360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming

Lan Xie[1], Zhimin Xu[1], Yixuan Ban[1,3], Xinggong Zhang[1,2,*], Zongming Guo[1,2]
[1]Institute of Computer Science & Technology, Peking University, Beijing, China
[2]Cooperative Medianet Innovation Center, Shanghai, China
[3]Beijing University of Posts and Telecommunications, Beijing, China
{xielan,xuzhimin,zhangxg,guozongming}@pku.edu.cn,banyixuan@bupt.edu.cn

ABSTRACT

Recently, there has been a significant interest towards 360-degree panorama video. However, such videos usually require extremely high bitrate which hinders their widely spread over the Internet. Tile-based viewport adaptive streaming is a promising way to deliver 360-degree video due to its on-request portion downloading. But it is not trivial for it to achieve good Quality of Experience (QoE) because Internet request-reply delay is usually much higher than motion-to-photon latency. In this paper, we leverage a probabilistic approach to pre-fetch tiles countering viewport prediction error, and design a QoE-driven viewport adaptation system, 360ProbDASH. It treats user's head movement as probability events, and constructs a probabilistic model to depict the distribution of viewport prediction error. A QoE-driven optimization framework is proposed to minimize total expected distortion of pre-fetched tiles. Besides, to smooth border effects of mixed-rate tiles, the spatial quality variance is also minimized. With the requirement of short-term viewport prediction under a small buffer, it applies a target-buffer-based rate adaptation algorithm to ensure continuous playback. We implement 360ProbDASH prototype and carry out extensive experiments on a simulation test-bed and real-world Internet with real user's head movement traces. The experimental results demonstrate that 360ProbDASH achieves at almost 39% gains on viewport PSNR, and 46% reduction on spatial quality variance against the existed viewport adaptation methods.

CCS CONCEPTS

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

KEYWORDS

360-degree video, tile-based adaptive streaming, rate and viewport adaptation, DASH, QoE-driven Optimization

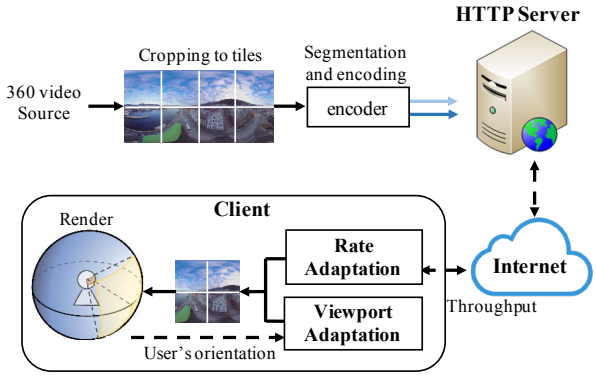


Figure 1: System Overview of Tile-based HTTP Adaptive Streaming for 360 Video.

1 INTRODUCTION

With the increasing demand on better user experience in interactive applications, virtual reality (VR) techniques have become increasingly important nowadays. One of the most attractive applications is 360-degree panorama video (360 video). By wearing head mounted displays (HMDs), such as Oculus Rift [3], users are able to obtain immersive experience, feeling free to control their orientation during video playback. To provide a good experience, the resolution of 360 video should be 6K or even higher. However, streaming 360 video at such resolution is not a trivial task, especially through the Internet. Besides, only a portion of the video is used at a time. Therefore, transmitting whole 360 video content and treating it as ordinary video, such as the strategy of YouTube [6], results in inevitable waste of bandwidth and computational resources. Using Dynamic Adaptive Streaming over HTTP (DASH) [24], *viewport adaptive streaming* is regarded as a promising way to deliver 360 video through the Internet. It is performed in such way that high quality is preserved within the Field of View (FOV), i.e. viewport, while others are delivered in low quality or even discarded.

There are mainly two categories of viewport adaptive streaming: asymmetric panorama and tile-based. The asymmetric panorama method [23], such as Truncated Pyramid Projection (TSP) [15] and Facebook's offset cubemap [16], transforms a 360 video into viewport-dependent multi-resolution panorama, which decreases the overall resolution without decreasing the quality of the viewport. This method is prevalent since it still provides 360-degree image. But the other side of the coin is that it would waste bandwidth resources because user's viewport is limited in FOV.

* Dr. Xinggong Zhang is the corresponding author of this paper.

MM '17, October 23–27, 2017, Mountain View, CA, USA

ACM ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3123291

Tile-based method is an emerging method which is more promising for viewport adaptive streaming [9–13, 19–21, 28]. As shown in Fig. 1, it crops 360 video frames into multiple tiles (or blocks) in space; then partitions and encodes the tiles into multi-bitrate segments. The client only pre-fetches the tiles within the predicted viewport. In tile-based HTTP adaptive streaming, the client has to do two adaptations: *rate adaptation* to adapt time-varying bandwidth, and *viewport adaptation* to cope with user's head movement. Although the tile-based streaming is more flexible, it is not trivial to provide high QoE streaming service. The challenges mainly arise from three aspects:

(1) *Tiles pre-fetching error*: Since the motion-to-photon latency requirement for VR is less than 20ms [26], which is much smaller than Internet request-reply delay. Therefore, it is necessary to pre-fetch tiles by viewport prediction. However, it is difficult to accurately predict user's orientation especially for long-term prediction (> 3s) [19]. If the video tiles that cover the new viewport is not streamed, blank block will be rendered in screen which leads to QoE drop;
(2) *Rebuffering/stall under small playback buffer*: Due to short-term constraint of viewport prediction, tile-based method should keep a small playback buffer (< 3s). How to avoid playback rebuffering/stall with small playback buffer is challenging for streaming over harsh Internet;
(3) *Border effects of mixed-bitrate tiles*: Due to the spatial partition of video, the mixed-bitrate tiles could result in visible border and quality inconsistence in combined-tiles rendering.

To address these challenges, in this work, we present a *probabilistic* tile-based adaptive streaming system, namely 360ProbDASH, that leverages probabilistic model of viewport prediction and expected quality optimization framework to maximize the quality of viewport adaptive streaming. Specifically, for small-buffer rate adaptation, we use a *target-buffer-based rate control* algorithm to maintain playback buffer around a small target size, aiming at avoiding rebuffering. To cope with the prediction error of user's orientation, we propose a *probabilistic model of viewport* to calculate the viewing probability of tiles. Then, we formulate a *QoE-driven optimization* problem: minimizing expected quality distortion and spatial variability of quality under the constraint of total transmission bitrate. By solving the problem, we can obtain the optimal tiles to stream.

