An Optimal Tile-Based Approach for Viewport-Adaptive 360-Degree Video Streaming

Article *in* IEEE Journal on Emerging and Selected Topics in Circuits and Systems · February 2019

DOI: 10.1109/JETCAS.2019.2899488

4 authors:

Nguyen Duc
Tohoku Institute of Technology
48 PUBLICATIONS **408** CITATIONS
SEE PROFILE

Tran Huyen
RIKEN
46 PUBLICATIONS **626** CITATIONS
SEE PROFILE

Anh T. Pham
The University of Aizu
250 PUBLICATIONS **3,329** CITATIONS
SEE PROFILE

Truong Cong Thang
The University of Aizu
192 PUBLICATIONS **2,400** CITATIONS
SEE PROFILE

An Optimal Tile-based Approach for Viewport-adaptive 360-degree Video Streaming

Duc V. Nguyen, *Student Member, IEEE,* Huyen T. T. Tran, *Student Member, IEEE,* Anh T. Pham, *Member, IEEE,* Truong Cong Thang, *Member, IEEE*

***Abstract*—In this paper, we propose a new adaptation approach for viewport-adaptive streaming of 360-degree videos over the Internet. The proposed approach is able to systematically decide versions of tiles according to user head movements and network conditions by taking into account not only viewport estimation errors, but also users' head movements in each segment duration. Experimental results show that the proposed approach can effectively adapt 360-degree videos to both varying network conditions and user head movements. Compared to existing approaches, the proposed approach can improve the average viewport quality by up to 3.8 dB and reduce the standard deviation of the viewport quality by up to 1.1 dB. Also, the impacts of the segment duration and the buffer size are investigated. It is found that long segment durations and large buffer sizes have significant impacts on the performances of tile selection methods.**

I. Introduction

360-degree video (360 video for short) is an integral part of Virtual Reality. It can provide immerse experience as a user can freely change his/her viewing direction while watching. Nevertheless, streaming of 360 video over the Internet is very challenging, especially because 360 video requires very high network bandwidth.

For 360 video streaming, one of the most popular approaches is tiling-based viewport-adaptive streaming [1]–[4]. In tiling-based viewport-adaptive streaming, a 360 video is spatially divided into small parts called *tiles*, each is encoded into multiple versions of different quality levels. Given the user's viewing direction and the network status, the most appropriate version of each tile is selected and delivered to a client running on the user's device. The client first decodes the tiles' versions, and then reconstructs the 360 video. Finally, the viewport corresponding to the user's current viewing direction is extracted and displayed [2]. Generally, for tile version selection, the *visible tiles* (i.e., tiles overlapping the viewport) are delivered at high quality, while the other tiles at low quality. Since only the visible tiles are seen by the user, a significant amount of bitrate can be reduced without causing negative impacts to the user.

In practice, viewport adaptation is sequentially applied to temporal segments called *adaptation intervals* or simply *segments* [5]. Also, the client should buffer some amount of video data before the video can start to play. The segmentation and client buffer cause a delay from when the tiles' versions of a segment are decided until when the playback of the frames in that segment starts. Thus, in order to decide the tiles' versions, future viewport positions need to be estimated. Since the user can freely change his/her viewing direction, errors in viewport estimation are likely to occur.

To deal with viewport estimation errors, the tiles surrounding the estimated viewport should also be delivered at high quality. Those tiles form the so-called *extension area* [6]. The remaining tiles form the so-called *background area* [6]. As the likelihood that the user will see the tiles in the background area is low, these tiles are usually delivered at the lowest quality, just in case there is sudden changes in head movements. For tile version selection, the two key questions are 1) *how big the extension area should be?* and 2) *what are the versions of the viewport and extension areas?*

There have been some previous studies proposing methods for tile version selection. A simple method, which selects the highest possible version for the visible tiles and the lowest version for other tiles, is used in previous work [1]–[4]. The tile versions may be different in resolution [1], [3], [4] or Quantization Parameters [2]. However, this method may suffer significant quality degradations due to viewport estimation errors, which is unavoidable due to the randomness of the user head movements [7], [8]. To deal with viewport estimation errors, some previous studies propose to further deliver the tiles surrounding the viewport area at high quality [8]–[10]. However, no specific algorithm for selecting tiles' versions is given in [8]. Though two tile selection algorithms are presented in [9], [10], they use constant values for some important parameters such as the extension width [9] and the portion of bandwidth allocated for the visible tiles [10]. Such constant values make it difficult to apply these algorithms in different scenarios.

In addition, most of previous studies employ HTTP Adaptive Streaming (HAS) for video data delivery [1], [2], [8], [10]. Yet, HAS results in a high delay due to its long segment duration and large client buffer [11]. So far, the impact of the delay on the performances of tile selection methods has never been investigated [1], [2], [8], [10]. However, a majority of 360 video streaming applications require very low latency. For example, telepresence applications require an end-to-end delay lower than 5ms [12]. In such a case, server driven adaptation is a better choice as the client-based approaches would result in unacceptable high latency.

Furthermore, most of the previous studies use bandwidth saving as the key performance metric [1], [2], [8]. However, the bandwidth saving cannot accurately reflect the quality

Duc V. Nguyen, Huyen T. T. Tran, Anh T. Pham, and Truong Cong Thang are with The University of Aizu, Aizuwakamatsu, Japan (e-mail:{d8182101, d8192106, pham, thang}@u-aizu.ac.jp).

perceived by the user. In [2], the average viewport PSNR is additionally used to demonstrate the effectiveness of a viewport-adaptive approach compared to conventional approaches. However, it is not possible to see how the video quality changes throughout a streaming session with the average viewport PSNR.

In this paper, we propose a new adaptation approach for tiling-based viewport-adaptive streaming that can systematically decide the version of each tile according to user head movements and network conditions. Our study has the following key features:

- The tile selection problem is formulated as an optimization problem with a new quality objective that is based on the visible portion of each tile.
- Viewport estimation errors and user head movements in each adaptation interval are jointly considered in the optimization problem.
- Two solution options for the problem are devised.
- The experiments are carried out with an actual test-bed.
- The behaviors of the proposed and reference methods are analyzed under different user's head movement patterns.
- The impact of the segment duration on the performances of tile version selection methods is investigated. It is found that the proposed method outperforms the reference methods when the delay increases.

Experimental results show that the proposed approach can effectively adapt to the user head movements and varying bandwidth conditions. The proposed approach can improve the average viewport quality by up to 3.8 dB and reduce the standard deviation of the viewport quality by up to 1.1dB compared to reference approaches. In addition, it is found that long segment durations and large buffer sizes cause severe impacts on the performances of tile selection methods.

A part of this work has been presented in [6]. Compared to [6], this paper has the following new points. First, a new option for tile version selection is proposed that can improve the viewport quality under highly predictable head movement traces. Second, the description and discussion are significantly extended. Third, the impact of the client buffer is investigated.

The remainder of this paper is structured as follows. The related work is presented in Section II. The proposed framework is given in Section III. The proposed approach is described in Section IV. The evaluations and discussions are drawn in Section V. Finally, conclusions and future work are provided in Section VI.

II. Related Work

In 360 video streaming, a 360 video is first projected onto a plane. In tile-based streaming, this projected video is then divided into tiles which are further encoded into multiple versions. In order to choose the optimal version of each tile, future viewport positions are estimated. Then, the selected tiles are transmitted to the client. Finally, some metrics are used to evaluate the performance of 360 video streaming. In this section, the related work to 360 video streaming is presented in detail.

Currently, typical planar formats for 360 videos include Cubemap (CMP), Equirectangular (ERP), Equal-Area (EAP), and Pyramid [13]. Those formats are different in size and the number/shape of faces [13]. Some previous studies have compared the performances of different projection formats. It is found that the EAP format yields around 8.3% bitrate saving relative to the ERP format [14], and the ERP and CMP formats outperform the Pyramid format [15]. Also, [16] suggests that the CMP format is better than the ERP and Pyramid formats. Nowadays, the ERP and CMP formats are the most widely used in practice [17].

In tiling-based approaches, a 360 video is spatially divided into tiles. The most popular tiling method is to partition every face of the video into tiles of equal size using a PxQ grid, such as 6x4 (ERP) [2], 8x8 (ERP) [18], 12x6 (ERP) [19], 2x2 (CMP) [20], [21], and 4x4 (CMP) [21]. In [22], the optimal tiling scheme is determined for each segment based on the user viewing behaviors. Also, some non-uniform tiling schemes have been used to exploit the content characteristics [23] and/or the user access preferences [1], [9], [10]. Each tile is then encoded into multiple versions of different bitrates [10], [18], QP values [2], [3], or resolutions [4].

So far, some encoding techniques for 360 video have been proposed such as region adaptive distortion calculation [24], adaptive QP selection [25], weighted-based rate control [26], spherical geometry padding [27], and fast intra estimation [28]. In addition, evaluation frameworks for 360 video coding have been designed in [14], [29].

In the literature, several viewing direction estimation methods have been proposed, such as average-based [8], linear regression-based [8], and neural network-based [7], [30]. It is found that future viewing direction can be predicted well (accuracy of more than 80%) within the next 1sec [7], [8], [30]. Yet, the estimation accuracy drops significantly for longer estimation window [8].

With respect to tile version selection, one of the simplest methods is to select the highest possible version for the visible tiles and the lowest version for the other tiles according to the estimated viewport [1]–[4]. Alternatively, utilizing Scalable Video Coding, the client fetches the base layers of all tiles, whereas only the visible tiles' enhancement layers are further fetched to enhance the viewport quality [20]. This method, however, suffers significant quality degradations due to estimation errors, which are unavoidable due to the randomness of head movements [8]. Moreover, a problem formulation has not been presented in [1]. In [7], the authors propose to trim each video frame according to the predicted viewport. The trimmed frame is then transmitted to the client. This method is actually not a tiling-based method.

To cope with viewport estimation errors, some previous studies propose to deliver the other tiles rather than the visible tiles at high quality [8]–[10]. However, no specific algorithm for selecting tiles' versions is given in [8]. Two tile selection algorithms are presented in [9] and [10]. Yet, these algorithms use the constant values of some important parameters such as the extension width [9] and the portion of bandwidth allocated for the visible tiles [10].

