Navigation Graph for Tiled Media Streaming

Jounsup Park
University of Illinois at Urbana-Champaign
Urbana, Illinois
jounsup@illinois.edu

Klara Nahrstedt
University of Illinois at Urbana-Champaign
Urbana, Illinois
klara@illinois.edu

ABSTRACT

After the emergence of video streaming services, more creative and diverse multimedia content has become available, and now the capability of streaming 360-degree videos will open a new era of multimedia experiences. However, streaming these videos requires larger bandwidth and less latency than what is found in conventional video streaming systems. Rate adaptation of tiled videos and view prediction techniques are used to solve this problem. In this paper, we introduce the Navigation Graph, which models viewing behaviors in the temporal (segments) and the spatial (tiles) domains to perform the rate adaptation of tiled media associated with the view prediction. The Navigation Graph allows clients to perform view prediction more easily by sharing the viewing model in the same way in which media description information is shared in DASH. It is also useful for encoding the trajectory information in the media description file, which could also allow for more efficient navigation of 360-degree videos. This paper provides information about the creation of the Navigation Graph and its uses. The performance evaluation shows that the Navigation Graph based view prediction and rate adaptation outperform other existing tiled media streaming solutions. Navigation Graph is not limited to 360-degree video streaming applications, but it can also be applied to other tiled media streaming systems, such as volumetric media streaming for augmented reality applications.

KEYWORDS

Tiled media streaming, View prediction, Rate adaptation

ACM Reference format:
Jounsup Park and Klara Nahrstedt. 2019. Navigation Graph for Tiled Media Streaming. In *Proceedings of Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, October 21–25, 2019 (MM '19),* 9 pages.
https://doi.org/10.1145/3343031.3351021

1 INTRODUCTION

Adaptive Bit-rate Streaming (ABS) has been used to control the quality of videos available in Video on Demand (VoD) services. Dynamic Adaptive Streaming over HTTP (DASH) [7] is one of the most successful video adaptive streaming platforms. DASH functions by generating multiple copies of the video segments with different encoding qualities along with a Media Presentation Description (MPD) file, which describes the videos information. DASH clearly distinguishes the role of the server and the clients. The server stores the video data as segments with multiple quality choices and uses the media presentation description file to provide video information to the clients. As the clients request the video data, the server provides them with information from the MPD file. The clients can then also request the video segments with the proper quality to allow for continuous play. Since the video segments are already encoded with a certain video encoder, the server can send the segments as soon as requests are received. The clients are responsible for rate adaptation. This allows the video to be viewed on various types of devices without changing the video server. Moreover, it is a Contents Delivery Network (CDN) friendly platform, which allows clients to fetch the video segments either from the server or from the cache, depending in which is closer to the clients. CDN can lower the network congestion and the latency by storing frequently viewed video segments in the cache.

360-degree videos have the potential to be the next major multimedia format because they can be used for Virtual Reality (VR) applications. However, in order to provide users with their desired level of Quality of Experience (QoE), they need a much larger data rate than VoD services currently provide. To show a similar number of pixels in the Field of View (FoV) or viewport (the video area that viewers actually view), 360-degree videos need 4 times higher resolution than conventional videos. Initially, 360-degree video streaming services streamed the whole video to the viewers and the viewers extracted the viewports from the video they wanted. This required a much larger bandwidth than conventional videos. Regional quality control of videos based on the feedback information from the client's viewport helps to lower the required bandwidth and makes the quality of the viewport better. Spatial Relationship Descriptor (SRD) [7] is introduced to support tile-based videos, such as multi-view point videos or omnidirectional videos [5][13]. A large video can be divided into smaller tiles that can be encoded independently using legacy video encoders, such as HEVC [8]. Different algorithms are introduced to improve the viewport quality by assigning more bits to the tiles in the viewport and fewer bits to non-visible tiles. Utility maximized rate control algorithms are introduced for the rate selection of tiles in unicast [13] and multicast [12] systems, where the utility means the expected quality of the viewport.

In addition to requiring a larger bandwidth, 360-degree video streaming systems require interaction between the server and the clients since clients' current viewport information must be reported to the server. However, there is a latency between the server and the clients that does not allow the viewport feedback information to be directly applied to regional quality control. Therefore, regional quality control needs to consider the clients' future viewport, which makes viewport prediction necessary for 360-degree video streaming systems. Viewport prediction can rely on historical viewport data [2][6][19][20] or video content-analysis result [21].

In this paper, we introduce the Navigation Graph concept, which captures viewing behaviors and helps perform rate adaptation based on view predictions. The Navigation Graph is a graph that consists of Vertices (set of tiles in a segment) and Edges (transition probabilities between vertices). We introduce how the Navigation Graph is generated and a probability model of the view transitions is built, how we deploy the Navigation Graph to perform a view prediction, and finally how tiled media rate adaptation is executed.

This paper is organized as follows: Section 2 covers the background and related work. Section 3 describes the Navigation Graph. Section 4 presents our view prediction algorithm and rate adaptation algorithm using the Navigation Graph. Section 5 presents our experimental results, and Section 6 concludes the paper.

2 BACKGROUND AND RELATED WORK

2.1 360-degree video streaming systems

360-degree videos are created by stitching together multiple videos taken by multiple outward looking cameras that capture the whole surrounding sphere. The sphere is then projected into the 2D plane to make the 360-degree video easier to process and store. This also makes it easy to use a conventional video encoder to compress the video. Most 360-degree videos are stored in the server in a rectangular shape.

DASH [7] is a popular video streaming platform that chooses the bit-rate of the video according to network conditions. DASH encodes the video into 1 to 15 seconds long segments, where each segment consists of a sequence of video frames. Usually, 360-degree videos have very high resolution in comparison to conventional videos. Therefore, the original videos are cut into smaller videos, called tiles, and encoded independently with different qualities (representations). The encoded and stored video data are called segments, and they are delivered to clients upon request. The bit-rate can change after each segment. SRD is also introduced to support tiled video streaming. SRD includes the tile information which is used to recover the viewport, for the client, i.e., each client watches only a certain region of each video frame. The encoder information, segment duration, SRD, and other necessary information are stored as a Media Description Presentation (MPD), and clients can request video segments based on the information in the MPD file using HTTP get requests. Since the adaptation heuristic is not specified in the MPD, clients can perform their own adaptation heuristics in order to watch a video with the best quality. Clients then have playback buffers to store video segments, which will prevent stalling under poor network conditions. The buffer status and measured network throughput are used in the adaptation heuristics [9].