While details are presented in the paper, some highlights of our contributions include the followings:

- By using a probabilistic model of viewport prediction, 360ProbDASH apparently reduces the side effects caused by wrong head movement prediction.
- The target-buffer-based rate control algorithm in 360ProbDASH significantly reduces video playback stall for 360 video streaming with small playback buffer.
- We mathematically formulate the rate and viewport adaptation into an QoE-driven optimization problem. By minimizing the quality distortion and spacial quality variance, clients achieve the best QoE for 360 video streaming.
- We implement a 360ProbDASH prototype and carry out extensive experiments on a simulation test-bed and real-world Internet with various real user's head movement traces. The results demonstrate that 360ProbDASH achieves at almost 39% gains on viewport PSNR, and 46% reduction on spatial quality variance against the existed viewport adaptation methods.

The rest of paper is organized as follows. Section II surveys related works on 360 video and tile-based streaming. In Section III, we presents the 360ProbDASH in details. Section IV described the system implementation. Performance evaluation and comparison is presented in Section V. Finally, Section VI concludes the paper and outlines future directions.

2 BACKGROUND AND RELATED WORK

Panorama or 360-degree video is constructed by stitching images from camera rigs. To play a 360-degree video, the video client, running on an HMD or a mobile phone, requires a graphic engine to render the video. As shown in Fig. 1, part of the panoramic video is reprojected onto the screen according to user's orientation and the HMD's FOV. Most application currently takes 360 video as ordinary content, and delivery them with full picture, such as widely used Equirectangular Projection (ERP) [22] and CubeMap Projection (CMP) [18], etc. However, transmission full view of 360 video in such format requires high bandwidth. Hence, viewport adaptive streaming is proposed to save transmission bandwidth which can be classified into two categories: asymmetric panorama-based streaming and tile-based streaming.

For asymmetric panorama-based streaming, a 360 video is transformed and encoded into several versions towards different perspectives. The asymmetric panorama contains 360-degree content with different resolution. Truncated Pyramid Projection (TSP) [15] and Facebook's offset cubemap [16] are the typical formats to represent asymmetric panorama, decreasing the overall bitrate without decreasing the quality of the viewport. During the video playback, the client requests one of the video versions according to user's orientation. The advantage of asymmetric panorama-based streaming is that even if the client wrongly predicts the user's orientation, content can still be rendered in user's viewport with low quality. However, such scheme results in inflexible and bandwidth waste in most cases.

In tile-based streaming, it has proven effective in domains such as online video lectures [17] and sport [11]. In terms of 360 video, Zare *et al.* [28] propose an HEVC-compliant tiles streaming approach utilizing the motion-constrained tile sets (MCTS) for delivery 360 video. They propose that the tiles currently viewed by the user are streamed in high resolution while the rest of the tiles are streamed in low resolution. However, transmission whole image results in waste of bandwidth. In [14], the authors propose a viewport adaptation scheme that the tiles are selected based on user's viewport without prediction. To better predict user's orientation in the near future, Feng *et al.* [19] propose to use bandwidth-based rate adaptation [8] and apply Linear Regression model to predict the user's orientation, and then stream only the visible portion. However, the accuracy drops to 70% when predicting the user's orientation in the future 2 seconds. As a result, blank block could be rendered

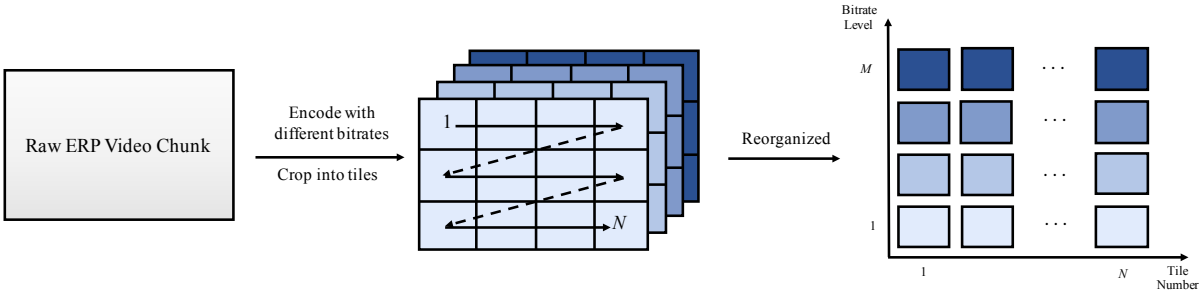


Figure 2: System Diagram of 360ProbDASH

in the user's screen if the client requests video tiles according to the wrong prediction result. Besides, in tile-based streaming, poor adaptation algorithm will result in apparent border and content quality inconsistent. In [25], the authors evaluate the perceptual effect of mixed-resolution tiles in tile-based video with low and medium motion. However, 60% of the participants express that they don't accept the quality degradation when the qualities of tiles are within great difference. Recently, MPEG has already standardized a new feature called Spatial Relationship Description (SRD) [7] to support tile-based streaming of DASH. By carefully design rate adaptation and viewport adaptation, we can leverage the tile-based HTTP adaptive streaming to provide high user's QoE of 360 video.

3 SYSTEM MODEL AND DESIGN

In this section, we present our probabilistic tile-based adaptive streaming system 360ProbDASH, that leverages probabilistic model of viewport prediction and expected quality optimization framework to maximize the quality of viewport adaptive streaming.

In tile-based HTTP adaptive streaming, a raw panoramic video in ERP format is temporally divided into several video chunks with same duration. As illustrated in Fig. 2, then for each chunk, it is spatially cropped into N tiles which are indexed in raster-scan order. Besides, each tile is encoded into segments with M kinds of bitrate levels versions. Consequently, there are $M \times N$ optional segments stored at the server which are ready for streaming. The main purpose of the proposed viewport adaptation algorithm is to determine which segments to be pre-fetched. Next, we formulate the problem under a QoE-driven optimization framework and present a probabilistic viewport adaptation model and a target-buffer-based rate adaptation algorithm.

