Low-latency FoV-adaptive Coding and Streaming for Interactive 360° Video Streaming

Yixiang Mao, Liyang Sun, Yong Liu and Yao Wang
Department of Electrical and Computer Engineering, Tandon School of engineering, New York University
{yixiang.mao,ls3817,yongliu,yw523}@nyu.edu

ABSTRACT

In 360° video interactive streaming, it is critical to minimize the end-to-end frame delay. It is also important to predict the user's field of video (FoV) and allocate more bits in regions within the predicted FoV. Towards both goals, we propose a low-delay FoV-adaptive coding and delivery system that is robust to bandwidth variations and FoV prediction errors. Each frame is coded only in the predicted FoV (PF), a border surrounding the predicted FoV (PF+), and a rotating intra (RI) region. To maximize the coding efficiency, the PF and PF+ regions are coded with temporal and spatial prediction, while the RI region is coded with spatial prediction only. The RI region enables periodic refreshment of the entire frame and provides robustness to both FoV prediction errors and frame losses. The total bit budget is adapted both at the segment level based on the predicted average bandwidth for the segment and at the frame level based on the sender buffer status, to ensure timely delivery. The system further adapts the sizes and coding rates of different regions for each video segment to maximize the average rendered video quality under the total bit budget. To enable such adaptation, we propose novel ways to model the quality-rate (Q-R) relations of coded regions that take into account of potentially misaligned coded regions in successive frames due to FoV dynamics. We examine the performance of the proposed system and three benchmark systems, under real-world bandwidth traces and FoV traces, and demonstrate that the proposed system can significantly improve the rendered video quality over the benchmark systems. Furthermore, the proposed system can achieve very low end-to-end frame delay while maintaining a low frame freeze probability and providing smooth video playback.

CCS CONCEPTS

• **Human-centered computing** → Virtual reality; • **Information systems** → **Multimedia streaming**.

KEYWORDS

360° video; FoV-adaptive streaming; tile-based coding; low latency

ACM Reference Format:
Yixiang Mao, Liyang Sun, Yong Liu and Yao Wang. 2020. Low-latency FoV-adaptive Coding and Streaming for Interactive 360° Video Streaming. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413751

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413751

1 INTRODUCTION

Virtual Reality (VR) and Augmented Reality (AR) technologies have become popular in recent years. Encoding and transmitting the omni-directional or 360° video is critical and challenging for those applications. The 360° video requires much higher bandwidth than the traditional planar video. A premium quality 360° video with 120 frames per second (fps) and 24K resolution can easily consume bandwidth in the range of Gigabits-per-second [18]. On the other hand, at any given time, a user only watches a small portion of the 360° scope within her Field-of-View (FoV). An effective way to reduce the bandwidth requirement of 360° video is through *FoV-adaptive streaming*, which codes and delivers the predicted FoV region at higher quality, and discards or codes at lower quality the remaining regions. Such strategy has been quite extensively studied for video-on-demand [4, 5, 15, 17, 18] and live video streaming applications[1, 7, 13, 19]. Interactive applications, such as conferencing, gaming, and remote collaboration, can also benefit from 360° video by creating an immersive environment for participants to interact with each other [20][23][11]. However, realtime coding and streaming of 360° video with extremely low latency, required for interactive applications, has not been sufficiently addressed. This work focuses on developing low-latency and FoV-adaptive coding and streaming strategies for interactive 360° video streaming. We assume the sender and the receiver are connected by a network path with dynamically varying throughput without short-latency guarantee. The sender is either the video source, or a proxy server relaying the source video. The receiver is either the end user device that directly renders the video, or a local edge server that renders the video and transmit to the end user [8].

Motion-compensated temporal prediction is critically important in video coding to maximize the video quality under the limited bandwidth and has been adopted successfully for planar video coding. For on-demand and live streaming of the 360° video, a video is typically divided into temporal segments (1 second or longer) and each segment is further divided into many small spatial tiles. In FoV adaptive streaming, the FoV distribution in each new segment is predicted, and only tiles that are likely to be viewed are delivered for each video segment. In this case, all the frames in a tile video are coded and temporal predictive coding can be readily applied, except the first frame, which is coded as an intra-frame.

For interactive streaming, to minimize the latency, the video should be coded and delivered at the frame level. This enables the sender to predict the FoV for each new frame and only code the regions covering the predicted FoV. On the one hand, this can

improve the FoV prediction accuracy and reduce the coded-but-not-viewed area. On the other hand, applying temporal predictive coding in such a scenario faces several unique challenges. The first challenge is that the coded regions in each frame depend on the predicted FoV and may not be aligned in successive frames. Such misalignment causes prolonged time lapse for temporal prediction for some tiles and leads to reduced coding efficiency. Properly taking into account of such reduced coding efficiency issues is important for accurate rate control, essential for minimizing the latency. The second challenge is how to periodically update all the regions in a frame to limit error propagation due to frame losses and the quality degradation in un-coded regions, which in turn affects temporal prediction accuracy. Commonly used periodic intra-frame design will cause periodic bit rate spikes and hence increase the latency. Without using temporal prediction (i.e. using only intra-coding for all the frames, as in some prior work [20][16]), it is much easier to perform FoV adaptive coding and rate control, but it would lead to significantly lower quality under the same network throughput.

In this paper, we develop a novel low-latency FoV-adaptive coding and streaming solution for interactive 360° *video. To maximize the quality-rate efficiency, we adopt motion-compensated temporal predictive coding, and address the challenges brought by temporal prediction in frame-wise FoV-adaptive coding.* The salient features of our proposed solution include:

(1) To meet the low latency requirement, realtime coding and streaming are conducted at the granularity of frames, instead of segments for video-on-demand and live streaming.
(2) The video source predicts the FoV of the receiver for a new frame and codes only a region covering the predicted FoV (denoted PF) plus a surrounding border (denoted PF+), with the border size adapted to the anticipated FoV prediction errors. Both regions will be coded using temporal prediction but at different rates. [1]
(3) To reduce the frame size burstiness (essential for minimizing delay) while maintaining the robustness against FoV prediction errors and frame losses, we also code a small rolling region using intra-coding (denoted RI) for each frame, enabling gradual refreshment of the entire 360° scope after a certain period. See Figure 1.
(4) We model the quality-rate (Q-R) relations of coded regions in a way that takes into account the spatially and temporally varying time lapse in temporal prediction due to FoV dynamics (see Figure 2). We further model the decay of the rendering quality of non-coded regions upon FoV prediction errors as a function of the time lapse since these regions are last coded.
(5) To maximize the rendered video quality, we periodically adapt the sizes and the allocated bits of different coding regions, guided by the developed Q-R models and the predicted network bandwidth and FoV prediction accuracy.