2.2 Rate Adaptation for Tiled Media

The simplest way to stream a 360-degree video would be to stream the whole 360-degree viewing area as a single video stream. However, it takes too much bandwidth and it is not efficient since clients actually view only a small portion of the whole 360-degree viewing area. There are many studies that use tiled media rate adaptation to try to transmit better tiles to the viewport and decrease the video quality of non-visible tiles. BAS360 [18] performs spatial and temporal rate adaptation. It generates multiple streaming units, which consist of tiles of consecutive segments and selects the streaming units with the lowest bandwidth waste rate to be transmitted to clients. This method actually helps to optimize the spatial and temporal variation in the video quality, where as most previous tiled media streaming algorithms have only optimized the spatial quality.

A cross-user learning based system (CLS) [20] gathers the users' fixation data on the server and gives more weight to the tiles with more fixations. It can also utilize prior viewers' fixation data to optimize the weighting coefficients. It also performs the clustering of the fixations to classify clients and chooses the weighting coefficients that should be used for each client. Sun et al. [15] propose a multi-tier streaming system generating two separate streams that deliver a base view and an enhanced view. It utilizes Scalable Video Coding (SVC) [16] to enhance viewport quality by opportunistically receiving enhanced layers.

Client-side playback buffer plays an important role preventing stall events that can happen because of sudden drop of network throughput [9]. Almquist et al. [1] examine the effects of prefetching of future video segments. Window-based adaptation (WBA) [11] uses tile-based rate adaptation including spatial and temporal rate selection, which can maximize expected utility in a window. "Window" denotes the tiles and the segments that will be played in the near future; these tiles and segments have a high impact on clients' viewport quality. WBA receives new chunks and updates existing chunks within a window to maximize the utility which is indicative of the expected quality of the viewport.

2.3 Viewport Prediction

A viewport prediction is critical for lowering the required bandwidth because prediction error leads to waste bandwidth. Because of the inevitable latency of the network, clients will receive video segments a few milliseconds after they requested them. Therefore, predicting the viewport becomes more difficult as the latency of the network increases. A linear regression using a client's past viewport data is used to predict future viewports assuming their head movement will continue follow similar patterns. A linear regression with a Gaussian prediction error [19] shows better prediction performance than the prediction found using only a linear regression. Sun et al. [15] propose a viewport prediction algorithm using recent samples that monotonically increase or decrease. Prediction errors occur whenever the user turns his/her head in the opposite direction.

While the viewport predictions rely solely on viewport data from the viewer him/herself, a cross-user viewport prediction [20] could further improve the prediction accuracy since the viewport trajectory of multiple viewers could be correlated. The viewing behavior is correlated with the video content, and it may help predict future viewports. For example, if the video is recorded by someone riding a roller coaster, the viewer may concentrate more on the vanishing point in the direction the roller coaster is heading. Therefore, we can find viewing patterns not only from prior viewers' viewport data but also from video analysis.

Then deep reinforcement learning can be used to predict head movements [21]: however, video analysis technology requires high

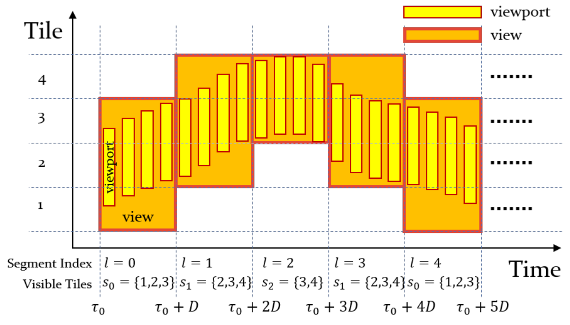

Tile
viewport
view
4
3
2
1
viewport
view
....
Segment Index $l = 0$ $l = 1$ $l = 2$ $l = 3$ $l = 4$
Visible Tiles $s_0 = \{1,2,3\}$ $s_1 = \{2,3,4\}$ $s_2 = \{3,4\}$ $s_1 = \{2,3,4\}$ $s_0 = \{1,2,3\}$
Time
τ_0 $\tau_0 + D$ $\tau_0 + 2D$ $\tau_0 + 3D$ $\tau_0 + 4D$ $\tau_0 + 5D$

Figure 1: Segments, Tiles, Viewports and Views

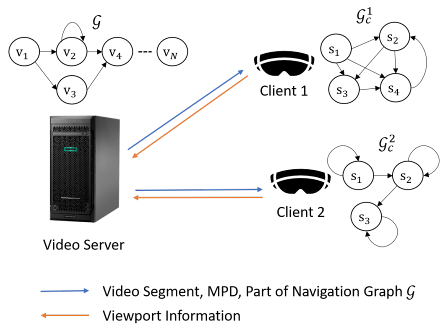


Figure 2: System Overview

computing power on the client's side, which is often not possible. Recently, videos are most frequently viewed on mobile devices that have limited battery and computing power. Recent mobile devices are becoming more powerful and can perform more complicated processing, but still the complicated processing required for viewport prediction can cause additional delays for future frame requests. In this paper, we introduce the Navigation Graph concept and show that it is useful for both single-user view prediction and cross-user view prediction. Moreover, the Navigation Graph can be used to encode important trajectories in videos using a powerful computer on the video server side and can then share this information with clients.

3 NAVIGATION GRAPH

DASH defines video segments that contain encoded video data. The duration D of the segments is the basic unit of rate adaptation and is usually between 1 to 15 seconds long. SRD is used to describe the spatial relationships between tiles for regional quality control based on viewers' viewports. Figure 1 shows the segments, tiles, viewports and views. The tiles cut the video into smaller spatial regions, and then, the viewport can span multiple tiles within a video frame. A video segment consists of multiple consecutive frames, and viewport can change every frame. We define a "view" as the union of all visible tiles within a segment duration D. Viewport usually means the visible part of the video, but the view is defined as a set of tiles in specific segment that is used for recovering viewports in the video segment. Since the viewers move continuously, the viewport can change within the duration of a segment. For example, a video segment $l = 1$ starts at the time $\tau = \tau_0 + D$, and the viewport spans tiles $(2, 3)$, but the viewport changes to a different region of the video that spans the tiles $(3, 4)$ at the end of the segment $l = 1$. Therefore, a set of tiles $\mathbf{s}_1 = (2, 3, 4)$ is required for a segment $l = 1$ to recover the viewports from the tiles across the whole segment $l = 1$. $\mathbb{S}$ describes a set of all possible combinations of tiles and $\mathbf{s}$ is one of the elements. We define the view $\mathbf{v}$ as a tuple consisting of a segment index l and a set of visible tiles $\mathbf{s}$, where $\mathbf{s}$ is the union of all visible tiles in the duration of a segment l. The temporal variation of the view must be considered to perform the future view prediction and manage the playback buffer. Therefore, a model that describes the relationship between the views in series of segments is required. We introduce the Navigation Graph $\mathcal{G}$, which is a directed graph describing view transitions.