3.1 Problem Formulation

As shown in Fig. 2, there are $M \times N$ optional segments, where $i \in \{1...N\}$ denotes tile sequence number and $j \in \{1...M\}$ denotes bitrate level. We define $r_{i,j}$ and $d_{i,j}$ are the bitrate and corresponding distortion of segment(i, j). Let p_i be the normalized viewing probability of i-th tile, such that $\sum_{i=1}^{N} p_i = 1$. In this problem, we want to find the set of streaming segments, $\boldsymbol{X} = \{x_{i,j}\}$, while $x_{i,j} = 1$ denote the segment of i-th tile at j-th bitrate level is selected for streaming and $x_{i,j} = 0$ otherwise.

To maximize the quality of viewport adaptive streaming, we define two QoE functions: 1) expected distortion $\Phi(\boldsymbol{X})$: it denotes the quality distortion of viewport under the consideration of viewing probability of tiles. 2) spatial quality variance $\Psi(\boldsymbol{X})$: it represents the quality smoothness in a viewport.

Our objective is to minimize the weighted distortion of these two QoE functions where η is the weight for spatial quality variance. Therefore, our optimization problem can be formulated as:

$$\begin{aligned} \min_{\boldsymbol{X}} \ & \Phi(\boldsymbol{X}) + \eta \cdot \Psi(\boldsymbol{X}) \\ \text{s.t.} \ & \sum_{i=1}^{N} \sum_{j=1}^{M} x_{i,j} \cdot r_{i,j} \le R, \\ & \sum_{j=1}^{M} x_{i,j} \le 1, \ x_{i,j} \in \{0,1\}, \ \forall i. \end{aligned} \tag{1}$$

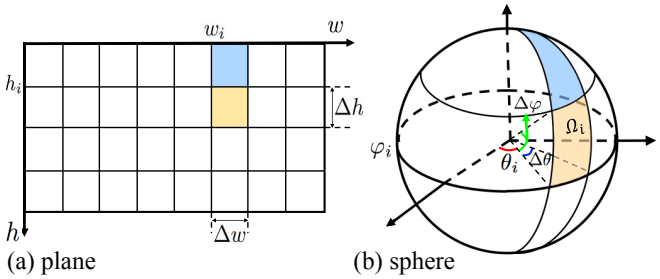
The first constraint in the optimization problem restricts the total bitrate of selected segments. To avoid playback stall, we set a transmission bitrate budget R which is calculated from our target-buffer-based rate control algorithm. Then, the second constraint gives the restriction on $x_{i,j}$. Obviously, it only needs to select at most one bitrate level of each tile.

By solving the optimization problem (1), the 360ProbDASH can select the segments to provide high user's QoE.

3.2 Expected Viewport Distortion and Variance

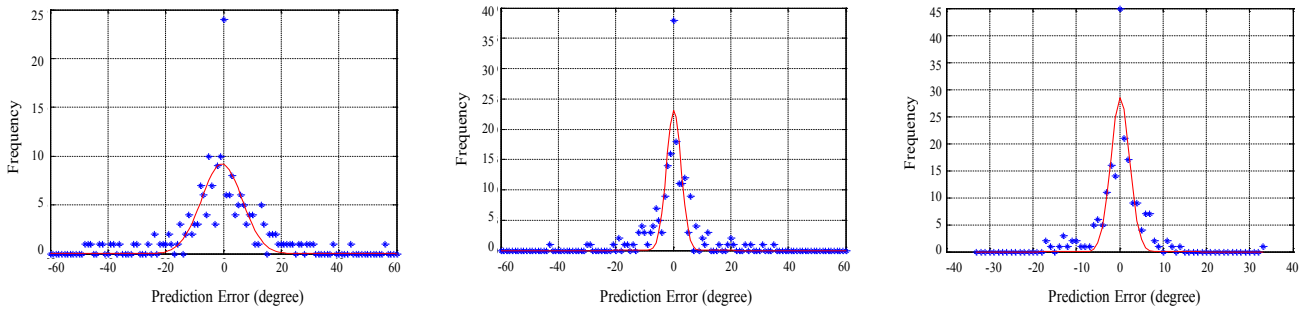
In this subsection, we present the calculation of expected viewport distortion and spatial quality variance in the optimization problem. We should take the sphere-to-plane mapping and viewing probability of tiles into consideration.

3.2.1 Spherical Distortion of segment. In 360 video, it usually uses S-PSNR [27] to evaluate the quality which is calculated via Mean Squared Error (MSE) of points on sphere. In our work, $d_{i,j}$ indicates the MSE corresponding to segment(i, j).



(a) plane (b) sphere

Figure 3: Spherical Mapping of Tiles



(a) Yaw (b) Pitch (c) Roll

Figure 4: The distribution of prediction error.

The overall spherical distortion of a segment is the sum of distortion over all pixels the segment covers. Therefore, it is required to calculate a tile's corresponding *spherical area*. This is because, as shown in Fig.3, even if tiles have same area in plane, their corresponding area on sphere are not the same.

We define the plane coordinates of i-tile as $h \in [h_i, h_i + \Delta h]$ and $w \in [w_i, w_i + \Delta w]$ where Δh and Δw are the height and width of a tile. According to the projection function of ERP [4], its corresponding spherical coordinates is $\varphi \in [\varphi_i, \varphi_i + \Delta\varphi]$ and $\theta \in [\theta_i, \theta_i + \Delta\theta]$. The polar angle φ_i (latitude), azimuthal angle θ_i (longitude) and their variations are calculated as:

$$\begin{aligned} \varphi_i &= \frac{\pi}{2} - h_i \cdot \frac{\pi}{H}, \qquad \Delta\varphi = \Delta h \cdot \frac{\pi}{H}, \\ \theta_i &= w_i \cdot \frac{2\pi}{W}, \qquad \Delta\theta = \Delta w \cdot \frac{2\pi}{W}, \end{aligned} \tag{2}$$

where H and W are the height and width of the 360 video in ERP format. Then, the spherical area of i-th tile is calculated as:

$$\begin{aligned} s_i &= \iint_{\Omega_i} \mathcal{R} d\varphi R \cos\varphi d\theta \\ &= \Delta\theta \mathcal{R}^2 [\sin(\varphi_i + \Delta\varphi) - \sin\varphi_i], \end{aligned} \tag{3}$$