Another method is to use the tiles' viewing probabilities to assist tile selection [19]. Yet, [19] is only tested under idealistic scenarios where 1) the head movements are assumed

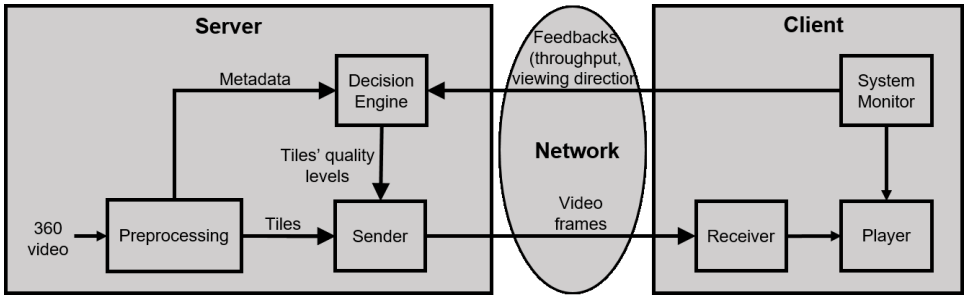


Fig. 1. System architecture.

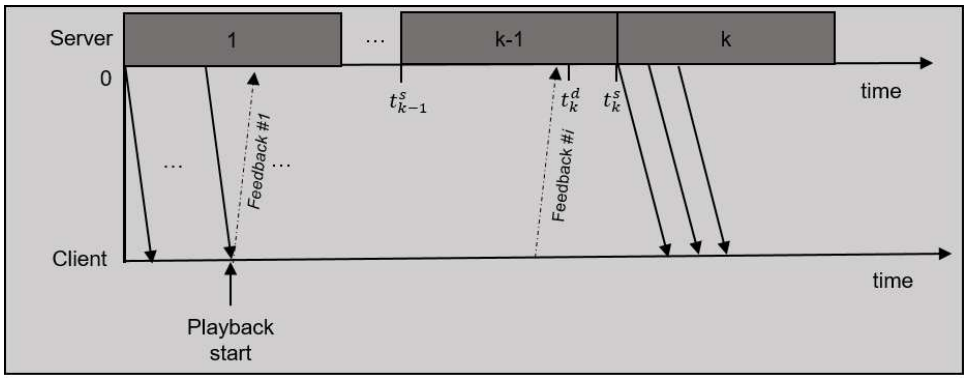


Fig. 2. System data flows.

to be constant at a given angular speed and 2) the network delay and head movement speed are known beforehand. In addition, some tiles may not be delivered to the client. Thus the user will experience blank block within the viewport if those tiles turn out to be visible tiles. Such cases can severely impact the user experience.

Although the work of [31] and our proposed method use the same idea to calculate the objective quality (or expected quality), the method for calculating the objective quality in our method is different from that of [31]. Specifically, in our proposed method, the viewport quality at each frame is first calculated given the estimated viewport position at that frame. Then, the objective quality is calculated a function of the frames' viewport quality. Meanwhile, in [31], the expected quality is calculated for the whole segment. In addition, the weights of the tiles are obtained from the viewing preferences of other viewers watching the same video. Though the results in [31] also show the viewport PSNR values over time, the streaming test-bed used to produce those results was not described. In addition, the performances of tile selection methods under different users head movement patterns have not been analyzed. In our paper, we present an analysis of the behaviors of the proposed and reference methods when 1) the viewport positions are static and 2) the viewport positions change frequently. It is shown that the considered methods behave differently under the two head movement patterns. Note that, such an analysis has not been performed in previous studies. The work in [31] has been extended to study the live 360 multicast in [32].

To deliver the tiles' versions over the networks in an efficient manner, important tiles can be rerouted to congestion-free network links using Software Defined Network [33] or first sent by the server using HTTP/2's priority feature [34]. Also, HTTP/2's server-push feature can be used to increase the achieved throughput in high RTT networks [9].

In most previous studies, bandwidth saving is used as the key performance metric [1]–[4], [30]. Yet, the bandwidth saving cannot accurately reflect the video quality perceived by users. In [14], a new metric called viewport PSNR (V-PSNR) has been proposed. Some recent studies have employed V-PSNR for evaluation such as [2], [10], [16], [19], [31], [32]. Yet, only the average V-PSNR values of the entire video [2], [19] or the segments [10] are reported.

III. Proposed Framework

A. System Architecture

Fig. 1 shows the general architecture of our proposed system where a server streams a 360 video to a client running on the user's device. The server includes a preprocessing module, a decision engine module, and a sender module. The preprocessing module first converts the input 360 video into a planar format using the ERP format [13]. Then, it spatially divides the converted video into multiple tiles of equal size. The preprocessing module then encodes each tile into multiple versions. The bitrate/quality information of the tiles' versions and other description information are stored in the metadata, which is used by the decision engine. Given the metadata and the client's feedbacks, the decision engine makes decisions on the tiles' versions. The sender delivers the selected tiles'

versions to the client. For low delay, the tiles of each frame are sent at the same time.

The client includes a receiver module, a player module, and a system monitor module. The receiver module receives and decodes the tiles' versions, then stitches the decoded tiles' versions to reconstruct the 360 video. The player extracts and displays the viewport corresponding to the user's current viewing direction. The system monitor is responsible for two tasks, which are 1) monitoring the network status (e.g., throughput) and the user viewing directions, and 2) periodically sending feedbacks to the server.

Fig. 2 shows the data flows in the proposed system. Similar to cloud gaming [35], the server sends video data to the client on frame basis. The adaptation is executed every adaptation interval (segment) that consists of L frames and contains τ seconds of the video. Segment k $(k \geq 1)$ is transmitted at time

$$t_k^s = (k-1) \times \tau. \tag{1}$$

To ensure that the decisions on tiles' versions of segment k are available before its transmission time, the decision engine decides the versions of the tiles of segment k at time

$$t_k^d = t_k^s - \delta t, \tag{2}$$

where δt is the maximum amount of time for making decisions, which depends on the server's processing speed as well as the complexity of the tile selection algorithm. At the client, as soon as the first B frames have completely received, the client starts playing the video. For every displayed video frame, the client (i.e., the system monitor) sends a feedback containing 1) the instant download throughput and 2) the user's current viewing direction (viewport position) to the server.

B. Problem Formulation

Suppose that, at a given time, the server needs to adapt a segment consisting of L frames to meet a bitrate constraint R^c. The segment is composed of *M* tiles, each is available in *N* versions where version 1 (*N*) has the lowest (highest) quality. Denote V_l the viewport position when the user watches the $l^{th}(1 \leq l \leq L)$ frame of the segment. Let v_m denote the version selected for tile m $(1 \leq m \leq M)$. The version v_m has a bitrate R_m and a distortion D_m. Note that, the bitrate and distortion are computed as the average values over all frames of the segment. The tile selection problem can be formulated as an optimization problem as follows.

Find $\{v_1, v_2, ..., v_M\}$ *to maximize a quality objective VQ which is a function of* $\{D_m\}_{1 \leq m \leq M}$ *and* $\{V_l\}_{1 \leq l \leq L}$

$$VQ = f\left(D_1, D_2, ..., D_M, V_1, V_2, ..., V_L\right) \tag{3}$$

and satisfy

$$\sum_{m=1}^{M} R_m \leq R^c. \tag{4}$$

To solve the above problem, our solution consists of the following aspects.

- Estimation of the bitrate constraint R^c (or throughput) and the viewport positions $V_l(1 \leq l \leq L)$. Note that in fact, any throughput and viewport estimation methods can be used in our framework.
- Computation of the quality objective *VQ*.
- Decision on the optimal version of each tile.

In the following section, we will address each of these aspects.

IV. Proposed Approach

In this section, the proposed approach to determine tiles' versions is described. Note that, the decision engine on the server is responsible for all important tasks, including throughput estimation, viewport estimation, and tile version selection. The solution below is for adapting a segment k at the server. We also assume that, at the decision making time, the server has just received the feedback of the l_{last}^{th} frame of the segment $k_{last} < k$.

A. Estimations of Throughput and Future Viewport Position

The estimated throughput $T^e(k)$ of the segment k is simply set to the throughput reported in the last client's feedback $T_{fb}(k_{last}, l_{last})$, i.e., $T^e(k) = T_{fb}(k_{last}, l_{last})$. The bitrate constraint R^c is then computed from $T^e(k)$ by

$$R^c = (1-\alpha) \times T^e(k), \tag{5}$$

where the safety margin α is in range $[0, 0.5]$ as shown in our previous work [36].

In this work, we use a linear regression method for viewport estimations. Different from [8], the estimation errors of past frames are taken into account. As mentioned above, the server needs to estimate the viewport position of each frame in segment k. Let $V^e(k,l)$ denote the estimated viewport position of the l^{th} $(1 \leq l \leq L)$ frame of the segment k. Let $E(k)$ denote the estimated viewport error of the segment k, which is set equal to the estimation error of the first frame of the segment k_{last}. Specifically, the value of $E(k)$ is given by

$$E(k) = V^e(k_{last}, 1) - V(k_{last}, 1). \tag{6}$$

Let $S_{avg}(k)$ denote the user's average head movement speed over the last L frames. The value of $S_{avg}(k)$ is given by

$$S_{avg}(k) = \frac{V(k_{last}, l_{last}) - V(k_{last}-1, l_{last})}{\tau}. \tag{7}$$

Note that, the values of $V(k_{last}, l_{last})$ and $V(k_{last}-1, l_{last})$ are obtained from the client's feedbacks. The estimated viewport position $V^e(k,l)$ of the l^{th} frame of the segment k is computed as follows.

$$V^e(k,l) = V(k_{last}, l_{last}) + (l-1) \times \frac{\tau}{L} \times S_{avg}(k) + \frac{l-1}{L-1} E(k). \tag{8}$$

B. Computation of Viewport Quality

In this part, we will compute the viewport quality value $VQ(k,l)$ of each frame of the segment k given the distortions $\{D_m(k)\}$ $(1 \leq m \leq M)$ and the estimated viewport positions $\{V^e(k,l)\}$ $(1 \leq l \leq L)$.

