AdaP-360: User-Adaptive Area-of-Focus Projections for Bandwidth-Efficient 360-Degree Video Streaming

Chao Zhou
SUNY Binghamton
czhou5@binghamton.edu

Shuoqian Wang
SUNY Binghamton
swang130@binghamton.edu

Mengbai Xiao
The Ohio State University
xiao.736@osu.edu

Sheng Wei
Rutgers University
sheng.wei@rutgers.edu

Yao Liu
SUNY Binghamton
yaoliu@binghamton.edu

ABSTRACT

360-degree video is an emerging medium that presents an immersive view of the environment to the user. Despite its potential to provide an immersive watching experience, 360-degree video has not achieved widespread popularity. A significant cause of this slow adoption is the high-bandwidth requirements of the format. The primary source of bandwidth inefficiency in 360-degree video streaming, un-addressed in popular transmission methods, is the discrepancy between the pixels sent over the network (typically the full omnidirectional view) and the pixels displayed in the head-mounted display's field of view. At worst, roughly 88% of transmitted pixels remain unviewed.

In this work, we motivate a user-adaptive approach to address inefficiencies in 360-degree streaming through an analysis of user-viewing traces. We design a greedy algorithm to generate projections of the spherical surface that allow the user-view trajectories to be efficiently transmitted. We further demonstrate that our approach can be applied to many popular 360-degree projection layouts. In BD-rate experiments, we show that the adaptive versions of the rotated spherical projection (RSP) and equi-angular cubemap (EAC) can save 26.2% and 24.0% bitrates on average, respectively, while achieving the same visual quality of rendered views compared to their non-adaptive counterparts in a realistic scenario. These adaptive projections can also achieve 53.1% bandwidth savings over the equirectangular projection.

CCS CONCEPTS

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

KEYWORDS

360-degree video streaming; area-of-focus projections; user-adaptive; BD-rate

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

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413521

ACM Reference Format:
Chao Zhou, Shuoqian Wang, Mengbai Xiao, Sheng Wei, and Yao Liu. 2020. AdaP-360: User-Adaptive Area-of-Focus Projections for Bandwidth-Efficient 360-Degree Video Streaming. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413521

1 INTRODUCTION

360-degree video provides users with an omnidirectional view of the environment. This omnidirectional view immerses the user in the recorded scene when viewed through a head-mounted display (HMD). This more-natural exploration of the surrounding scene can be useful to users in settings that benefit from greater immersion.

Widespread adoption of 360-degree streaming applications, however, is inhibited by the high bandwidths required for the high-quality 360-degree video delivery. High bandwidths are required not only due to the high definition video desired by end users, but also due to the large percentage of wasted bandwidth – unique to 360-degree videos. While 360-degree videos encode full omnidirectional views surrounding a camera, views rendered on HMDs have only limited fields-of-view (FoVs). For example, to render a view with 90° by 90° FoV, as little as 11.7% pixels of a full 360-degree video frame are needed. The remaining 88% transmitted data is not viewed and thus wasted.

One method of addressing this bandwidth inefficiency involves improving the efficiency when projecting pixels from the sphere to a 2D image, e.g., the equi-angular cubemap (EAC) [2]. However, efficient projections themselves do not fully solve the problem: the full omnidirectional views are still transmitted to the viewer, resulting in a high percentage of unviewed pixels.

Another family of methods are offset projection approaches (e.g., the offset cubemap [3]). These methods encode a pre-determined set of versions for a single temporal video segment, with each version representing a single direction of the omnidirectional view in higher quality. As a result, the percentage of unviewed pixels is reduced when the user's viewing direction is perfectly aligned with the high-quality direction. However, these approaches suffer from two main drawbacks. *First*, each version has a single high-quality direction while a user's view may change and form a trajectory even over a short time period (e.g., a few seconds). As a result, when the user's view deviates from the high-quality direction, the percentage of unviewed pixels increases, and the quality of rendered views quickly drops. *Second*, to account for various viewing directions, a large number of versions of a single temporal video segment need

to be encoded (e.g., 22 versions [28]), which significantly increases the storage overhead.

Finally, tile-based approaches partition the video segments spatially into multiple tiles and only download the tiles predicted to be viewed [12, 13, 17–19, 24, 27]. These approaches are effective at reducing the unviewed fraction of downloaded pixels. However, they create technical challenges at the client side. Here, streaming clients must navigate a large decision space to select a subset of tiles or tile bitrates. Further, these clients must stage individual tile downloads so that all of these downloads meet the segment playback deadline. Moreover, downloaded tiles must be decoded in time for playback and view rendering, and improper tile decoding scheduling can lead to playback stall [20].

In this paper, we propose a general method that adapts existing spherical projections to user viewing behaviors. It aligns high-quality visual content with user's viewing trajectories when watching 360-degree videos. In addition, this adaptation method avoids tiling videos, enabling practical 360-degree video streaming systems. Overall, this work makes the following contributions:

- We analyzed 360-degree head movement datasets and found that it is possible to create area-of-focus projections whose high-quality regions align with user-view trajectories.
- We propose AdaP-360 – a user-adaptive area-of-focus scheme that can be combined with many existing spherical projections for encoding 360-degree frames. In this scheme, each frame is constructed to have one high- and one low-quality region.
- Our proposed AdaP-360 scheme selects and orients the high-quality regions to efficiently cover past user-view trajectories of each temporal video segment under a storage constraint. These high-quality areas capture typical user "view-trajectories" across a temporal segment, increasing view efficiency.
- We apply AdaP-360 to three popular spherical projections: i) the equi-angular cubemap projection (EAC), ii) the rotated spherical projection (RSP), and iii) the barrel projection (BRL).

To evaluate our approach, we conducted extensive experiments using actual user traces. BD-rate results from these experiments show that AdaP-360 can achieve up to 55.7% bitrate savings on average compared to the equirectangular projection while achieving the same visual quality of rendered views.