where $\mathcal{R}$ is the radius of the sphere, such that $\mathcal{R} = W/2\pi$. Hence, the overall spherical distortion of segment(i, j) is calculated as:

$$D_{i,j} = d_{i,j} \cdot s_i. \tag{4}$$

3.2.2 Expected Viewport Distortion and Quality Variance. Let p_i denotes the viewing probability of i-th tile. The Expected Viewport Distortion and Spatial Quality Variance should be:

$$\begin{aligned} \Phi(\boldsymbol{X}) &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} D_{i,j} \cdot x_{i,j} \cdot p_i}{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} \cdot s_i}, \\ \Psi(\boldsymbol{X}) &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} \cdot p_i \cdot (D_{i,j} - s_i \cdot \Phi(\boldsymbol{X}))^2}{\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} \cdot s_i}. \end{aligned} \tag{5}$$

3.3 Probabilistic Model of Viewport

In 360ProbDASH, we need to pre-fetch segments by predicting viewport. However, the prediction may be inaccurate which will lead to viewport deviation. Hence, in viewport adaptation, we propose a probabilistic model to predict viewport.

3.3.1 Linear Regression Prediction of Orientation. We denote the user's orientation (Euler angle), as yaw (α), pitch (β) and roll (γ) and leverage Linear Regression (LR) model to do prediction. We denote t_0 as the current time of system. By using the historical samples in window $(t_0 - 1, t_0]$, we apply Least Square Method (LSM) to calculate the trends of head movements. We denote the slope over yaw, pitch and roll as m_α, m_β and m_γ. Therefore, the estimated value of yaw, pitch and roll at time of $t_0 + \delta$ can be predicted using Linear Regression model as:

$$\begin{cases} \hat{\alpha}(t_0 + \delta) = m_\alpha \delta + \alpha(t_0), \\ \hat{\beta}(t_0 + \delta) = m_\beta \delta + \beta(t_0), \\ \hat{\gamma}(t_0 + \delta) = m_\gamma \delta + \gamma(t_0). \end{cases} \tag{6}$$

3.3.2 Distribution of Prediction Error. Viewport is hard to predict accurately, especially for long-term prediction. Thus, we use a probabilistic model to represent prediction error. We collect 5 user head movement traces and investigate the probability distribution of short-term prediction error. Fig. 4 shows the distribution of LR prediction error under $\delta = 3s$. As a consequence, we can reasonably assume the prediction error of yaw, pitch and roll follows Gaussian Distribution such as $e_\alpha \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha^2)$, where mean μ_α and standard deviation σ_α can be learned from statistics. Therefore, the probability of an arbitrary yaw α, pitch β and roll γ is exact the real value and can be derived as:

$$\begin{cases} P_{\text{yaw}}(\alpha) = \frac{1}{\sigma_\alpha\sqrt{2\pi}} \exp\{-\frac{[\alpha - (\hat{\alpha} + \mu_\alpha)]^2}{2\sigma_\alpha^2}\}, \\ P_{\text{pitch}}(\beta) = \frac{1}{\sigma_\beta\sqrt{2\pi}} \exp\{-\frac{[\beta - (\hat{\beta} + \mu_\beta)]^2}{2\sigma_\beta^2}\}, \\ P_{\text{roll}}(\gamma) = \frac{1}{\sigma_\gamma\sqrt{2\pi}} \exp\{-\frac{[\gamma - (\hat{\gamma} + \mu_\gamma)]^2}{2\sigma_\gamma^2}\}. \end{cases} \tag{7}$$

Since yaw, pitch and roll are independent of each other, the correct probability that an Euler angle (α, β, γ) is exactly the user's orientation is calculated as:

$$P_E(\alpha, \beta, \gamma) = P_{\text{yaw}}(\alpha) P_{\text{pitch}}(\beta) P_{\text{roll}}(\gamma). \tag{8}$$

3.3.3 Viewing Probability of Points on Sphere. To calculate the viewing probability of a tile, we need to firstly calculate the viewing probability of points on sphere according to the probability of user's orientation. We define the viewing probability of a spherical point (φ, θ) that user may view as $P_s(\varphi, \theta)$. Since a spherical point could be viewed among multiple viewports, we define $\boldsymbol{L}(\varphi, \theta)$ as the set

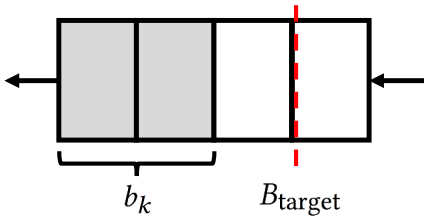


Figure 5: Dynamics of small playback buffer.

of user's orientations from which point (φ, θ) can be seen. Thus, the viewing probability of a spherical point $P_s(\varphi, \theta)$ equals to the average probability of orientations in $\boldsymbol{L}(\varphi, \theta)$ as:

$$P_s(\varphi, \theta) = \frac{1}{|\boldsymbol{L}(\varphi, \theta)|} \sum_{(\alpha, \beta, \gamma) \in L(\varphi, \theta)} P_E(\alpha, \beta, \gamma), \tag{9}$$

3.3.4 Viewing Probability of Tiles. At last, the viewing probability of a specific tile, i.e. p_i, can be calculated by averaging the probability of spherical points contained in the tile. Let $\boldsymbol{U_i}$ denote the set of spherical points corresponding to i-th tile. Thus, the viewing probability of i-th tile is:

$$p_i = \frac{1}{|\boldsymbol{U_i}|} \sum_{(\varphi, \theta) \in \boldsymbol{U_i}} P_s(\varphi, \theta). \tag{10}$$

3.4 Target-Buffer-based Rate Control

Since long-term head movement prediction results in high prediction error, it is not able to employ a large playback buffer to smooth bandwidth variation. Aiming at providing continuous playback under a small buffer, we propose a target-buffer-based rate control algorithm.