[1]Note that if one remote site in a two- or multi-party conference has multiple participants, we can take the union of the FoVs of all the participants as the ground truth "FoV" of this site, and predict the future FoV union. Typically all the users tend to focus on the same region (e.g. the speaker of the remote site), and the union may be only slightly larger than the single user's FoV. Occasionally, the users' FoVs may diverge significantly; In this case, we have to code a very large region, making FoV adaptive coding less efficient.

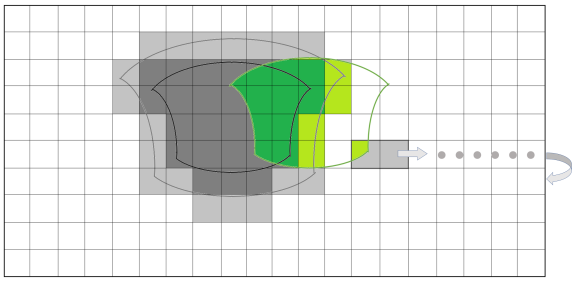

Figure 1: Different regions in a frame using the ERP format. Dark grey: PF tiles, coded at the rate R_e. Light grey: PF+ and RI tiles, coded at the rate R_b. Green: user's FoV, which may fall in PF, PF+, RI, and un-coded tiles.

(6) To minimize the transmission latency and avoid self-congestion, we design push-based frame delivery with short sender and receiver buffers. We further adjust the frame-level bit budget in realtime and control source buffer overflow to maximize the frame delivery rate before the display deadline.

We evaluate the performance of the proposed system using real-world bandwidth traces and FoV traces, in core metrics including the rendered quality, the frame delay, and the freezing probability. We further compare the proposed system with three benchmark systems. Our trace-driven simulations demonstrate that the proposed system can achieve significantly higher rendered quality than all benchmark systems using either intra- or inter-coding, and lower delay than the traditional inter-coding (using periodic I-frames) benchmark. Our simulation results further reveal that the gain from region size adaptation in the proposed system is limited, and hence can be foregone in practice, for reduced complexity.

In Section 2, we present the proposed video coding framework, formulate the expected rendered quality taking into account the FoV hit rates in different regions and the frame delivery rate, and describe our approach for joint optimization of region size and rate allocation. In Section 3, we present the streaming system design, including our approach for bandwidth prediction and FoV prediction, adaption of region size and rate allocation at the segment level, and furthermore adjustment of the bit budget at the frame level. Section 4 presents the setup of the trace-driven simulations and compares the performance of the proposed system with the benchmark systems. Section 5 summarizes the main takeaways from our studies.

2 LOW DELAY, FOV-ADAPTIVE 360° VIDEO CODING USING SPATIAL AND TEMPORAL PREDICTION

2.1 The Proposed Video Coding Scheme

To achieve low-latency interactive streaming, we propose a novel FoV-adaptive coding structure. We represent each 360° video frame using the Equirectangular projection (ERP) format. Instead of the traditional group of pictures (GOP) structure that inserts intra frames periodically, we only encode the first frame of the entire 360° scope as an intra frame. For each subsequent frame, we predict the FoV for that frame, and code only the predicted FoV region projected onto the ERP (to be denoted "PF") and a small border outside the predicted FoV (called "PF+"). The PF+ is coded in case the actual

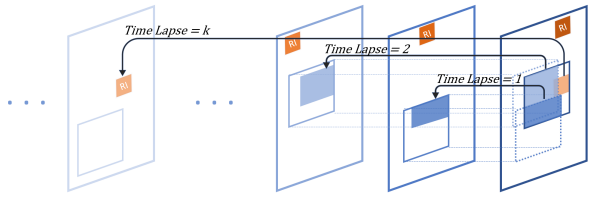

Figure 2: Illustration of the variable time lapses between the coded tiles. The square region covered by a solid-line border in each frame indicates the coded region covering PF and PF+. Different tiles in the coded region in the current frame have different time lapses to the latest frame when their corresponding tiles were coded.

FoV is slightly off from the PF. Both the PF and PF+ regions are coded using temporal predictive coding (aka inter-coding) based on the previously decoded reference frame. The PF will be coded using a high rate, while the PF+ will be coded using a lower rate. To prevent some areas from never getting refreshed, we also code and transmit a rotating intra region (RI) using only spatial prediction (aka intra-coding). The RI region will be rolled in successive frames from top to bottom and left to right in the ERP. The RI is introduced to ensure that all pixels in the ERP will be refreshed using intra-coding after a certain period. For example, if an RI includes only a 1/72 of the ERP frame, the entire frame would be refreshed after every 72 frames. Such periodic refreshment provides robustness both to FoV prediction errors as well as to frame losses due to transmission packet losses. The RI region will be coded using a lower rate, since it has a low likelihood to be viewed. Figure 1 illustrates the video coding frame structure. Note that the first frame should be encoded using a high quantization level to reduce the total rate, and hence initial buffering time.

To enable different regions be coded using different methods and at different rates, we divide an entire ERP frame into multiple non-overlapping tiles, each covering a small rectangular region on the ERP. All the tiles that intersect with the PF region will be coded at R_e, and all the remaining tiles that intersect with the PF+ region will be coded at R_b. As shown in Figure 1, a FoV or a predicted FoV on the 360° sphere does not correspond to a rectangular region in the ERP frame. In general, the number of tiles needed to cover a PF could differ from frame to frame. Similarly, the number of tiles for the PF+ region would also differ. We set the RI to be a rectangular region in the ERP, consisting of an integer number of tiles. To simplify rate allocation, we code the RI region using the same rate R_b as the PF+ region. Because RI region is intra-coded, it would have lower quality than the inter-coded PF+ region, even though they are allocated the same average rate. Note that for some frames, some or all the tiles in the RI may fall within the PF or PF+ region, and in that case, those tiles are coded in the intra-mode, so as to stop decoding error propagation due to potential frame losses after the intra refreshment period. In the decoder, which is also part of the encoder to derive the reference frame for temporal prediction, only coded tiles (those in the PF, PF+, and RI region) will be updated based on the received bits for this frame. Other tiles will stay the same as the previously decoded frame. Although more sophisticated error concealment methods may be adopted for enhancing these regions, we choose to use this simple approach.

2.2 Quality-Rate Models

To perform joint optimization of the region size and rate allocation, we need to model the expected quality-rate (Q-R) relations for different regions that consider the temporal variation of the FoV location. We first introduce the quality metric we use, and then describe the "ideal" Q-R models derived from video coding experiments that do not consider the prolonged temporal prediction time lapse for some tiles due to the changes in FoV. Next we describe how to consider the rate increase due to such time lapses. Finally we show how to model the quality decay of tiles that are not coded.