2 BACKGROUND AND RELATED WORK

2.1 Spherical Projections

360-degree videos are best represented as pixels on a spherical surface. Thus, they are encoded by first projecting spherical pixels of each omnidirectional view frame to a 2D rectangular plane. Planar pixels are then encoded using standard video codecs such as H.264 [22] and HEVC [21].

A commonly used spherical projection is called the **equirectangular projection (EQ)** (shown in Figure 1(a)). In this projection, spherical pixels are projected onto a rectangular plane based on their yaw and pitch on the sphere with respect to a reference coordinate system[1]. Spherical projections, however, can over-represent spherical pixels. The number of planar pixels that appear on a projection for each spherical pixel depends on both the chosen planar projection and the location of the pixel. For example, the equirectangular projection maps a single pixel at the north pole (i.e., pitch 90°) to a full row of pixels in the plane. This polar pixel thus uses the same number of planar pixels as all pixels from the sphere's equator (i.e., pitch 0°) in the equirectangular projection.

Researchers at Google characterized the uneven distribution of projected spherical pixels using "uniformity comparisons" [2]. They used a saturation map to show if too many or too few pixels are used by the projection to encode different areas on the sphere. Their findings led them to design a new spherical projection scheme: the **equi-angular cubemap (EAC) projection**. EAC is a variation of the standard cubemap projection [7]. Unlike the standard cubemap, EAC applies a transformation to pixel position allowing spherical pixels to more-closely match their corresponding areas on the sphere. EAC produces six cube faces. These faces are then laid out on a 2D plane for video encoding. To achieve the best video compression, the six cube faces are laid out in two bands, a top band containing continuous video content on the left, front, and right faces, and a bottom band containing continuous video content on the top, back, and bottom faces. This layout is also called the "baseball layout", wrapping the sphere like a two-piece leather covering the core of a baseball. Figure 1(c) illustrates how the six faces are laid out in the baseball layout. Based on our findings, 360-degree videos on YouTube are encoded using EAC in the baseball layout today.

In a parallel effort to address the inefficiency of the equirectangular projection, Facebook researchers proposed the **Barrel projection (BRL)** [4]. BRL is motivated by the observation that while the equirectangular projection over-samples areas near the poles, its center [-45°, 45°] pitch portion is roughly uniform. The barrel projection thus encodes the top and bottom quarter of the equirectangular projection near the poles in two circular areas, using many fewer pixels than the center portion. These two circles are then positioned adjacent to the center [-45°, 45°] pitch portion of an equirectangular projection. Figure 1(b) shows the barrel projection.

The **rotated sphere projection (RSP)** [25] also takes advantage of the uniform pixel density at the center band of the equirectangular projection. It includes the central [-45°, 45°] pitch and [-135°, 135°] yaw portions of two differently-configured equirectangular projections. The two equirectangular projections are rotated by 180° in yaw and 90° in roll directions relative to one another. Figure 1(e) shows how the two portions are laid out on the projected image. RSP is similar to EAC in that the top portion in RSP roughly corresponds to the top row in EAC, though different pixel sampling methods are used in these projections. RSP is also similar to the barrel projection in that both projections use the center [-45°, 45°] pitch portion of the equirectangular projection. However, the top portion of RSP contains [-135°, 135°] yaw portions while the main portion of the barrel projection contains the full 360° yaw portions.

2.2 Bitrate-Adaptive 360-Degree Streaming

To facilitate video streaming under varying bandwidth conditions, encoded 360-degree videos are delivered to users via dynamic adaptive streaming over HTTP (DASH) [15], the de facto standard for online video streaming. With DASH, a 360-degree video is divided

[1]For example, consider a reference coordinate system where the xz-plane is located on the equator of the sphere and that the north and south poles are located along the y-axis, then we can assume yaw and pitch on the sphere are equivalent to longitude and latitude as long as the yaw rotation is applied before the pitch rotation and that the north pole has a pitch value of 90°.

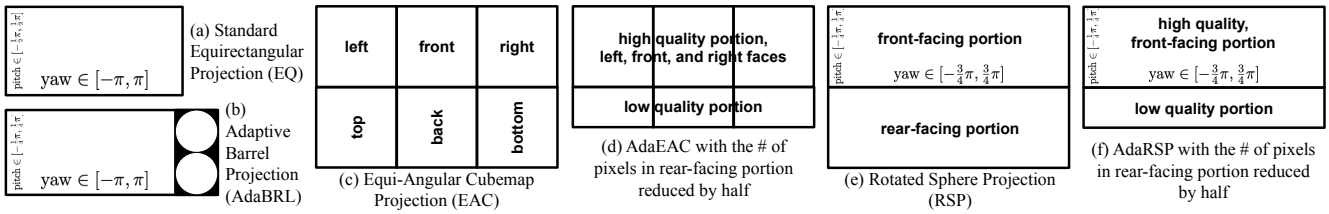


Figure 1: Spherical projections discussed and proposed in this paper. Note that we omit the barrel projection in this figure as its layout is the same as AdaBRL.

temporally into segments, each containing up to a few seconds of video content. Each segment is encoded in different bitrates. During streaming, the client downloads each segment in a bitrate that best matches its available bandwidth.

At playback time, the video player renders views on the display using omnidirectional frames decoded from the downloaded segment. The content inside the rendered view is based on both user's viewing direction (e.g., Euler angle <yaw, pitch, roll>) and the display's field of view (FoV).

2.3 View Inefficiency in 360-Degree Streaming

To allow the user to freely move his/her head during the playback and always render desired view, 360-degree video segments encode full omnidirectional frames. However, both human eyes and 360-degree video viewing devices have limited FoVs. During playback, although all pixels on the omnidirectional view are transmitted, only the portion displayed in the user view's FoV is rendered. This discrepancy between transmitted pixels and viewed pixels means that a significant amount of transmitted pixels are unviewed, and the associated transmitted data is wasted. For example, consider an equirectangular-projected video frame, to render a 90° by 90° view centered at the 0° latitude line, only 11.7% of the pixels in the full frame is required. The remaining 88% of downloaded pixels are not viewed.