Fig. 5 shows the dynamics of small playback buffer in the client. The buffer occupancy is generally tracked in *seconds of video*. We group a set of segments with same timestamps as one chunk stored in the buffer. At adaptation step k, we define b_k as the buffer occupancy when the k-th set of segments are downloaded completely. We denote the total bitrate as R_k and the average bandwidth as C_k. Therefore, the buffer occupancy when finishing downloading the segments is calculated as:

$$b_k = b_{k-1} - \frac{R_k \cdot T}{C_k} + T. \tag{11}$$

If we want to prevent rebuffering, the buffer must be controlled to avoid running out of chunks. Due to the small buffer constraint, we set a target buffer level B_{target} to ensure continuous playback. We prefer to let the buffer occupancy stay at B_{target}, that is $b_k = B_{\text{target}}$. Combined with Eq. (11), the total bitrate of selected segment should satisfy:

$$R_k = \frac{C_k}{T} \cdot (b_{k-1} - B_{\text{target}} + T), \tag{12}$$

where C_k is network bandwidth which can be estimated from historic segments downloading. We set a lower bound $R_{\min}$ to R. Then, we can modify Eq. (12) into Eq. (13):

$$R_k = \max\{\frac{C_k}{T} \cdot (b_{k-1} - B_{\text{target}} + T), R_{\min}\}. \tag{13}$$

The bitrate calculated from Eq. (13) is used as the total bitrate budget constraint in our optimization problem (1).

4 IMPLEMENTATION

To evaluate the performance of 360ProbDASH, we have implemented a prototype. We present the implementation details in this section. The system architecture is shown in Fig. 6 which contains several key components.

4.1 Server Implementation

As shown in Fig. 6, in the server side, a 360 video should be processed and stored at a HTTP server. It contains the following key components:

(1) Video Cropper: It spatially crops the video frames into a desired number of tiles. This module also can be integrated to video encoder by modifying encoder.
(2) Encoder: This module partitions and encodes each tile into multiple bitrate segments.
(3) MPD Generator: MPD is an XML document containing information about media segments. To support tile-based streaming, we add SRD to describe the spatial relationship of segments. Specifically, in Representation element, we add *longitude* and *latitude* attribute corresponding to the representation. Besides, we add *quality distortion* attribute and *size* attribute of each segment in order to support our proposed 360ProbDASH to do adaptation.
(4) Apache HTTP Server: The HTTP server stores media segments and corresponding MPD. It provides 360ProbDASH service to the clients.

4.2 Client Implementation

We implement the video player based on the open source project *dash.js* [1], which is a DASH-compliant video player in JavaSript. To support our 360ProbDASH, we integrate additional modules into the adaptation algorithm of *dash.js*, namely:

(1) QoE-driven Optimizer: This module determines the optimal segments to download which are involved in the HTTP GET requests. It takes the output of *Target-buffer-based Rate Controller* module, *Viewport Probabilistic Model* module and *QR Map* module.
(2) Target-buffer-based Rate Controller: To avoid playback stall, we need to control the buffer stay at a target level. Thus, this module calculates the total transmission bitrate budget of segments according to Eq. (13) which takes the output of *Bandwidth Estimation* module.
(3) Viewport Probabilistic Model: This module calculates the viewing probability of each tile considering the user's orientation prediction error. It takes the output of *Orientation Prediction* module and SRD information and calculates the result according to Eq. (10).
(4) QR Map: This module generates a quality-rate (QR) map for all segments according to the attributes in MPD.
(5) Bandwidth Estimation: We can record the download duration of segments and then obtain the corresponding time-varying throughput. This can be achieved by the *onProgress* callback of the standard XMLHttpRequest API of the web

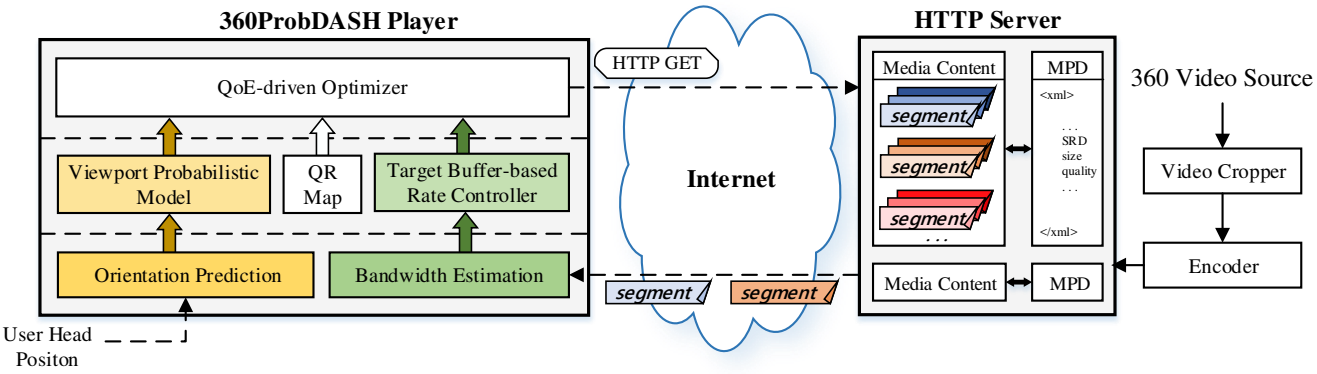


Figure 6: System Architecture

browser. Since the bandwidth estimation is not the key point of this paper, we just use the average result of previous 3 seconds.

(6) Orientation Prediction: The monitoring of user's orientation can be achieved by the *DeviceOrientation* callback of the standard Web API. Then, we apply Linear Regression model to predict the user's orientation.

In addition, to play 360 videos, we develop a graphic engine as a player wrapper based on Web Graphics Library (WebGL). Therefore, the user can watch 360 video through our 360ProbDASH.

5 PERFORMANCE EVALUATION

To evaluate the performance of 360ProbDASH, we carry out extensive simulation experiments and real-world Internet experiments under various head movement traces and network conditions.