2.2.1 Objective quality metric. Due to the lack of well-accepted subjective quality metrics for 360° video, we will assume the perceptual quality is proportional to the weighted-to-spherically-uniform peak-signal-to-noise ratio (WS-PSNR),

$$\text{WS-PSNR} = 10\log_{10}\frac{\text{MAX}_\text{I}^2}{\text{WS-MSE}}, \tag{1}$$

$$\text{WS-MSE} = \frac{1}{\sum_{i,j} w(i,j)}\sum_{i,j}[I(i,j)-K(i,j)]^2 w(i,j), \tag{2}$$

where $i, j \in$ projected FoV, $w(i,j) = \cos((\frac{i}{m}-\frac{1}{2})\pi)$, $I(i,j)$ and $K(i,j)$ are the pixel value of the encoded and raw sequence at the coordinate (i,j) on the ERP frame. This is a 360° video objective metric recommended by JVET [3]. The geometrical distortion of ERP is taken into account by assigning different weights to different pixel locations in the ERP. Given the actual FoV, we calculate WS-MSE only over the project FoV pixels on the ERP, where these pixels can be in either PF, PF+, RI, or remaining region. This will give us the actual rendered quality.

2.2.2 "Ideal" Quality-rate Models For Different Coded Regions. To derive the "ideal" Q-R models, we perform video coding experiments on several JVET 360° video testing sequences using the HEVC reference software (HM) [9] under JVET common test condition (CTC). The PF and PF+ tiles are coded using the low-delay-P mode, while the RI tiles are coded using the intra mode. We assume the FoV location is fixed throughout the entire sequence, so that every tile in the PF or PF+ region is updated in each frame. To generate empirical quality-rate curves, we code the video using different quantization parameters (QP), and determine the corresponding WS-PSNR and the total rate over the PF, PF+, and RI region separately. More details about the coding experiments can be found in [14].

To circumvent the variability of the relation between the spherical area to be covered and the actual number of tiles, which depends on the actual location of the FoV, we define the bit rate in terms of the total bits needed to cover a unit area on the sphere. Therefore the normalized bit rate has a unit of bits/degree^2. For PF and PF+, we first derive the Q-R curves for selected FoV locations and then use weighted average of these FoV curves to derive the average Q-R curves. The Q-R curve for the RI is obtained by averaging results from all possible RI locations. The Q-R curves for the PF+ and RI are further evaluated for different PF+ and RI sizes. The resulting Q-R curves for two sequences with very different characteristics are shown in Figure 3. Note that the Q-R functions for the PF+ region depends on the PF+ border width: the coding efficiency is higher for a wider border due to the fact that fewer pixels in the coded PF+ tiles are wasted in such a case. On the other hand, the Q-R function

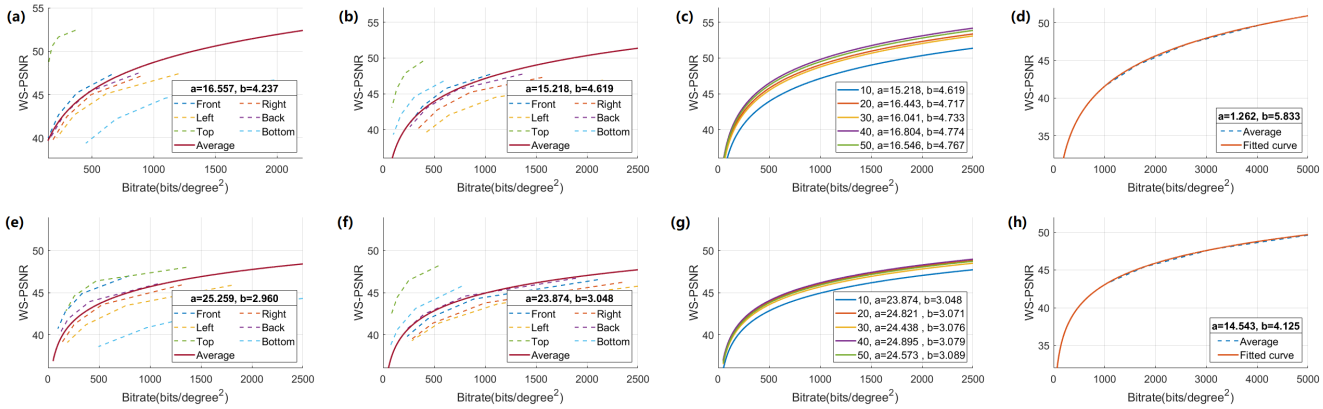

Figure 3: Q-R models: (a)-(d) are for "Trolley", (e)-(h) are for "Chairlift". (a)(e): WS-PSNR vs. normalized rate for the PF region for six different viewing directions and the averaged WS-PSNR vs. normalized rate relation. (b)(f): WS-PSNR vs. normalized rate for the PF+ region when the PF+ region size is 10°. (c)(g) are the averaged WS-PSNR vs. normalized rate when PF+ region sizes are 10°, 20°, 30°, 40°, 50°, respectively; (d)(h) WS-PSNR vs. normalized rate curves for the RI region. This relation is independent of the RI size.

for the RI region does not depend on the RI size because all the tiles in a RI are considered equally useful.

As shown in Figure 3, each Q-R curve can be modeled quite well by a logarithmic relation:

$$Q(R) = a + b \log R. \tag{3}$$

We will use $Q_{PF}(R)$, $Q_{PF+}(R)$ and $Q_I(R)$ to denote the Q-R models for the PF, PF+ and RI regions, respectively. We will use a_e, b_e to indicate the parameters for $Q_{PF}(R)$, a_b, b_b for $Q_{PF+}(R)$, and a_I, b_I for $Q_I(R)$. These parameters are generally content-dependent. Furthermore, a_b, b_b vary with the PF+ region size.

2.2.3 Rate-increase Factor. The Q-R models shown in Figure 3 are obtained by assuming the FoV and consequently the PF and PF+ regions do not change. Because the actual FoV and hence the PF and PF+ regions typically change in time, a tile in the PF or PF+ region in the current frame may not be coded in the previous frame. In fact, the corresponding tile may not have been updated in several frames. We define *the coding time lapse*, denoted by τ, as the frame distance between the current frame and the frame that a tile is last coded (See Figure 2). A larger τ will generally lead to less accurate temporal prediction. To reach the same reconstruction quality, a higher rate is likely needed than when $\tau = 1$. The *rate-increase factor* $\rho(\tau)$ is defined as the ratio of the rate required for a given τ vs. the rate when $\tau = 1$. We have conducted video coding experiments to measure this rate-increase factor and details can be found in [14]. As shown in Figure 4(a), $\rho(\tau)$ can be well fitted by a reverse exponential decay function:

$$\rho(\tau) = 1 + c * [1 - e^{-d*(\tau-1)}]. \tag{4}$$

Note that the rate-increase factor depends on the target quality (which in turn depends on the QP). Hence the parameters c and d are generally content- and QP-dependent.