To reduce view inefficiency, the research community has proposed tiling, dividing the projected video frames spatially into tiles [12, 13, 17–19, 24, 27]. During streaming, the client only downloads high-quality tiles for the regions it expects the user to view. Despite their theoretical attractiveness, tiling methods pose practical difficulties. First, noticeable seams can show up in views containing adjacent tiles of different qualities, leading to poor user-reported visual quality [26]. Second, the tile selection problem is significantly more difficult than choosing a single temporal segment bitrate level as in standard DASH adaptation. In addition, the tile selection must be computed in the online streaming setting, further increasing practical difficulties [14, 20]. Finally, each tile must be separately-decoded. Scheduling decoding to meet playback and view-rendering deadlines can be challenging [20].

To achieve view efficiency without tiling, Facebook proposed the **offset cubemap projection (OFFSET)** [3]. The offset cubemap projection distorts spherical pixels so that one cube face encodes a small pixel-concentration area on the sphere in high-quality. If the user's view is near the center of the high-quality face, high view efficiency can be achieved. However, as the offset cubemap's pixel-concentration area on the sphere is very small, e.g., 30° by 30° FoV, even small view direction changes (e.g., head movements) during playback of the video segment can cause the view to deviate from the pixel-concentration area, increasing number of wasted pixels, and decreasing user-observed visual quality [28]. In addition, offset cubemap uses 22 default pixel-concentration directions, resulting in significant server-side storage overhead.

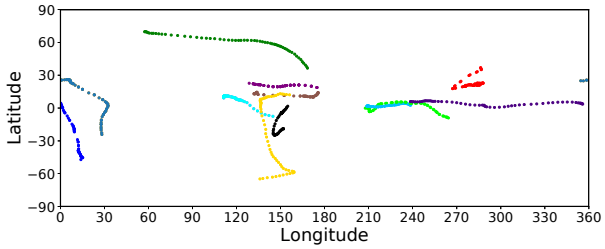


Figure 2: View trajectories of 12 users (randomly selected from a dataset of 58 users' head movements [11]) watching 2 seconds in a video. Every dot in this figure represents the view orientation of one frame. Dots of the same color belong to the same user's trajectory.

3 MOTIVATION: VIEW TRAJECTORIES

360-degree video allows users to freely view content in any direction. User-view orientations often form trajectories which can cover a long distance. Figure 2 plots 12 users' view orientations watching the same temporal portion of a video for 2 seconds. Here, we represent a view orientation as the view center's longitude and latitude coordinates on the sphere. Each user's view orientations are plotted in a different color. View trajectories can be seen through sequences of the same color in the plot.

These long trajectories limit the visual qualities that can be achieved in practice by offset projection approaches having areas of focus that extend radially from a single point. Because these symmetric orientations do not align well with observed user behavior, rendered views are more likely to include less high-quality content, which means the user view will deviate from the high-quality area for many segments.

We further analyzed the view trajectories of all video segments in a video and present the results in Figure 3. In this figure, we consider the view trajectories of 58 users watching a 58-second long video divided into 58 segments. (Each segment contains 1 second video content.) The view trajectories are extracted from a publicly-available head movement dataset [11].

We first analyzed the centroid of all view orientations in the video segment and the view orientation in each video frame (for a 30 frame-per-second video, a 1 second segment contains 30 frames in total). Figure 3 (left) shows the cumulative distribution of maximum angles from centroids of each trajectory for all 58 segments and 58

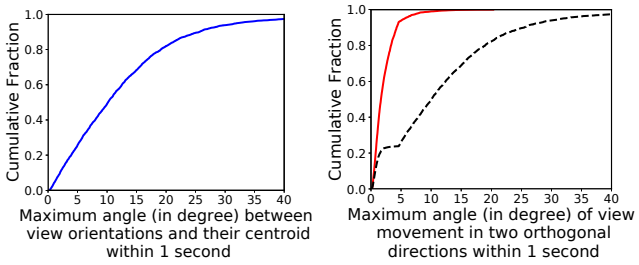


Figure 3: Left: The cumulative fraction of maximum view orientation angles from the centroid of each trajectory. Right: The cumulative fractions of maximum view orientation angles from the minimum (solid red) displacement planes (with plane normal p_{min}) and planes with normal $p_{min} \times p_{max}$ (dashed black) for each view trajectory.

users. We can see that view orientations can significantly deviate from the centroid of the view trajectory within a 1-second period.

It is further possible to select two orthogonal directions where the angular displacements of orientation vectors in a trajectory are maximal in one direction but minimal in the other. To find these directions, we set up the following optimization problem:

$$p_{\min} = \text{argmin}_p \sum_i (p^T \cdot v_i)^2$$

$$p_{\max} = \text{argmax}_p \sum_i (p^T \cdot v_i)^2$$

$$\sum_i (p^T \cdot v_i)^2 = p^T \left(\sum_i v_i \cdot v_i^T \right) p = p^T M p$$

, where v_i represents a user-view orientation in frame i, $||v_i|| = 1$, and $||p|| = 1$. Here, $p_{\min}$ and $p_{\max}$ can be found by taking the eigenvectors of M associated with the minimum or maximum eigenvalues, respectively.

Figure 3 (right) shows cumulative distributions of the maximum angles from planes having normal p_{min} and $p_{min} \times p_{max}$ (cross product of p_{min} and p_{max}) for each trajectory in the dataset. Here, p_{min} is the eigenvector associated with the minimum eigenvalue, and p_{max} is associated with the maximum. For over 93% of these 1-second long view trajectories, the maximum angle with their respective planes having normal p_{min} is smaller than 5 degrees (in red solid line). On the other hand, for only 26% view trajectories, the maximum angle with their respective plane having normal $p_{min} \times p_{max}$ is smaller than 5 degrees (in black dashed line).

Based on the observations that user-view trajectories can be effectively decomposed into minimal and maximal directions, in the next section, we describe our design for an adaptive area-of-focus projection scheme that better captures this typical viewing pattern.

