Jiang, Xiaolan; Naas, Si Ahmed; Chiang, Yi-Han; Sigg, Stephan; Ji, Yusheng

SVP: Sinusoidal Viewport Prediction for 360-Degree Video Streaming

Published in:
IEEE Access

DOI:
10.1109/ACCESS.2020.3022062

Published: 01/01/2020

Document Version
Publisher's PDF, also known as Version of record

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 20, 2020, accepted August 27, 2020, date of publication September 7, 2020, date of current version September 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3022062

SVP: Sinusoidal Viewport Prediction for 360-Degree Video Streaming

XIAOLAN JIANG[1,2], SI AHMED NAAS[3], (Student Member, IEEE), YI-HAN CHIANG[4], (Member, IEEE), STEPHAN SIGG[3], (Member, IEEE), AND YUSHENG JI[2,1], (Senior Member, IEEE)

[1]Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo 101-8430, Japan
[2]Information Systems Architecture Research Division, National Institute of Informatics (NII), Tokyo 101-8430, Japan
[3]Department of Communications and Networking, Aalto University, 02150 Espoo, Finland
[4]Department of Electrical and Information Systems, Osaka Prefecture University, Osaka 599-8531, Japan

Corresponding author: Yi-Han Chiang (chiang@eis.osakafu-u.ac.jp)

This work was supported in part by the JSPS KAKENHI under Grant Number JP20H00592 and in part by the NII International Internship Program.

ABSTRACT The rapid growth of user expectations and network technologies has proliferated the service needs of 360-degree video streaming. In the light of the unprecedented bitrates required to deliver entire 360-degree videos, tile-based streaming, which associates viewport and non-viewport tiles with different qualities, has emerged as a promising way to facilitate 360-degree video streaming in practice. Existing work on viewport prediction primarily targets prediction accuracy, which potentially gives rise to excessive computational overhead and latency. In this paper, we propose a sinusoidal viewport prediction (SVP) system for 360-degree video streaming to overcome the aforementioned issues. In particular, the SVP system leverages 1) sinusoidal values of rotation angles to predict orientation, 2) the relationship between prediction errors, prediction time window and head movement velocities to improve the prediction accuracy, and 3) the normalized viewing probabilities of tiles to further improve adaptive bitrate (ABR) streaming performance. To evaluate the performance of the SVP system, we conduct extensive simulations based on real-world datasets. Simulation results demonstrate that the SVP system outperforms state-of-the-art schemes under various buffer thresholds and bandwidth settings in terms of viewport prediction accuracy and video quality, revealing its applicability to both live and video-on-demand streaming in practical scenarios.

INDEX TERMS 360-degree video, viewport prediction, live streaming, video on demand.

I. INTRODUCTION

The ever-increasing demand of immersive multimedia experience has stimulated content providers (e.g. YouTube and Vimeo) to roll out 360-degree video streaming. To fully embrace the panoramic and high-resolution multimedia experience, the unavailability of network bandwidth is an unsolved challenge. Streaming 360-degree videos requires unprecedentedly high bitrates when compared to video streaming of fixed viewing directions. Therefore, the ability to adaptively stream 360-degree videos in accordance with the dynamics of network bandwidth is decisive for its wide spread.

Commercialized 360-degree video streaming services mostly deliver entire 360-degree videos at constant quality. Since a user focuses on the so-called field of view (FoV) or *viewport* of a sphere at a time, delivering the entirety of 360-degree video results in a waste of network bandwidth. Tile-based streaming [1] [2]–[5] can resolve this issue by partitioning temporal video segments as spatially independent tiles and then associating viewport and non-viewport tiles with different qualities. Network bandwidth can thus be more efficiently utilized to facilitate 360-degree video streaming.

Smooth 360-degree video streaming requires a certain amount of video segments buffered for continuous playback. Existing solutions [4]–[7], [9], [10], [14]–[16] suggest to pre-fetch all tiles of each segment, with tiles in the predicted viewport pre-fetched at a higher quality. Viewport prediction algorithms can be categorized into trajectory- and content-based methods as follows.

- Trajectory-based methods [4]–[13]: existing methods predict future viewport based on a trajectory of either his own (single-user) or other users' (cross-user)

The associate editor coordinating the review of this manuscript and approving it for publication was P. K. Gupta.

[1]In practice, tile-based streaming has been standardized as part of dynamic adaptive streaming over HTTP (DASH) [1].

historical rotations. Single-user methods predict future viewport based on user's past head rotation trajectory. Cross-user methods assume that different users have similar viewing behaviors on the same video, and determine the tile viewing probabilities of each individual user based on historical fixations of other users who have watched the same video. However, a trajectory-based prediction could be inaccurate due to its angle periodicity (e.g. -180° and 180° indicate the same direction). Moreover, cross-user algorithms cannot be deployed in live streaming scenarios which, by definition, lack historical information (see FIGURE 1). In addition, accuracy degrades with diversity in interest (viewing angle) for cross-user methods.

- Content-based methods [14]–[22]: existing algorithms typically use a saliency map [23] or a flow net [24] to extract image-features from the video content first, and then use a complicated model to predict future viewport with image-features and past rotations. Although content-based methods may yield a higher accuracy, the computational overhead of these algorithms exceeds the resources in practical deployments on mobile devices. Even if content-based algorithms can be deployed on the server, they may encounter scalability issues when there are millions of users watching millions of videos at the same time. In addition, VR video streaming systems already consume much computational power on encoding/decoding and rendering the 360 videos, a heavy viewport prediction model may make the whole experience worse than a lightweight model.

To address the above trajectory-based (inaccuracy and inapplicability to live streaming) and content-based (excessive computational overhead) concerns, we are motivated to improve the trajectory-based single-user method to build up a practical and accurate viewport prediction system applicable to both video on demand (VoD) and live streaming scenarios. To this end, we conduct data analysis on several datasets and find out that the angle periodicity issue on *yaw* (i.e. horizontal) direction can be avoided by converting degrees to the corresponding sinusoidal values.

Motivated by this observation, we design the sinusoidal viewport prediction (SVP) system for 360-degree video streaming in three stages: 1) orientation prediction, 2) error handling, and 3) tile probability normalization. First, we use linear regression (LR) least square method to predict the sinusoidal values of rotation angles. Next, we train a linear support vector regression (LinSVR) model to estimate the potential prediction errors to virtually enlarge the viewport area to cover more actual viewport tiles. Finally, we calculate the normalized viewing probability of tiles to improve adaptive bitrate (ABR) streaming performance. Overall, the contributions of this paper are:

- We identify that using sinusoidal values of rotation angles on *yaw* direction can improve the smoothness and linearity, thereby reducing prediction errors.