$$\mathcal{G} = (\mathbf{V}, \mathbf{E}). \tag{1}$$

The vertices are defined as

$$\mathbf{V} = \{\mathbf{v} | \mathbf{v} = (l, \mathrm{s}), l \in \{1, 2, ..., L\} \text{ and } \mathrm{s} \in \mathbb{S}\}, \tag{2}$$

where l is a segment index, L is the number of segments in a video and $\mathbb{S}$ is the set of s configuring the views that clients have seen at least once. $\mathbf{E}$ is a set of edges connecting the vertices, where at least one transition happens.

$$\mathbf{E} = \left\{(\mathbf{v}_i, \mathbf{v}_j) | \mathbf{v}_{i,j} \in \mathbf{V}, w(\mathbf{v}_i, \mathbf{v}_j) = p(\mathbf{v}_j | \mathbf{v}_i), i, j \in 1, ..., N\right\}, \tag{3}$$

where $w(\mathbf{v}_i, \mathbf{v}_j)$ is a weight function. We also define the matrix $\hat{\mathbf{E}}$ which consists of the transition probabilities from one vertex to other vertices connected by edges in $\mathbf{E}$,

$$\hat{\mathbf{E}} = \mathbb{R}^{N \times N} = \left\{ \begin{array}{cccc} p(\mathrm{v}_1|\mathrm{v}_1) & p(\mathrm{v}_1|\mathrm{v}_2) & \cdots & p(\mathrm{v}_1|\mathrm{v}_N) \\ p(\mathrm{v}_2|\mathrm{v}_1) & p(\mathrm{v}_2|\mathrm{v}_2) & \cdots & p(\mathrm{v}_2|\mathrm{v}_N) \\ \vdots & \vdots & \ddots & \vdots \\ p(\mathrm{v}_N|\mathrm{v}_1) & p(\mathrm{v}_N|\mathrm{v}_2) & \cdots & p(\mathrm{v}_N|\mathrm{v}_N) \end{array} \right\} \tag{4}$$

where N is the number of vertices. The maximum number of vertices is $2^T \times L$, where the T is the number of tiles and L denotes the number of segments. However, N counts only the vertices that have been visited at least once. The Navigation Graph is expended whenever it encounters a new view. For example, if only one viewer has watched the video one time, there will be L vertices in the Navigation Graph, because single viewer will watch only one view per segment. However, as more viewers watch the same video, the Navigation Graph will expend and the number of vertices N will range between L and $C \times L$ depending on how various the viewing patterns of clients are, where C is the number of clients.

We believe that the Navigation Graph concept has a great potential to be used by many different applications, but, in this paper, we focus specifically on building a view transition model that is useful for view prediction with the Navigation Graph.

3.1 View Transition Model

The Navigation Graph can build a view transition model both on the client side and the server side. Since clients give their view information to the server through the feedback channel, the video server can use clients' view information to update the Navigation Graph. The Navigation Graph concept helps to share the view information with a video server and other clients. Clients only require to feedback segment index and T bits binary number to share the view information. Another way to indicate the viewport is using a quaternion [17] consisting of four floating numbers and a time stamp, but with this method, it is not clear how often the clients should report their viewport and which specific tiles are required to render the viewport. This could be a source of prediction error, because the viewport can change within the duration of a segment, which can last from 1 sec to 15 sec.

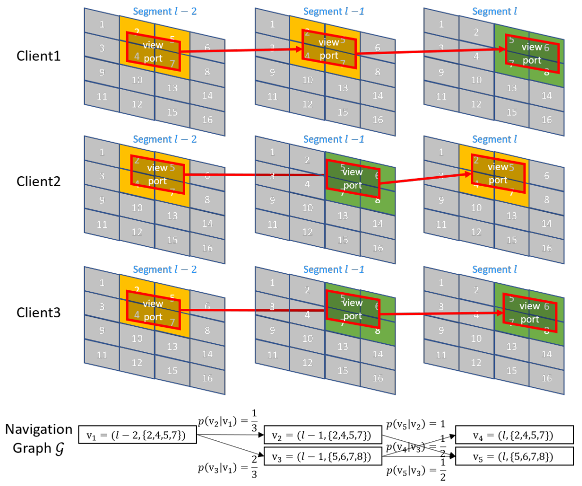
Clients can build their own Navigation Graph by using their own view information. Figure 2 shows that the video server owns the global Navigation Graph, which is gathering all of the clients' view information for any particular video. Each client also has a Navigation Graph generated from their own view information. Therefore, the client's Navigation Graph learns the client's distinctive viewing patterns and the Navigation Graph on the server learns the viewing patterns of all of the clients watching the same video.

3.1.1 *Navigation Graph on Video Server.*

The video server can generate a view transition model by collecting multiple viewers' viewport feedback information. When a 360-degree video is newly made and has never been seen by anyone, there are no vertices or edge in $\mathcal{G}$. As people start to request the video, the server also starts to generate a Navigation Graph. All viewers who watch the video contribute to updating the Navigation Graph, therefore a new viewer can take advantages of past viewers' data. The server compares the current view $\mathbf{v}_c$ received from the client with the vertices in a set $\mathbf{V}$. If the same vertex exists in $\mathbf{V}$ that is the same as $\mathbf{v}_c$, then the server only updates the edges $\mathbf{E}$ and the transition matrix $\hat{\mathbf{E}}$ which describes the transition probabilities. Otherwise, the Navigation Graph increases N by 1 and adds the vertex $\mathbf{v}_c$ into $\mathbf{V}$ as a $\mathbf{v}_N$, and updates the edges $\mathbf{E}$ and the transition matrix $\hat{\mathbf{E}}$. The transition probability in $\hat{\mathbf{E}}$ is updated as

$$p(\mathbf{v}_\mathrm{c}|\mathbf{v}_\mathrm{p}) = \frac{\text{number of clients moving their view from } \mathbf{v}_\mathrm{p} \text{ to } \mathbf{v}_\mathrm{c}}{\text{number of clients visiting } \mathbf{v}_\mathrm{p}}. \tag{5}$$