4 AdaP-360: USER-ADAPTIVE AREA-OF-FOCUS PROJECTIONS

In this section, we propose AdaP-360 – a general user-adaptive projection generation method. Our method leverages historical user view data to select sets of high-quality regions that can efficiently cover past user-view trajectories of each temporal video segment.

Figure 4: Pixels in red-shaded positions are needed for rendering a view trajectory that moves from <yaw=-45°, pitch=0°> to <yaw=0°, pitch=0°>. *Left:* in an equirectangular-projected frame, the red-shaded positions are 18% of all pixels in the frame. *Right:* when the view trajectory perfectly aligns with the horizontal center line of the adaptive RSP (AdaRSP) frame's high quality portion, the red-shaded positions now occupy 32% of all pixels in the frame.

This method generates adaptive projections derived from a variety of underlying spherical projections such as EAC, RSP, and BRL. Each layout is configured to devote more pixels to one area-of-focus region (creating a high-quality region) and sub-sample pixels in the remaining region (creating a low-quality region).

4.1 Overview

Align area-of-focus region with view trajectory. Our main innovation in the user-adaptive area-of-focus projection is that we select a high-quality region (i.e., the area-of-focus) that best aligns with the user-view trajectory. We can then decrease the quality of the other region, saving bandwidth on unviewed pixels. For example, the top portion of RSP covers a 90° by 270° area. Thus, if a view trajectory perfectly aligns with the horizontal center line of the top portion, the whole trajectory can be covered by the high-quality region. Figure 4 illustrates this example. As a greater percentage of pixels are used to render views, the percentage of unviewed (i.e., wasted) pixels decreases, allowing us to achieve bandwidth savings without sacrificing visual quality.

Select AdaP-360 configurations for encoding and server-side storage. While the set of possible user-view trajectories is huge, only limited storage is available at the streaming server. Thus, it is infeasible to create one area-of-focus projection for each view trajectory. Instead, area-of-focus projections should be selected such that each high-quality region can cover not just one user-view trajectory of a segment, but view trajectories from groups of users with similar behaviors. To do so, we design an efficient algorithm to select a small set of adaptive area-of-focus projections (**AdaP-360**) that best represent groups of historical trajectories over a segment. Once these AdaP-360 configurations are determined, video segments are transcoded accordingly and stored at the streaming server for clients to request.

Choose the best AdaP-360 to download during streaming. During streaming playback, the client must select and download a video segment encoded in an adaptive area-of-focus projection that roughly aligns the user view with the high-quality region of the projection. Otherwise, the visual quality of rendered views will suffer. In the worst-case scenario, the user's view can be centered at the down-sampled low-quality portion.

To address this problem, we predict each position of user's view trajectory using linear regression. This regression formulation takes a window of past user-view orientations as input. During streaming,

Table 1: Variables used in the problem formulation.

$x_{p,j}$	A binary variable indicating whether to assign an AdaP-360 configuration p to a view trajectory j.
y_p	A binary vector representing whether an AdaP-360 configuration p should be selected, i.e., encoded for streaming.
$s_{p,j}$	A variable representing the cost of assigning view trajectory j to AdaP-360 configuration p. For example, a weighted sum of the number of pixels in the view rendered using the low-quality region.
K	An integer limiting the maximum number of AdaP-360 configurations that can be selected and encoded for a video segment.

a client requests the adaptive area-of-focus projection whose high quality region best-matches the user's predicted view trajectory when the segment is rendered.

We also note that our linear-regression-based view prediction is a simple one – it does not leverage any deep features of the video content. We show that our proposed AdaP-360 approach works well even with simple prediction approaches. We expect better bandwidth savings can be achieved with more-effective view predictions.

4.2 Adaptive Area-of-Focus Projections

We apply our user-adaptive scheme to three efficient and popular spherical projections: the equi-angular cubemap (EAC) projection (currently used by YouTube), the rotated sphere projection (RSP), and the barrel projection (BRL) (currently used by Facebook). We refer to the user-adaptive versions of these projections as AdaEAC, AdaRSP, and AdaBRL, respectively. Figures 1(d), 1(f), and 1(b) illustrate these projections.

To create an area-of-focus projection from the RSP and EAC projections, we sub-sample vertical pixels (with respect to the frame layout) in the bottom portion (representing the "rear-facing" direction), allowing both top and bottom portions to be laid out in a single frame while still maintaining the rectangular shape of the final layout. The BRL projection is an area-of-focus projection by its definition: it contains a high-quality region covering the center [-45°, 45°] pitch portion of the sphere and a low-quality region covering the remaining areas.

Based on the properties of these projections, we can define an area-of-focus projection by the **equatorial plane** of the high-quality portion and an **orientation** vector that lies on this equatorial plane. We represent the equatorial plane using the plane normal n, where n is a unit vector. We refer to the orientation vector by the symbol o. For RSP, a spherical pixel at o is projected to the center of front-facing band of the layout (the top rectangular section of the planar image). For EAC, a spherical pixel at o is projected to the center of the front cube face. For BRL, this pixel is projected to the center of the equirectangular band.

4.3 Problem Formulation: AdaP-360 Configuration Selection

We address the adaptive area-of-focus projection selection problem by formulating an optimization problem that takes historical user views of 360-degree segments as input and returns a set of AdaP-360 configurations (e.g., (n, o) vectors defining the projection) that can best serve these historical views. The set of selected AdaP-360 configurations are restricted so that they fit within a pre-defined server-side storage budget.

User-view history inputs to the algorithm are defined to be trajectories of view orientations for each frame in a video segment. We represent view trajectories as $j = [j^{(1)}, j^{(2)}, \cdots, j^{(n)}]$, where $j^{(n)}$ is a quaternion representing the rotation of user's view at the n^{th} frame of the video segment. We define the set of past view trajectories as $\mathcal{J}$. We then consider a discrete set $\mathcal{P}$ of candidate area-of-focus projections by limiting ourselves to configurations whose high-quality regions perfectly align with a view in $\mathcal{J}$.