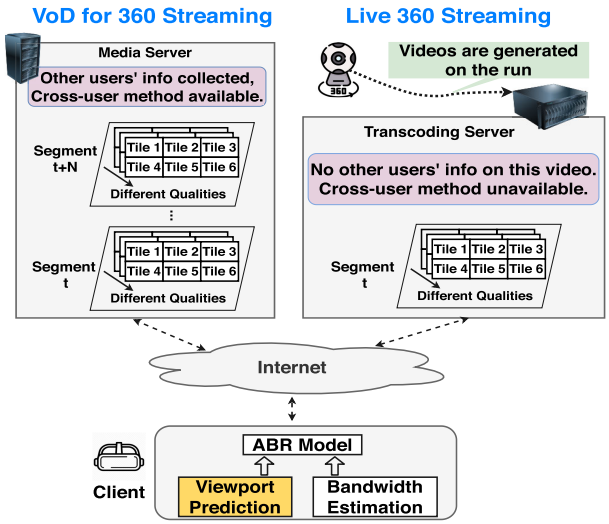


FIGURE 1. An illustration of tile-based streaming for 360-degree videos in VoD and live scenarios.

- We further improve the prediction accuracy by observing that head movement velocity and prediction time window positively correlate with prediction errors.
- We normalize viewing probabilities of tiles to improve the streaming performance of ABR.

The rest of the paper is organized as follows. Sec. II introduces the basic knowledge of tile-based streaming and discusses related work on viewport prediction. In Sec. III, we conduct data engineering. Then, we introduce the proposed viewport prediction system in Sec. IV. Performance evaluation and comparison are presented in Sec. V. Finally, we conclude the paper in Sec. VI.

II. BACKGROUND AND RELATED WORK

A. TILE-BASED 360-DEGREE VIDEO STREAMING

In traditional HTTP-based adaptive streaming, a video is temporally partitioned into segments. Video segments are further spatially divided into tiles, so that each temporal segment is composed of several spatial tiles, to support viewport-adaptive streaming (FIGURE 1). Since the client needs to buffer a number of video segments to ensure continuous playback, it is necessary to pre-fetch video segments according to the results of viewport prediction, the aim of which is to determine which tiles cover the user's future viewport in a deterministic or probabilistic manner. Then, bitrate selection is used to select the quality level of tiles based on the predicted viewport and estimated available bandwidth. Whenever video segments are downloaded, the 360-degree video is rendered onto the screen of a head mounted display (HMD) using graphic engines.

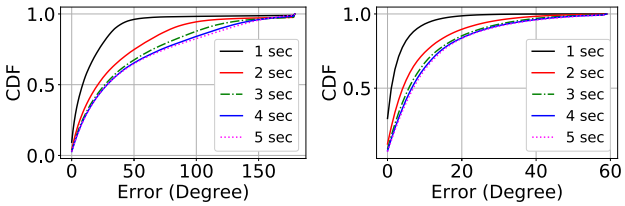
B. VIEWPORT PREDICTION

Viewport prediction is an important technique that can be utilized by various applications, e.g. augmented/virtual reality (AR/VR) and volumetric video streaming [25], [26]. Viewport prediction can be categorized into two classes: trajectory- and content-based methods. Trajectory-based

methods predict future viewport either with user's own (single-user) or other users' (cross-user) historical head rotations. Single-user methods usually estimate the user's future head rotations by solving a regression problem. Qian *et al.* [4] use logistic regression to predict future viewport. Xie *et al.* [5] propose a probabilistic model assuming the prediction error follows a Gaussian distribution. Jiang *et al.* [6] apply a long short term memory (LSTM) based model to predict future head rotations. Zhang *et al.* [7] utilize an ensemble of three LSTM models to further improve the prediction accuracy. However, all these methods are not accurate in the *yaw* direction due to the periodicity issue. Heyse *et al.* [8] propose a contextual bandit based approach to estimate user's future viewport. The accuracy of this method can be lower than those regression-based methods, since it does not make use of historical information.

Cross-user methods assume that users have similar region-of-interest (ROI) when watching the same video, and henceit is possible to exploit multi-users' ROI behavior to predict viewport. Xie *et al.* [9] group users with density-based spatial clustering of applications with noise (DBSCAN) in the server, then on the client end, classify the user to the corresponding cluster with a support vector machine (SVM) classifier, and finally obtain the viewing probability from the cluster. Ban *et al.* [10] first predict future fixations with LR, then utilize K-Nearest-Neighbor (KNN) to find the K nearest fixations of other users around the LR result to improve accuracy. Petrangeli *et al.* [11] first identify user clusters with a kind of spectral clustering algorithm, then fit a regression model for each cluster, finally predict with the regression model from the user's corresponding cluster. Li *et al.* [12] utilize LSTM and Attentive Mixture Experts (AME) techniques to train a model to predict based on both the target user's historical fixations and other users' fixations. Nasrabadi *et al.* [13] first cluster users based on their quaternion rotations, and then classify the target user to the corresponding cluster and estimate the future fixation as the cluster center. If no available cluster for the target user, the last sample will be used as the future viewport. Although these methods can have relatively high accuracy in the long term, it cannot be deployed in a live streaming scenario where no other users have watched the same video before. Moreover, cross-user methods can perform worse than single-user methods in a short-term prediction.

To improve the prediction accuracy, content-based methods utilize both rotations and video contents as features. Zhang *et al.* [15] propose a generative adversarial network (GAN) to generate multiple future frames conditioned on the single current frame and then anticipate the corresponding future gazes in the upcoming few seconds. Xu *et al.* [14] propose a deep reinforcement learning (DRL) based approach to better model the users' attention with video contents. The works in [16]–[19] design a hybrid architecture of CNN and LSTM models. They use a convolutional neural network (CNN) to extract video content features from saliency maps or original images, and use LSTM to extract motion



(a) on *yaw* direction (b) on *pitch* direction

FIGURE 2. The prediction errors by LR (see Sec. V-A5) with respect to various time window lengths on the AV dataset (see Sec. V-A2).

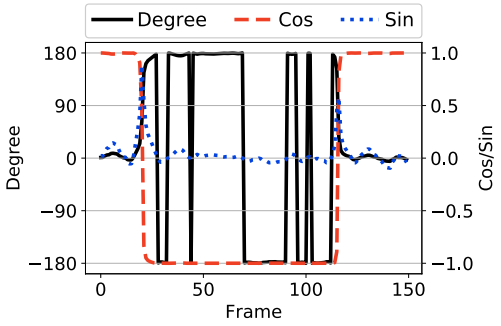


FIGURE 3. Head movement trace on *yaw* direction. The angle is represented by degree, and cosine/sine of the angle.

patterns from history rotations. Then, they predict the future viewport based on CNN and LSTM features. Feng *et al.* [20] utilize optical flow and Gaussian mixture model (GMM) for motion detection and feature tracking, then predict user's future viewport by leveraging a dynamic user interest model. Feng *et al.* [21] leverage CNN based model to predict future viewport in live streaming by modifying the training/testing process and the workflow of the CNN application. To further improve the prediction accuracy, Feng *et al.* [22] employ a hybrid deep learning model which involves both CNN and LSTM models. All these methods would have a burden to the practical deployment in a real system since they consume excessive computing resources.

III. SINUSOIDS VERSUS PREDICTION ACCURACY

Viewport prediction is critical in implementing tile-based 360 video streaming. Understanding why current models cannot predict future orientation accurately is important in order to design a better model. We conduct a study to understand the origin of the orientation prediction error, and propose a method to improve prediction accuracy.

