our solution to volumetric streaming is consistent with modern HTTP streaming and requires only minor modifications to, for example, DASH, such as information in the manifest to support the notion of a volume and its coordinate system relative to the user, as well as tiles and segment indexes for the tiles. Simulation results show that even in the absence of tiles and rate-utility optimization, our algorithm has both higher throughput and fewer rebuffering events than existing algorithms, in both stable and variable network conditions. Simulation results also show that our algorithm handles user interaction for volumetric video as expected.

REFERENCES

[1] A. Rebatta, "295 Tbps: Internet Traffic and Capacity in 2017," https://blog.telegeography.com/295-tbps-internet-traffic-and-capacity-in-2017, Sep. 2017.

[2] "Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," ISO/IEC 23009-1:2014.

[3] E. Liotou, K. Samdanis, E. Pateromichelakis, N. Passas, and L. Merakos, "QoE-SDN APP: A Rate-guided QoE-aware SDN-APP for HTTP Adaptive Video Streaming," *IEEE Journal on Selected Areas in Communications*, pp. 1–1, 2018.

[4] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in Proc. IEEE Int'l Conf. Communications, May 2017.

[5] P. Rondao Alface, J.-F. Macq, and N. Verzijp. "Interactive Omnidirectional Video Delivery: A Bandwidth-Effective Approach." Bell Labs Technical Journal 16.4 (2012): 135-147.

[6] H. Kimata, S. Shimizu, Y. Kunita, M. Isogai, and Y. Ohtani, "Panorama video coding for user-driven interactive video application," *2009 IEEE 13th International Symposium on Consumer Electronics*, 2009.

[7] Liu, Xing, Qingyang Xiao, Vijay Gopalakrishnan, Bo Han, Feng Qian, and Matteo Varvello. "360 Innovations for Panoramic Video Streaming." In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pp. 50-56. ACM, 2017.

[8] Corbillon, Xavier, Francesca De Simone, Gwendal Simon, and Pascal Frossard. "Dynamic Adaptive Streaming for Multi-Viewpoint Omnidirectional Videos." In *ACM Multimedia Systems Conference 2018*, no. CONF. 2018.

[9] T.-Y. Huang, R. Johari, and N. McKeown, "Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming." Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking. ACM, 2013.

[10] J. Jiang, V. Sekar, and H. Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012.

[11] L. De Cicco, et al. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." Packet Video Workshop (PV), 2013 20th International. IEEE, 2013.

[12] Li, Zhi, et al. "Probe and adapt: Rate adaptation for HTTP video streaming at scale." IEEE Journal on Selected Areas in Communications 32.4 (2014): 719-733.

[13] J. Park, J.-N. Hwang, Q. Li, Y. Xu, and W. Huang. "Optimal DASH-multicasting over LTE." *IEEE Transactions on Vehicular Technology* (2018).

[14] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," *2016 IEEE International Conference on Big Data (Big Data)*, 2016.

[15] J. L. Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," *Proceedings of the 7th International Conference on Multimedia Systems - MMSys 16*, 2016.

[16] L. Dacunto, J. V. D. Berg, E. Thomas, and O. Niamut, "Using MPEG DASH SRD for zoomable and navigable video," *Proceedings of the 7th International Conference on Multimedia Systems - MMSys 16*, 2016.

[17] J. Hou, L.-P. Chau, Y. He, and P. A. Chou, "Sparse Representation for Colors of 3D Point Clouds via Virtual Adaptive Sampling," Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Mar. 2017.

[18] R. L. de Queiroz and P. A. Chou, "Transform Coding for Point Clouds Using a Gaussian Process Model," IEEE Trans. Image Processing, Vol. 26, no. 7, July 2017.

[19] E. Pavez and P. A. Chou, "Dynamic polygon cloud compression," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[20] W.W. Zhang, Y.G. Wen, Z.Z. Chen and A. Khisti, "QoE-Driven Cache Management for HTTP Adaptive Bit Rate Streaming over Wireless Networks," *IEEE Transactions on Multimedia,* vol. 15, no. 6, pp.1431-1445, 2013.

[21] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," IEEE Trans. Image Processing, Vol. 25., no. 8. Aug. 2016.