Using the variables described in Table 1, we can formulate the area-of-focus projection selection problem as follows:

$$\begin{aligned}
\text{minimize:} \quad & \sum_p \sum_j s_{p,j} x_{p,j} \\
\text{subject to:} \quad & \sum_p x_{p,j} = 1 && \forall j \\
& x_{p,j} \le y_p && \forall p, \forall j \\
& \sum_p y_p \le K \\
& x_{p,j} \in \{0,1\} && \forall p, \forall j \\
& y_p \in \{0,1\} && \forall p
\end{aligned}$$

The formulated optimization problem aims to select the best set of AdaP-360 configurations such that: (1) each view trajectory of a video segment must be assigned to exactly one of the selected configurations; (2) the visual qualities of rendered views are maximized; and (3) the selected number of AdaP-360 configurations is at most K (this limit acts as a constraint on the total storage associated with each segment.)

4.4 Greedy AdaP-360 Configuration Selection

To efficiently select AdaP-360 configurations, we convert the formulation into a set cover problem and use a greedy algorithm to solve it. In the converted formulation, given past view trajectories (i.e., a set of items to be covered) and candidate AdaP-360 configurations that can be used to render view trajectories in high quality (i.e., collections of sets covering items), select K AdaP-360 configurations (i.e., K sets) such that each past view trajectory (i.e., each item) is covered by exactly one selected AdaP-360 (i.e., one selected set).

A challenge in this converted formulation is how to determine if a candidate AdaP-360 configuration can render a view trajectory in high quality. Intuitively, if a view trajectory can be rendered using more pixels in the high-quality region of an adaptive area-of-focus projection, then this user view can achieve better visual quality. Based on this intuition, we propose the following approach for approximating visual quality of rendered views. We first create a spherical mesh with 1280 triangular faces by repeatedly refining faces of a normalized regular icosahedron. For each candidate AdaP-360 configuration, we calculate the set of triangular faces whose centers fall within the high-quality, area-of-focus portion. Similarly, we can calculate the set of triangular faces required for rendering each view trajectory. The visual quality of rendered views can thus

1: Generate candidate AdaP-360 configurations, e.g., with different high quality focus areas
2: **for all** past user trajectories **do**
3: Compute which candidate AdaP-360 configurations can render views in the trajectory with high visual quality
4: **while** the number of selected AdaP-360 configurations is smaller than K **do**
5: Select the AdaP-360 configuration that allows the most remaining view trajectories to be rendered with high quality

Figure 5: Pseudocode for the Greedy AdaP-360 Configuration Selection Algorithm.

be approximated by calculating the percentage of triangular faces required by a view trajectory that also fall within the AdaP-360's high-quality portion. For a candidate AdaP-360 configuration to cover a view trajectory with high visual quality, we require that the percentage of triangular faces covered by the high-quality portion to be higher than Θ – the goodness threshold. Figure 5 outlines the greedy algorithm for AdaP-360 configuration selection.

For the remaining trajectories that cannot be "well covered" by the selected K AdaP-360 configurations, we fallback to using the non-adaptive version. For example, if RSP is used as the underlying projection, the fallback version would use the same number of pixels to represent the top and bottom portions of the projection (as shown in Figure 1(e)), encoding both portions in the same quality.

4.5 Requesting the Best AdaP-360 During Streaming

A set of K adaptive area-of-focus projections selected by our greedy algorithm are pre-computed and stored for clients to download during streaming. For bandwidth-efficiency, only a single projection should be requested and downloaded. Ideally, we want to choose the projection that best matches the user's future view trajectory.

To do so, the streaming client will first fetch information regarding AdaP-360 configurations (e.g., (n, o) vectors that define an area-of-focus projection) and use this information to calculate the set of triangular faces whose centers fall within the high-quality portion. It will then make predictions of the user's view trajectory over the requested segment and calculate the set of triangular faces required by the predicted view trajectory. Finally, it will choose to download the AdaP-360 whose high-quality portion can cover the most faces required by the view trajectory.

5 IMPLEMENTATION

AdaP-360 creation and rendering. We modified an open source library, `Transform360` [8]. `Transform360` can transform an input equirectangular frame to the BRL, the adaptive barrel (AdaBRL), and the offset cubic (OFFSET) projections. To generate our proposed adaptive projections and baseline projections, we implemented four additional video filters within the `Transform360` library. These additional filters allow us to transform a standard equirectangular input frame to RSP, AdaRSP, EAC[2], and AdaEAC output frames. All output frames are then encoded as normal video frames using the encoding pipeline of FFmpeg[9].

To render 360-degree videos given various projections, we created another tool, `Render360`[3], based on FFmpeg360 [16] we developed before. It renders user views of a frame for a series of view orientations given an input 360-degree video. It can take input videos encoded using a variety of projections: standard equirectangular, RSP, AdaRSP, EAC, AdaEAC, BRL, AdaBRL, and OFFSET. We encode these rendered views into "view-videos" using lossless encoding. The rendered "view-videos" allow us to compare the visual quality, e.g., in terms of viewport peak signal-to-noise ratio (V-PSNR) and viewport video multi-method assessment fusion (V-VMAF) [1, 6].

Greedy AdaP-360 selection. The greedy AdaP-360 selection algorithm requires two parameters: the coverage threshold, Θ, and the number of differently-configured area-of-focus projections, K. We require that for a candidate AdaP-360 configuration to render a view trajectory with high visual quality, it must cover at least 90% of the area required by the view trajectory, i.e., $\Theta = 0.9$. We select six AdaP-360 configurations, i.e., $K = 6$. With the greedy selection, the AdaP-360 configuration selected during the first iteration covers the most trajectories, followed by the second, third, etc.

View prediction. To predict the view trajectory for a future segment, we trained a simple linear regressor using the scikit-learn implementation of SGDRegressor [5]. While view prediction is not the focus of this paper, we show that our proposed AdaP-360 approach is effective even in less-than-ideal client prediction scenarios. That is, AdaP-360 can tolerate certain prediction errors as long as the view is still aligned with high quality portions of the current AdaP-360 configuration.