The time lapse of each tile is spatially and temporally variant and depends on the FoV dynamics. To adapt the coding rates and the region sizes at the beginning of each segment, we adjust the Q-R functions derived in Section 2.2.2 for the PF and PF+ regions based on the distribution of the τ in the previous segment. Specifically,

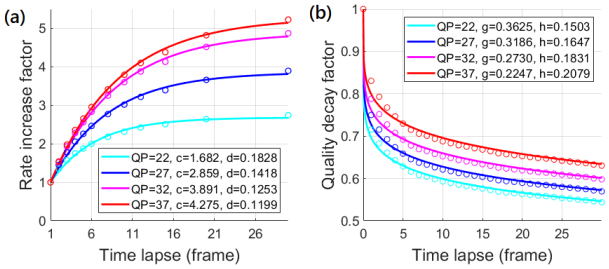

Figure 4: (a): The rate-increase factor as a function of the coding time lapse. (b): The quality-decay factor. Results are for "Chairlift".

suppose the rate required to achieve a given quality Q is $R(Q)$ according the "ideal" Q-R model for the PF or PF+ region, we modify the rate to

$$\tilde{R}(Q) = \left(\sum_{\tau} p(\tau)\rho(\tau)\right) R(Q), \tag{5}$$

where $p(\tau)$ denotes the probability of τ measured among all the tiles in the PF or PF+ region.

2.2.4 Quality-decay factor. The actual FoV for a frame may fall outside the PF, PF+ and RI regions for this frame. Pixels in such un-coded tiles are not refreshed. The rendered quality of such a tile is effectively the WS-PSNR between the current tile and when it is last coded. Generally this quality depends on the time lapse since this tile is last coded, τ, as well as the quality when it is last coded. The *quality-decay factor* $\kappa(\tau)$ is defined as the ratio of the quality of such a tile vs. the quality when it is last coded. We have conducted video coding experiments to measure this quality-decay factor and the details are described in [14]. As shown in Figure 4(b), $\kappa(\tau)$ can be well fitted by a modified exponential decay model:

$$\kappa(\tau) = e^{-g*\tau^h}, \tag{6}$$

where the parameters g and h are also content- and QP-dependent.

2.3 Optimizing rate allocation and region sizes

2.3.1 Expected video quality. With our coding scheme, the perceived quality of a rendered pixel depends on whether this pixel is coded in the PF, PF+, RI or is uncoded. We use α_{PF} to denote the

hit rate of the predicted FoV region, which is the probability that a rendered pixel will fall in the PF region that is not overlapping with the RI. We can similarly define the probabilities that a rendered pixel falls in the PF+ (not overlapped by RI) and the RI as α_{PF+} and α_I, respectively. These hit rates depend on the adopted FoV prediction algorithm and the time horizon of prediction. α_{PF+} and α_{RI} also depend on the sizes of the PF+ and RI.

If the FoV pixels fall in the PF, PF+, and RI regions, with probabilities α_{PF}, α_{PF+}, and α_I, respectively, they will be decoded with qualities $Q_{PF}(R_e)$, $Q_{PF+}(R_b)$ and $Q_I(R_b)$, correspondingly. However, if they fall outside these regions, they will be decoded with quality $\kappa(\tau)Q_{\text{last}}$, where τ is the time lapse since this tile is last coded, as explained in Section 2.2.4. The time lapse τ is in general spatially and temporally varying. The quality Q_{last} can in general be either $Q_{PF}(R_e)$, $Q_{PF+}(R_b)$ or $Q_I(R_b)$. However, because the chance that a rendered pixel falls in the un-coded region is small, we choose to use $\kappa_{\min}Q_I(R_b)$, with $\kappa_{\min} = \kappa(\tau_{max})$, as the worst-case estimate, where $\tau_{\max}$ is the intra-refresh period, inversely proportional to the RI size. Thus the average rendering quality can be formulated as

$$Q_1 = \alpha_{PF}Q_{PF}+\alpha_{PF+}Q_{PF+}+\alpha_I Q_I+(1-\alpha_{PF}-\alpha_{PF+}-\alpha_I)\kappa_{\min}Q_I. \quad (7)$$

The quality Q_1 is valid only if the packets containing bits for this frame are delivered in time, with probability γ. We call γ the *frame delivery rate*. If the frame is not delivered in time, with probability $1-\gamma$, and if the previous decoded frame is simply repeated, then a worst case estimate of the average quality would be $Q_2 = \kappa_{\min}Q_I$. Therefore, the overall average quality can be approximated as

$$\begin{aligned}\bar{Q}(R_b, R_e, A_{PF+}, A_{RI}) &= \gamma Q_1 + (1-\gamma)Q_2 \\ &= \gamma(\alpha_{PF}Q_{PF}(R_e) + \alpha_{PF+}Q_{PF+}(R_b) + \alpha_I Q_I(R_b)) \\ &+ (1-\gamma(\alpha_{PF}+\alpha_{PF+}+\alpha_I))\kappa_{\min}Q_I(R_b).\end{aligned} \quad (8)$$

In (8), A_{PF}, A_{PF+} and A_I denote the sizes of the PF, PF+ and RI region, respectively, in terms of the square degree on the sphere. Note that the hit rate α_{PF+} and the Q-R function $Q_{PF+}(R)$ depends on A_{PF+}, and the hit rate α_{RI} depends on A_I. Therefore, the average quality in Equation (8) is a function of A_{PF+}, A_{RI}, R_e and R_b, for given FoV prediction accuracy, characterized by α_{PF}, and the frame delivery rate γ.

2.3.2 Optimization Problem Formulation and Solution. Given the target number of bits for a frame B_t, the normalized rates R_b and R_e, and the areas A_{PF}, A_{PF+} and A_I have to be chosen to satisfy:

$$\lambda_{PF}A_{PF}R_e + (\lambda_{PF+}A_{PF+} + A_I)R_b \le B_t, \quad (9)$$

where λ_{PF} and λ_{PF+} indicate the average fractions of tiles in PF and PF+, respectively, that are not covered by the RI region, both equal to the ratio of the number of RI tiles (related to the RI size) vs. the total number of tiles in the ERP.