5.1 Setup

In the experiments, we imitate user's head motion by embedding real user's head movement trace into the player and actively manipulate the network conditions to observe how different schemes react to the network fluctuations. Specifically, we examine the performance on video sequence and 5 user's head movement traces on this video, which are generously provided by AT&T [19]. The sequence is about 3 minutes long with the resolution 2880×1440 in ERP format. We chop it into small chunks with a constant duration of 1 second ($T = 1$). Then, for each chunk, we further partition it into 6x12 tiles ($N = 72$). The bitrate levels of each segment are set as {20kbps, 50kbps, 100kbps, 200kbps, 300kbps}. The video codec is the widely used open source encoder x264 [5]. All video segments are packaged by the tool *MP4Box* [2]. It is worth noting that the exact size of each segment could be different from its coding rate, especially for the chunks with short duration. To avoid its impacts on rate adaptation, we also include segment size into MPD file. At the client, the buffer size is set to $B_{\max} = 3$s. The target buffer level is set to $B_{\text{target}} = 2.5$s and $R_{\min}$ is set to 200kbps. The weight in our objective function is set to $\eta = 0.0015$ which shows a good trade-off between expected quality and spatial variance in quality. We use 5 head movement traces apart from the test head movement traces as the training set, on which we train the parameters in the Gaussian Distribution of prediction error. The result is:

Table 1: Gaussian Distribution Settings

Yaw	Pitch	Roll
$\mu_\alpha = -0.54$, $\sigma_\alpha = 7.03$	$\mu_\beta = 0.18$, $\sigma_\beta = 2.55$	$\mu_\gamma = 2.16$, $\sigma_\gamma = 0.15$

To validate the efficiency of the proposed 360ProbDASH, we select three typical 360 video streaming methods as the comparisons:

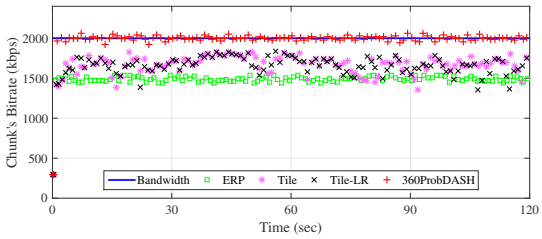
- ERP: Its format is same as ordinary video. This method is widely developed on Internet, such as YouTube [6].
- Tile: It only requests tiles corresponding to the user's current viewport, such as in [14]. But it doesn't apply any viewport prediction algorithm. This is the baseline method of tile-based streaming.
- Tile-LR: This tile-based method [19] uses Linear Regression to predict future viewport and requests corresponding tiles. The bitrate of each tile is allocated equally.

In performance comparison, we take the following measurement metrics into consideration:

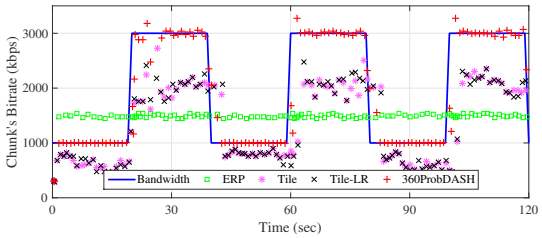
- Stall Ratio: This metric represents the playback continuity which calculates the percentage of the duration of stall over the total video streaming time.
- Viewport PSNR (V-PSNR) [27]: This metric directly indicates the quality of content in the user's viewport.
- Spatial Quality Variance: If the quality of content is not smooth, it will decrease the user's QoE [25]. We calculate this value according to the coefficient of variation (CV) of quality of content in the viewport.
- Viewport Deviation: Blank block may be rendered on the screen if the viewport adaptation algorithm picks wrong tiles. This metric calculates the percentage of the blank area over the viewport area.

5.2 Playback Continuity and Bitrate

In the simulation experiments, we actively manipulate the network conditions to evaluate the performance of playback continuity and downloading bitrate under fixed bandwidth and long-term bandwidth variations.



(a) Fixed Bandwidth: 2Mbps



(b) Varying Bandwidth: 1-3Mbps

Figure 7: Bitrate under Different Network Conditions

Fig.7(a) shows the bitrate of total requested segments per second under fixed bandwidth (2Mbps) scenario. For the three comparison methods of ERP, Tile and Tile-LR, the bitrate of tiles is adapted by estimated bandwidth. For Tile and Tile-LR schemes, the bitrate of tiles is allocated equally and quantized according to the discrete available bitrate. As a consequence, the total bitrate of segments is lower than the available bandwidth which results in low bandwidth utilization. On the contrast, 360ProbDASH allocates the total bitrate to tiles by their weight according to our QoE optimizer. Thus, it eliminates the effects of bitrate quantization.

Fig. 7(b) demonstrates the performance under log-term bandwidth variations (1Mbps→3Mbps per 20s). The chunk's bitrate of ERP is at a low level since bitrate quantization. Besides, we can observe that both Tile, Tile-LR and 360ProbDASH can react to bandwidth changes timely. But 360ProbDash achieves higher bitrate because of the proposed target-buffer-based rate control algorithm and probabilistic viewport adaptation.

Table 2 summarizes average bandwidth utilization (Bw. Utiliz.) and playback stall ratio. We can see that 360ProbDASH achieves the highest bandwidth utilization. Besides, it decides the total bitrate according to buffer occupancy and estimated bandwidth which can avoid playback stall effectively. On the contrast, ERP, Tile and Tile-LR adapts bitrate only based on historical bandwidth, they could overestimate the network capacity which leads to playback stall under a small buffer.

5.3 Viewport Quality and Spatial Variance

In the simulation experiments, we evaluate the metrics of V-PSNR and spatial quality variance under real head movement traces. The bandwidth through these tests is fixed at 2Mbps.

The average V-PSNR among the head movement traces is shown in Fig. 8. All tile-based schemes achieves higher V-PSNR than ERP. This echoes the conclusion that tile-based method is more efficient

Table 2: Bandwidth Utilization and stall

Alg.	Bandwidth=2Mbps		Bandwidth=1-3Mbps	
	Bw. Utiliz. (%)	Stall (%)	Bw. Utiliz. (%)	Stall (%)
ERP	74.45	0	64.89	10.93
Tile	80.44	0	64.27	1.33
Tile-LR	79.69	0	66.06	1.27
360ProbDASH	97.51	0	93.55	0

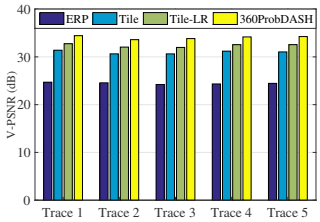


Figure 8: Average V-PSNR

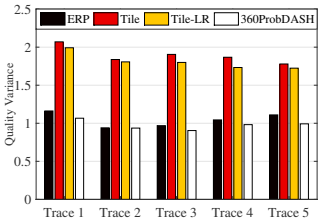


Figure 9: Average Spatial Quality Variance

in 360 video streaming with less bandwidth waste. Besides, the proposed 360ProbDASH achieves the highest viewport quality since we obtain the optimal bitrate of tiles considering its bitrate-quality relationship under probabilistic model.