6 EVALUATION

We compared our method against other baselines experimentally to determine the bandwidths needed to achieve equivalent visual qualities. We run these comparisons in three settings. *First*, we evaluate the performance in an ideal scenario where future view trajectories follow the same pattern as past trajectories used in AdaP-360 selection. *Next*, we evaluate how our AdaP-360 method generalizes when presented with unobserved user traces. *Finally*, we evaluate our method under a realistic streaming setting where the predicted user-view trajectory determines which version of AdaP-360 configurations to download. The realistic streaming setting uses a simple linear regressor for view prediction.

Dataset. We evaluated our AdaP-360 scheme using two public 360-degree video head movement datasets [11, 23]. The first dataset (D1) [11] contains view orientations of 58 users watching 5 different 360-degree videos. These 5 videos in D1 are available in 4K quality. The second dataset (D2) [23] includes view orientations of 48 users watching 9 360-degree videos. The 9 videos in D2 are available in 2K quality.

We use these equirectangular videos available in the datasets as groundtruth videos. For each video, we evaluated the performance of our proposed method over a randomly selected 10 consecutive seconds of content. Each video's 10 seconds content was further divided temporally into 10 1-second long segments.

[2]The `Transform360` implementation of the EAC projection does not support baseball layout of the six cube faces. We implemented a new filter so that the generated EAC frames are in the same layout as used on YouTube.

[3]`Render360` is available at: https://github.com/bingsyslab/render360-adap360.

Our user-adaptive method requires a "training set" of past user-view trajectories. Hence, we used trajectories from 50 users and 40 users in D1 and D2, respectively, as past user-view trajectories to generate $K = 6$ AdaP-360 projections. Traces from the remaining 8 users from each dataset are used as our test sets.

Comparative methods. We compared AdaP-360 projections (i.e., AdaRSP, AdaEAC, and AdaBRL) against state-of-the-art non-adaptive projections: the equirectangular projection (EQ), the equi-angular cubemap projection (EAC), the rotated sphere projection (RSP), the barrel projection (BRL), and the offset cubic projection (OFFSET) used in Oculus 360-degree video streaming [28].

The aspect ratio of RSP, EAC and OFFSET videos are 3:2. For AdaRSP and AdaEAC, when we reduce the number of pixels in the bottom, low-quality region by half, the aspect ratio becomes 2:1. To place the main and circular portions of AdaBRL videos on a same rectangular plane, AdaBRL videos are encoded with 5:2 aspect ratio. For fair streaming bandwidth comparison, we transcoded groundtruth videos to various projections with roughly the same number of pixels. We transcoded videos in D1 to AdaRSP, AdaEAC and EQ in 2880x1440 resolution, RSP, EAC, and OFFSET in 2520x1680 resolution, and AdaBRL in 3840x1536 resolution. Since groundtruth videos in D2 are in much lower quality compared to videos in D1 (2K vs. 4K quality), we transcoded videos in D2 to AdaRSP, AdaEAC and EQ in 1280x640 resolution, RSP, EAC, and OFFSET in 1128x752 resolution, and AdaBRL in 1400x560 resolution.

6.1 Bandwidth Savings

Metrics. To evaluate bandwidth savings of our method, we used the Bjøntegaard-Delta bitrate (BD-rate) [10] metric, commonly adopted by the video compression community [13, 26]. BD-rate calculates the average difference in bandwidth under the same visual quality. It is derived from the rate-distortion (RD) curve. In our experiments, "Rate" on the RD-curve represents the bandwidth used to download a video segment. For "Distortion", we used two visual quality metrics: the viewport peak signal-to-noise ratio (V-PSNR) and viewport video multi-method assessment fusion (V-VMAF). Both visual quality metrics are calculated using views rendered by the downloaded video segment and views rendered by the groundtruth video.

Methodology. To create these RD-curves, we encoded videos in different projections using four quantization parameter (QP) values: {22, 27, 32, 37}. To obtain the "distortion" portion of the curve, we computed V-PSNR and V-VMAF between the views rendered from client-downloaded projections during streaming and groundtruth views. These views were rendered with 90° by 90° FoV. For videos in D1, we generated views with 960x960 resolution. For videos in D2, given that they are in lower resolution than D1, we considered views with 560x560 resolution. We computed V-PSNRs only between the Y luminance component in each rendered view's YUV representation. The groundtruth views were rendered from the original equirectangular videos from two datasets.

6.1.1 Savings under the ideal scenario. The ideal scenario occurs when future view trajectories follow the past view trajectories. We thus evaluate BD-rate for all training set users for all videos.

To generate user views for the Ada* methods, we selected the best among the 6 available adaptive projections to render a given user's view of each segment. Figures 6 (a) and 6 (b) show the RD-curve for 50 training user traces in D1. D2 results are shown in Figures 6 (g) and 6 (h). Results from both datasets show that Ada* projections can achieve the same visual quality metrics, V-PSNR and V-VMAF, with significantly smaller bandwidth compared to EQ, OFFSET, RSP and EAC projections.

Specifically, the RD-curve of OFFSET is very close to EQ. BD-rate result shows that, in D1 (D2) OFFSET uses 2.3% (7.9%) more bandwidth than EQ to achieve the same visual quality. This is because offset cubic projections used in Facebook's Oculus streaming has only a single high-quality direction with a very small pixel-concentration area. As a result, when the view deviates from the high-quality direction, the visual quality drops significantly.

BD-rate calculated between AdaRSP and RSP, between AdaEAC and EAC, and between AdaEAC and EQ show that in D1 (D2), AdaRSP achieves on average 24.9% (29.5%) bandwidth savings compared to RSP, AdaEAC achieves on average 22.8% (27.2%) bandwidth savings compared to EAC, 49.0% (55.4%) savings compared to EQ, and 49.6% (57.8) savings compared to OFFSET while maintaining the same visual quality. AdaBRL performs slightly better than BRL but not as well as AdaRSP or AdaEAC. This performance difference is caused by the high-quality band of AdaBRL covering more spherical area compared to AdaRSP and AdaEAC, leading to more unviewed pixels and lower visual quality.