Given the total rate target for a frame B_t, our goal is to find optimal rate allocation (R_b and R_e) and region sizes (A_{PF+}, A_I) to maximize the quality in Eq. (8) subject to the rate constraint in Eq. (9). In general, α_{PF+} and α_I increase with A_{PF+} and A_I, and $\kappa_{\min}$ also increases with A_I. However, the rates R_e and R_b decrease with A_{PF+} and A_I due to the total bandwidth constraint.

For a given ERP frame size and tile size, and a fixed FoV size, we only consider a limited set of feasible sizes for PF+ and RI. For each region size combination, the only free variables of Eq. (8) are R_e and R_b. Using the bandwidth constraint in Eq. (9), we can write $R_b = (B_t - \lambda_{PF}A_{PF}R_e)/(\lambda_{PF+}A_{PF+} + A_I)$. Then, the optimal R_e can be solved by setting $\frac{\partial \bar{Q}}{\partial R_e} = 0$. Using the logarithmic Q-R models shown in Section 2.2.2, we can obtain the following analytical solution:

$$R_e = \frac{X}{X+Y}\frac{B_t}{\lambda_{PF}A_{PF}},\ R_b = \frac{Y}{X+Y}\frac{B_t}{\lambda_{PF+}A_{PF+}+A_I}, \quad (10)$$

where

$$X = \gamma\alpha_{PF}b_e, \quad (11)$$

$$Y = \gamma\alpha_{PF+}b_b + \gamma\alpha_I b_I + \kappa_{\min}b_I - \gamma\kappa_{\min}b_I(\alpha_{PF}+\alpha_{PF+}+\alpha_I). \quad (12)$$

We exhaustively search among all possible region size combinations (and their corresponding optimal rate allocation) to find the optimal region size and rate allocation that maximizes $\bar{Q}$.

3 LOW-DELAY RATE-ADAPTIVE STREAMING

3.1 Proposed streaming system overview

The proposed interactive video streaming system uses a server push-based streaming solution to control the video encoding phase (Figure 5). The system performs rate adaptation periodically at the segment level (In our simulations, each segment is 1 second long and contains 30 video frames). At the beginning of each new segment, based on the FoV and bandwidth history feed-backed continuously from the user, the server predicts the average bandwidth as well as the hit rates for different regions for the new segment. The server then performs optimization to determine the sizes of PF+ and RI, and the average rate R_b and R_e for all the frames in this segment. Each video frame is then encoded sequentially in real time. In our simulation we assume that the video frame rate is 30 Hz, and coding each frame takes 1 frame interval (33 ms). Once each frame is encoded, it will be appended to the sender buffer, shown as Process 1 in Figure 5. To reduce the latency, once the sender buffer exceeds a maximum buffer size, $BF_{\max}$ (which is set to 10 frames in our simulation), the encoder will skip the newly captured frames, until the sender buffer size is less than $BF_{\max}$. If there are encoded frames in the sender buffer, the server will send as many frames as possible based on the predicted bandwidth to the client, shown as Process 2 in Figure 5.

We assume the one-way transmission delay is 15 ms, which is reasonable for transmission within a country like the US. Upon receiving any new frame, the decoder will decode it with respect to the current reference frame (the last decoded frame) and put the decoded frames into the display buffer, shown as Process 3 in Figure 5. This newly decoded frame then becomes the updated reference frame. Only tiles in the PF, PF+ and RI regions will be updated. Note that we still decode a received frame even if it is late for display, so that we could update the reference frame in the decoder buffer, to avoid mismatch between encoder and decoder reference frames. We assume that decoding each frame takes 1/3 of the frame interval (11ms).

The display checks whether there is any decoded frame in the display buffer at an interval smaller than the frame interval (1/3 frame interval in our simulation), displays one frame after each interval, shown as Process 4 in Figure 5. Checking the display buffer at this higher frequency enables speeding up the playback when several delayed frames arrive in a burst. When a frame arrives later than the maximum display deadline (20 frames in our simulations), we skip this frame, and display the latest frame that meets the display deadline. The display repeats the last received frame until a new frame arrives before its deadline, leading to a frame freeze.

Note that the frame freeze can be caused by either the skipped frame at the sender or the late frames at the receiver. With the simulation set up described above, our trace-driven simulations have shown that the end-to-end frame delay is very low, with mean <100 ms, as detailed in Table 1 and 2.

3.2 Predicting FoV and bandwidth

3.2.1 Frame-Level FoV Prediction. FoV prediction is critical to the performance of FoV-adaptive streaming. In the past, linear regression, weighted linear regression, and truncated linear prediction [5, 17, 18] as well as neural network based methods [2, 6, 12] have been proposed. Most of these methods can predict the short-term FoV (within the future 1 second) well with an accuracy of more than 90% [17][12][6]. In our system, the prediction horizon equals the total frame delay, which is on average less than 100 ms. Therefore, simple regression methods are expected to work well. We adopt the truncated linear prediction method [18], which only use the last few past samples among a maximum number of past samples that can be approximated well by a linear function to predict a future sample. As shown in Figure 6, this method can generate very accurate prediction except when the FoV suddenly changes. Furthermore, the predicted trace has a very short time lag from the actual trace after sudden changes.

3.2.2 Segment-Level Bandwidth Prediction. In the beginning of each new segment, we predict the average sustainable throughput from the sender to the receiver based on the measured throughput in the past segments fedback by the receiver. In a prior work [10] on rate adaptation for video calls, an online linear adaptive filter called recursive least square (RLS) was found to achieve higher prediction accuracy than exponentially weighted moving average. Therefore, we adopt the RLS method in this work and use the measured bandwidth in previous two segments for the prediction. As shown in Figure 6, the prediction accuracy is very good with a short time lag after a sudden change. We would like to note that the proposed streaming system can easily adopt more accurate FoV and bandwidth prediction methods to further improve the performance.

3.3 Adaptation of coding rates and region sizes

Based on the estimated bandwidth available for transmitting the next segment and the current sending buffer size, we determine the average target bit budget for this segment. Then we determine the target average video rates in different regions and region sizes of all frames in the next segment by maximizing the expected video quality in Equation (8), using the method described in Section 2.3.2. We use the average hit rates and frame delivery rate determined in previous segment as the estimate for these variables for the next segment. We also use the time lapse distribution $p(\tau)$ computed from the previous segment to update the Q-R models as described in Section 2.2. The target rate of each frame inside a segment is further adjusted, depending on the sending buffer status measured at the time of encoding.