A. PREDICTION ERROR ANALYSIS

FIGURE 2 shows the CDF of the prediction errors of both *yaw* and *pitch* for 1-5 secs. Prediction errors on the *yaw* axis are larger than towards *pitch* direction. One intuitive reason is that people move more in horizontal direction and remain steady in the vertical direction. Larger movement results in larger prediction error. Another reason is the angle periodicity issue (where $-180° = 180°$) in *yaw* direction. As shown in FIGURE 3, the black line (motion curve represented in degrees) shows a significant change when the head moves

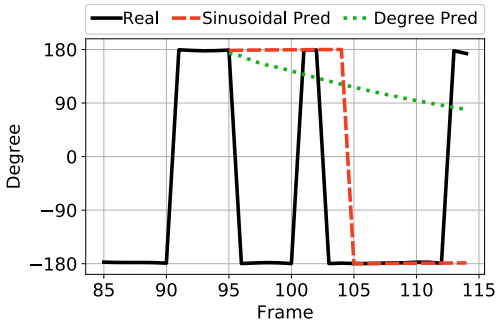


FIGURE 4. An illustration of predicted values of LR models trained with both the degrees and the sinusoidal values.

only a little, from slightly smaller than 180° to slightly larger than −180° or inversely. A model trained by degrees may be inaccurate due to such a problem caused by the periodicity.

B. CONVERSION OF DEGREES TO SINUSOID

Representing angles with Cosine and Sine avoids the periodicity issue. In FIGURE 3, observe that while user's head is moving between 180° and −180°, the cosine (red dashed line) and sine (blue dotted line) of the angle remain stable.

FIGURE 4 shows the predicted values of LR models trained with both the degrees and the sinusoidal values. Here, we use Frame 85 to 95 as the input to predict the future *yaw* degrees in the next 20 frames. We can find that model trained with sinusoidal values (red dashed line) can avoid the periodicity issue and get much less error than the model trained with degrees (green dotted line).

C. WHY SINUSOID IS BETTER?

To quantitatively explain why converting degree to sinusoidal values leads to a more accurate linear model, a good way is to represent the data trace properties of the degree and sinusoidal values by numerical metrics. In the light of the fact that a smooth and linear curve can be more easily learned, we consider the smoothness and the linearity[2] as the metrics to measure the data trace.

For a data trace $X = [x_1, \ldots, x_N]$, to obtain smoothness, we first derive the difference trace as $D = [d_1, \ldots, d_{N-1}]$; $d_i = x_{i+1} - x_i$; $\bar{d} = \frac{1}{N-1}\sum_{i=1}^{N-1} d_i$. We define the smoothness as the inverse of the standard deviation of D.

$$\text{Smoothness}(X) = \frac{1}{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}\left(d_i - \bar{d}\right)}}. \tag{1}$$

We use the absolute value of Pearson correlation coefficient between X and a linear vector $Y = [y_1, \ldots, y_i, \ldots, y_N]$ as the linearity of X, that calculated as:

$$\text{Linearity}(X) = \frac{\sum_{i=1}^{N} |(x_i - \bar{x})(y_i - \bar{y})|}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}, \tag{2}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

[2]In addition to the smoothness and linearity metrics, there may be other representative indicators (e.g. by learning latent patterns from sinusoidal values) for non-linear models, which will be left to our future works.

TABLE 1. Smoothness and linearity of degree, cosine and sine on *yaw* direction of the head motion datasets (see Sec. V-A2) where Y is set to $[1, 2, \ldots, N]$.

Dataset	Metric	Degree	Cos	Sin
AV	Smoothness	10.34	21.21	20.04
	Linearity	0.069	0.197	0.115
THU	Smoothness	5.61	7.86	14.06
	Linearity	0.037	0.065	0.081
UT	Smoothness	7.27	8.30	13.98
	Linearity	0.036	0.087	0.0045

A larger absolute coefficient value relates to higher linearity, and a larger smoothness value corresponds to a smoother curve. TABLE 1 shows the smoothness and linearity of normalized degree, cosine, and sine on three head motion datasets. By representing degree with cosine and sine, the smoothness of data traces can increase to 99.5%, 95.4%, and 53.2%, while the linearity can improve to 126.1%, 97.3%, and 27.1%, for the AV, THU and UT datasets, respectively.

IV. THE SVP SYSTEM DESIGN

To incorporate the concept of sinusoidal viewport prediction into 360-degree video adaptive streaming, we propose the SVP system (as depicted in FIGURE 5) which consists of three stages: orientation prediction, error handling, and tile probability normalization. The SVP system first predicts future head rotations for each frame based on the collected historical head motion traces. Considering potential viewport prediction errors, the error handling module then divides tiles into viewport (VP), adjacent (AD) and outside (OUT) areas. Since the orientation prediction error is dependent on prediction time window length[3] and head movement velocity, we train a LinSVR [28] model to predict the orientation prediction error, thereby determining the size of adjacent areas. For each frame, we set a tile viewing probability for each tile, where VP tiles have the highest probability and OUT tiles get the lowest one. In the final stage, we normalize the tile probabilities in the next segment (containing M video frames) based on the viewing probabilities of tiles in the M frames. In the following, we are going to give more detailed descriptions of these three stages.

A. ORIENTATION PREDICTION

The orientation prediction process is depicted on the left part in FIGURE 5. After collecting head rotations $[yaw_{t-N+1}, \ldots, yaw_t]$ and $[pitch_{t-N+1}, \ldots, pitch_t]$ of N past video frames, we convert them to cosine and sine vectors and then we apply linear regression (Least Square Method) to predict the cosine and sine values of both *yaw* and *pitch* in the next frame. Here, the *yaw* degree can be calculated by means of the atan2[4] function as:

$$yaw_{t+1} = \text{atan2}(\sin(y_{t+1}), \cos(y_{t+1})), \tag{3}$$

which is similar for calculating the *pitch* degree.

[3]The prediction time window length refers to the duration of playback time of video segments prefetched in the buffer.