where the $\mathbf{v}_p$ is a prior vertex that the client had visited before she moved to the current vertex $\mathbf{v}_c$. Since there are no prior views at the beginning of the video, $\mathbf{E}$ has elements only after the first view transition happens. Vertices $\mathbf{V}$ and Edges $\mathbf{E}$ are updated every time the server receives the view information from the clients. Figure 3 shows a Navigation Graph made by three clients' view data. The three clients visited a vertex $\mathbf{v}_1$, therefore, the denominator of $p(\mathbf{v}_2|\mathbf{v}_1)$ and $p(\mathbf{v}_3|\mathbf{v}_1)$ is 3. Client 1 stays in the same set of visible tiles while the other two clients switch their sets of visible tiles in segment $l-1$, and that is why the numerator of $p(\mathbf{v}_2|\mathbf{v}_1)$ is 1 and the numerator of $p(\mathbf{v}_3|\mathbf{v}_1)$ is 2. Edges and transition probabilities connecting vertices $\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ and $\mathbf{v}_5$ are also generated in the same manner. The Navigation Graph is stored in the server along with the MPD files, and part of the Navigation Graph is transmitted to clients upon their request to help them to predict future views and ultimately to allow for the best rate of tile. Cross-user view prediction is performed on the client's end by receiving the part of the Navigation Graph from the server (transition probabilities from the current segment l to the future segments $l+K$ that the client is trying to predict). Clients only need to report one segment index and a T bits binary number to indicate which tiles are required to render the viewports in the segment. In an extreme case, a single vertex may consist of all the tiles in a segment if the client is looking around the whole area of the 360-degree video within the duration of a segment. Therefore, reporting the current view as a vertex $\mathbf{v}_c$ including all the tiles that are needed to generate the client's viewports over the duration of a segment simplifies the feedback process.



Segment $l-2$
Segment $l-1$
Segment l
Client1
Client2
Client3
view port
Navigation Graph $\mathcal{G}$
$\mathrm{v}_1 = (l-2, \{2,4,5,7\})$
$\mathrm{v}_2 = (l-1, \{2,4,5,7\})$
$\mathrm{v}_3 = (l-1, \{5,6,7,8\})$
$\mathrm{v}_4 = (l, \{2,4,5,7\})$
$\mathrm{v}_5 = (l, \{5,6,7,8\})$
$p(\mathrm{v}_2|\mathrm{v}_1) = \frac{1}{3}$
$p(\mathrm{v}_3|\mathrm{v}_1) = \frac{2}{3}$
$p(\mathrm{v}_5|\mathrm{v}_2) = 1$
$p(\mathrm{v}_4|\mathrm{v}_3) = \frac{1}{2}$
$p(\mathrm{v}_5|\mathrm{v}_3) = \frac{1}{2}$

Figure 3: Navigation Graph in Video Server for Cross-User View Prediction

3.1.2 *Navigation Graph on Clients.*

The simplified Navigation Graph in the client's end is used for single-user view prediction (SU). The simplified Navigation Graph for clients is defined as

$$\mathcal{G}_c = (\mathbb{S}, \mathbf{E_c}), \tag{6}$$

where the set $\mathbb{S}$ consists of sets of visible tiles $\mathbf{s}$ that are defined in the previous section (Figure 1). All vertices $\mathbf{s_m} \in \mathbb{S}, m = 1, \ldots, M$ are the vertices that a client has visited at least once. $\mathbf{E_c}$ consists of edges connecting all vertices in $\mathbb{S}$ and the transition probability matrix is defined as

$$\hat{\mathbf{E}}_c = \mathbb{R}^{M \times M} = \begin{Bmatrix} p(\mathrm{s}_1|\mathrm{s}_1) & p(\mathrm{s}_1|\mathrm{s}_2) & \cdots & p(\mathrm{s}_1|\mathrm{s}_M) \\ p(\mathrm{s}_2|\mathrm{s}_1) & p(\mathrm{s}_2|\mathrm{s}_2) & \cdots & p(\mathrm{s}_2|\mathrm{s}_M) \\ \vdots & \vdots & \ddots & \vdots \\ p(\mathrm{s}_M|\mathrm{s}_1) & p(\mathrm{s}_M|\mathrm{s}_2) & \cdots & p(\mathrm{s}_M|\mathrm{s}_M) \end{Bmatrix} \tag{7}$$

where M is the number of vertices. If the client is staying at the same viewing position during the whole video, $M = 1$, since the client will watch only one set of tiles. However, M can be at most L (number of segments) if the client moves her head every segment to different set of tiles across all segments of the video. The difference between the original Navigation Graph $\mathcal{G}$ and the simplified version $\mathcal{G}_c$ is the configuration of the vertices. The vertices for the simplified Navigation Graph consist of the set of tiles $\mathbb{S}$ and do not include a segment index. $\hat{\mathbf{E}}_\mathbf{c}$ elements present the transition probabilities $p(\mathbf{s}_\mathrm{c}|\mathbf{s}_\mathrm{p})$ from vertices $\mathbf{s}_p$ to vertices $\mathbf{s}_c$, where

$$p(\mathbf{s}_\mathrm{c}|\mathbf{s}_\mathrm{p}) = \frac{\text{number of transitions from } \mathbf{s}_\mathrm{p} \text{ to } \mathbf{s}_\mathrm{c}}{\text{number of times visiting the set of visible tiles } \mathbf{s}_\mathrm{p}}. \tag{8}$$

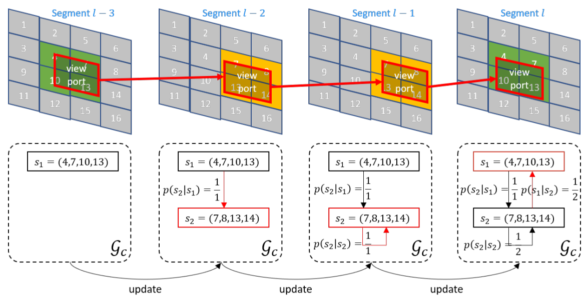


Figure 4: Navigation Graph in Clients for Single-User View Prediction

Figure 4 shows how the Navigation Graph is constructed by four consecutive video segments that a client has seen. When the video starts at $l-3$, the Navigation Graph only has one vertex that consists of visible tiles s_1. The client changes her head position in segment $l-2$ and requires a different set of tiles s_2. Since there is no corresponding vertex s_2 in $\mathbb{S}$, s_2 is newly added into $\mathbb{S}$, and an edge is created to connect s_1 and s_2. The transition probabilities are updated by (8).

This information is helpful to understand how the client behaves in a certain head position. If a viewer tilts his/her head in order to look upward, he/she may soon become uncomfortable and change the position of his/her head. In this case, the head movement and change in the view is not related to the video content, but due to the head position itself. In contrast, if the viewer is looking forward, which is a more comfortable head position, then he/she will probably stay in the position longer. This may differ between clients; therefore, clients have her own Navigation Graph to capture the characteristics of the viewers. We provide detailed information about how a transition matrix is used for view prediction for future segments in the next section.