3.3.1 Assigning total bit budget considering sending buffer status. Due to the error in the average bandwidth prediction and the actual bandwidth fluctuation within a segment time, there could be occasional packet backlogs. To avoid the backlogged bits accumulating in the sender buffer over time, when encoding a new segment s, we calculate the target bit budget by subtracting the bits q_s remaining in the sender buffer before coding the segment from the predicted total bits that can be sent for the segment $\hat{b}_s$. Then we apply a bandwidth utilization ratio η to keep the probability to exceed the network capacity low. The work in [10] showed that $\eta = 66\%$ can keep this probability lower than 5%. The final bit budget for encoding segment s is

$$b_s = \eta(\hat{b}_s - q_s). \tag{13}$$

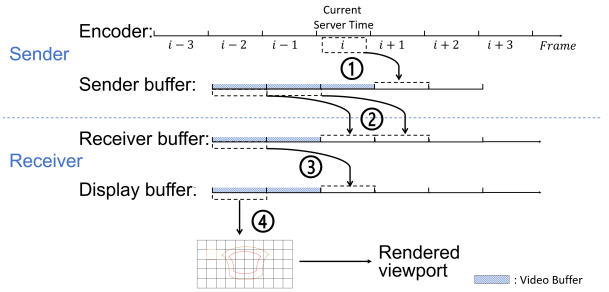

Figure 5: Illustration of the streaming system.

3.3.2 Frame-level Bit Budget Update. The bandwidth inside a segment can be unstable, especially over wireless connection time varying. History based bandwidth prediction at the segment level does not consider such variations. We adjust the bit budget for each frame based on the remaining bit budget of that segment and the current buffer occupation. Assuming that there are N frames in a segment, when coding frame n ($n = 0$ for the first frame in the segment), on average $\frac{n}{N} b_s$ bits should have been used. However, it is possible that the actual number of bits already generated S_n could be either smaller or larger than this estimation. To be conservative, we set the remaining bit budget as

$$b_s(n) = b_s - max\left(S_n, \frac{n}{N} b_s\right). \tag{14}$$

Then, we check the sender buffer occupation $BF(n)$; a rate higher than that in (14) is preferred if the sender buffer is almost empty, whereas a lower rate is more appropriate if the buffer is close to the maximum buffer length $BF_{\max}$. Therefore, if the sender buffer is not full, the target bit rate for this frame is adjusted as

$$B_t(n) = \frac{b_s(n)}{N - n} a \exp(-b * BF(n)/BF_{\max}), \tag{15}$$

where a and b are parameters that can be adjusted empirically. In our simulations, we set $a = 1.20$, $b = 1.00$, and $BF_{\max} = 10$ frames. As noted before, if the buffer is full, this frame is skipped.

4 TRACE-DRIVEN SIMULATION RESULTS

4.1 Traces and Benchmarks

To evaluate the proposed coding and streaming system, we develop a simulation system using real user FoV traces and wireless bandwidth traces. We use the cellular link capacity traces as described in [10]. We choose a particularly challenging trace with a $std/mean$ ratio of 0.673, shown in Figure 6. Furthermore, we scale up the bandwidth dynamic range to 50 to 200 Mbps to match the video rate range for the 8K testing video.

We evaluate the performance of the streaming system for two JVET 360° video testing sequences "Trolley" and "Chairlift". "Trolley" is a stable 360° video shot by a fixed camera. "Chairlift" is a dynamic 360° video shot by a moving camera. Both videos are represented in ERP format with 8192×4096 spatial resolution (8K), and 30 frames per second. "Trolley" uses 8 bits per color component

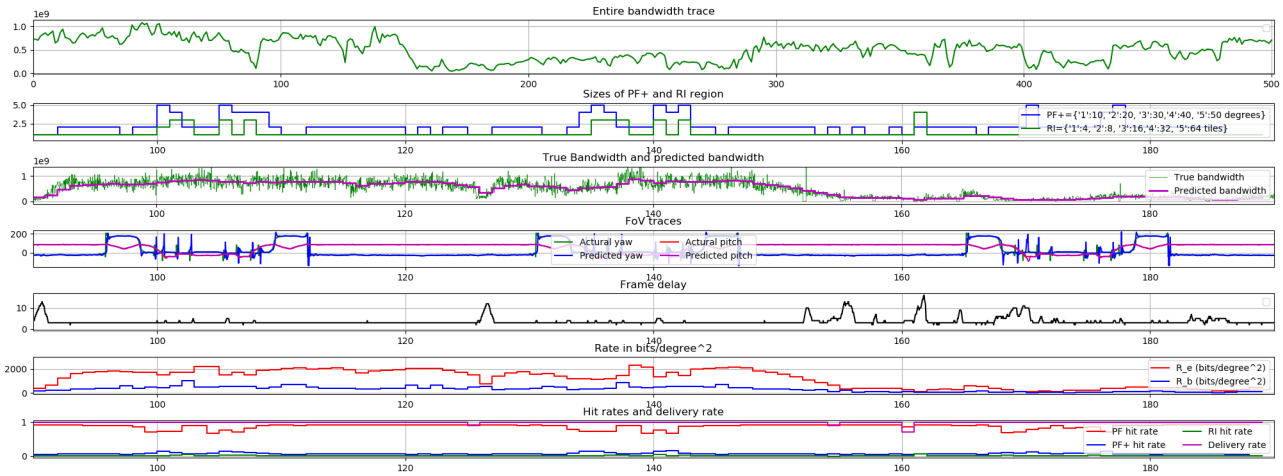

Figure 6: Top: the entire bandwidth trace; Below: various performance indices over a short duration.

while "Chairlift" uses 10 bits. We have investigated the impact of the tile size on the coding efficiency and found that a tile size of 256×256 pixels achieves the best trade-off between the coding efficiency and granularity for region size adaptation for 8K video[14]. Coding results shown in Figure 3 and 4 are all obtained with this tile size ,including the parameters for the "ideal" Q-R model, rate-increase factor model, and quality-decay factor model. For practical applications, we assume that such model parameters can be pre-determined for a few video categories (e.g. stationary vs. moving cameras, slow vs. fast object motion in the captured scene) and the category of the video can be determined in the beginning of a streaming session. Instead of actually doing video encoding and decoding, we use the "ideal" Q-R models and models for the rate-increase and quality-decay factors to determine the rate and quality of each tile (which may be coded as PF, PF+, RI or not coded). In our simulation, we continuously update a record of the time lapse since it is last coded for each tile in each video frame, and the quality when it is last coded. To determine the actual number of bits to inter-code a tile for a given target rate, we increase the target rate by the rate-increase factor based the time lapse. To evaluate the average WS-PSNR of the rendered video in a displayed frame, we use the time lapse of each rendered tile and the quality when it is last coded to determine the quality of this tile.