[4]https://www.mathworks.com/help/matlab/ref/atan2.html

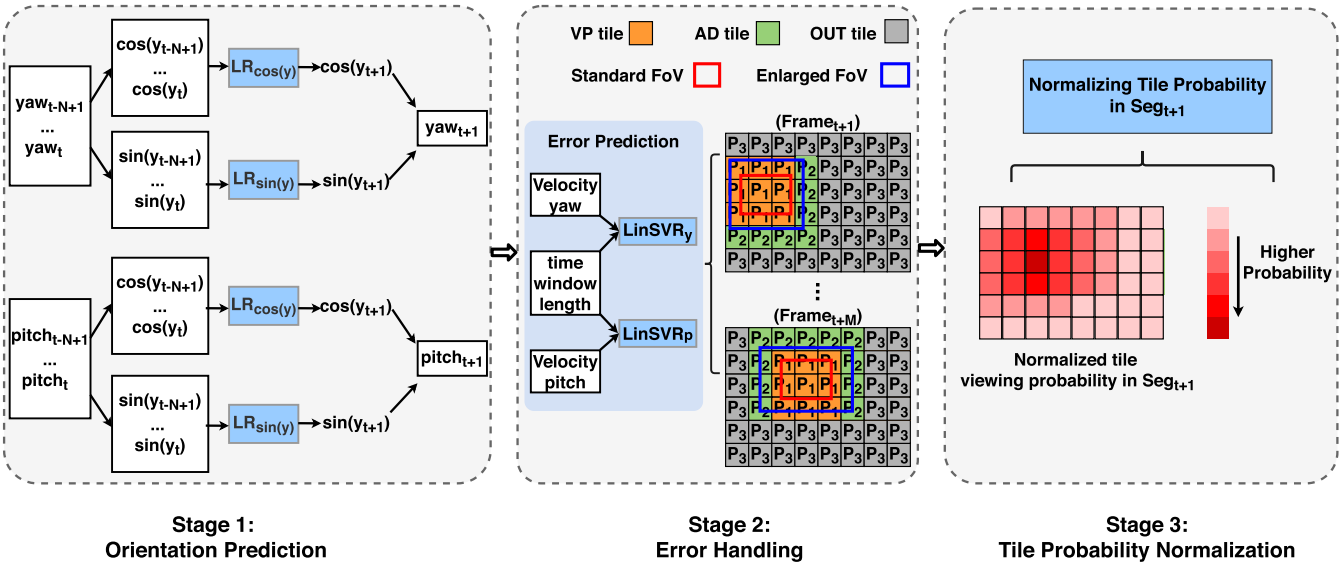


FIGURE 5. An illustration of the SVP system, which consists of three stages: orientation prediction, error handling, and tile probability normalization. The SVP system predicts based on head rotations collected by an HMD, and finally outputs viewing probability for each tile to the ABR model (corresponding to FIGURE 1).

Although this model can only be used to predict the future orientation of one video frame from now, we can easily predict rotations in the next M frames by taking the last prediction results as inputs. Then, we can determine the viewport areas according to the collected orientations in one segment length. Even though this method can predict at relatively high accuracy, prediction errors may also take place, especially when the prediction time window is too long or the user's head movement is too fast. Therefore, in the following, we will introduce how we handle the potential prediction errors on both *yaw* and *pitch* directions.

B. ERROR HANDLING

We handle the potential prediction errors of head orientations by virtually enlarging the viewport range. In practice, the angles on a spherical surface can be defined by *yaw* and *pitch* values (in degrees), both of which are discretized and mapped to pixels on a rectangular image in accordance with

$$x = \frac{width \cdot (yaw + 180)}{360}, \quad y = \frac{height \cdot (90 - pitch)}{180}, \tag{4}$$

where *width* and *height* refer to the dimension of the equirectangular image, whose center is located at $(yaw, pitch) = (0, 0)$, and the *pitch* angle increases in the upward direction.

The middle part on FIGURE 5 illustrates the three areas: VP area (in yellow), AD area (in green), and OUT area (in gray). Consider that a typical setting of vertical and horizontal FoVs (i.e. red rectangles) are 90 and 110 degrees [4], [6], [27], respectively. We first mark the tiles covered by the actual FoV as the VP area. To handle the potential errors, we mark the tiles covered by the enlarged FoVs (i.e. blue rectangles) from 90 to $90 + V$ degrees on *pitch* direction, and from 110 to $110 + H$ on *yaw* direction as the AD area. Then, all the remaining tiles will be viewed as OUT areas It is noted that the expansion is symmetrical (i.e., $V/2$ and $H/2$ on both positive and negative directions of *yaw* and *pitch*, respectively). However, the AD areas may be asymmetrical with respect to the corresponding VP areas, which may happen whenever a standard FoV is not centered at the central point of a VP area.

In our previous work [6], H and V have been naively set to fixed values. However, in this case, the enlarged FoV may not cover the actual viewport or may cover too many tiles outside the viewport. To overcome this issue, a good way is to estimate prediction errors, thereby adaptively determining the sizes of AD and OUT areas. In particular, by testing the prediction model on a real-world dataset (see FIGURE 6), we find that the prediction error is highly positively correlated with prediction window length and current head movement speed. For both *yaw* and *pitch* directions, the prediction errors increase when the prediction window length is longer or the motion velocity is larger. Based on this observation, we use the LinSVR model to estimate the prediction error H (*yaw*) and V (*pitch*) based on head movement velocity and time window length (as depicted in Stage 2 of FIGURE 5).

Since we are trying to assign higher quality for viewport and adjacent areas and lower quality for outside area, we set the viewing probability for tiles in each area of per frame as follows: p_1 for VP tiles, p_2 for AD tiles, and p_3 for OUT tiles, with $p_1 > p_2 > p_3$. Next, we will introduce how to get the normalized tile viewing probability in one video segment which contains M frames.

C. TILE PROBABILITY NORMALIZATION

After error handling, the viewing probability of each tile in the next video segment is determined based on the tile probs in each frame. The right part in FIGURE 5 illustrates the viewing probability of one video segment in a 8×6 tilling manner as an example. We denote the viewing probability for the i-th tile in the k-th frame as v_i^k. Therefore, the total viewing probability for the i-th tile in the next

video segment is calculated as:

$$\bar{v}_i = \sum_{k=1}^{M} v_i^k, \tag{5}$$

based on which the normalized viewing probability of the i-th tile can be expressed as (e.g. [9])

$$p_i = \frac{\bar{v}_i}{\sum_{i=1}^{M} \bar{v}_i}. \tag{6}$$

V. PERFORMANCE EVALUATION

In this section, we evaluate the proposed viewport prediction system under various head motion datasets in terms of prediction error and accuracy. Then, we show how the proposed system can improve video quality under both fixed and dynamic bandwidth settings.

A. SIMULATION SETTINGS

1) EXPERIMENTAL SETUP

To validate user perceived quality under the proposed SVP system, we utilize a trace-driven emulation environment where a large corpus of network traces and viewport motion traces can be tested [6]. For each prediction, 10 past frames' rotations are used. In addition, the exact tile probabilities p_1, p_2 and p_3 for VP, AD and OUT tiles, respectively, should be obtained according to the design of ABR algorithms. In our experiments, we set p_1, p_2 and p_3 to 1.0, 0.5, 0.0, respectively, for simplicity.

2) VIEWPORT MOTION DATASETS

We test the performance on three open viewport motion datasets. Different datasets are collected by different hardware, software and group of people and videos.

- AV [29]: 48 users watching 20 different entertaining omnidirectional videos on an HTC Vive HMD.
- THU [30]: A head tracking dataset composed of 48 users (24 males and 24 females) watching 18 sphere videos from 5 categories of contents.
- UT [31]: The authors gathered and produced 28 videos based on the taxonomy, and recorded viewport traces from 60 participants watching the videos.