3.2 Trajectory Encoding

The view transition correlates with object trajectories in videos since the viewers tend to look at the more interesting parts of a video. An experiment in [20] shows that some tiles are more popular because they contain more interesting parts of the videos, which could be objects, salient parts, or vanishing points. To find these points, machine learning technologies are used with very powerful computing hardware, which is not usually found on mobile devices. The videos are stored on the server, and servers usually have better computing power than the mobile devices. Therefore, we can assume that the server can perform video analysis after it gets new video content and can encode the important paths on the Navigation Graph, so that clients receive this information immediately after they request a video segment. It helps clients to predict future views more efficiently without performing complicated video analysis.

Moreover, non-video content can also lead to head movement. For example, audio guidance can cause a viewer to move according to the instruction they hear, which cannot be captured by video contents analysis. Since there are scene changes in the videos, viewers can easily lose direction in the 360-degree video. Therefore, the video creator can encode a preferred trajectory in the Navigation Graph to allow the viewer to easily find the place they should be looking when the scene changes.

There are many different video analysis techniques that could affect the view prediction results in many different ways. Therefore, we focus on building view transition models using the Navigation Graph in this paper, and we will discuss the view prediction based on the trajectory encoding in future research.

4 VIEW PREDICTION AND RATE ADAPTATION WITH NAVIGATION GRAPH

The Navigation Graph based view transition model is useful for modeling viewer's spatial and temporal behavior to help predict future views. Prediction results directly affect the rate adaptation of tiled media. We first discuss view prediction methods using the Navigation Graph concept. Second, a utility maximized rate adaptation algorithm is provided, which uses future view information, estimated throughput, and current buffer status.

4.1 View Prediction Methods

We introduce two ways to predict future views based on the Navigation Graph, which are Single-user view prediction (SU) and Cross-user view prediction (CU).

4.1.1 Single-user View Prediction (SU). Without the help of the server, clients can perform the prediction themselves using past view transition data encoded as a Navigation Graph $\mathcal{G}_c$. It is important to understand the viewing behavior of a viewer to predict future views. Some viewers prefer to watch videos without much movement, but some viewers prefer to move their head frequently. A viewer's viewing pattern also depends on their current viewing direction. The Navigation Graph $\mathcal{G}_c$ has a recorded history of view transitions; therefore, it can give probabilistic information about future views based on past viewing behavior.

The probability that a viewer will change her head position from the current set of visible tiles $\mathbf{s}_c$ to another set of visible tiles $\mathbf{s}_m$ for $1 \leq m \leq M$ is the column vector of the $\hat{\mathbf{E}}_c$, which is defined as a vector $\mathbf{d}_1 = \mathbb{R}^{1\times M}$ whose elements are $p(s_m|\mathrm{s}_c)$, for $1 \leq m \leq M$. For example, $\hat{\mathbf{E}}_c$ is updated after the video started, and the viewer is currently (segment l) at the vertex $\mathbf{s}_1$. The first column of $\hat{\mathbf{E}}_c$ is $\mathbf{d}_1$, which describes the probabilities that the viewer will have a set of tiles $s_m, m = 1, ..., M$, in segment $l+1$. We are also interested in the future views in segments far from the current segment. To predict a set of visible tiles in segment $l+2$, we can perform a matrix multiplication $\hat{\mathbf{E}}_c \times \mathbf{d}_1$ to get $\mathbf{d}_2$. In general, we can define a k^{th} future transition probability as

$$\mathbf{d}_k = \hat{\mathbf{E}}_c^{(k-1)}\mathbf{d}_1. \qquad (9)$$

Since the vertices s_m consist of many tiles, the probability $p_{t,k}$ of needing a specific tile t in future segment k is given as

$$p_{t,k} = \sum_{\forall m,\, t\in s_m} d_k^m \qquad (10)$$

where the t is the index of tile and d_k^m indicates the m^{th} element of vector $\mathbf{d}_k$. We can generate a prediction matrix $P_s = \mathbb{R}^{T\times K}$ that

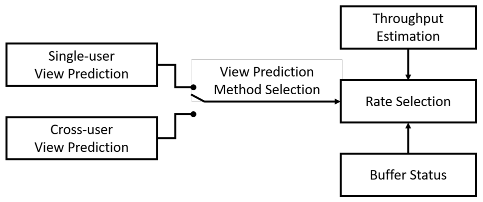

Throughput Estimation
Single-user View Prediction
View Prediction Method Selection
Rate Selection
Cross-user View Prediction
Buffer Status

Figure 5: View Prediction and Rate Selection

has elements $p_{t,k}$ using the SU, where K is the index of the farthest future segment we want to predict the view.

4.1.2 Cross-user View Prediction (CU). The Navigation Graph in the media server is updated by all prior viewers' view transitions. Information about previous viewers' view transitions is helpful for understanding how other viewers change their view in the same segment. The Navigation Graph provides the statistics for how many times other viewers move from the current view to the subsequent views. Since not all viewers behave in the same way, the Navigation Graph gives the probabilistic clues of the next view to the viewers. The probability that a viewer will change her view from the current view $\mathbf{v}_c$ to the next view $\mathbf{v}_n$ is the column vector of $\hat{\mathbf{E}}$, which is defined as $\mathbf{b}_1 = \mathbb{R}^{1\times N}$, whose elements are $p(\mathrm{v}_n|\mathrm{v}_c)$, for $1 \le n \le N$. We can also define a k^{th} transition vector $\mathbf{b}_k$, which represents the transition probabilities from the current vertex to the vertices in the future segment $l + k$; the $k = 1, ..., K$ transition happens after the current segment l as follows:

$$\mathbf{b}_k = \hat{\mathbf{E}}^{(k-1)}\mathbf{b}_1. \tag{11}$$

However, the current vertex could be a newly generated vertex if previous viewers have never seen the set of tiles in the segment, which means that there would be no outgoing connection to the vertex, so we could not get any information about the next view from the vertex. We can alternatively find the transition probability using the closest vertex in $\mathbf{V}$ from the current vertex. The closest vertex can be found by counting the number of common tiles that has the same segment index with the current vertex.

Since the vertices consist of many tiles, the probability of needing a specific tile t is given as

$$p_{t,k} = \sum_{\forall n,\, t\in \mathrm{v}_n} b_k^n \tag{12}$$

Again, we can get a prediction matrix $P_c = \mathbb{R}^{T\times K}$ that has elements $p_{t,k}$.