As the user only watches a portion of the full 360 video (i.e., the viewport) at each time instant, the contribution of

a tile to the viewport quality is dependent on how that tile overlaps the viewport. Thus, the viewport quality distortion value $VD(k,l)$ of the l^{th} frame $(1 \leq l \leq L)$ is computed as a weighted sum of the distortion values of all tiles as follows.

$$VD(k,l) = \sum_{m=1}^{M} w_m(V^e(k,l)) \times D_m(k). \quad (9)$$

Here, $w_m(V)$ is the weight of tile m given the viewport position V. $w_m(V)$ is calculated as the fraction of the overlapped area of tile m to the viewport area given the viewport position V. If tile m does not overlap the viewport, then $w_m(V) = 0$. In this paper, the distortion is measured in terms of the Mean Square Error (MSE), so the viewport quality value $VQ(k,l)$ of the l^{th} frame $(1 \leq l \leq L)$ is converted into PSNR as follows.

$$VQ(k,l) = 10 \times log_{10}\left(\frac{255^2}{VD(k,l)}\right). \quad (10)$$

The quality objective VQ in the above problem formulation can be computed from the obtained values of $VQ(k,l)$ $(1 \leq l \leq L)$.

C. Tile Selection Method

In this part, we will present our method to select the version of each tile. The basic idea is to extend the viewport to deal with viewport estimation errors. However, different from previous studies, the proposed method can systematically decide 1) the coverage of the extension area and 2) the versions of the viewport and extension areas. Note that, the tiles in the background area always have the lowest version.

If the user head movements are very difficult to estimate (or estimation error is high), the viewport should be extended in all directions so that any estimation errors could be tolerated. However, when the user head movements are highly predictable, extending the viewport in all directions will waste the available bandwidth since only a part of the extension area is actually useful. In this work, we consider two options for extending the viewport area, as illustrated in Fig. 3. In the following, we will present the method to decide the versions of tiles for each option.

1) Option 1: In this option, the viewport will be extended in all directions. The width of the *extension area* should be dependent on the viewport estimation error. The higher the estimation error is, the wider the extension area should be. An extension area with the width of I $(1 \leq I \leq I_{max})$ is formed by extending the viewport area by I tiles in all directions. Note that, the extension area does not include the tiles of the viewport area. The maximum width of the extension area I_{max} is dependent on the tiling scheme. The extension area of the width of I is divided into I ranges. Specifically, Range 1 is identical to the extension area with the width of 1 tile. Range i $(2 \leq i \leq I)$ contains tiles that are in the extension area with the width of i but not in the extension area with the width of $(i-1)$. All tiles belonging to the same range have the same version.

Given the expected viewport quality value of every frame, the overall quality value of the segment can be estimated. As the viewport quality will change over the segment due to user head movements, the quality objective ***VQ(k)*** is computed as the average of the viewport quality values of the first and last frames as follows.

$$VQ(k) = \frac{1}{2} \times (VQ(k,1) + VQ(k,L)). \quad (11)$$

Let v_0 be the version selected for the visible tiles, v_i the version selected for extension range $i(1 \leq i \leq I)$. Note that, v_i can take values from 1 (lowest quality) to N (highest quality). The general procedure to decide the tiles' versions can be summarized as follows.

- **Step 1**
  - Compute bitrate constraint R^c using Eq. (5).
  - Compute the estimated viewport positions $\{V_k^e(l)\}_{1 \leq l \leq L}$ using Eqs.(6)(7)(8).
- **Step 2**
  For each extension width $I(1 \leq I \leq I_{max})$
  For each set of $(v_0, v_1, ..., v_I)$ so that $v_0 \geq v_1 \geq ... \geq v_I$
  1) Compute the quality objective ***VQ(k)*** using Eqs. (9)(10)(11).
  2) Compute the total bitrate of all tiles including that in the background area.
- **Step 3**: Select the extension width I and the set of $(v_0, v_1, ..., v_I)$ that result in the highest quality objective ***VQ(k)*** and satisfy the condition in Eq. (4).

In the above algorithm, the viewport area is always ensured to have the highest quality. For the extension ranges, the version is gradually decreasing when moving away from the viewport center. When the width of the extension area is equal to I_{max}, there is no background area.

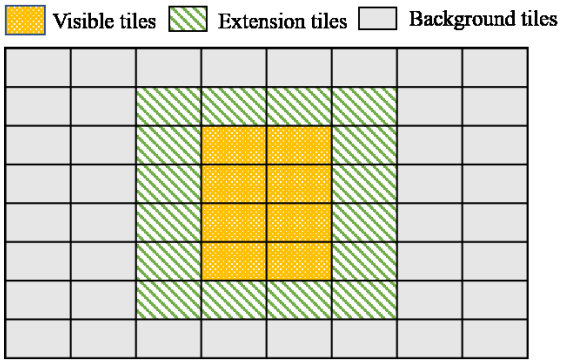
2) Option 2: Similar to the option 1, the extension area is first formed by extending the viewport in all directions. However, each range of the extension area is further divided into a number of sub-areas. Fig. 4 shows an example where every extension range is divided into 4 sub-areas. The tiles belonging to the same sub-area will have the same version. Let v_0 the version selected for the visible tiles, $v_{i,j}$ the version selected for the sub-area $j(1 \leq j \leq J_i)$ of range $i(1 \leq i \leq I)$ where J_i denotes the number of sub-areas of the extension range i.

If the number of sub-areas is high, searching over all possible selections of sub-areas' versions could be so time-consuming that is not feasible for real-time adaptation. Thus, we add two constraints regarding the version of each sub-area in order to reduce the processing time. First, the viewport area always has the highest possible version. Second, the tile version of any sub-area of an extension range i is higher than that of any sub-area of the range $(i+1)$. The two constraints are mathematically defined by equations (12) and (13).

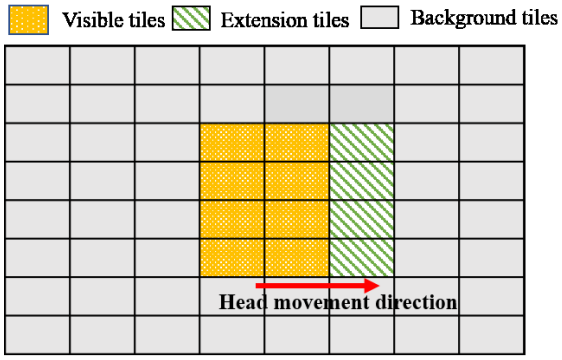
$$v_0 \geq v_{i,j}, \text{for all } i,j \neq 0 \quad (12)$$

$$v_{i,j} \geq v_{i+1,j'} \text{for all } i,j,j' \quad (13)$$

These two constraints ensure that the quality is gradually decreasing from the viewport center to the periphery.



(a) Option 1: Extending in all directions.



(b) Option 2: Extending according to the predicted viewport positions.

Fig. 3. Illustrations of two options for extending the viewport area.

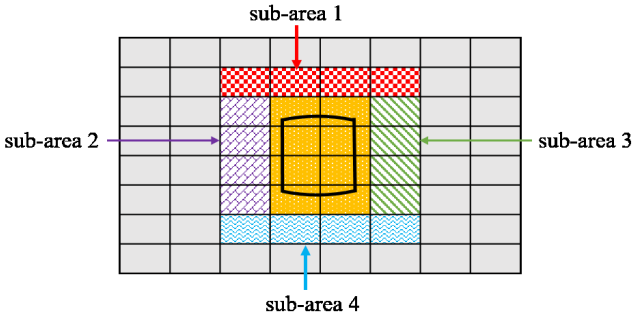


Fig. 4. An example of the sub-areas.

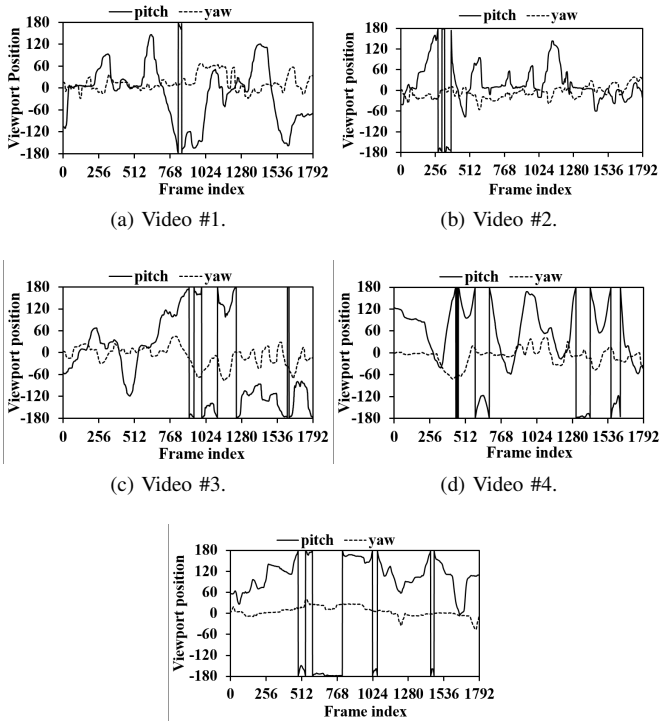
As for the objective quality, it is computed as the average of the estimated viewport quality values of all video frames of the segment k as follows.

$$VQ(k) = \frac{1}{L} \times \sum_{l=1}^{L} VQ(k,l). \tag{14}$$

The general procedure to decide the versions of tiles can be summarized as follows.

- **Step 1**
  - Compute bitrate constraint R^c using Eq. (5).
  - Compute the estimated viewport positions $\{V_l^e\}_{1 \le l \le L}$ using Eqs. (6)(7)(8).
- **Step 2** For each set of $\{v_{i,j}\}_{1 \le i \le I_{max}, 1 \le j \le J}$ satisfying the constraints in Eqs. (12)(13)(4)
  1) Compute the quality objective *VQ(k)* using Eqs. (9)(10)(14).
  2) Compute the total bitrate of all tiles including that in the background area.
- **Step 3**: Select the set of $(v_0, v_1, ..., v_{I_{max} * J})$ that results in the highest quality objective *VQ(k)*.