6.1.2 Savings under the unobserved users scenario. We then evaluate if the set of 6 adaptive projections selected based on past view history can be used for future user views. Here, we use traces from the 8 test set users in the datasets and assume that the client-side view trajectory prediction is perfect.

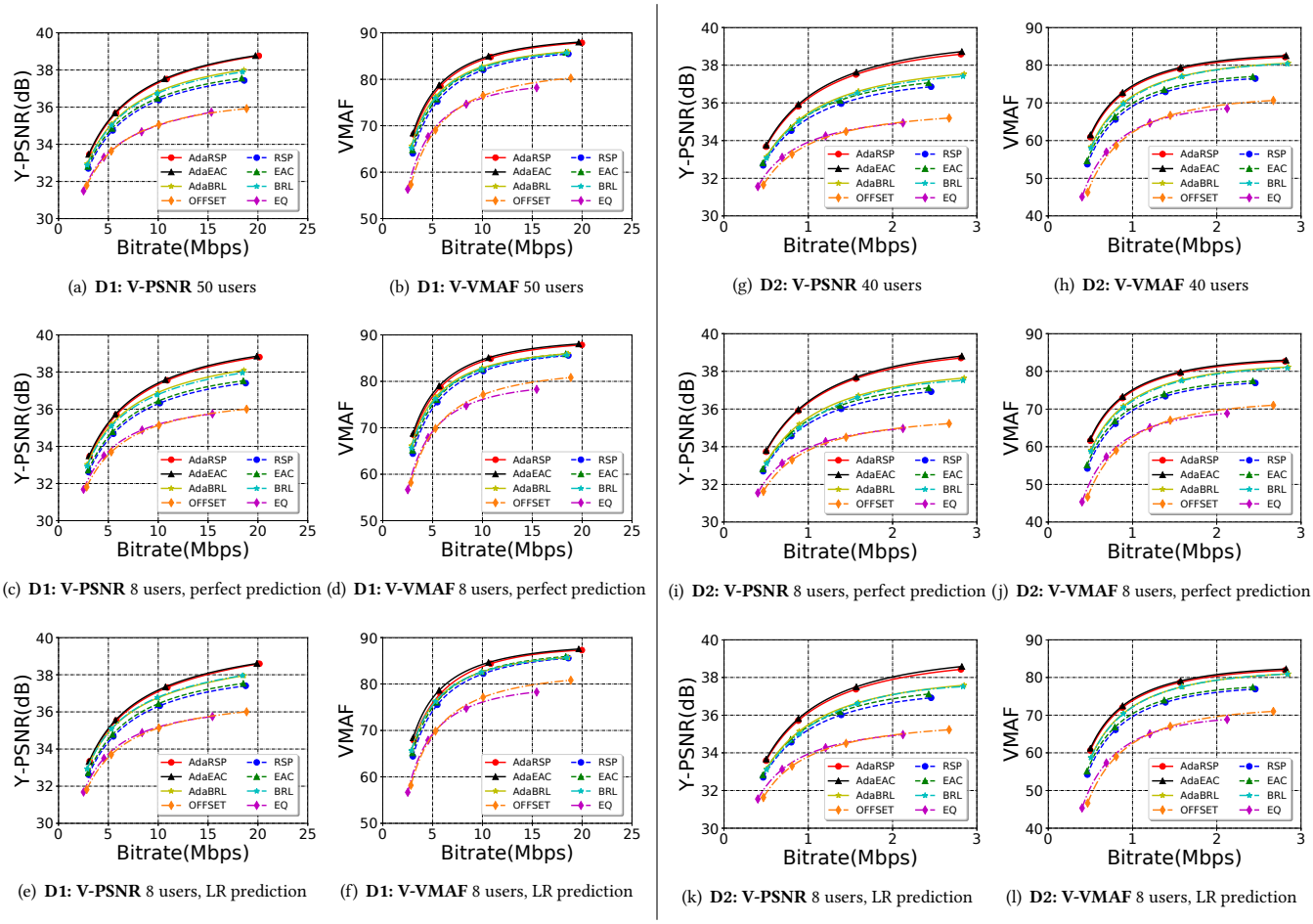
Figures 6 (c) and 6 (d) show the RD-curve for the 8 test set users in D1. D2 results are shown in Figures 6 (i) and 6 (j). Since we assume perfect knowledge of future view trajectory, the client can download the best AdaP-360 for view rendering. BD-rate results show that for D1 (D2), on average, AdaRSP can achieve 27.0% (30.3%) bandwidth savings compared to RSP, AdaEAC can achieve 25.5% (27.5%) bandwidth savings compared to EAC, 44.9% (55.7%) savings compared to EQ, 48.0% (58.6%) savings compared to OFFSET while achieving the same visual quality.

6.1.3 Savings under the client-side prediction scenario. We finally evaluate if bandwidth savings can be achieved under practical prediction. We consider the 8 test-set users' view traces and predict these users' future view trajectories using our simple linear regression (LR) model. In this scenario, the streaming client requests and downloads the best (out of 6 available) AdaP-360 based on the predicted view trajectory.

Figures 6 (e) and 6 (f) show the RD-curve for this scenario for D1. D2 results are shown in Figures 6 (k) and 6 (l). This BD-rate result shows that even using a basic prediction method to decide which future view to render, in D1 (D2), AdaRSP can achieve 22.9% (26.2%) bandwidth savings compared to RSP, AdaEAC can achieve 20.7% (24.0%) bandwidth savings compared to EAC, 41.5% (53.1%) savings compared to EQ, and 45.5% (55.7%) savings compared to OFFSET while achieving the same visual quality.