4.1.3 View Prediction Method Selection. Both methods (SU and CU) can contribute to the view prediction. However, the prediction accuracy of the two methods differs in different situations. We measure the prediction accuracy of segment $l+k-1$ for SU and CU, and decide which prediction method should be used for the prediction of the segment $l + k$. We run both methods every prediction cycle to evaluate which method work better and use only one prediction method at a time (SU or CU). The reason we switch between the two methods without doing a weighted sum is that the prediction method with lower precision could introduce a prediction error. The idea behind switching between the two methods is that the view prediction should be done based on the user's current behavior. If the user's current behavior is related to the video content, the CU will work better. However, if the user is not interested in the video content, SU will work better.

4.2 Utility Maximized Rate Selection

View Prediction results are given to the rate selection algorithm along with estimated throughput and current buffer status. In this section, we present an algorithm for utility maximization, where the utility is defined as the expected view quality for next k segments. The utility maximization problem is described as

$$\max U(\mathbf{q}) = \sum_{\forall t,k} u\left(q_{t,k}\right) p_{t,k} \tag{13}$$

subject to

$$R(\mathbf{q}) = \sum_{\forall t,k} r\left(q_{t,k}\right) \le R, \tag{14}$$

where $q_{t,k} \in \{1, \ldots, Q\}$ are the selected quality representations of tile t for k^{th} future segment, Q is the number of different copies of same tile with different encoding parameters, $u(q_{t,k})$ is the utility that we can achieve by allocating the $q_{t,k}$, and $r(q_{t,k})$ is the datarate that is required to receive the quality representation $q_{t,k}$. We can instead solve the easier problem of maximizing the Lagrangian as follows:

$$U(\mathbf{q}) - \lambda R(\mathbf{q}) = \sum_{\forall t,k} \left[u\left(q_{t,k}\right) p_{t,k} - \lambda r\left(q_{t,k}\right)\right] \tag{15}$$

for some $\lambda > 0$. Moreover,

$$\begin{aligned} &max_{\mathbf{q}}[U(\mathbf{q}) - \lambda R(\mathbf{q})] \\ &\quad = max_{\mathbf{q}} \sum_{\forall t,k} \left[u\left(q_{t,k}\right) p_{t,k} - \lambda r\left(q_{t,k}\right)\right] \\ &\quad = \sum_{\forall t,k} max_{\mathbf{q}} \left[u\left(q_{t,k}\right) p_{t,k} - \lambda r\left(q_{t,k}\right)\right], \end{aligned} \tag{16}$$

so the maximization problem can be solved independently for each element for all k and t. For each λ, the solution is given as [11]

$$q_{t,k} = \mathrm{argmax}_{q_{t,k}} \frac{u(q_{t,k})p_{t,k}}{r(q_{t,k})}, \tag{17}$$

which means that the clients should request video data chunks $q_{t,k}$ quality that have the largest $\frac{u(q_{t,k})p_{t,k}}{r(q_{t,k})}$ value first. We used the log of the assigned rate as a utility $u(q_{t,k}) = \log r(q_{t,k})$ in this paper.

5 PERFORMANCE EVALUATION

5.1 Video Data Set and Network Traces

Nine 360-degree videos along with 48 users' view traces [4] are used for the performance evaluation. The videos are divided into tiles, and the tiles are encoded with five different quantization parameter (QP) values (22, 27, 32, 37, and 42) using ffmpeg [3]. Clients perform the SU without help from the server, and the CU with the help of the Navigation Graph on the server. Switching between SU and CU (SU+CU) scheme is also tested to see how the two schemes help each other to improve the view prediction performance. In this paper, we consider two different network conditions. The first condition is a stable network condition in which the mean packet arrival rate is constant. If the client is connected to the Internet through a wired network and there are only a few users, the network throughput may be very stable. The second condition is a variable network in which the mean packet arrival rate changes frequently. If the client is connected to the Internet through a wireless network and there are many users within a small area competing for the same wireless network resources, the network throughput may be very unstable. A real HSDPA network trace [14] is used in this condition.

5.2 View Prediction

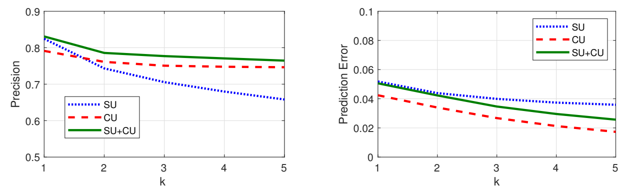
We first measure the performance of view prediction methods (SU, CU, and SU+CU) without considering the bandwidth variation. SU starts to predict future view from second segment, since there is no prior data to build the Navigation Graph in the first video segment. The Navigation Graph is very sparse at that point, but it gets larger and more accurate as more the viewing data are gathered. CU requires other viewers data, therefore, 43 users' view traces are used to train the Navigation Graph on the server, and the remaining 5 users' view traces are used to perform view predictions.

The prediction precision and the prediction error are measured. First, the precision for k^{th} future view prediction is measured as

$$\sum_{t=1}^{T} \min \left\{ p_{t,k}, g_{t,k} \right\} \tag{18}$$

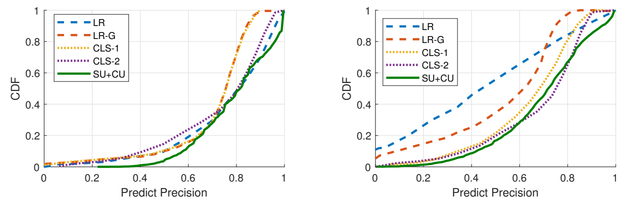
where $g_{t,k}$ is a normalized ground truth that indicates the visible tiles as 1/(number of visible tiles) and non-visible tiles as 0, and $p_{t,k}$ is a prediction probability value for tile t in the k^{th} future view. Figure 6a shows the mean precision of view prediction result for k segments, $k = 1, \ldots, 5$. SU+CU outperforms the other schemes because it can opportunistically choose the better prediction scheme. Since the uncertainty of the prediction makes the precision worse, which means that it has higher chance of requesting non-visible tiles, the entropy value (level of uncertainty) is used to select the scheme that will be used for future k segments when there is no prior knowledge about precision performance. However, after performing the first predictions, clients can get a precision value (accuracy of the prediction) for SU and CU, and the client can switch the prediction schemes between SU and CU. SU performs very well for predicting the near future ($k = 1$), but its predictive capabilities decline faster than CU with larger k because it relies on the Markov property, which means that only the current status is considered in the prediction of future views. CU shows better prediction performance for $k = 2, \ldots, 5$ future views because the Navigation Graph on the server provides the viewing history of the previous clients. The prediction precision of SU+CU is better than SU or CU. Second, the prediction errors are also measured (Figure 6b) to see whether the necessary tiles are requested on time. The prediction error is defined as the average of the number of tiles that has $p_{t,k} = 0$ but $g_{t,k} > 0$ over the total number of tiles needed to render the view. Therefore, it represents the percentage of visible blank areas in the view. The prediction errors under each condition are less than 6%. Our system tends to have lower prediction error for distant future prediction because the higher uncertainty of distant future prediction causes the Navigation Graph to request more redundant tiles.