The above algorithm is implemented using nested *for* loops, each for a sub-area. Compared to the full-search approach, our algorithm can reduce the computation time by approximately 99%. In our experiments, the processing time of the above algorithm is always less than 1msec on an Ubuntu 14.04 LTS 64bit machine with 8GB RAM, Intel Core i5-2500 CPU 3.3GHz.



(a) Video #1. (b) Video #2.

(c) Video #3. (d) Video #4.

(e) Video #5.

Fig. 5. The head movement traces of the five videos.

V. Evaluation

A. Experiment Settings

In our experiment, we use five 360 videos from Youtube with different content types. Table I summarizes the content features of the four videos. The considered videos have 1792 frames, a resolution of 3840x1920 (4K), and a frame rate of 30fps. The Field of View (FoV) of the viewport is 90 degrees both horizontally and vertically. The video is divided into $M = 64$ tiles (i.e., 8x8 tiling) as in [18], each has a resolution of 480x240. The maximum width of the extension

TABLE I
DESCRIPTIONS OF FIVE 360 VIDEOS USED IN OUR EXPERIMENTS.

Video	Name	YouTubeID	Start offset	Content Description
#1	Yakitori	pQyt6H7GlcY	40s	Exploring night streets in Tokyo. Medium moving camera. No main focus
#2	Diving	2OzlksZBTiA	40s	Diving scene. Slowly moving camera, no clear horizon. No main focus expected within the sphere.
#3	RollerCoaster	8lsB-P8nGSM	65s	Rollercoaster. Fast moving camera fxed in front of a moving roller-coaster. Strong main focus following the rollercoaster trail.
#4	Timelapse	CIw8R8thnm8	0s	Timelapse of city streets. Fixed camera, clear horizon with a lot of fast moving people/cars, many scene cuts. Focus expected along the equator line.
#5	Venice	s-AJRFQuAtE	0s	Virtual aerial reconstruction of Venice. Slowly moving camera. No main focus expected within the sphere.

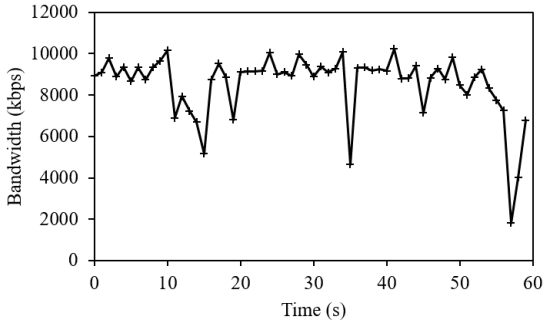


Fig. 6. The used bandwidth trace.

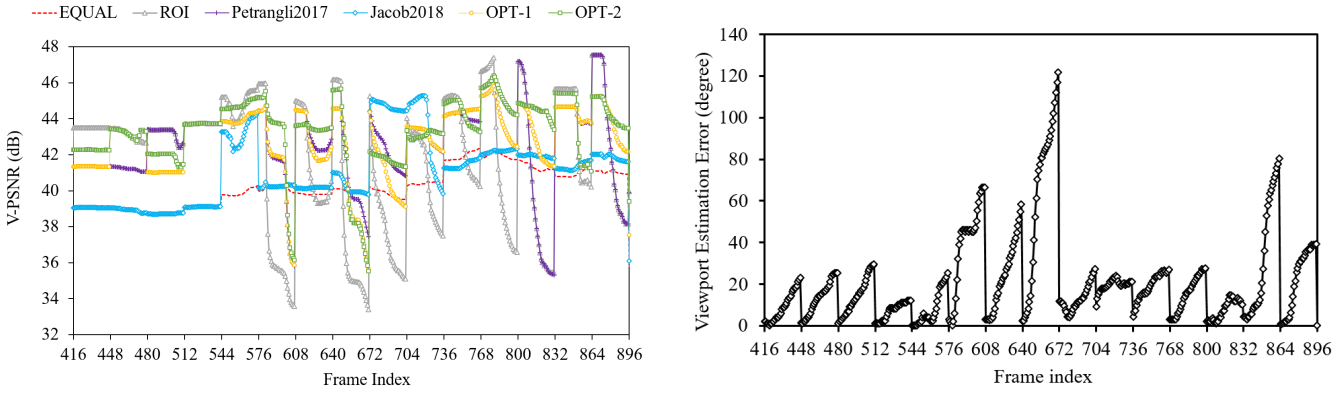
area I_{max} is 3. Each tile is encoded into $N = 7$ versions corresponding to 7 QP values of 24, 28, 32, 36, 40, 44, and 48 using HEVC format. To enable fast processing time, the low-delay B profile with the Group of Picture (GoP) size of 4 frames is used. According to our previous study [36], the value of the safety margin should be in [0, 0.5] range. In this paper, the safety margin α in Eq. (5) is set to 0.2 according to [36]. Each streaming session lasts for 59.7s. Each segment contains $L = 32$ frames. As aforementioned, the processing time of our proposed method is always less than 1ms. Thus, in order to use the latest feedback information from the client, the value of δt in Eq. (2) is set to 33ms, which is equal to the playback duration of one video frame at the frame rate of 30fps. In the option 2 of the proposed method, the value of J is set to 4 to cover four main changes in viewing direction of left, right, top, and bottom. In the future work, we will consider using different values of J. The buffer size B is set to 1. That means the client starts playing after it has fully received the first frame.

We implemented the proposed system based on Gaming Anywhere, an open-source cloud gaming platform [35]. The client is written in C++ and running on a Ubuntu 14.04 LTS 64bit machine with 4GB RAM, Intel Core i5-3210M CPU 2.5GHz. The server is running on another Ubuntu 14.04 LTS 64bit machine with 8GB RAM, Intel Core i5-2500 CPU 3.3GHz. The client is connected to the server via a router. The network conditions are emulated using DummyNet [37] in which the Round-Trip Time delay is set to 50ms. We use two head movement traces, one contains the periods of steady movements while the other has a lot of changes in moving directions. The used head movement traces are plotted in Fig. 5. One head movement trace is used for each video of the five videos. Here, the viewport position at a given time is determined by a (pitch, yaw) pair in degrees with $-180 \leq$ pitch ≤ 180 and $-90 \leq$ yaw ≤ 90 [13]. During a streaming session, the selected version of each tile and the corresponding viewport position are logged. After the streaming session finishes, the tiles' versions are combined to reconstruct the 360 video. Then the viewport is extracted, and the actual viewport PSNR (V-PSNR) of each frame is calculated.

The proposed approach is compared to four reference approaches. The first approach, denoted *ROI* (i.e., [1]–[3]), selects the lowest version for the background area and the highest possible version for the viewport area; no extension area is considered. The second approach, denoted *EQUAL*, selects the same version for all tiles. The third method, denoted *Petrangli2017*, classifies the tiles into three groups, namely *viewport*, *adjacent*, and *outside* [9]. Different from our proposed method, the viewport group in this method consists of the visible tiles corresponding to not only the estimated viewport postion but also the current viewport position. As the name suggests, the adjacent group consists of tiles that are adjacent to the viewport. It can be noted that the adjacent group is a special case of the extension area in our method when the extension width is one tile. This method always tries to maximize the version of the viewport group. The remaining bandwidth will be used to improve the version of the adjacent group, then the outside group. The fourth method, denoted *Jacob2018*, selects the versions of tiles so as to minimize a weighted sum of the distortions of tiles. The distortions of tiles are measured using the MSE metric. The weights of tiles are calculated from the viewing history [31]. In our implementation, five head traces from five different users watching the same video are used to calculate the weights of tiles. The two options of the proposed method are respectively denoted by *OPT-1* and *OPT-2*. For fairness, the reference approaches of *ROI*, *EQUAL*, and *Petrangli2017* are implemented using the same viewport estimation method as that used in the proposed method.

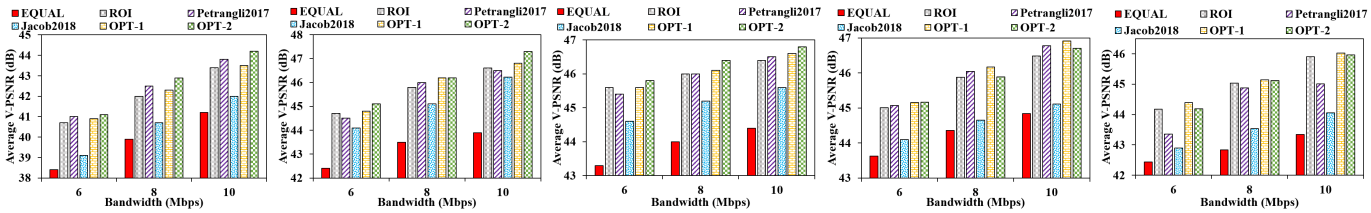
B. Constant Bandwidth Cases

In this part, we will investigate the performances of the proposed method and the four reference methods under



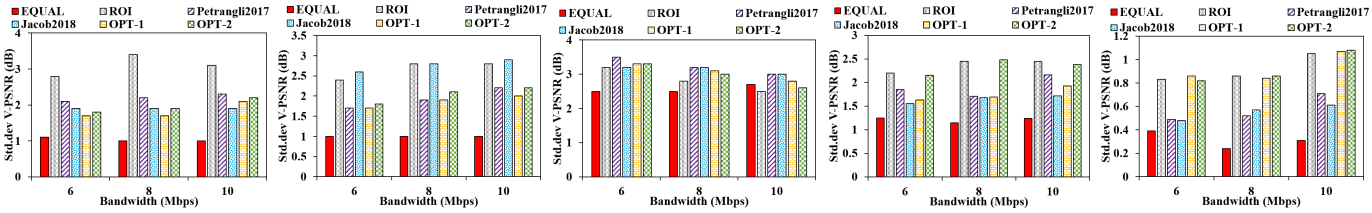
(a) Frames' V-PSNR values(dB). (b) Viewport estimation errors (degree).