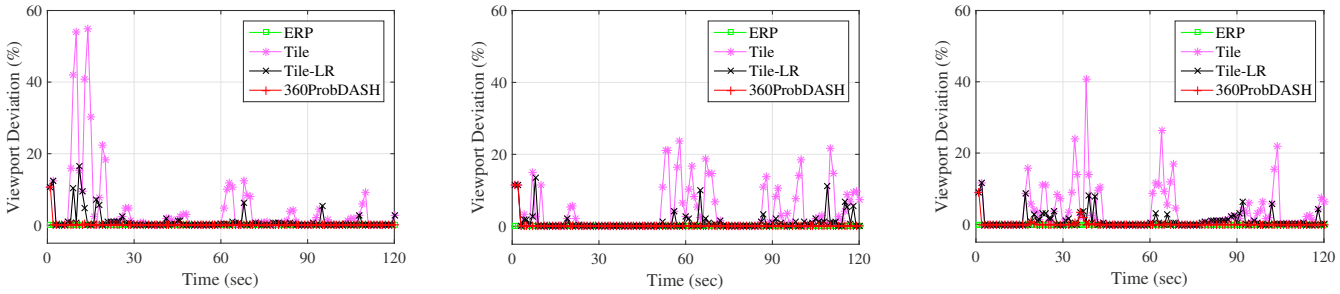
Fig. 9 compares the spatial quality variance of different methods. Our proposed approach has the most smooth quality variance in viewport. This is because, we design the optimization problem to minimize the spatial quality variance. The ERP also has a small variance in quality since it requests whole 360 content in the same one frame which smooth the quality to some extent.

Fig. 10 shows the viewport deviation ratio plotted against time. Due to the space limitation, we only show the result on three traces. The viewport deviation can represent the user' head prediction error to an extent. Obviously, ERP request the full view of 360 video which result in zero blank area ratio. We can see that the blank area can be more than 40% in non-prediction method Tile. It will definitely decrease user's QoE. Our proposed scheme outperforms the Tile and Tile-LR schemes, since we take the probability of prediction error into consideration which results in almost no blank area.

Table 3: Performance Comparison and Our Improvement

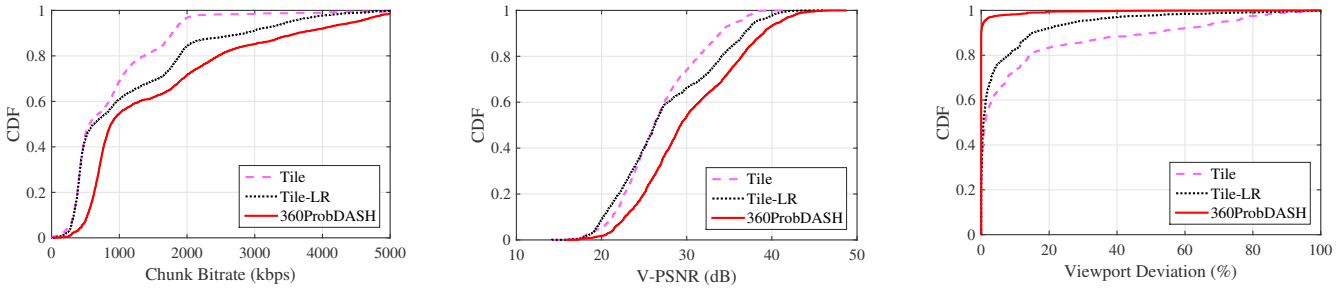
Metrics		ERP	Tile	Tile-LR	360ProbDASH
V-PSNR	Value (dB)	24.45	31.16	32.38	**34.06**
	Relative Imp.	+39.3%	+9.31%	+5.19%	-
Spacial Quality Variance	Value (CV)	1.05	1.89	1.81	**0.97**
	Relative Imp.	+8.25%	+48.67%	+46.41%	-
Viewport Deviation	Value (%)	**0**	3.62	0.96	0.13
	Absolute Imp.	-0.13%	+3.49%	+0.83%	-

Table 3 summarizes the performance results and improvement (Imp.) of 360ProbDASH over the three comparison methods. Our proposed 360ProbDASH achieves 39.3% V-PSNR improvement at most, and 46.41% quality variance reduction compared to Tile-LR which is the tile-based method with viewport prediction. This validates the proposed 360ProbDASH is able to achieve better V-PSNR with smooth spatial quality. Besides, the viewport deviation ratio of 360ProbDASH is the smallest among all tile-based schemes.



(a) Trace 1 (b) Trace 2 (c) Trace 3

Figure 10: Viewport Deviation under Different User's Head Movement Traces



(a) CDF of Total Request Segment Bitrate (b) CDF of V-PSNR of Request Segment (c) CDF of Viewport Deviation of Request Segment

Figure 11: Internet Experiment results

5.4 Real-world Experiments over Internet

To further evaluate the performance under severe network conditions, we conduct a series of experiments over real-world Internet. An Apache HTTP Server is setup in *Hong Kong*, and a client is located at a campus in *Beijing*. They are connected through Internet links. Due to the fact that Internet bandwidth is time-varying, we run the three tile-based methods sequentially (Tile, Tile-LR, 360ProbDASH) and repeat them for 10 rounds. In each two rounds, we import the same head movement trace into the client for different methods. Thus, 5 head movement traces are used throughout the entire experiments. The performances are illustrated in Fig. 11.

As shown in Fig. 11(a), our proposed 360ProbDASH can improve the video bitrate and bandwidth utilization. This is because we use target-buffer-based control to decide the total bitrate budget and allocate them to tiles by weight according to our QoE-driven optimizer. On the contrast, Tile and Tile-LR suffer from low bandwidth utilization because of bitrate quantization.