The prediction precision of the proposed Navigation Graph based scheme (SU+CU) scheme is compared with existing solutions, which



(a) Average Precision **(b) Average Prediction Error**

Figure 6: View Prediction with the Navigation Graph



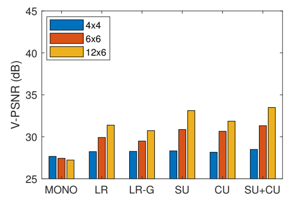
(a) View Prediction of 1 sec after current segment **(b) View Prediction of 5 sec after current segment**

Figure 7: CDFs of View Prediction Precision

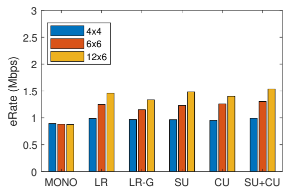
are Linear Regression (LR), Linear Regression with Gaussian distributed error (LR-G) [19], CLS-1, and CLS-2 [20]. For fair comparison with existing schemes, we test the proposed algorithm with same three videos and the same tile configuration (12×6) used in [20]. Figure 7 shows that the proposed Navigation Graph based prediction has better precision results than CLS-2, which is a state-of-the-art method. LR works well for 1 second future view prediction, but the Navigation Graph based prediction has better precision than LR. The view prediction for 5 seconds in the future with LR does not work very well because the viewer usually does not keep the same viewing pattern for 5 seconds. The Navigation Graph again shows the best precision performance in predicting the view for 5 seconds in the future. The Navigation Graph based prediction outperforms for near future prediction and distant future prediction.

5.3 Rate Selection

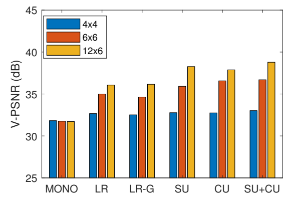
Good view prediction leads to the allocation of better video tiles to the view, and higher prediction error leads to a higher chance of seeing a blank area. However, the WBA [11][10] allows updates to existing tiles if this improves the expected view quality more than receiving new tiles. Therefore, prediction error or precision cannot fully describe the view quality. Rate selection algorithms choose the tiles' representations based on the view prediction result, the estimated network throughput, and the buffer status. We measure the average viewport PSNR (V-PSNR) and the effective rate (eRate) to see how the view prediction result and the tile configurations affect the actual user experience. V-PSNR [22] is measured by comparing pixels in the original video used to render the viewport and the pixels in the transmitted tiles used to render the viewport. eRate measures the datarate that is actually used to render the viewport. Figure 8 show the V-PSNR and the eRate for three stable network



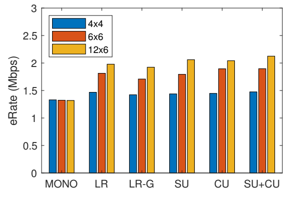
(a) V-PSNR (BW=2Mbps)



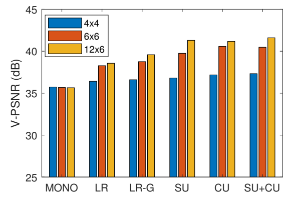
(b) Effective Rate (BW=2Mbps)



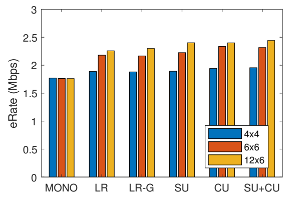
(c) V-PSNR (BW=3Mbps)



(d) Effective Rate (BW=3Mbps)

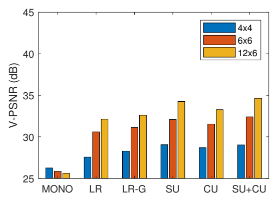


(e) V-PSNR (BW=4Mbps)

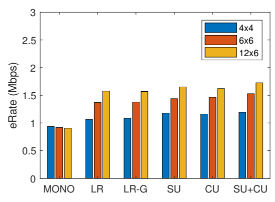


(f) Effective Rate (BW=4Mbps)

Figure 8: V-PSNR and eRate in Stable Network Conditions with Different Tile Configurations

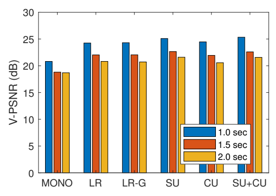


(a) V-PSNR

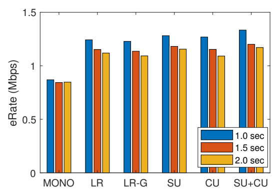


(b) Effective Rate

Figure 9: V-PSNR and eRate in Real Mobile Traces with Different Tile Configurations (Segment Duration = 1.0 sec)



(a) V-PSNR



(b) Effective Rate

Figure 10: V-PSNR and eRate in Real Mobile Traces with Different Segment Durations (Tile Configuration = 6×6)

conditions (2Mbps, 3Mbps, and 4Mbps) and a variable network condition. We test three different tile configurations (4×4, 6×6, and 12×6). With smaller tiles (12×6), we can achieve better V-PSNR and eRate, because we can remove more redundant areas and assign better representations for the visible part of the video. We can see that the Navigation Graph based prediction schemes (SU, CU, and SU+CU) outperform MONO, Linear Regression (LR) or Linear Regression with Gaussian distributed error (LR-G). MONO does not predict the view: therefore, it assumes all tiles have the same probability of being seen in the future.

Figure 9 shows the V-PSNR and eRate for the three tile configurations under six different view prediction schemes with realistic variable network traces. The Navigation Graph based prediction schemes (SU, CU and SU+CU) outperform the other schemes in terms of V-PSNR and eRate because they have better view prediction performance. SU+CU is the best, but SU also shows very similar performance to SU+CU. This means that the view prediction of the near future affects the real viewport quality more than the view of prediction of the distant future since SU can predict segments in the near future better than CU can.

The most computationally expensive part of the Navigation Graph is searching for the vertex, which takes $\log M$ where M is the number of vertices in the client-side Navigation Graph. The number of vertices depends on the number of tiles and actual user behavior. In our experiments, the actual number of vertices in the Navigation Graph range between 4-34, 13-100, and 28-262 for 4×4, 6×6 and 12×6 tile configurations respectively. Figure 10 shows the V-PSNR and eRate with different segment durations (1.0 sec, 1.5 sec and 2.0 sec). We can see that a smaller segment duration helps to deliver better video. Longer segments have a larger number of required tiles, because clients have more opportunities to change their viewport within a segment with the longer segment duration. In other words, shorter segments help to track view changes more precisely.