Fig. 7. V-PSNR values and viewport positions of video frames #512-896 of all considered methods when the network bandwidth is 8Mbps under the head trace #1 (Video #1).



(a) Video #1 (b) Video #2. (c) Video #3. (d) Video #4. (e) Video #5.

Fig. 8. Average V-PSNR values of the proposed and four reference methods at three bandwidth values.



(a) Video #1 (b) Video #2. (c) Video #3. (d) Video #4. (e) Video #5.

Fig. 9. Std.dev V-PSNR values of the proposed and four reference methods at three bandwidth values.

constant bandwidth scenarios. Specifically, we consider three bandwidth values of 6Mbps, 8Mbps, and 10Mbps. Fig. 7 shows the V-PSNR values of the video frames #416-896 of the considered methods under the head trace #1 of video #1 when the network bandwidth is 8Mbps.

It can be seen that when the viewport positions are quite static (i.e., frames #416-576), the V-PSNR values of the *ROI*, *Petrangli2017*, *OPT-1*, and *OPT-2* methods are similar. This is because that the viewport positions can be well estimated during this interval (i.e., Fig. 7b). As a result, the proposed method (both two options) behaves like the *ROI* method by selecting the highest possible version for the visible tiles and the lowest version for the extension tiles. The current and estimated viewport positions will be very close to each other when the viewport positions are stable. Consequently, the viewport group of the *Petrangli2017* method and the viewport area of the proposed method will be almost identical. Meanwhile, the *EQUAL* method has significantly lower V-PSNR values since it selects the same version for all tiles. As a significant amount of bandwidth is consumed by the invisible tiles, the versions of the visible tiles are significantly reduced. It can be noted that the *Jacob2018* method results in the same V-PSNR curve as that of the *EQUAL* method for frames #416-544. This is because that none of the reference traces starts those segments at the same tile as that of the considered trace. Thus, it is not possible to determine the navigation likelihood of tiles using the reference traces. In this case, all tiles are assigned the same weight. As a result, the *Jacob2018* method selects the same version for all tiles. From frame #544 to frame #575, we can see that the frame V-PSNR of the *Jacob2018* method is only slightly lower than that of the *OPT-1* option of the proposed method. This is because that the user looks at a similar viewing direction to those of the reference traces. This result indicates that the performance of the *Jacob2018* method is strongly dependent on the reference traces. Especially, it seems that this method

TABLE II
AVERAGE PERFORMANCE IMPROVEMENTS OF THE PROPOSED METHOD COMPARED TO THE FOUR REFERENCE METHODS V-PSNR AT DIFFERENT BANDWIDTH VALUES IN TERMS OF AVERAGE V-PSNR (DB).

	BW=6Mbps				BW=8Mbps				BW=10Mbps			
Video	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**
#1	2.5	0.2	-0.1	1.9	2.4	0.3	-0.2	2.1	2.3	0.1	-0.3	2.3
#2	2.4	0.1	0.3	0.7	2.7	0.4	0.2	1.1	2.9	0.2	0.3	0.6
#3	2.3	0.0	0.2	1.0	2.1	0.1	0.1	0.9	2.2	0.2	0.1	1.0
#4	1.5	0.1	0.1	1.1	1.8	0.3	0.1	1.5	2.1	0.4	0.1	1.8
#5	2.0	0.2	1.0	1.5	2.3	0.1	0.3	1.6	2.7	0.1	1.0	2.0
Average	**2.1**	**0.1**	**0.3**	**1.2**	**2.3**	**0.2**	**0.1**	**1.4**	**2.4**	**0.2**	**0.2**	**1.5**
Max	**2.5**	**0.2**	**1.0**	**1.9**	**2.7**	**0.4**	**0.3**	**2.1**	**2.9**	**0.4**	**1.0**	**2.3**

(a) The *OPT-1* option.

	BW=6Mbps				BW=8Mbps				BW=10Mbps			
Video	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**
#1	2.7	0.4	0.1	2.1	3.0	0.9	0.4	2.7	3.0	0.8	0.4	3.0
#2	2.7	0.4	0.6	1.0	2.7	0.4	0.2	1.1	3.4	0.7	0.8	1.1
#3	2.5	0.2	0.4	1.2	2.4	0.4	0.4	1.2	2.4	0.4	0.3	1.2
#4	1.5	0.1	0.1	1.1	1.5	0.0	-0.1	1.2	1.9	0.2	-0.1	1.6
#5	1.8	0.0	0.8	1.3	2.3	0.1	0.2	1.6	2.6	0.1	1.0	1.9
Average	**2.2**	**0.2**	**0.4**	**1.3**	**2.4**	**0.4**	**0.2**	**1.6**	**2.7**	**0.4**	**0.5**	**1.8**
Max	**2.7**	**0.4**	**0.8**	**2.1**	**3.0**	**0.9**	**0.4**	**2.7**	**3.4**	**0.8**	**1.0**	**3.0**

(b) The *OPT-2* option.

TABLE III
AVERAGE PERFORMANCE IMPROVEMENTS OF THE PROPOSED METHOD COMPARED TO THE FOUR REFERENCE METHODS V-PSNR AT DIFFERENT BANDWIDTH VALUES IN TERMS OF STD.DEV V-PSNR (DB).

	BW=6Mbps				BW=8Mbps				BW=10Mbps			
Video	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**
#1	-0.6	1.1	0.4	0.8	-0.7	1.7	0.5	1.2	-1.1	1.0	0.2	1.2
#2	-0.7	0.7	0.0	0.9	-0.9	0.9	0.0	0.9	-1.0	0.8	0.2	0.9
#3	-0.8	-0.1	0.2	-0.1	-0.6	-0.3	0.1	0.1	-0.1	-0.3	0.2	0.2
#4	-0.4	0.6	0.2	-0.1	-0.5	0.8	0.0	0.0	-0.7	0.5	0.2	-0.2
#5	-0.5	0.0	-0.4	-0.4	-0.6	0.0	-0.3	-0.3	-0.8	0.0	-0.4	-0.5
Average	**-0.6**	**0.4**	**0.1**	**0.2**	**-0.7**	**0.6**	**0.1**	**0.4**	**-0.7**	**0.4**	**0.1**	**0.3**
Max	**-0.4**	**1.1**	**0.4**	**0.9**	**-0.5**	**1.7**	**0.5**	**1.2**	**-0.1**	**1.0**	**0.2**	**1.2**

(a) The *OPT-1* option.

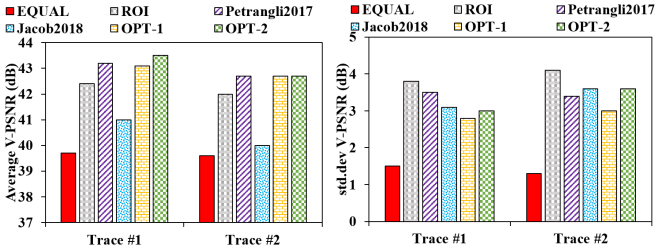
	BW=6Mbps				BW=8Mbps				BW=10Mbps			
Video	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**	**EQUAL**	**ROI**	**Petrangli2017**	**Jacob2018**
#1	-0.7	1.0	0.3	0.7	-0.8	0.6	-0.1	0.8	-0.8	-0.1	0.2	-0.1
#2	-0.8	0.6	-0.1	0.8	-0.8	-0.1	0.2	-0.1	-0.9	0.1	-0.3	-0.6
#3	-0.8	-0.1	0.2	-0.1	-0.9	0.1	-0.3	-0.6	-0.4	0.0	-0.3	-0.3
#4	-0.9	0.1	-0.3	-0.6	-0.4	0.0	-0.3	-0.3	0.0	0.0	0.0	0.0
#5	-0.4	0.0	-0.3	-0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Average	**-0.7**	**0.3**	**0.0**	**0.1**	**-0.6**	**0.1**	**-0.1**	**0.0**	**-0.4**	**0.0**	**-0.1**	**-0.2**
Max	**-0.4**	**1.0**	**0.3**	**0.8**	**0.0**	**0.6**	**0.2**	**0.8**	**0.0**	**0.1**	**0.2**	**0.0**

(b) The *OPT-2* option.

does not work well with a small number of reference traces.

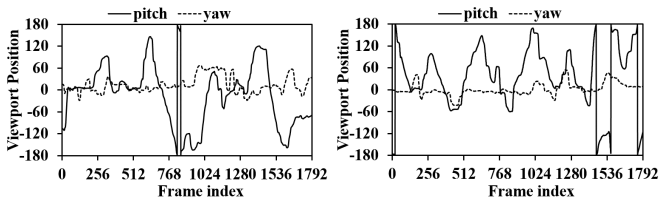
When the viewport positions start changing quickly (i.e., frames #576-896), the *ROI* method experiences drastic viewport quality reductions within each segment. The V-PSNR values are usually high at the beginning of each segment, but decrease significantly later. This can be explained as follows. As can be seen in Fig. 7b, the viewport estimation error are very small at the beginning of each segment. Yet, it increases significantly during each segment. The high viewport estimation errors cause some invisible tiles, which is associated with the estimated viewport position, become visible. Because the *ROI* method selects the lowest version for those tiles, the viewport quality is reduced significantly. We can see that the *Petrangli2017* method can achieve similar frame V-PSNR values to that of the *OPT-1* option of the proposed method for most of the frames. Yet, the frame V-PSNR of this method drops drastically during segment #25 (frames #800-831) and segment #27 (frames #864-895).

Meanwhile, the proposed method can provide much more stable V-PSNR values than those of the *ROI* method. This indicates that our proposed method is effective in dealing with the viewport estimation errors. It can also be noted that, the *OPT-2* option achieves higher and more stable V-PSNR values than those of the *OPT-1* option for most of the segments. This is because that the viewport position can be well predicted in this case (i.e., Fig. 7b). In addition, there are some segments in which the *OPT-1* option is comparable or better than the *OPT-2* option (e.g., segment #20 (frames #640-671) and segment #26 (frames #832-863)). This is due to the fact that the head movement direction suddenly changes at



(a) Average V-PSNR (dB). (b) Std.dev V-PSNR (dB).

Fig. 10. A comparison of the considered methods under a real bandwidth trace (Video #1).