6.2 Decoding and Rendering Time

We logged the time used to decode a video segment and render views for given user-view orientations on a Linux computer with an Intel i7-7700K CPU and an NVIDIA GeForce GTX 1080 GPU. As



(a) **D1: V-PSNR** 50 users (b) **D1: V-VMAF** 50 users (g) **D2: V-PSNR** 40 users (h) **D2: V-VMAF** 40 users

(c) **D1: V-PSNR** 8 users, perfect prediction (d) **D1: V-VMAF** 8 users, perfect prediction (i) **D2: V-PSNR** 8 users, perfect prediction (j) **D2: V-VMAF** 8 users, perfect prediction

(e) **D1: V-PSNR** 8 users, LR prediction (f) **D1: V-VMAF** 8 users, LR prediction (k) **D2: V-PSNR** 8 users, LR prediction (l) **D2: V-VMAF** 8 users, LR prediction

Figure 6: RD-curves constructed using encodings of the all videos in Dataset 1 (D1, left) and Dataset 2 (D2, right) to render views of 90° by 90° FoV. Two distortion metrics are used: viewport PSNR (V-PSNR) and viewport VMAF [1] (V-VMAF). *Top row:* the RD-curve for users whose view history is used for AdaP-360 configuration selection. *Middle row:* the RD-curve under perfect client-side view prediction for the 8 users held-out from server-side AdaP-360 configuration selection. *Bottom row:* the RD-curve using linear regression (LR) for client-side view-prediction for the remaining 8 users.

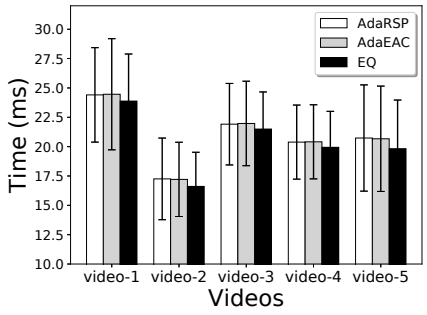


Figure 7: Average per-frame decoding and rendering time of videos in Dataset 1.

there are 30 frames in each video segment, we calculate and report the average per-frame decoding and rendering time. The results of videos in Dataset-1 are shown in Figure 7. Error bars in this figure indicate decoding and rendering duration at 2.5% and 97.5% percentiles. The figure shows no significant difference in decoding and rendering time among AdaRSP, AdaEAC, and EQ schemes.

7 CONCLUSION

We created a method for generating area-of-focus frames of common 360-degree video projections that can be adapted, based on user view histories, to deliver more efficient 360-degree video streams. Our method is applicable to a wide variety of popular projections and is simple to understand and implement. We evaluated the approach on adaptive versions of the BRL, RSP, and EAC projections. Results from these evaluations show that compared to the equirectangular projection, our method can achieve on average 55.4% bandwidth savings under ideal conditions and 53.1% savings in a realistic scenario where a streaming client must predict future user-view trajectories.

8 ACKNOWLEDGEMENT

We appreciate constructive comments from anonymous referees. This work is partially supported by National Science Foundation under grant CNS-1618931.

REFERENCES

[1] 2016. Toward A Practical Perceptual Video Quality Metric. https://medium.com/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652.

[2] 2017. Bringing pixels front and center in VR video. https://blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/.

[3] 2017. End-to-end optimizations for dynamic streaming. https://code.facebook.com/posts/637561796428084/end-to-end-optimizations-for-dynamic-streaming/.

[4] 2017. Enhancing high-resolution 360 streaming with view prediction. https://engineering.fb.com/virtual-reality/enhancing-high-resolution-360-streaming-with-view-prediction/.

[5] 2018. sklearn.linear_model.SGDRegressor. https://github.com/scikit-learn/scikit-learn/blob/a86709fdc379f7d7db76a75f39572890e4ddcad1/sklearn/linear_model/stochastic_gradient.py.

[6] 2018. VMAF: The Journey Continues. https://medium.com/netflix-techblog/vmaf-the-journey-continues-44b51ee9ed12.

[7] 2019. Cubic Projection. http://wiki.panotools.org/Cubic_Projection.

[8] 2019. Transform360. https://github.com/facebook/transform360.

[9] 2020. FFmpeg. http://www.ffmpeg.org/.

[10] Gisle Bjontegaard. 2001. Calculation of average PSNR differences between RD-curves. *VCEG-M33* (2001).

[11] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 2017. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 199–204.

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

[13] Mario Graf, Christian Timmerer, and Christopher Mueller. 2017. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over HTTP: Design, Implementation, and Evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 261–271.

[14] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Rubiks: Practical 360-Degree Streaming for Smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 482–494.

[15] ISO/IEC 23009-1:2014 2014. *ISO/IEC 23009-1:2014 Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*. Standard. International Organization for Standardization.

[16] Yao Liu, Chao Zhou, Shuoqian Wang, and Mengbai Xiao. 2019. FFmpeg360 for 360-degree videos: edge-based transcoding, view rendering, and visual quality comparison: poster. In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*. 337–339.

[17] Anahita Mahzari, Afshin Taghavi Nasrabadi, Aliehsan Samiei, and Ravi Prakash. 2018. FoV-Aware Edge Caching for Adaptive 360Â° Video Streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 173–181.

[18] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 1689–1697.

[19] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An HTTP/2-Based Adaptive Streaming Framework for 360 Virtual Reality Videos. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 306–314.

[20] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 99–114.

[21] Gary J Sullivan, Jens Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668.

[22] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. 2003. Overview of the H. 264/AVC video coding standard. *IEEE Transactions on circuits and systems for video technology* 13, 7 (2003), 560–576.

[23] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 193–198.

[24] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo. 2017. 360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 315–323.

[25] Yan Ye, Elena Alshina, and Jill M Boyce. 2017. JVET-G1003: Algorithm Description of Projection Format Conversion and Video Quality Metrics in 360Lib Version 4. *Joint Video Exploration Team (JVET)* (2017).

[26] Matt Yu, Haricharan Lakshman, and Bernd Girod. 2015. Content adaptive representations of omnidirectional videos for cinematic virtual reality. In *Proceedings of the 3rd International Workshop on Immersive Media Experiences*. ACM, 1–6.

[27] 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 2016 ACM on Multimedia Conference*. ACM, 601–605.

[28] Chao Zhou, Zhenhua Li, and Yao Liu. 2017. A Measurement Study of Oculus 360 Degree Video Streaming. In *Proceedings of the 8th International Conference on Multimedia Systems*. ACM.