As illustrated in Fig. 11(b), during 80% time, the proposed 360ProbDASH achieves 3-5dB V-PSNR gain compared to other methods. This demonstrates that the probabilistic model viewport prediction has improved viewport quality significantly. The results that Tile-LR's PSNR is higher than that of Tile also validate the effect of viewport prediction.

Fig. 11(c) compares the accuracy of viewport prediction. Tile-LR is better than Tile since it predicts the user's orientation in the near future. The proposed 360ProbDASH takes the prediction error into consideration which has the lowest viewport deviation ratio among the three tile-based schemes. Therefore, our probabilistic model can avoid missing tiles to a great extent.

6 CONCLUSIONS

Tile-based HTTP adaptive streaming is a promising way to deliver 360 video through the Internet. However, it invokes problem of pre-fetching tiles error, playback stall and border of mixed-bitrate tiles. In this work, we propose a probabilistic tile-based adaptive streaming system, called 360ProbDASH. In rate adaptation, it applies a target-buffer-based control algorithm to ensure continuous playback within small buffer. In viewport adaptation, it constructs a probabilistic model to cope with the viewport prediction error. Then, we formulate a QoE-driven optimization problem: minimizing expected quality distortion of tiles and spatial variability of quality under the constraint of total transmission bitrate. We implement a 360ProbDASH prototype and carry out extensive experiments on a simulation test-bed and real-world Internet with various real user's head movement traces. Our approach outperforms other methods in V-PSNR, spatial quality smoothness and playback continuity. In our future work, we plan to apply saliency model which takes the content into account to assist viewport adaptation more precisely.

ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China under contract No. 61471009 and Culture Development Funding under Grant No.2016-288.

REFERENCES

[1] dash.js. https://github.com/Dash-Industry-Forum/dash.js/wiki.
[2] MP4Box. https://gpac.wp.mines-telecom.fr/mp4box/.
[3] Oculus Rift. https://www.oculus.com/rift/.
[4] Omnidirectional Media Application Format (OMAF). http://mpeg.chiariglione.org/standards/mpeg-a/omnidirectional-media-application-format.
[5] x264. http://www.videolan.org/developers/x264.html.
[6] YouTube Live in 360 Degrees Encoder Settings. https://support.google.com/youtube/answer/6396222.
[7] ISO/IEC 23009-1:2014/Amd 2:2015. Spatial Relationship Description, Generalized URL Parameters and Other Extensions.
[8] S. Akhshabi, A. C. Begen, and C. Dovrolis. 2011. An Experimental Evaluation of Rate-adaptation Algorithms in Adaptive Streaming over HTTP. In *Proceedings of the second annual ACM Conference on Multimedia Systems (MMSys)*. ACM, 157–168.
[9] Gene Cheung, Zhi Liu, Zhiyou Ma, and Jack ZG Tan. 2017. Multi-Stream Switching for Interactive Virtual Reality Video Streaming. In *arXiv preprint arXiv:1703.09090*.
[10] Lucia D'Acunto, Jorrit van den Berg, Emmanuel Thomas, and Omar Niamut. 2016. Using MPEG DASH SRD for Zoomable and Navigable Video. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys)*. ACM, 34.
[11] Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Carsten Griwodz, and Pål Halvorsen. 2016. Tiling in Interactive Panoramic Video: Approaches and Evaluation. *IEEE Transactions on Multimedia (TMM)* 18, 9 (2016), 1819–1831.
[12] Mohammad Hosseini. 2017. View-aware Tile-based Adaptations in 360 Virtual Reality Video Streaming. In *IEEE Virtual Reality (VR)*. IEEE, 423–424.
[13] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 107–110.
[14] Masayuki Inoue, Hideaki Kimata, Katsuhiko Fukazawa, and Norihiko Matsuura. 2010. Interactive Panoramic Video Streaming System over Restricted Bandwidth Network. In *Proceedings of the ACM International Conference on Multimedia (MM)*. ACM, 1191–1194.
[15] ISO/IEC JTC1/SC29/WG11/M. 2016. VR/360 Video Truncated Square Pyramid Geometry for OMAF.
[16] End-to-end Optimizations for Dynamic Streaming. https://code.facebook.com/posts/637561796428084.
[17] Mina Makar, Aditya Mavlankar, Piyush Agrawal, and Bernd Girod. 2010. Real-time Video Streaming with Interactive Region-of-interest. In *IEEE International Conference on Image Processing (ICIP)*. IEEE, 4437–4440.
[18] King-To Ng, Shing-Chow Chan, and Heung-Yeung Shum. 2005. Data Compression and Transmission Aspects of Panoramic Videos. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)* 15, 1 (2005), 82–95.
[19] 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*. ACM, 1–6.
[20] Ngo Quang Minh Khiem, Guntur Ravindra, Axel Carlier, and Wei Tsang Ooi. 2010. Supporting Zoomable Video Streams with Dynamic Region-of-interest Cropping. In *Proceedings of the first annual ACM Conference on Multimedia Systems (MMSys)*. ACM, 259–270.
[21] Robert Skupin, Yago Sanchez, Cornelius Hellge, and Thomas Schierl. 2016. Tile Based HEVC Video for Head Mounted Displays. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 399–400.
[22] J. P. Snyder. 1993. Flattening the Earth: Two Thousand Years of Map Projections. *University of Chicago Press* (1993).
[23] Kashyap Kammachi Sreedhar, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. Viewport-Adaptive Encoding and Streaming of 360-Degree Video for Virtual Reality Applications. In *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 583–586.
[24] ISO/IEC JTC1/SC29/WG11 W13533. 2012. MPEG DASH: The Standard for Multimedia Streaming over the Internet.
[25] Hui Wang, Vu-Thanh Nguyen, Wei Tsang Ooi, and Mun Choon Chan. 2014. Mixing Tile Resolutions in Tiled Video: A perceptual Quality Assessment. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop (NOSSDAV)*. ACM, 25.
[26] Richard Yao, Tom Heath, Aaron Davies, Tom Forsyth, Nate Mitchell, and Perry Hoberman. 2014. Oculus VR Best Practices Guide. *Oculus VR* (2014).
[27] Matt Yu, Haricharan Lakshman, and Bernd Girod. 2015. A Framework to Evaluate Omnidirectional Video Coding Schemes. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 31–36.
[28] Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *Proceedings of the ACM International Conference on Multimedia (MM)*. ACM, 601–605.