(a) Trace #1. (b) Trace #2.

Fig. 11. The two head movement traces of Video #1.

those segments. As a result, the estimated viewport positions are very inaccurate as can be seen in Fig. 7(b). In such a case, the *OPT-2* option suffers from significant quality degradations since it chooses the extension tiles according to the estimated viewport. Meanwhile, the *OPT-1* option can tolerate the errors as the viewport is extended in all directions, thereby achieving higher V-PSNR.

Fig. 8 and Fig. 9 compare the proposed method with the reference methods, in terms of average and standard deviation (std.dev) V-PSNR, using five videos and at three different bandwidth values of 6Mbps, 8Mbps, and 10Mbps. It can be seen that the proposed method always achieve the highest V-PSNR values while providing stable viewport quality. This result indicates that the proposed method is effective in improving the viewport quality for various types of content in different network conditions.

Table II and Table III summarize the improvements of the proposed method over the four reference methods. The last two rows in Table II and Table III show the average and maximum gains over all videos at each bandwidth value. In particular, the *OPT-1* of the proposed method can improve the average V-PSNR by up to 2.9dB compared to the *EQUAL* method, up to 0.4dB compared to the *ROI* method, up to 1.0dB compared to the *Petrangli2017* method, and up to 2.3dB compared to the *Jacob2018* method. The *OPT-2* option can improve the average V-PSNR by up to 3.4dB compared to the *EQUAL* method, 0.9dB compared to the *ROI* method, 1dB compared to the *Petrangli2017* method, and 3.0dB compared to the *Jacob2018* method. As shown in Table III, the proposed method can reduce the std.dev V-PSNR compared to all reference methods except for the *EQUAL* method. Note that, despite of having the lower std.dev V-PSNR values than the proposed method, the *EQUAL* method results in very low average V-PSNR values. It can be noted that the *OPT-2* option has a slightly higher improvement than the *OPT-1* option in terms of average V-PSNR, but has lower improvement in terms of std.dev V-PSNR. The reason is that the *OPT-1* extends the viewport area in all directions, whereas the viewport area is extended in only one specific direction in the *OPT-2* option.

C. Variable bandwidth case

In this subsection, we will evaluate the performance of our proposed method under a real bandwidth trace. For that purpose, we use a bandwidth trace recorded using a mobile client under 4G network [38]. The bandwidth trace is shown in Fig. 6. Here, we only consider the Video #1. Beside the head movement trace shown in Fig. 5, an additional head movement trace of the Video #1 is used. The two head movement traces used in this experiment are shown in Fig. 11 The average V-PSNR and std.dev V-PSNR values of the proposed and four reference methods are shown in Fig. 10. Similar to the constant bandwidth cases, the *OPT-2* option achieves the highest average V-PSNR value under the head trace #1, whereas the *OPT-1* option performs the best under the head trace #2. The performance improvements of the proposed method are summarized in Table IV. Under the head trace #1, the *OPT-2* option can increase the average V-PSNR by up to 3.8dB. Also, it can reduce the std.dev V-PSNR by 0.8dB compared to the *ROI* method. The gains in terms of average V-PSNR compared to the *Petrangli2017* and *Jacob2018* methods are respectively 0.3dB and 2.5dB. Under the head trace #2, the *OPT-1* option achieves the highest average V-PSNR value, which is 3.1dB higher than that of the *EQUAL* method. Besides, the *OPT-1* option can reduce the std.dev V-PSNR by 1.1dB compared to the *ROI* method. These results indicate that the prosed method outperforms the reference methods under the variable bandwidth trace.

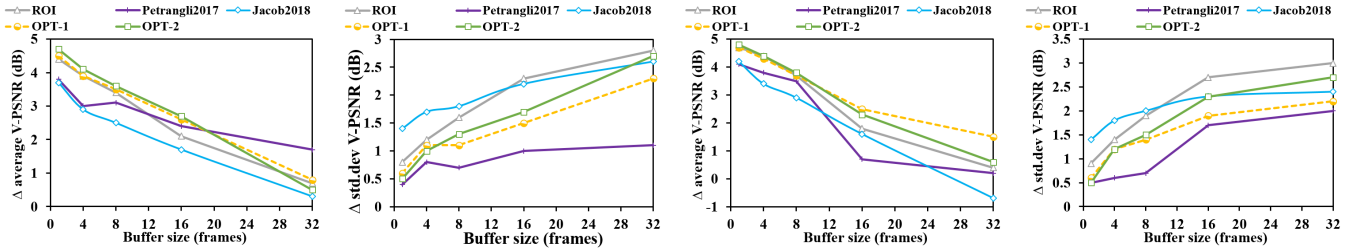
D. Impacts of Content Characteristics

In this subsection, we will investigate the impacts of content characteristics to the performance of the proposed method.

We can see that different content types have different head movement characteristics that in turn affect the performances of the proposed and reference methods. Specifically, the proposed method can clearly improve the viewport quality for Video #2 and Video #3. However, the improvement gains compared to the reference methods become smaller in case of Video #4 and Video #5. This can be explained as follows. As shown in Fig. 5, the user usually changes the viewing directions while watching Videos #2 and Video #3. Thus, it is necessary to deliver a number of extension tiles at high quality. As a result, the two options of the proposed method can improve the viewport quality compared to that of the *ROI* method that does not consider any extension tiles. Moreover, it can be seen that the *OPT-2* option achieves higher average V-PSNR than the *OPT-1* option for all the cases except when the bandwidth is 8Mbps with Video #2. It is interesting to see that the *Petrangli2017* method has

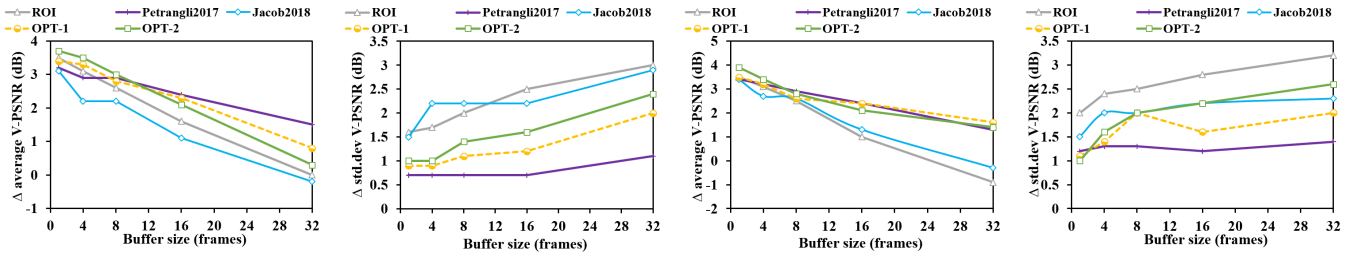
TABLE IV
PERFORMANCE IMPROVEMENTS OF THE PROPOSED METHOD UNDER A REAL BANDWIDTH TRACE (VIDEO #1).THE GAINS IN CASE OF THE TRACE #1 ARE OF THE *OPT-2* OPTION. THE GAINS IN CASE OF THE TRACE #2 ARE OF THE *OPT-1* OPTION.

Trace	Metrics		EQUAL	ROI	*Petrangli2017*	*Jacob2018*	OPT-1	OPT-2
Trace #1	Average V-PSNR	Value (dB)	39.7	42.4	43.2	41	43.1	43.5
		Gain (dB)	+3.8	+1.1	+0.3	+2.5	-	-
	Std.dev V-PSNR	Value (dB)	1.5	3.8	3.5	3.1	2.8	3
		Gain (dB)	-1.5	+0.8	+0.5	+0.1	-	-
Trace #2	Average V-PSNR	Value (dB)	39.6	42	42.7	40	42.7	42.7
		Gain (dB)	+3.1	+0.7	0.0	+2.7	-	-
	Std.dev V-PSNR	Value (dB)	1.3	4.1	3.4	3.6	3	3.6
		Gain (dB)	-1.7	+1.1	+0.4	+0.6	-	-



(a) Trace #1, Δ*V-PSNR*$_{avg}$. (b) Trace #1, Δ*V-PSNR*$_{std}$. (c) Trace #2, Δ*V-PSNR*$_{avg}$. (d) Trace #2, Δ*V-PSNR*$_{std}$.

Fig. 12. The impacts of the buffer size and the segment duration when $L = 4$ frames (Video #1).



(a) Trace #1, Δ*V-PSNR*$_{avg}$. (b) Trace #1, Δ*V-PSNR*$_{std}$. (c) Trace #2, Δ*V-PSNR*$_{avg}$. (d) Trace #2, Δ*V-PSNR*$_{std}$.

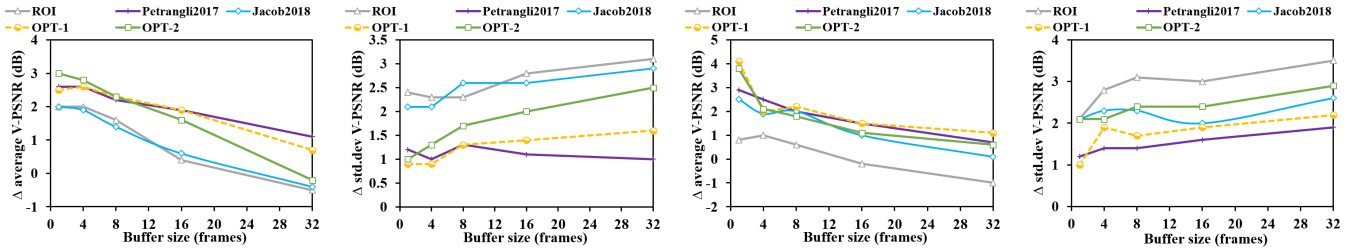
Fig. 13. The impacts of the buffer size and the segment duration when $L = 16$ frames (Video #1).

lower average V-PSNR values than that of the *ROI* method when the bandwidth is 6Mbps and 10Mbps with Video #2. In addition, the average V-PSNR of the *Jacob2018* method is the second lowest among the considered methods/options. This is because that the *Jacob2018* method calculates the navigation likelihood of the tiles using prior navigation traces. As a consequence, this method does not work well under the head movement traces that contain a significant number of new viewing directions.