For the lack of real FoV traces for these two videos, we choose two groups of representative traces from open source 360° video FoV trace datasets [21, 22]. For "Trolley", we use the traces collected when users watched a video called "Weekly Idol-Dancing" shot by a fixed camera [22]. For "Chairlift", we use the traces collected when users viewed for "GoPro VR-Tahiti Surf" shot by a moving camera [21]. To eliminate the slight random jitters in these traces, the raw FoV traces are smoothed using a Kalman filter, and the filtered traces are used as the testing FoV traces. Because the duration of a FoV trace is shorter than our bandwidth trace, we repeat the FoV trace by concatenating the flipped FoV trace to itself repeatedly. We run the simulation using 48 users' FoV traces, and report the average results.

We first compare our system to two intra-coding benchmarks. Both use tile-based intra-coding. BM1 [20] codes and transmits a vertical slice to cover the predicted FoV in each frame (the vertical slice covers a $140 \times 180°$ region when the user FoV size is $90 \times 90°$). BM2 only code the tiles in an extended PF region using intra-coding. The extended region is equivalent to the union of the PF and PF+ regions in our proposed system, with a fixed PF+ size to cover a 50° border. For both benchmark systems, we use the Q-R models derived for the RI region to determine the WS-PSNR of coded tiles given the allocated bit budget per frame, and we further use the proposed quality-decay model to determine the WS-PSNR of the un-coded region in the FoV. We also compare our system to an inter-coding benchmark (BM3). Rather than using rotating intra regions, it codes the entire first frame in each segment as an intra frame. The remaining frames are inter-coded in the same extended PF region as in BM2 with the same rate. All benchmark systems use the same approaches as the proposed system, for FoV prediction, bandwidth prediction, and segment- and frame-level rate budget assignment. For BM3, the target rate for the I-frame is set to 2.5 times of that for the P-frame, to reach a good trade-off between the delay (caused by the increased rate of the I-frame) and quality degradation of the I-frame.

4.2 Performance metrics

For each system and each video, we report the average rendering quality (average WS-PSNR over all rendered pixels) over all displayed frames as well as the average frame delay, the freeze frequency and duration, and the frame delivery rate. In addition to the average WS-PSNR, we also evaluate the spatial and temporal quality variation, as such variation can affect the perceptual quality. The *temporal quality discontinuity* is the mean absolute difference between the rendering qualities of every two adjacent frames. The *spatial quality discontinuity* is the mean absolute difference between the rendering qualities of a tile and each of its neighboring tiles in the displayed FoV.

4.3 Evaluation results

In our simulation, we assume the PF size FoV covers a 90×90° region on the sphere. The PF+ sizes FoV can choose from a candidate set of {10°, 20°, 30°, 40°, 50°}. The RI size can choose from {4, 8, 16, 32, 64} tiles.

Figure 6 shows the region size and rate adaptation relative to the FoV trace and bandwidth trace over a chosen time interval with significant changes in FoV and bandwidth. It also shows the predicted vs. true bandwidth and FoV traces, respectively, and the frame delay traces. Note that because we predict the average bandwidth and

adapt the region size and rate allocation per second, the plots for these are constant within each second. We also present the average hit rates of various coded regions and the frame delivery rate per second. From this figure, we can see that frame-level FoV prediction using the proposed truncated linear predictor is mostly very accurate, except when there are sudden transitions of FoV. We can see the PF+ and RI size remain small while FoV prediction is accurate. When FoV prediction is less accurate, the PF+ and RI sizes increase. Furthermore, frame delay increases when the predicted bandwidth is higher than the actual bandwidth. When the bandwidth drops to very low occasionally, the frame delivery rate drops, and a larger RI size is used to shorten the refresh period of the RI region.

Metric	BM1	BM2	BM2	Prop.	Simp.
WS-PSNR in FoV (dB)	36.29	38.22	44.66	48.31	48.23
Temporal discontinuity (dB)	0.247	0.206	0.567	0.229	0.265
Spatial discontinuity (dB)	0.269	0.002	0.008	0.318	0.442
Average delay (ms)	92.64	92.64	107.01	95.03	94.80
Delay STD/Average	0.380	0.378	0.451	0.431	0.429
% of freeze frames (%)	0.260	0.260	0.588	0.296	0.287
Average freeze duration (ms)	22.65	22.27	43.01	31.19	30.47
Display interval average (ms)	33.43	33.43	33.47	33.43	33.43
Display interval STD (ms)	12.20	12.14	15.81	12.60	12.59
Average hit rate, PF (%)	N/A	N/A	N/A	90.42	90.44
Average hit rate, PF+ (%)	N/A	N/A	N/A	7.78	8.55
Average hit rate, RI (%)	N/A	N/A	N/A	0.91	0.88
Average hit rate, total (%)	54.30	99.78	99.83	99.11	99.86

Table 1: Trolley video (fixed camera).

Tables 1 and 2 compare the proposed system with the three benchmark systems using average metrics over all the FoV traces, for the two videos. Compared to the two benchmark systems using intra-coding (BM1 and BM2), the proposed system is able to achieve significantly higher average WS-PSNR (6-10dB higher) through the use of FoV adaptive inter-coding. However it has slightly higher spatial discontinuity, because it codes the PF and PF+ region using different rates. The BM2 system has better quality than BM1 because BM2 generally codes and transmits a smaller area surrounding the predicted FoV than the BM1. BM1's low quality is also in part because it uses a fixed width vertical slice, which does not cover the entire FoV when the FoV is towards the north or south pole, leading to a low FoV hit rate. Because all the systems use the same approach for bandwidth estimation and bit budget allocation, these systems are similar in terms of average frame delay, and freeze duration, and percentages of freeze frames. Most importantly, all the systems have very low frame delays with mean < 100 ms and a small variance. The freeze probabilities (< 0.5%) and freeze duration (≈ 1 frame time) are also very low. The proposed system has slightly higher delay and freeze because of the video rate variation within each segment caused by the variable coding time lapses due to the FoV dynamics. Recall that we predict the time lapse distribution in the new segment based on the distribution in the past segment; When the FoV suddenly changes from stationary to dynamic, such prediction is not accurate, leading to the actual video rate higher than allocated, causing extra delay and occasionally freezing. The benchmark systems do not experience such rate variation, because they use intra-coding.

Metric	BM1	BM2	BM3	Prop.	Simp.
WS-PSNR in FoV (dB)	37.27	38.67	42.97	45.25	45.21
Temporal discontinuity (dB)	0.191	0.147	0.387	0.164	0.185
Spatial discontinuity (dB)	0.189	0.002	0.009	0.218	0.305
Average delay (ms)	92.64	92.64	109.25	98.41	98.07
Delay STD/Average	0.380	0.378	0.501	0.439	0.435
% of freeze frames (%)	0.260	0.260	0.667	0.382	0.367
Average freeze duration (ms)	22.65	22.27	57.33	45.77	44.24
Display interval average (ms)	33.43	33.43	33.47	33.42	33.42
Display interval STD (ms)	12.20	12.14	15.86	13.12	13.09
Average hit rate, PF (%)	N/A	N/A	N/A	89.98	90.02
Average hit rate, PF+ (%)	N/A	N/A	N/A	7.74	8.61
Average hit rate, RI (%)	N/A	N/A	N/A	0.98	0.94
Average hit rate, total (%)	77.67	99.45	99.59	98.71	99.57

Table 2: Chairlift video (moving camera).

Compared to the benchmark inter-coding system (BM3), the proposed system achieves higher quality and lower delay. The higher average WS-PSNR (2-4dB higher) is because it codes the PF+ region at a lower rate than the PF region, and it codes a smaller percentage of tiles using the intra-mode. BM3 suffers from a higher frame delay and more frequent freezing because the periodic I-frame has a target rate higher than the P-frame. We set the target I-frame rate to be 2.5 times of the target P-frame rate, to avoid a bigger rate spike, which would have further increased frame delay and freezing. On the other hand, this made the I-frame quality to be lower than the P-frame, leading to higher temporal discontinuity than other methods.

Between the two videos, "Chairlift" has lower quality primarily because of its fast moving content and hence lower Q-R efficiency (see Figure 3). "Chairlift" also suffers from slightly higher delay and freeze due to the more significant video rate variation caused by the more dynamic FoV traces.

To evaluate the gain from adapting both the region sizes and rate allocation in the proposed system, we also examine a simplified version of the proposed system, where the region sizes are fixed (PF+ is set to 50°, while RI is 4 tiles). As shown in Table 1 and 2, the performance degradation from using fixed sizes for the PF+ and RI is quite small. Hence, this simplified system may be preferred for practical implementation.

5 CONCLUDING REMARKS

This work focuses on overcoming the challenges in integrating temporal predictive coding into low-latency, FoV-adaptive coding and streaming of interactive 360° video. Through accurate quality-rate modeling that explicitly considers the reduced coding efficiency due to the prolonged temporal prediction time lapse, the system can achieve accurate rate control at both the segment and frame levels and optimize rate allocation to maximize the rendering quality. By introducing rotating intra-regions, the system can periodically stop both error propagation due to frame losses as well as quality degradation in un-coded regions, without causing bit rate spikes that increase frame delay. Together with push-based frame delivery and target rate adaptation at both segment and frame levels, the proposed system has been shown to be capable of reducing the mean end-to-end delay to below 100ms under challenging bandwidth traces. Compared to benchmark systems, the proposed system improves the average WS-PSNR by 2 to 10 dB, which should correspond to substantial improvement in the perceived quality.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1816500.

REFERENCES

[1] Ridvan Aksu, Jacob Chakareski, and Viswanathan Swaminathan. 2018. Viewport-driven rate-distortion optimized scalable live 360° video network multicast. In *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. 1–6.

[2] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 1161–1170.

[3] Jill Boyce, Elena Alshina, Adeel Abbas, and Yan Ye. 2017. JVET common test conditions and evaluation procedures for 360 video. *Joint Video Exploration Team of ITU-T SG* 16 (2017).

[4] Xavier Corbillon, Alisa Devlic, Gwendal Simon, and Jacob Chakareski. 2017. Optimal set of 360-degree videos for viewport-adaptive streaming. In *Proceedings of the 25th ACM international conference on Multimedia*. 943–951.

[5] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *2017 IEEE international conference on communications (ICC)*. IEEE, 1–7.

[6] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Netwbibliographyork and Operating Systems Support for Digital Audio and Video*. ACM, 67–72.

[7] Carsten Griwodz, Mattis Jeppsson, Håvard Espeland, Tomas Kupka, Ragnar Langseth, Andreas Petlund, Peng Qiaoqiao, Chuansong Xue, Konstantin Pogorelov, Micheal Riegler, et al. 2018. Efficient live and on-demand tiled hevc 360 vr video streaming. In *2018 IEEE International Symposium on Multimedia (ISM)*. IEEE, 81–88.

[8] Xueshi Hou, Yao Lu, and Sujit Dey. 2017. Wireless VR/AR with edge/cloud computing. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 1–8.

[9] IK Kim, K McCann, K Sugimoto, B Bross, WJ Han, and G Sullivan. 2014. High efficiency video coding (HEVC) test model 14 (HM 14) encoder description. Document: JCTVC-P1002. *JCT-VC, Jan* (2014).

[10] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. 2016. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–11.

[11] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. 2015. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. 151–165.

[12] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang. 2019. Very Long Term Field of View Prediction for 360-degree Video Streaming. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 297–302.

[13] Xing Liu, Bo Han, Feng Qian, and Matteo Varvello. 2019. LIME: understanding commercial 360° live video streaming services. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 154–164.

[14] Yixiang Mao, Liyang Sun, Yong Liu, and Yao Wang. 2020. *Supplementary Document: Video Coding Experiments*. https://s18798.pcdn.co/videolab/wp-content/uploads/sites/10258/2020/08/Supplementary_Documents_MM2020_360interactive-2.pdf

[15] Duc V Nguyen, Huyen TT Tran, Anh T Pham, and Truong Cong Thang. 2019. An optimal tile-based approach for viewport-adaptive 360-degree video streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 29–42.

[16] Mikko Pitkänen, Marko Viitanen, Alexandre Mercat, and Jarno Vanne. 2019. Remote VR Gaming on Mobile Devices. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2191–2193.

[17] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[18] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2019. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 43–57.

[19] Liyang Sun, Yixiang Mao, Tongyu Zong, Yong Liu, and Yao Wang. 2020. Flocking-based live streaming of 360-degree video. In *Proceedings of the 11th ACM Multimedia Systems Conference*. 26–37.

[20] Marko Viitanen, Jarno Vanne, Timo D Hämäläinen, and Ari Kulmala. 2018. Low latency edge rendering scheme for interactive 360 degree virtual reality gaming. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1557–1560.

[21] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A dataset for exploring user behaviors in VR spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 193–198.

[22] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze prediction in dynamic 360 immersive videos. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5333–5342.

[23] Liyang Zhang, Syed Obaid Amin, and Cedric Westphal. 2017. VR video conferencing over named data networks. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*. 7–12.