As to the training and testing in our performance evaluation, we first evenly sample 10 frames per segment for all three datasets. To evaluate **SVP**'s performance on unseen videos and users, we select 70% videos in the AV dataset for training, and remaining 30% videos in the AV dataset and all videos in the dataset of THU and UT for testing. In the experiments of comparison scheme **CLS**, 90% of the users are randomly selected in each video for training, while the remaining 10% of users are used for testing.

3) NETWORK TRACE DATASET

The test bandwidth dataset we use contains the 4G networks traces along several routes in and around the city of Ghent, Belgium [32]. All bandwidth logs are recorded based on the types of transportation: car, train, tram, bus, bicycle and foot. A total of 40 logs were collected, covering a 5-hour active monitoring.

4) VIDEO DATASET

A 360-degree video with a 237-sec duration [33] is employed for our simulations. The raw 3840 × 2016 video is extracted from the original clip and re-encoded using the kvazaar encoder [34], which is an open-source software for H.265. The video is available in a 1-sec segment version, tiled as 12 × 6 for each segment. According to the recommended settings [35], we consider six levels of qualities, corresponding to the bitrates of {1, 2.5, 5, 8, 16, 40} Mbps.

5) COMPARISON SCHEMES

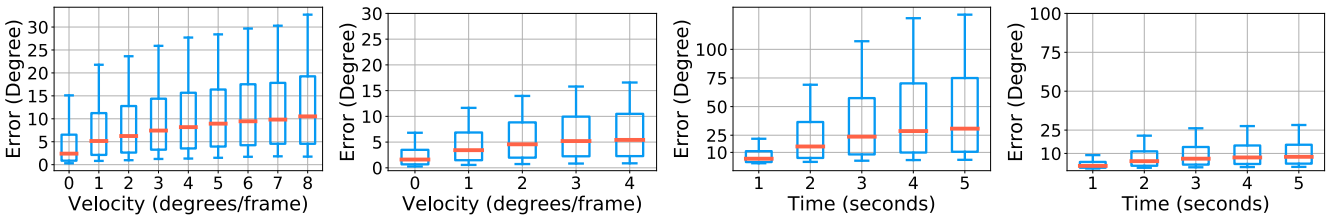
- **LR-G** [36]: Linear regression model trained by gradient descent method.
- **LSTM** [6]: Long short term memory neural network, which is commonly used for time series prediction.
- **MLP** [37]: Multi-layer perception neural network.
- **LR** [4]: This tile-based streaming uses linear regression (Least Square Method) to predict future viewport, and then divides the tiles into three areas: viewport, adjacent, and outside areas. Each tile in the same area will be assigned the same quality [38].
- **CLS** [9]: group users with density-based spatial clustering of applications with noise DBSCAN) in the server, then on the client end, classify the user to the corresponding cluster with a support vector machine (SVM) classifier, and finally obtain the viewing probability from the cluster.
- **TJ** [11]: first identify user clusters with a kind of spectral clustering algorithm, fit a regression model for each cluster, and then predict with the regression model from the user's corresponding cluster.
- **CUB** [10]: first predict future fixations with LR, then utilize K-Nearest-Neighbor (KNN) to find the K nearest fixations of other users around the LR result to improve accuracy.
- **LC** [13]: first cluster users based on their quaternion rotations, classify the target user to the corresponding cluster and then estimate the future fixation as the cluster center. If no available cluster for the target user, then predict future fixation as the last sample.

B. VIEWPORT PREDICTION ACCURACY

1) PREDICTION ACCURACY OF ORIENTATION

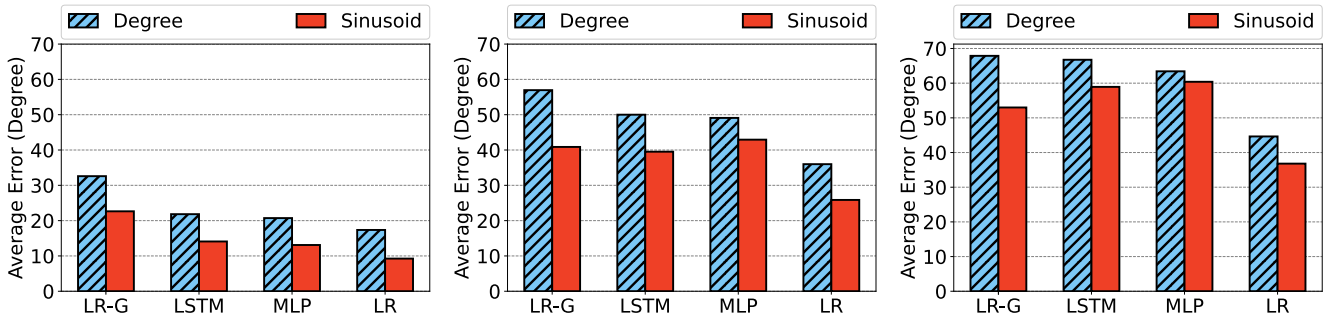
Here, we show how the prediction performance is improved by replacing degrees with their sinusoidal values on both *yaw* and *pitch* directions.

Prediction errors on yaw direction. FIGURE 7 shows the mean prediction errors in *yaw* direction for 1- to 5-sec prediction time windows. First, prediction errors of all methods with sinusoidal values (i.e. in red bars) are lower than that with degrees (i.e. in blue bars) for all prediction time windows. In particular, we see that the prediction errors of **LR-G**, **LSTM**, **MLP** and **LR** with sinusoidal values (i.e. in red bars) can be reduced by 31%∼47%, 13%∼28%, and 5%∼22%, in FIGURE 7a, 7b and 7c, respectively. This is because sinusoidal values can effectively avoid the periodicity issue



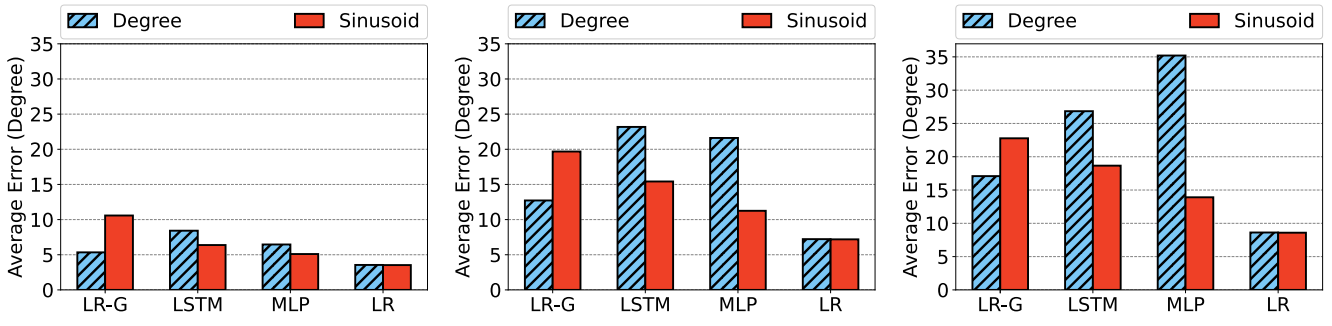
(a) Velocity-Error (*yaw*) (b) Velocity-Error (*pitch*) (c) Time-Error (*yaw*) (d) Time-Error (*pitch*)

FIGURE 6. **The relationship between prediction errors, time window lengths and head movement velocities in the AV dataset.**



(a) 1-sec time window (b) 3-sec time window (c) 5-sec time window

FIGURE 7. **The prediction errors on *yaw* direction in 1-, 3-, 5-sec time window.**



(a) 1-sec time window (b) 3-sec time window (c) 5-sec time window

FIGURE 8. **The prediction errors on *pitch* direction in 1-, 3-, 5-sec time window.**

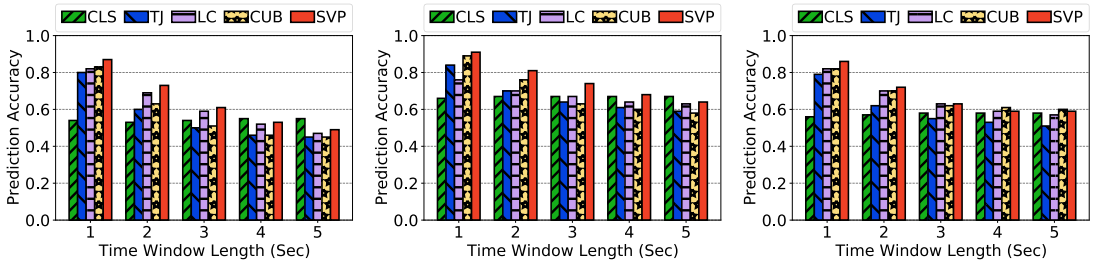
on *yaw* direction. In addition, LR can always achieve the least prediction errors, which motivates us to design the orientation prediction part of the SVP system based on linear regression as in LR.

Prediction errors on pitch direction. Prediction errors on *pitch* direction are shown on FIGURE 8. Since there is no periodicity issue on *pitch* direction, the smoothness and linearity of the sinusoidal data traces are not effectively changed, we cannot anticipate the performance improvement for all comparison schemes. Even though LR-G with sinusoidal values become worse than that with degrees due to its weak learning ability, we do observe that LR with sinusoidal values achieves the least prediction errors. Therefore, we utilize LR in Stage 1 of the SVP system. Note that the prediction errors on *yaw* direction are more pronounced (than *pitch* direction), which requires more accurate viewport prediction designs in practice.

2) PREDICTION ACCURACY OF TILES

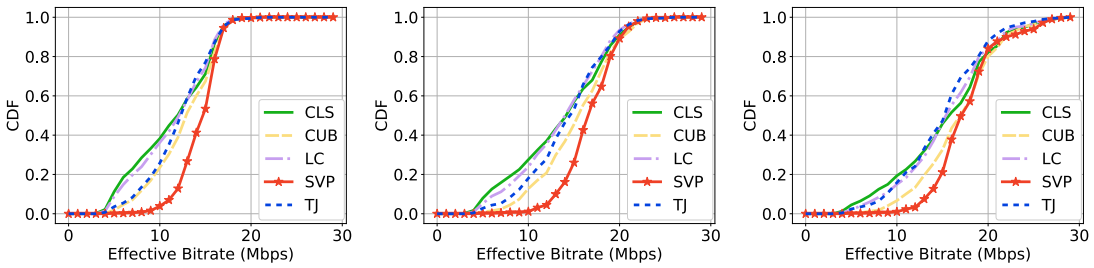
We evaluate the tile prediction accuracy here and define tile prediction accuracy as the percentage of actual viewport tiles that are predicted as viewport tiles. As LR-G, LSTM and MLP yield much higher prediction errors than LR, we simply compare SVP with CLS, CUB, LC, and TJ in the following.

Prediction accuracy on different datasets. FIGURE 9 shows the average accuracy of these methods on three datasets in 1-, 2-, 3-, 4-, and 5-sec prediction time windows. Obviously, we see that, on all datasets, SVP outperforms CLS, CUB, LC and TJ in small time window lengths (i.e. 1∼3 sec) thanks to its accurate prediction of orientation. In larger time windows (i.e. 4∼5 sec), the tile prediction accuracy of SVP reduces due to the accumulated prediction errors of orientation. This is because inaccurate viewport prediction of the previous frames inevitably leads to larger prediction errors of the next frame. Nevertheless, SVP remains comparable with



(a) The AV dataset (b) The THU dataset (c) The UT dataset

FIGURE 9. Tile prediction accuracy on different datasets.



(a) Bandwidth = 30Mbps (b) Bandwidth = 35Mbps (c) Bandwidth = 40Mbps

FIGURE 10. Effective bitrate comparison (fixed bandwidth, 3-sec buffer threshold).

respect to the cross-user methods **CLS**, **CUB**, **LC** and **TJ**. Note that the tile prediction accuracy of **CLS** is insensitive to time window lengths since **CLS** does not rely on previous frames for predicting the next frame.

C. VIDEO QUALITY ASSESSMENT

To evaluate the proposed SVP system, we design a target buffer-based bitrate selection algorithm inspired by [38]. The bitrate selection algorithm operates in four steps. First, it calculates the total bitrate budget BR for the next segment. Then, to avoid blank blocks in the user's viewport, the lowest quality l is assigned to all the tiles initially. Next, it updates the available bitrate budget BR' as the difference between the total bitrate budget BR and the sum bitrate allocated to all tiles. Finally, it assigns the highest possible quality q to the tile with highest viewing probability under the constraint of BR', and then updates BR' by $BR' - (R_q - R_l)$, where R_q and R_l represent the tile sizes of the selected highest possible quality and that of the initially allocated lowest quality, respectively. The final step repeats for all the remaining tiles.

In 360-degree video streaming, viewport prediction influences the bitrate allocation of tiles and perceptual quality of contents in viewport. Since only a portion of each video is viewed by the user, the perceptual quality is decided by the tiles in the viewport. Therefore, we define the performance metric of *effective bitrate* as the sum bitrates of the tiles in the viewport.

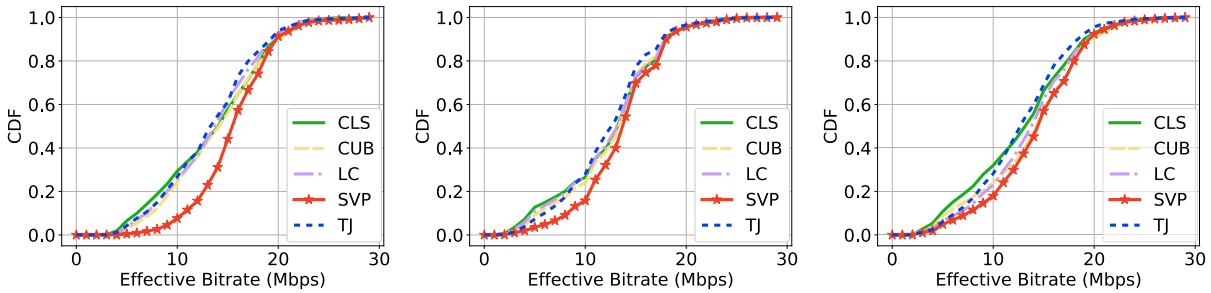
1) FIXED BANDWIDTH SETTINGS

Effective bitrates under fixed bandwidth settings. FIGURE 10 shows the CDF of an effective bitrate of different methods during streaming sessions for all three viewport motion trace datasets. Apparently, **SVP** yields higher effective bitrate than **CLS, TJ, LC and CUB** for all bandwidth settings as it achieves higher tile prediction accuracy. In particular, at the bandwidth setting of 30Mbps, **SVP** outperforms **CLS, CUB, LC, and TJ** by around 25.4%, 12.5%, 23.6% and 17.7%, respectively; at the bandwidth setting of 40Mbps, **SVP** outperforms **CLS, CUB, LC, and TJ** by around 13.6%, 4.0%, 12.9% and 15.7%, respectively. That is, the performance of **SVP** is more pronounced when the available bandwidth goes insufficient.

2) DYNAMIC BANDWIDTH SETTINGS

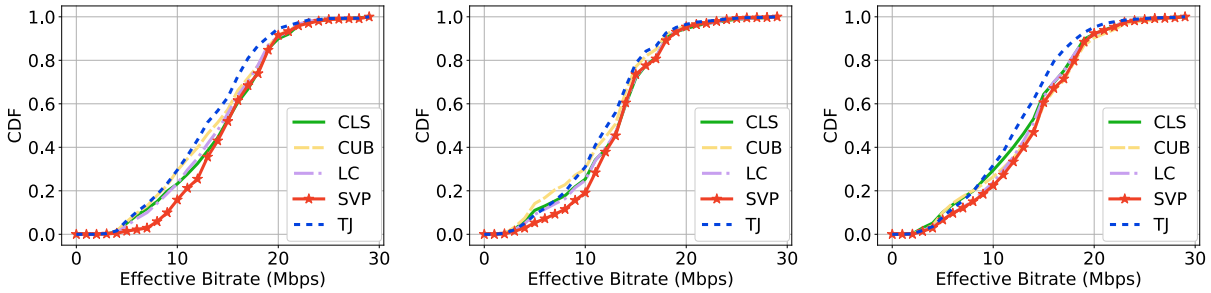
To further evaluate the performance under various network conditions, we conduct a series of experiments under real-world network conditions. To make the simulation results repeatable, we employ the bandwidth trace of the 4G dataset collected at Belgium (see Sec. V-A3 for detailed description). In addition, we evaluate two buffer thresholds to see the variation of effective bitrates, where each buffer threshold serves as the upper limit of time window lengths.

Effective bitrates under dynamic bandwidth settings. FIGURE 11 shows the CDF of effective bitrates of different methods on real-world network throughput with the 3-sec buffer threshold. Evidently, **SVP** performs better than **CLS, CUB, LC, TJ** in that it achieves higher tile prediction accuracy. Specifically, **SVP** can outperform **CLS, CUB, LC, TJ** by 8.6%∼14.6%, 3.2%∼10.7%, 4.7%∼14.5%, and 10.0%∼16.4% respectively, across the datasets (described in Sec. V-A2). On the other hand, when the buffer threshold goes larger (i.e. 6-sec), the prediction time window length will be longer. However, large buffer thresholds can result in inaccurate prediction, which reduces the effective bitrate of **SVP** as well as its performance gain over the comparisons as shown in FIGURE 12. Since live and VoD 360-degree video



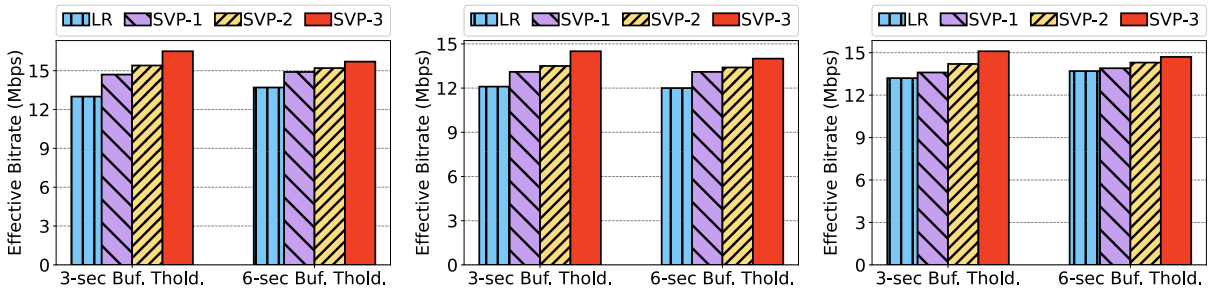
(a) The AV dataset (b) The THU dataset (c) The UT dataset

FIGURE 11. Effective bitrate comparison (dynamic bandwidth, 3-sec buffer threshold).



(a) The AV dataset (b) The THU dataset (c) The UT dataset

FIGURE 12. Effective bitrate comparison (dynamic bandwidth, 6-sec buffer threshold).



(a) The AV dataset (b) The THU dataset (c) The UT dataset

FIGURE 13. Effective bitrate improvement by each stage of SVP (dynamic bandwidth, 3- and 6-sec buffer thresholds).

streaming usually require small buffer thresholds, revealing the applicability of **SVP** in practice.

Per-stage effective bitrate improvement. To see how much performance gain can be brought by the stages of the SVP system, we denote **SVP-***n* as the **SVP** system from stage 1 to stage n, where $n \in \{1, 2, 3\}$. In FIGURE 13, we see that **SPV-1** outperforms **LR** by 1.5%∼13.1% thanks to the accurate orientation prediction. For **SPV-2** and **SPV-3**, the performance gains will be further increased by 2.0%∼4.8% and 2.8%∼7.4%, respectively, revealing that the proposed error handling and tile probability normalization are both effective in elevating effective bitrates.

VI. CONCLUSION

Tile-based streaming has emerged as a promising way to deliver high-quality 360-degree videos with limited bandwidth. In this paper, we propose the 3-stage SVP system to predict orientation, handle prediction errors and normalize viewing probabilities. By interpreting original rotations as sinusoidal values, the SVP system can effectively reduce prediction errors on *yaw* direction. We conduct simulations based on several real-world datasets, and our simulation results show that the SVP system can achieve better prediction accuracy and video quality than other comparison schemes under various buffer thresholds and bandwidth settings.

REFERENCES

[1] (May 2020). *MPEG-DASH*. [Online]. Available: http://mpeg.chiariglione.org/standards/mpeg-dash

[2] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360° video streaming with a better understanding of quality perception," in *Proc. ACM SIGCOMM*, Aug. 2019, pp. 394–407.

[3] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," in *Proc. ACM Multimedia Conf. (MM)*, 2017, pp. 943–951.

[4] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. 5th Workshop Things Cellular Oper., Appl. Challenges (ATC)*, 2016, pp. 1–6.

[5] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 315–323.

[6] X. Jiang, Y.-H. Chiang, Y. Zhao, and Y. Ji, "Plato: Learning-based adaptive streaming of 360-degree videos," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, Oct. 2018, pp. 393–400.

[7] Y. Zhang, Y. Guan, K. Bian, Y. Liu, H. Tuo, L. Song, and X. Li, "EPASS360: QoE-aware 360-degree video streaming over mobile devices," *IEEE Trans. Mobile Comput.*, early access, Mar. 4, 2020, doi: 10.1109/TMC.2020.2978187.

[8] J. Heyse, M. T. Vega, F. De Backere, and F. De Turck, "Contextual bandit learning-based viewport prediction for 360 video," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Mar. 2019, pp. 972–973.

[9] L. Xie, X. Zhang, and Z. Guo, "CLS: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 564–572.

[10] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, "CUB360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2018, pp. 1–6.

[11] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *Proc. IEEE Int. Conf. Artif. Intell. Virtual Reality (AIVR)*, Dec. 2018, pp. 157–160.

[12] C. Li, W. Zhang, Y. Liu, and Y. Wang, "Very long term field of view prediction for 360-degree video streaming," in *Proc. IEEE Conf. Multimedia Inf. Process. Retr. (MIPR)*, Mar. 2019, pp. 297–302.

[13] A. T. Nasrabadi, A. Samiei, and R. Prakash, "Viewport prediction for 360° videos: A clustering approach," in *Proc. 30th ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, Jun. 2020, pp. 34–39.

[14] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting head movement in panoramic video: A deep reinforcement learning approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2693–2708, Nov. 2019.

[15] M. Zhang, K. T. Ma, J. H. Lim, Q. Zhao, and J. Feng, "Deep future gaze: Gaze anticipation on egocentric videos using adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4372–4381.

[16] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze prediction in dynamic 360° immersive videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5333–5342.

[17] Q. Yang, J. Zou, K. Tang, C. Li, and H. Xiong, "Single and sequential viewports prediction for 360-degree video streaming," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[18] X. Li, S. Wang, C. Zhu, L. Song, R. Xie, and W. Zhang, "Viewport prediction for panoramic video with multi-CNN," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2019, pp. 1–6.

[19] X. Chen, A. T. Z. Kasgari, and W. Saad, "Deep learning for content-based personalized viewport prediction of 360-degree VR videos," *IEEE Netw. Lett.*, vol. 2, no. 2, pp. 81–84, Jun. 2020.

[20] X. Feng, V. Swaminathan, and S. Wei, "Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking," in *Proc. ACM IMWUT*, Jun. 2019, pp. 1–22.

[21] X. Feng, Z. Bao, and S. Wei, "Exploring CNN-based viewport prediction for live virtual reality streaming," in *Proc. IEEE Int. Conf. Artif. Intell. Virtual Reality (AIVR)*, Dec. 2019, pp. 183–186.

[22] X. Feng, Y. Liu, and S. Wei, "LiveDeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Mar. 2020, pp. 800–808.

[23] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360° videos," in *Proc. ECCV*, Sep. 2018, pp. 488–503.

[24] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2462–2470.

[25] T. Xu, B. Han, and F. Qian, "Analyzing viewport prediction under different VR interactions," in *Proc. 15th Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2019, pp. 165–171.

[26] B. Han, Y. Liu, and F. Qian, "ViVo: Visibility-aware mobile volumetric video streaming," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, Apr. 2020, pp. 1–13.

[27] S. Afzal, J. Chen, and K. K. Ramakrishnan, "Characterization of 360-degree videos," in *Proc. Workshop Virtual Reality Augmented Reality Netw.-VR/AR Netw.*, Apr. 2017, pp. 1–6.

[28] (May 2020). *LinearSVR*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVR.html.

[29] S. Fremerey, A. Singla, K. Meseberg, and A. Raake, "AVtrack360: An open dataset and software recording people's head rotations watching 360° videos on an HMD," in *Proc. 9th ACM Multimedia Syst. Conf.*, Jun. 2018, pp. 403–408.

[30] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in VR spherical video streaming," in *Proc. 8th ACM Multimedia Syst. Conf.*, Jun. 2017, pp. 193–198.

[31] A. T. Nasrabadi, A. Samiei, A. Mahzari, R. McMahan, R. Prakash, M. Farias, and M. Carvalho, "A taxonomy and dataset for 360° videos," in *Proc. 10th ACM Multimedia Syst. Conf.*, Jun. 2019, pp. 273–278.

[32] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks," *IEEE Commun. Lett.*, vol. 20, no. 11, pp. 2177–2180, Nov. 2016.

[33] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proc. 8th ACM Multimedia Syst. Conf.*, Jun. 2017, pp. 199–204.

[34] (May 2020). *Kvazaar: An Open-Source HEVC Encoder*. [Online]. Available: http://github.com/ultravideo/kvazaar

[35] (May 2020). *Recommended Upload Encoding Settings: YouTube Help*. [Online]. Available: http://support.google.com/youtube/answer/1722171

[36] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: http://arxiv.org/abs/1609.04747

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[38] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. Turck, "An HTTP/2-based adaptive streaming framework for 360° virtual reality videos," in *Proc. 25th ACM Int. Conf. Multimedia*, Oct. 2017, pp. 306–314.

XIAOLAN JIANG received the B.S. degree from the School of Computer Science and Technology, University of Science and Technology of China, Anhui, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Informatics, Graduate University for Advanced Studies (SOKENDAI), and the National Institute of Informatics, Tokyo, Japan. His research interests include 360 degree video streaming, live video streaming, deep reinforcement learning, and caching in future Internet.

SI AHMED NAAS (Student Member, IEEE) received the B.S. and master's degrees from the University of Sciences and Technology Houari Boumediene, Algeria, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, Finland. His research interests include emotion recognition, deep learning, information retrieval, and AI-based mobile optimization.

YI-HAN CHIANG (Member, IEEE) received the Ph.D. degree from the Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, in 2017. He was a Postdoctoral Researcher at the National Institute of Informatics, Japan, in 2018 and 2019, respectively. He is currently an Assistant Professor with Osaka Prefecture University, Osaka, Japan. His research interests include energy efficiency of cellular systems, mobile edge computing, wireless content caching, and the age of information of wireless networks.

STEPHAN SIGG (Member, IEEE) is an Associate Professor with the Department of Communications and Networking, Aalto University. He has published more than 130 peer-reviewed articles. His research interests include pervasive and ubiquitous computing, in particular, with respect to activity recognition and usable security. He has served on the organizing and technical committees of numerous prestigious conferences including the IEEE PerCom, ACM Ubicomp, and the IEEE ICDCS. He is a TPC member of IEEE PerCom since 2016, and a member of the Editorial Board of the *ACM IMWUT* journal since 2020.

YUSHENG JI (Senior Member, IEEE) received the B.E., M.E., and D.E. degrees in electrical engineering from The University of Tokyo. She joined the National Center for Science Information Systems (NACSIS), Japan, in 1990. She is currently a Professor with the National Institute of Informatics (NII) and the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan. Her research interests include network architecture, resource management in wireless networks, and mobile computing.