On the other hand, the performances of the *ROI* method, the *OPT-1* and *OPT-2* options of the proposed method are comparable under Video #5 as shown in Fig. 8d. The reason is that the user viewing directions are mostly stable throughout the viewing session of this video as can be seen in Fig. 5d. Thus, the *ROI* method can achieve good performance since there is almost no need for the extension tiles. This also explains why the two options of the proposed methods have very similar performances. In contrast, it can be noted that the *Petrangli2017* method has quite lower average V-PSNR values than those of the *ROI* method for all three bandwidth values though the viewport area of this method is very similar to that of the *ROI* method. The lower performance of the *Petrangli2017* method is caused by the fact that this method requires all tiles in the viewport area must have the same version. This may cause a significant amount of bandwith be unnecessarily allocated to the adjacent area. We can see that the *OPT-1* option of the proposed method achieves higher average V-PSNR values than the *OPT-2* option for Video #4. This result is similar to that of the head trace #2 of Video #1. Again, that the reason is that the head movement trace of Video #4 contains many changes in viewing directions that results in high viewport estimation errors.

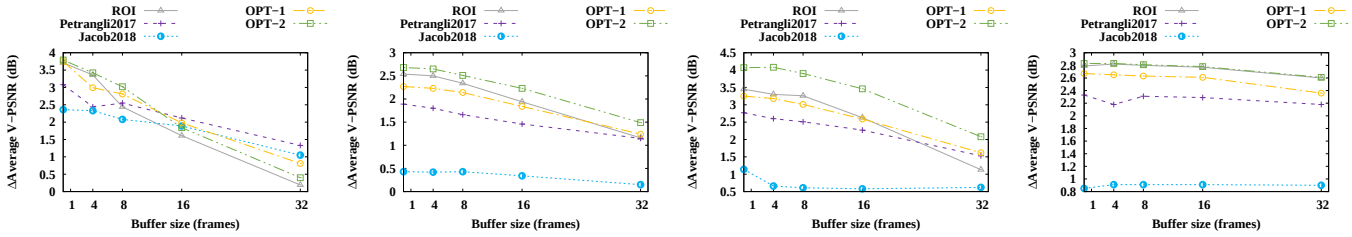
E. Impacts of Segment Duration and Buffer Size

In this part, we will investigate the performances of the proposed, *Petrangli2017*, *Jacob2018* and *ROI* methods under the different settings of segment durations and buffer sizes. For that purpose, we consider 3 segment durations of 4, 16, and 32 frames, and 5 buffer sizes of 1, 4, 8, 16, and 32 frames. Note that, the *EQUAL* method is not affected by viewport estimation errors. Thus, its performance will be independent of the segment duration and the buffer size. Again, the two head movement traces shown in Fig. 11 are used for the Video #1. To clearly show the results, we use



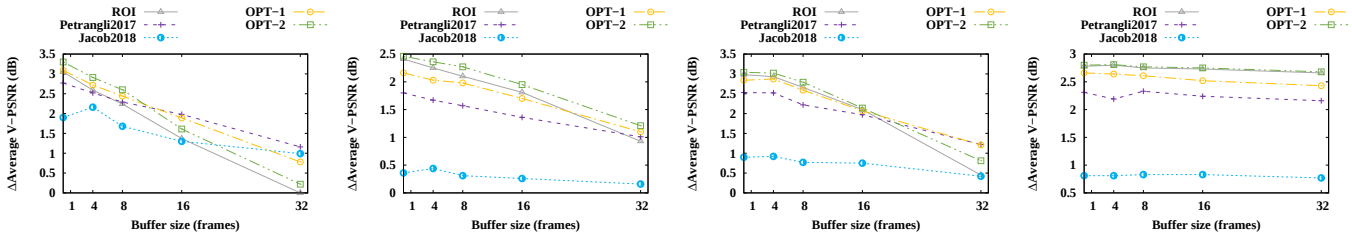
(a) Trace #1, Δ*V-PSNR*$_{avg}$. (b) Trace #1, Δ*V-PSNR*$_{std}$. (c) Trace #2, Δ*V-PSNR*$_{avg}$. (d) Trace #2, Δ*V-PSNR*$_{std}$.

Fig. 14. The impacts of the buffer size and the segment duration when $L = 32$ frames (Video #1).



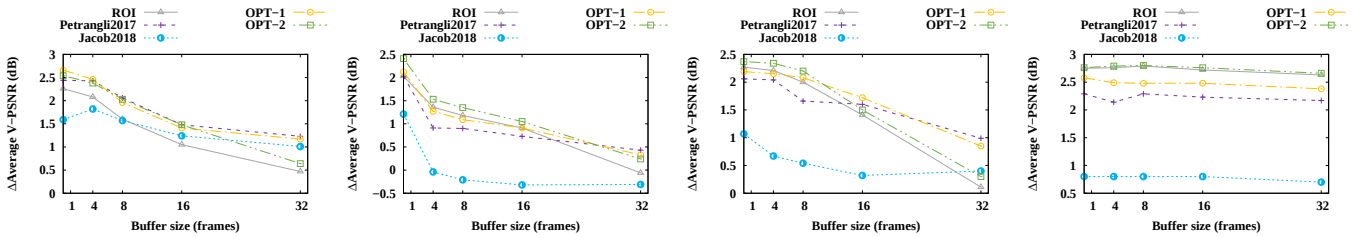
(a) Video #2. (b) Video #3. (c) Video #4. (d) Video #5.

Fig. 15. The impacts of the buffer size and the segment duration when $L = 4$ frames of Videos #2, #3, #4, and #5.



(a) Video #2. (b) Video #3. (c) Video #4. (d) Video #5.

Fig. 16. The impacts of the buffer size and the segment duration when $L = 16$ frames of Videos #2, #3, #4, and #5.



(a) Video #2. (b) Video #3. (c) Video #4. (d) Video #5.

Fig. 17. The impacts of the buffer size and the segment duration when $L = 32$ frames of Videos #2, #3, #4, and #5.

two performance metrics namely 1) delta average V-PSNR and 2) delta std.dev of V-PSNR. The delta average V-PSNR, denoted by Δ*V-PSNR*$_{avg}$, is the difference in the average V-PSNR between a given option/method and the *EQUAL* method. The delta std.dev V-PSNR, denoted by Δ*V-PSNR*$_{std}$, is the difference in the std.dev V-PSNR between a given option/method and the *EQUAL* method.

Fig. 12, Fig. 13, and Fig. 14 show the Δ*V-PSNR*$_{avg}$ and Δ*V-PSNR*$_{std}$ values at different settings of segment durations and buffer sizes of the proposed and *ROI* methods when the bandwidth is 8Mbps under the two head movement traces. It can be seen that, given a segment duration, the higher the buffer size is, the lower the Δ*V-PSNR*$_{avg}$ value would become. Also, the Δ*V-PSNR*$_{std}$ values tend to increase with the buffer size. This can be explained that, longer buffer size leads to higher delay, that in turn increases the viewport estimation errors. It can also be noted that, the *ROI* method experiences the strongest quality degradation as the buffer size increases. For $L = 16$ and $L = 32$ cases, the Δ*V-PSNR*$_{avg}$ values of the *ROI* method become lower than zero when $B = 32$ frames. Hence, the *ROI* method is likely worse than

the *EQUAL* method as it has much higher variations as shown in Fig. 13b, Fig. 13d, Fig. 14b, and Fig. 14d. Meanwhile, the two options of the proposed method can still achieve higher average *V-PSNR* values than those of the *EQUAL* method (i.e., Fig. 13a and Fig. 14a)

The results of videos #2, #3, #4, and #5 are shown in Fig. 15, Fig. 16, and Fig. 17. Similar to the Video #1, the trend is that the higher the buffer size is, the lower the Δ*V-PSNR*$_{avg}$ becomes. It can be seen that when the buffer size increases the Δ*V-PSNR*$_{avg}$ reduces quickly in case of Video #2. This is because that the user frequently changes the viewing direction while watching this video. This makes the estimation errors increase rapidly when the delay (buffer size) increases. On the other hand, the Δ*V-PSNR*$_{avg}$ decreases very slowly in case of Video #5. The reason is that the estimation errors are small across different buffer sizes, thanks to the stable viewport potions during the head movement trace. It can also be seen that the proposed method outperforms the reference methods when the delay increases for most of the cases.

In this study, we consider a server-based scenario in which the bitrate and quality information of all tile versions are available to the decision engine beforehand. In HAS paradigm, the decision engine resides at the client. At the beginning of a streaming session, the client retrieves a metadata containing tile description information from the server. Thus, the proposed algorithm can work seamlessly in HAS paradigm if the bitrate and quality information of all tiles versions is provided in the metadata. In [19], the authors propose to include bitrate and quality information of tiles versions in the metadata. Otherwise, the bitrate and quality information can be estimated at the client as proposed in [39]. In this case, the models for bitrate and quality estimation must be sent to the client in advance, e.g., in the metadata.

VI. Conclusion

In this paper, we have proposed a low-delay system for viewport-adaptive streaming of 360-degree videos. Our proposed approach decides the versions of tiles based on estimation errors and user head movements during each segment duration. Though experiments, it is found that the proposed approach can provide not only high but also stable quality. Specifically, the proposed approach can improve the average viewport quality by up to 3.8dB while reducing the standard deviation by up to 1.1dB compared to the two reference approaches when the segment duration is 32 frames. The impacts of the segment duration and buffer size on the performance of the proposed approach is also investigated. It is found that the performance of the proposed approach decreases as the segment duration and the buffer size increases. In future work, we will improve the proposed method to better cope with high viewport estimation errors.

References

[1] M. Hosseini and V. Swaminathan, "Adaptive 360 VR Video Streaming: Divide and Conquer !" in *Proc. 2016 IEEE International Symposium on Multimedia (ISM)*, San Jose, US, Dec 2016, pp. 107–110.

[2] M. Graf, C. Timmerer, and C. Mueller, "Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP," in *Proc. 8th ACM on Multimedia Systems Conference (MMSys'17)*, Taipei, Taiwan, June 2017, pp. 261–271.