6 CONCLUSION

The Navigation Graph concept is introduced for tiled media streaming systems. It helps to build a model of temporal and spatial view transitions. It also supports encoding trajectories of objects or story lines as a media descriptor. This paper presents the features of the Navigation Graph and the detailed procedures for generating the Navigation Graph. View prediction and rate adaptation algorithms using the Navigation Graph are shown to improve the viewport quality of the 360-degree video streaming systems. The evaluation results show that the proposed Navigation Graph based view prediction and rate adaptation algorithms outperform other existing view prediction and tiled media rate adaptation schemes. The Navigation Graph is not limited to 360-degree video streaming systems, but it can be applied to any other tiled media, such as volumetric media streaming systems.

ACKNOWLEDGMENTS

This work is supported by the Grainger College of Engineering, Coordinated Science Laboratory and the Ralf and Catherine Fisher Gift Funding. We would like to acknowledge Dr. Michael Zink and Dr. Ramesh Sitaram from University of Massachusetts, Amherst for their initial discussions about the concept of Navigation Graphs.

REFERENCES

[1] Mathias Almquist, Viktor Almquist, Vengatanathan Krishnamoorthi, Niklas Carlsson, and Derek Eager. 2018. The Prefetch Aggressiveness Tradeoff in 360&Deg; Video Streaming. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys '18)*. ACM, New York, NY, USA, 258–269. https://doi.org/10.1145/3204949.3204970

[2] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. 1161–1170. https://doi.org/10.1109/BigData.2016.7840720

[3] F. Bellard. https://www.ffmpeg.org. (FFmpeg https://www.ffmpeg.org).

[4] W. Chenglei, T. Zhihao, W. Zhi, and Y. Shiqiang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 193–198. https://doi.org/10.1145/3083187.3083210

[5] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen. 2016. Tiling in Interactive Panoramic Video: Approaches and Evaluation. *IEEE Transactions on Multimedia* 18, 9 (Sep. 2016), 1819–1831. https://doi.org/10.1109/TMM.2016.2586304

[6] A. Ghosh, V. Aggarwal, and F. Qian. 2018. A Robust Algorithm for Tile-based 360-degree Video Streaming with Uncertain FoV Estimation. *CoRR* abs/1812.00816 (2018). arXiv:1812.00816 http://arxiv.org/abs/1812.00816

[7] M. Hosseini and V. Swaminathan. 2016. Adaptive 360 VR Video Streaming Based on MPEG-DASH SRD. In *2016 IEEE International Symposium on Multimedia (ISM)*. 407–408. https://doi.org/10.1109/ISM.2016.0093

[8] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. 2013. An Overview of Tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing* 7, 6 (Dec 2013), 969–977. https://doi.org/10.1109/JSTSP.2013.2271451

[9] J. Park. 2019. Video Streaming over the LWA Systems. In *2019 International Conference on Computing, Networking and Communications (ICNC)*. 597–601. https://doi.org/10.1109/ICCNC.2019.8685626

[10] J. Park, P. A. Chou, and J. Hwang. 2018. Volumetric Media Streaming for Augmented Reality. In *2018 IEEE Global Communications Conference (GLOBECOM)*. 1–6. https://doi.org/10.1109/GLOCOM.2018.8647537

[11] J. Park, P. A. Chou, and J. Hwang. 2019. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (March 2019), 149–162. https://doi.org/10.1109/JETCAS.2019.2898622

[12] J. Park, J. Hwang, and H. Wei. 2018. Cross-Layer Optimization for VR Video Multicast Systems. In *2018 IEEE Global Communications Conference (GLOBECOM)*. 206–212. https://doi.org/10.1109/GLOCOM.2018.8647389

[13] R. A. Patrice, M. Jean-Franois, and V. Nico. 2012. Interactive Omnidirectional Video Delivery: A Bandwidth-Effective Approach. *Bell Lab. Tech. J.* 16, 4 (March 2012), 135–147. https://doi.org/10.1002/bltj.20538

[14] R.Haakon, E. Tore, V Paul, G. Carsten, and H. Paul. 2012. Video Streaming Using a Location-based Bandwidth-lookup Service for Bitrate Planning. *ACM Trans. Multimedia Comput. Commun. Appl.* 8, 3, Article 24 (Aug. 2012), 19 pages. https://doi.org/10.1145/2240136.2240137

[15] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. 2019. A Two-Tier System for On-Demand Streaming of 360 Degree Video Over Dynamic Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (March 2019), 43–57. https://doi.org/10.1109/JETCAS.2019.2898877

[16] P. Wydrych, K. Rusek, and P. Cholda. 2013. Efficient Modelling of Traffic and Quality of Scalable Video Coding (SVC) Encoded Streams. *IEEE Communications Letters* 17, 12 (2013), 2372–2375. https://doi.org/10.1109/LCOMM.2013.110613.132163

[17] C. Xavier, D. Francesca, and S. Gwendal. 2017. 360Degreee Video Head Movement Dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. ACM, New York, NY, USA, 199–204. https://doi.org/10.1145/3083187.3083215

[18] M. Xiao, C. Zhou, V. Swaminathan, Y. Liu, and S. Chen. 2018. BAS-360 : Exploring Spatial and Temporal Adaptability in 360-degree Videos over HTTP/2. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 953–961. https://doi.org/10.1109/INFOCOM.2018.8486390

[19] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo. 2017. 360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming. In *Proceedings of the 25th ACM International Conference on Multimedia (MM '17)*. ACM, New York, NY, USA, 315–323. https://doi.org/10.1145/3123266.3123291

[20] L. Xie, X. Zhang, and Z. Guo. 2018. CLS: A Cross-user Learning Based System for Improving QoE in 360-degree Video Adaptive Streaming. In *Proceedings of the 26th ACM International Conference on Multimedia (MM '18)*. ACM, New York, NY, USA, 564–572. https://doi.org/10.1145/3240508.3240556

[21] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang. 2018. Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018), 1–1. https://doi.org/10.1109/TPAMI.2018.2858783

[22] M. Yu, H. Lakshman, and B. Girod. 2015. A Framework to Evaluate Omnidirectional Video Coding Schemes. In *2015 IEEE International Symposium on Mixed and Augmented Reality*. 31–36. https://doi.org/10.1109/ISMAR.2015.12