[3] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications," in *Proc. 2016 IEEE Picture Coding Symposium (PCS 2016)*, Nuremberg, Germany, Dec. 2016, pp. 601–605.

[4] R. Skupin, Y. Sanchez, D. Podborski, C. Hellge, and T. Schierl, "Hevc tile based streaming to head mounted displays," in *Proc. 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, CA, US, Jan 2017, pp. 613–615.

[5] T. C. Thang, Q. D. Ho, J. W. Kang, and A. T. Pham, "Adaptive streaming of audiovisual content using MPEG DASH," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, 2012.

[6] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "A new adaptation approach for viewport-adaptive 360-degree video streaming," in *Proc. 2017 IEEE International Symposium on Multimedia (ISM)*, Taichung, Taiwan, Dec 2017, pp. 38–44.

[7] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *Proc. 2016 IEEE International Conference on Big Data (Big Data)*, Washington, DC, USA, Dec 2016, pp. 1161–1170.

[8] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. 5th Workshop on All Things Cellular: Operations, Applications and Challenges (ATC'16)*, NY, USA, October 2016, pp. 1–6.

[9] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An http/2-based adaptive streaming framework for 360 virtual reality videos," in *Proc. 2017 ACM Multimedia Conference*, Mountain View, California, USA, Oct. 2017, pp. 1–9.

[10] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360 video streaming using tiles for virtual reality," in *Proc. 2017 IEEE International Conference on Image Processing (ICIP 2017)*, Beijing, China, Sept. 2017, pp. 2174–2178.

[11] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *Proc. 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2011)*, Lucca, Italy, 2011, pp. 1–8.

[12] K. S. Kim, D. K. Kim, C. Chae, S. Choi, Y. Ko, J. Kim, Y. Lim, M. Yang, S. Kim, B. Lim, K. Lee, and K. L. Ryu, "Ultrareliable and low-latency communication techniques for tactile internet services," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 376–393, Feb. 2019.

[13] Y. Ye, E. Alshima, J. Boyce, "JVET-E1003: Algorithm descriptions of projection format conversion and video quality metrics in 360Lib," in *Jt. Video Explor. Team ITU-T SG 16 WP3 ISO/IEC JTC 1/SC 29/WG 11 5th Meet.* Geneva, Switzerland: Joint Video Exploration Team, 2017.

[14] M. Yu, H. Lakshman, and B. Girod, "A Framework to Evaluate Omnidirectional Video Coding Schemes," in *Proc. 2015 IEEE International Symposium on Mixed and Augmented Reality*, Fukuoka, Japan, 2015, pp. 31–36.

[15] K. Kammachi-Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *Proc. 2016 IEEE International Symposium on Multimedia*, San Jose, US, 2016, pp. 583–586.

[16] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. 2017 IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–7.

[17] YouTube, "Improving vr videos," https://youtube-eng.googleblog.com/2017/03/improving-vr-videos.html, accessed: 2018-05-23.

[18] P. R. Alface, J.-F. Mac, and V. Nico, "Interactive Omnidirectional Video Delivery: A Bandwidth-Effective Approach," *Bell Labs Technical Journal*, vol. 16, no. 4, pp. 135–148, 2012.

[19] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming," in *Proc. 2017 ACM Multimedia Conference*, Mountain View, California, USA, Oct. 2017, pp. 315–323.

[20] A. T. Nasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using scalable video coding," in *Proc. 2017 ACM Multimedia Conference*, Mountain View, California, USA, Oct. 2017, pp. 1689–1697.

[21] Y. Sanchez, R. Skupin, C. Hellge, and T. Schierl, "Random access point period optimization for viewport adaptive tile based streaming

of 360 video," in *Proc. 2017 IEEE International Conference on Image Processing (ICIP)*, Beijing, China, Sept 2017, pp. 1915–1919.
[22] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "OpTile: Toward Optimal Tiling in 360-degree Video Streaming," in *Proc. 2017 ACM Multimedia Conference*. Mountain View, California, USA, Oct. 2017, pp. 708–716.
[23] Z. Tu, T. Zong, X. Xi, L. Ai, Y. Jin, X. Zeng, and Y. Fan, "Content adaptive tiling method based on user access preference for streaming panoramic video," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, Jan 2018, pp. 1–4.
[24] Y. Li, J. Xu, and Z. Chen, "Spherical domain rate-distortion optimization for 360-degree video coding," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, Hong Kong, China, July 2017, pp. 709–714.
[25] Y. Sun and L. Yu, "Coding optimization based on weighted-to-spherically-uniform quality metric for 360 video," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, St. Petersburg, FL, USA, Dec 2017, pp. 1–4.
[26] B. Li, L. Song, R. Xie, and W. Zhang, "Weight-based bit allocation scheme for VR videos in HEVC," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, St. Petersburg, FL, USA, Dec 2017, pp. 1–4.
[27] Y. He, Y. Ye, P. Hanhart, and X. Xiu, "Geometry padding for motion compensated prediction in 360 video coding," in *2017 Data Compression Conference (DCC)*, Snowbird, UT, USA, April 2017, pp. 443–443.
[28] Y. Wang, Y. Li, D. Yang, and Z. Chen, "A fast intra prediction algorithm for 360-degree equirectangular panoramic video," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, St. Petersburg, FL, USA, Dec 2017, pp. 1–4.
[29] X. Xiu, Y. He, Y. Ye, and B. Vishwanath, "An evaluation framework for 360-degree video compression," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, St. Petersburg, FL, USA, Dec 2017, pp. 1–4.
[30] C.-L. Fan, J. Lee, and K.-T. Chen, "Fixation Prediction for 360 Video Streaming in Head-Mounted Virtual Reality," in *Proc. 27th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, Taipei, Taiwan, June 2017, pp. 67–72.
[31] J. Chakareski, R. Aksu, X. Corbillon, G. Simon, and V. Swaminathan, "Viewport-driven rate-distortion optimized 360 video streaming," in *2018 IEEE International Conference on Communications (ICC)*, MO, USA, May 2018, pp. 1–7.
[32] R. Aksu, J. Chakareski, and V. Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360 video network multicast," in *Proc. IEEE ICME Int'l Workshop on Hot Topics in 3D (Hot3D)*, CA, USA, Jul. 2018, pp. 1–7.
[33] S. Zhao and D. Medhi, "SDN-Assisted Adaptive Streaming Framework for Tile-Based Immersive Content Using MPEG-DASH," in *Proc. 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Berlin, Germany, Nov. 2017, pp. 1–6.
[34] M. Nguyen, D. H. Nguyen, C. T. Pham, N. P. Ngoc, D. V. Nguyen, and T. C. Thang, "An adaptive streaming method of 360 videos over http/2 protocol," in *2017 4th NAFOSTED Conference on Information and Computer Science*, Hanoi, Vietnam, Nov 2017, pp. 302–307.
[35] C.-y. Huang, C.-h. Hsu, Y.-C. Chang, and K.-T. Chen, "GamingAnywhere: An Open Cloud Gaming System," in *Proceedings of the 4th ACM on Multimedia Systems Conference (MMSys'13)*, Oslo, Norway, 2013, pp. 36–47.
[36] D. V. Nguyen, H. T. T. Tran, P. N. Nam, and T. C. Thang, "A QoS-adaptive framework for screen sharing over Internet," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, Vienna, Austria, July 2016, pp. 972–974.
[37] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," in *SIGCOMM Computer Communications Review*, vol. 27, no. 1, 1997, pp. 31–41.
[38] A. Bokani, M. Hassan, S. S. Kanhere, J. Yao, and G. Zhong, "Comprehensive mobile bandwidth traces from vehicular networks," in *Proc. 7th International Conference on Multimedia Systems (MMSys'16)*, Klagenfurt, Austria, May 2016, pp. 1–6.
[39] T. C. Thang, H. T. Le, H. X. Nguyen, A. T. Pham, J. W. Kang, and Y. M. Ro, "Adaptive video streaming over HTTP with dynamic resource estimation," *Journal of Communications and Networks*, vol. 15, pp. 635–644, 2013.

Duc V. Nguyen received the B.E. and M.E. degrees from University of Aizu in 2014 and 2016. He is currently a PhD student and research assistant in the Computer Communications Lab., the University of Aizu. His research interests include video streaming, Virtual Reality, and networking. He is a recipient of Japanese government scholarship (MonbuKagaku-sho) for graduate study since 2014.

Huyen T. T. Tran received the B.E. degree from Hanoi University of Science and Technology in 2014. She is currently a graduate student and research assistant in the Computer Communications Lab., the University of Aizu. Her research interests include Quality of Experience (QoE), multimedia networking, and content adaptation. She is a recipient of Japanese government scholarship (MonbuKagaku-sho) for graduate study since 2015.

Anh T. Pham received the B.E. and M.E. degrees, both in Electronics Engineering from the Hanoi University of Technology, Vietnam in 1997 and 2000, respectively, and the Ph.D. degree in Information and Mathematical Sciences from Saitama University, Japan in 2005. From 1998 to 2002, he was with the NTT Corp. in Vietnam. Since April 2005, he has been on the faculty at the University of Aizu, where he is currently Professor and Head of Computer Communications Laboratory with the Division of Computer Engineering. Professor Pham's research interests are in the broad areas of communication theory and networking with a particular emphasis on modeling, design and performance evaluation of wired/wireless communication systems and networks. He has authored/co-authored more than 140 peer-reviewed papers, including 40+ journal articles, on these topics. Professor Pham is senior member of IEEE. He is also member of IEICE and OSA.

Truong Cong Thang received the B.E. degree from Hanoi University of Science and Technology, Vietnam, in 1997 and the Ph.D. degree from KAIST, Korea, in 2006. From 1997 to 2000, he worked as a network engineer in Vietnam Post & Telecom (VNPT). From 2007 to 2011, he was a Member of Research Staff at Electronics and Telecommunications Research Institute (ETRI), Korea. He was also an active member of Korean and Japanese delegations to standard meetings of ISO/IEC and ITU-T from 2002 to 2014. Since 2011, he has been an Associate Professor of University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards.