Optimizing Fixation Prediction Using Recurrent Neural Networks for 360° Video Streaming in Head-Mounted Virtual Reality

Cheng-Ling Fan, Shou-Cheng Yen, Chun-Ying Huang, *Member, IEEE*, and Cheng-Hsin Hsu, *Senior Member, IEEE*

***Abstract*—We study the problem of predicting the viewing probability of different parts of 360° videos when streaming them to Head-Mounted Displays (HMDs). We propose a fixation prediction network based on Recurrent Neural Network (RNN), which leverages sensor and content features. The content features are derived by Computer Vision (CV) algorithms, which may suffer from inferior performance due to various types of distortion caused by diverse 360° video projection models. We propose a unified approach with overlapping virtual viewports to eliminate such negative effects, and we evaluate our proposed solution using several CV algorithms, such as saliency detection, face detection, and object detection. We find that overlapping virtual viewports increase the performance of these existing CV algorithms that were not trained for 360° videos. We next fine-tune our fixation prediction network with diverse design options, including: (i) with or without overlapping virtual viewports, (ii) with or without future content features, and (iii) different feature sampling rates. We empirically choose the best fixation prediction network and use it in a 360° video streaming system. We conduct extensive trace-driven simulations with a large-scale dataset to quantify the performance of the 360° video streaming system with different fixation prediction algorithms. The results show that our proposed fixation prediction network outperforms other algorithms in several aspects, such as: (i) achieving comparable video quality (average gaps between -0.05 and 0.92 dB), (ii) consuming much less bandwidth (average bandwidth reduction by up to 8 Mbps), (iii) reducing the rebuffering time (on average 40 sec in bandwidth-limited 4G cellular networks), and (iv) running in real-time (at most 124 ms).**

***Keywords*-360° video, Virtual Reality, HMD, prediction, machine learning, RNN, tiled streaming**

I. Introduction

Commodity Head-Mounted Displays (HMDs), e.g., Oculus Rift [1], HTC Vive [2], Samsung Gear VR [3], have become more and more popular. While 10.1 million HMDs were sold in 2016, a market research firm predicts a staggering 58% annual growth rate of HMDs sales, predicting for almost 100 million units to be shipped in 2021 [4]. HMDs *dictate* new media content for an immersive experience. For example, omni-directional cameras record scenes from all directions into *360° videos*, which are streamed and played to a viewer wearing an HMD. The viewer may rotate his/her head during the playout to view different parts of the 360° video, *as if* he/she was at the scenes captured by the omnidirectional camera.

360° videos have, in fact, gradually gotten into our daily life. Merely after 1.5 years of rolling out 360° video supports, Facebook reports that more than 1 million 360° videos have been posted [5]. Such a rapid adoption can be attributed to the improved viewing experience: a user study shows that: (i) 360° videos attract 8 times more web clicks, and (ii) viewers of 360° videos watch 29% longer on average, compared to conventional videos [6]. The momentum of the increasing popularity of watching 360° videos with HMDs shows no indication of slowing down in the coming years.

Streaming 360° videos to HMDs is, however, quite challenging for two reasons:

- 360° videos contain much more information than conventional ones, and thus 360° videos have higher resolutions and are encoded at higher bitrates. 360° video streaming systems, therefore, are vulnerable to insufficient and unstable bandwidth due to limited line speed, significant cross traffic, or high wireless dynamics. Hence, efficiently reducing the transmitted data without degrading the video quality is crucial for the success of 360° video streaming systems.
- 360° videos need to be projected from a spherical to flat surface before being compressed, since video codecs only support rectangle videos. Different projection models have different pros and cons, e.g., complexity, pixel density, and shape distortion [7]. Therefore, existing Computer Vision (CV) algorithms designed and trained for 2D images/videos do not perform well with 360° images/videos. Extra care is needed to apply the rich body of existing CV algorithms on to 360° videos for improving the performance of 360° video streaming and other similar systems.

We tackle the above two challenges in this article as follows. First, HMD viewers only get to see a small viewable region, called *viewport*, of each 360° video at any moment. Therefore, streaming whole 360° videos wastes precious bandwidth on many unwatched regions. A better solution is to predict *viewer fixation*, which can be quantified by the viewing probability of different regions, and only transmit the regions with high viewing probability. In particular, we propose to use a Re-

Manuscript received June 2018; revised February 2019; revised June 2019; accepted July 2019.

C. Fan, S. Yen, and C. Hsu are with Department of Computer Science, National Tsing Hua University. C. Huang is with Department of Computer Science, National Chiao Tung University.

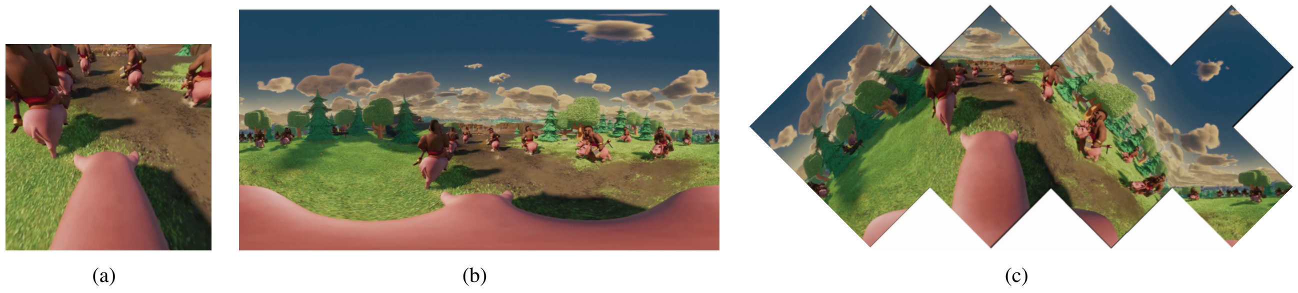


Fig. 1. A sample image: (a) seen in HMDs; distorted images due to: (b) equirectangular projection and (c) rhombic dodecahedron projection.

current Neural Network (RNN) that considers both sensor and content features to predict the viewer fixation, where the sensor features are extracted from the HMDs and the content features are detected from the video content via CV algorithms. To systematically organize video content, we adopt *tiling* [8] to split videos into rectangular regions, or *tiles*, where tiles can be independently encoded/decoded.

Since the content features, such as saliency map [9] and motion map [10], are outputs of CV algorithms, they are vulnerable to distortion attributed to projection models. Such distortion can be further classified into: (i) shape distortion and (ii) ill segmentation, which are illustrated in Fig. 1. Compared to the image seen in an HMD (Fig. 1(a)), the image from equirectangular projection (Fig. 1(b)) suffers from shape distortion, which is especially severe for objects close to the north and south poles. On the other hand, the image from rhombic dodecahedron projection (Fig. 1(c)) suffers from ill segmentation, where an object is cut into smaller pieces in different parts of the projected image. One solution approach is to adopt CV algorithms specifically designed for 360° videos in a given (say equirectangular) projection model. However, such CV algorithms would not work for 360° videos in other projection models. Moreover, compared to CV algorithms designed for 2D images/videos, there are only very few CV algorithms proposed and trained with 360° videos [11], [12], [13], [14]. While the number of these CV algorithms may increase over time, they will still be outnumbered by the CV algorithms for 2D images/videos. To overcome such limitations, we systematically generate *virtual viewports* ahead of the streaming time. Virtual viewports are *carefully* chosen *simulated* viewports in HMDs, which are projected back to the *sphere*. Therefore virtual viewports are not vulnerable to distortion caused by projection models. By sending virtual viewports to existing CV algorithms, we improve the quality of their outputs, as well as that of the subsequent fixation network.

A. Contributions

This article is extended from Fan et al. [15] to improve the performance of the fixation prediction network by: (i) considering future content as features, (ii) eliminating negative effects from projection, and (iii) training and testing on a larger dataset. On top of that, we carefully develop the virtual viewport approach into a unified approach to turn existing CV algorithms applicable to 360° videos.

More specifically, we make the following contributions in this article.

- We propose a fixation prediction network for 360° videos streamed to HMDs, which takes both sensor and content features as inputs and predicts the viewer fixation.
- We propose a unified approach based on overlapping virtual viewports to turn CV algorithms designed and trained for 2D images/videos applicable to 360° videos. We then use this approach to enhance the performance of our fixation prediction network. The resulting algorithm outperforms a state-of-the-art algorithm in the literature in prediction accuracy.
- We employ the fixation prediction network and virtual viewport approach to optimize our 360° video streaming system. To evaluate its performance, we use a large dataset of 50 HMD viewers and ten 360° videos to drive extensive NS-3 simulations. The results show the superior performance of our solution, e.g., compared to the baseline approaches, our solution: (i) achieves comparable video quality (average gaps between -0.05 and 0.92 dB), (ii) consumes much less bandwidth (average bandwidth reduction by up to 8 Mbps), (iii) reduces the rebuffering time (on average 40 sec in bandwidth-limited 4G cellular networks), and (iv) runs in real-time (at most 124 ms). Our solution approach may also be adopted by other 360° video related systems.

B. Paper Organization

The rest of this article is organized as follows. Sec. II surveys the literature. An overview of 360° video streaming systems is given in Sec. III. We introduce our fixation prediction network in Sec. IV. The dataset and neural network implementations are described in Sec. V. Sec. VI presents a unified approach to apply existing CV algorithms on 360° videos, and studies how it enhances the performance of our fixation prediction algorithm. Sec. VII evaluates our proposed solutions using detailed trace-driven simulations. Sec. VIII concludes this article and presents future work.

II. Related Work

360° video streaming has received much attention in recent years [16]. In this section, we survey the literature in two directions: (i) fixation, which is further classified into 2D image/video saliency, 360° image/video saliency, and head movements prediction; and (ii) 360° video streaming, which includes 360° video streaming systems and their optimization.

A. Fixation Prediction

2D image/video saliency. Conventional fixation prediction is built on salient object detection, which has been done on different content types, such as still images [9]. Image saliency can be derived from low-level features, e.g., contrast, textures, and edges [17], [18]. Several studies propose to detect image saliency based on region-based models, which leverages graph-based segmentations [19] or linear iterative clustering [20]. Wang et al. [21] analyze the image saliency by combining color and contrast with spatial priors. For example, the saliency maps are aligned to image edges or correlated with color distributions. Given training samples with ground truth, the learning-based methods are getting popular because of higher accuracy. Liu et al. [22] train learning-based models using a binary labeled dataset with low-level features. Recently, deep learning has become the most popular method to perform saliency detection. For example, Convolution Neural Network (CNN) is able to learn from low-level features parallelly, which is suitable for vision processing. Li and Yu [23] adopt a pre-trained CNN for extracting features in different scales and perform regression on the inputs to produce saliency maps. These studies, however, are designed for 2D conventional images.

In terms of 2D video saliency, Mavlankar and Girod [24] predict the future viewing trajectory based on extrapolation and enhance the performance by analyzing the characteristics of video content, such as optical flow and motion vectors. Recently, More and more supervised learning methods are adopted for fixation detection [25], [26], [27] to achieve better feature extraction and prediction accuracy. In particular, Chaabount et al. [26] predict the video saliency by developing a CNN with residual motion as the features. Furthermore, they adopt transfer learning to cope with the lack of large video datasets. Their results show the positive effectiveness of the transfer learning. Alshawi et al. [27] analyze the correlation between the saliency of pixels and their spatial/temporal neighbors, where the correlation is much affected by the video characteristics. Nguyen et al. [25] note a close relationship between image (static) and video (dynamic) saliency. Based on the observation, they adopt both the information of image saliency (static) and camera motion to predict video saliency (dynamic).

360° image/video saliency. The saliency detection algorithms specifically designed for 360° videos are proposed very recently. Assens et al. [11] develop a deep CNN to predict the scan-paths on 360° images. They train the prediction network on a public 360° image dataset [28], which consists of 60 360° images and 63 participants with the trajectories of both their head and eye movements. In particular, they perform transfer learning by initializing the network weights from several 2D image datasets. They further analyze the prediction under different sampling strategies and their results reveal that limiting the distance between fixations can improve the prediction accuracy. Monroy et al. [12] first map the 360° images to six faces of a cubic projection, which reduces the distortion close to poles compared to the equirectangular projection. Each face is detected by a conventional saliency detection network with spherical coordinates for locating the face on the sphere. Finally, six detected saliency faces are combined into a single saliency map for the 360° image. Similarly, Cheng et al. [29] also map the 360° videos into cubic projection for eliminating the distortion. Some tricks, e.g., wider angle for each face and temporal model development, are introduced to improve the saliency prediction accuracy. Zhang et al. [30] develop a spherical CNN with spherical Mean Squared Error (MSE) loss function, which takes the angle to the center of the sphere into considerations. Besides, the starting position for the viewer to watch the 360° video is also considered as an important feature in their model. These studies [11], [12], [29], [30] develop and train neural networks to predict the saliency for the 360° videos. We note that the resulting prediction algorithms are *locked in* with projection models, compared to our unified approach. Nevertheless, as a future task, we may integrate their proposed solutions with our proposed fixation prediction network for a given projection model.

Fixation/head movement prediction in HMDs. Our work goes beyond saliency detection to predict viewer fixation for 360° video streaming to HMDs. There are a few recent studies that share similar goal to ours. Ban et al. [31] propose to predict the viewer's head movements in three steps. First, an initial prediction is performed based on the viewer's previous viewing position using linear regression. Second, K-Nearest-Neighbors (KNN) are calculated to find the nearest K view points among all other viewers. This is under an assumption that most viewers would be interested in the same objects/areas in 360° videos. Last, the viewport region of the K nearest view points are calculated to finalize the predicted viewing probability of each tile. This study is considered as one of our baselines for comparisons. Not only consider the previous viewing positions, Xu et al. [32] further take video content into account. In particular, they develop the head movement prediction network based on Reinforcement Learning (RL) considering the previous viewer orientation and frame content. The network aims to predict which direction among the eight directions (top, left, bottom, right, and the direction between each two of above directions) the current viewer will move to. However, their study only predicts the future head moving direction, which may be insufficient to be applied to tiled streaming systems. Nguyen et al. [33] employ the same fixation prediction network architecture as Fan et al. [15] that considers both the orientation and the detected saliency on frames. They make three major changes: (i) they develop their own image saliency network trained on their own 360° video viewing dataset, (ii) they do not consider the motion map in our work, and (iii) they represent the orientation data as orientation maps instead of the raw sensor values of *yaw*, *roll*, and *pitch* used in our work.

B. 360° Video Streaming

360° video streaming system. Gaddam et al. [34] propose an interactive panoramic video streaming system with a tile-based encoder. They gradually decrease the quality from the

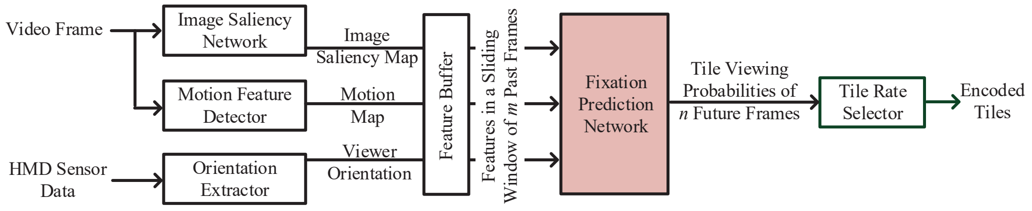


Fig. 2. Architecture of the proposed 360° video streaming server. A tile-based streaming example is shown.

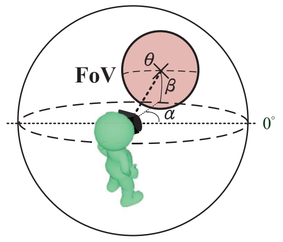


Fig. 3. The model of HMD viewport.

center of the current viewed region. This approach saves bandwidth while providing smooth quality degradation. Spatial Relationship Description (SRD) [35], an extension of DASH (Dynamic Adaptive Streaming over HTTP), enables arbitrary spatial access on video streaming. Several studies [36], [37] adopt SRD as a new way to realize tile-based video streaming for higher flexibility in terms of spatial relationship and encoder/decoder supports. Zhou et al. [38] analyze an undocumented projection model, offset cubic projection, implemented in Oculus HMD streaming system. They find that the projection offers comparable video quality when it saves up to 16.4% in bitrate. Lo et al. [39] build a 360° video streaming system on cellular networks and compare the performance of transmitting all tiles versus viewport tiles only. Graf et al. [40] describe different tiled streaming strategies for 360° videos and conduct experiments to measure the bitrate overhead and bandwidth consumption under different tiling strategies.

Optimization. Alface et al. [41] propose to multicast 16K-resolution videos to users by streaming at 4K-resolution for user viewport and 1K-resolution for the whole panorama shared by all viewers. Xiao et al. [42] propose to download the 360° videos with optimal tile bitrates produced by an Integer Linear Programming (ILP) problem. Their proposed solution improves the bandwidth consumption compared to general fixed-tiling solution by up to 47%. Duanmu et al. [43] formulate the buffer scheduling problem of prioritized 360° video streaming to improve video quality under restricted bandwidth. They propose to guarantee smooth playout by giving the highest priority to the base tiles of the whole panorama at a lower resolution. The residual bandwidth is used for enhanced tiles of predicted viewports at a higher resolution. Sanchez et al. [44] model the rate-distortion relation according to the temporal and spatial characteristics of the video content. Their results demonstrate that the tiling schemes need to be carefully optimized for 360° videos. Several studies [36], [37] propose to only stream tiles within user viewports to the client using DASH. In contrast to requesting tiles one by one, Petrangeli et al. [45] propose to leverage push-based HTTP/2 protocol to reduce network overhead. Qian et al. [46] study 360° video streaming over cellular networks. They conduct measurement study on famous streaming platforms: YouTube and Facebook. Their work adopts a prediction algorithm similar to Mavlankar and Girod [24] for saving bandwidth consumption. Bao et al. [47] develop a hybrid unicast and multicast framework for 360° video streaming. They take the viewers' historical head movements and network conditions to determine the communication model of each area. In particular, the overlapped regions are transmitted using multicast so as to reduce the bandwidth consumption. Lo et al. [48] consider edge-assisted 360° video streaming, in which the edge server either: (i) combines the tiles into a new 360° video stream or (ii) transcodes 360° video into a viewport video stream. They also propose an algorithm to dynamically switch each client between these two edge processing approaches according to the video characteristics and resource constraints for optimal overall video quality. In contrast to these systems work, our article focuses on the optimization of fixation prediction algorithms.

III. Overview

We present an overview of the 360° video streaming systems, which is optimized in the rest of this article.

A. 360° Video Streaming Systems

Fig. 2 presents our proposed architecture of a 360° streaming server [15], in which we focus on the software components related to *fixation prediction*. We have identified two content features: image saliency map [9] and motion map [10] from 360° videos; and a sensor feature: orientation from HMDs. We describe the software components in the following:

- **Image saliency network** is a deep neural network trained to derive the *image saliency map*, which shows the parts of the image that attract viewers the most.
- **Motion feature detector** analyzes the Lucas-Kanade optical flow [49] of consecutive frames, because viewers may be attracted by moving objects.
- **Orientation extractor** derives the viewer orientation data including yaw, pitch, and roll, from HMD sensors.
- **Feature buffer** stores the features, including the saliency map, motion map, and viewer orientation in a sliding window, which are used for fixation prediction.
- **Fixation prediction network** uses content features (image saliency maps and motion maps) and sensor features (viewer orientation) as inputs to predict the viewing probability of different regions of the next n frames.
- **Tile rate selector** performs rate allocation among video *tiles*, which are rectangular and independently decodable regions of a video frame. 360° video streaming systems may be classified into two classes: tile-based [50], [51], [36], [37] and transcoder-based [52], [53]. In tile-based

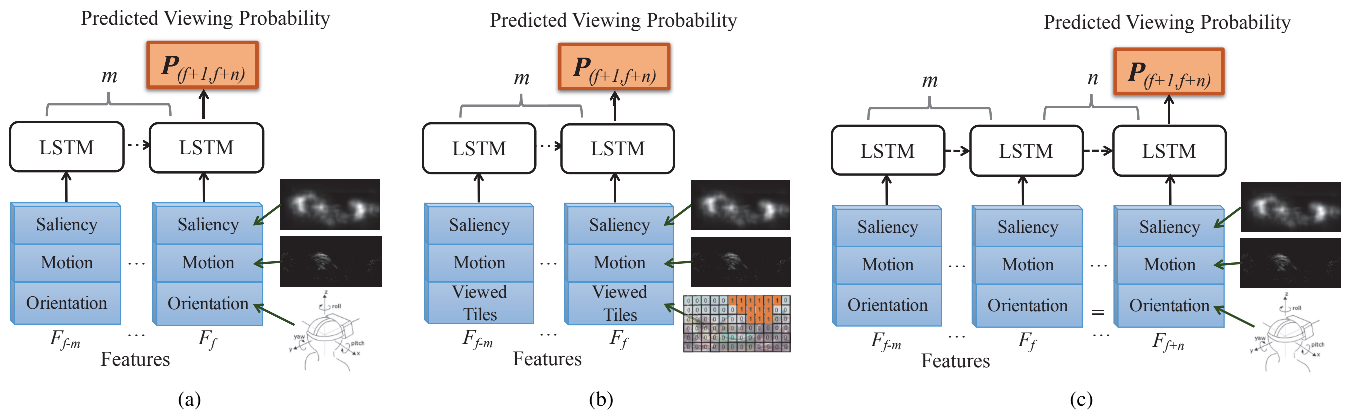


Fig. 4. Our proposed fixation prediction networks: (a) orientation-based network, (b) tile-based network, and (c) future-aware network.

systems, the server encodes 360° videos into tiles, while the client dynamically requests tiles at specific bitrates for adaptation. In transcoder-based systems, the server dynamically transcodes the viewport of each viewer on-the-fly. For the sake of brevity, we assume tile-based systems are used, although our solution is also applicable to transcoder-based systems.

The interactions among these components are as follows. The video frames are sent to the image saliency network and motion feature detector for generating the image saliency map and the motion map, respectively. Generating these two maps is potentially resource demanding, and we assume that they are created offline for pre-recorded videos. The HMD sensor data are transmitted to the orientation extractor to derive the viewer orientation. The feature buffer maintains a sliding window that stores the latest image saliency maps, motion maps, and viewer orientations as the inputs of the fixation prediction network. The fixation prediction network predicts the future viewing probability of each tile. The tile rate selector optimally selects the rates of the encoded video tiles.

B. *Viewport and Modeling*

Different HMDs may have different viewport sizes, which need to be systematically derived. We conduct experiments of playing a 360° video with artificial *grids* to viewers, and collect questionnaires to understand how to model the viewport of commodity HMDs. We find that the viewport of existing HMDs, including Oculus Rift, HTC Vive, and Samsung Gear can all be modeled as a circle on sphere. Viewports could be in different shapes on 2D projected planes. For example, a viewport appears as an ellipse on an equirectangular projected plane. Fig. 3 presents the viewport model of HMDs. The viewer stands at the center of the sphere. Let α and β be the yaw and pitch of the HMD viewport center, which are reported from the sensors equipped by HMDs. Furthermore, we let θ be the diameter of a viewport in degrees. Therefore, we describe the viewport in the spherical space as $f_s = (\alpha, \beta, \theta)$. The measured θ values are about 100° (Oculus Rift), 67° (HTC Vive), and 67° (Samsung Gear). We use 100° in our experiments if not otherwise specified. We note that the parameters of other HMDs may be derived using our experiment design.

IV. Fixation Prediction Networks

The core component of our proposed 360° streaming server is the fixation prediction network, which is detailed in this section.

A. *Overview*

The fixation prediction network is based on an RNN, which is suitable to learn useful information from a time series of video frames. However, basic RNNs suffer from the problem of gradient vanishing during back-propagation [54]. This prevents the RNN from learning long-term dependencies effectively. Hence, we chose to use the LSTM (Long Short Term Memory) network [55]. LSTM solves the problem by using gates in its neurons, and learns more long-term dependencies among video frames.

In this article, we propose three neural networks: (i) orientation-based, (ii) tile-based, and (iii) future-aware. The orientation-based network takes the orientation values of the past frames, which are read from HMD sensors, as the sensor features. The tile-based network considers the viewing probabilities of tiles, which have already been projected from the raw orientation values, of the past frames as the sensor features. Both networks take the saliency and motion maps of the past frames as the content features. Our preliminary study [15] demonstrates the higher prediction accuracy and efficiency of the orientation-based network compared to the tile-based network. Therefore, we extend the orientation-based network into the future-aware network in this article. The future-aware network considers the content features of not only the past frames but also future frames. This is feasible because all the video frames are pre-stored on the server, thus the content features can be extracted and saved beforehand.

In addition to the future-aware network, we enhance the networks presented in our previous study [15] in two ways. First, we reduce the feature sampling rate to 1 frame-per-second (fps). The intuition behind this decision is that the changes of video content and viewer orientation are typically small over a short time period. Therefore, although lower sampling rates may impose small negative impacts on prediction accuracy, they significantly reduce the resource consumption. Second, the proposed networks predict the viewing probability of each tile within a number of frames instead of just a single frame. The rationale is that in most practical streaming systems, each client asks for a few consecutive frames in a request. For example, a DASH client asks for a *segment* of video frames in each request. Our pilot experiments show that the two enhancements lead to: (i) at least three times of training time reduction and (ii) on average 1.4% accuracy boost, compared to our original networks [15]. We present the three resulting networks below, while more performance results are given in Sec. VII.

B. Orientation-Based Network

Fig. 4(a) presents the orientation-based network. Let F_f be the features of frame f, which include the image saliency map, motion map, and viewer orientation. The saliency maps and motion maps are downsampled to 64x64 to avoid excessive computation loads. The viewer orientation is the sensor data, which consists of *x, y, z, yaw, roll*, and *pitch*, read from HMD sensors. These features are concatenated and fed into the network. Let m and n be the number of past frame samples that contribute the features to the network and the number of the predicted frames, respectively. We let $P^t_{f+1,f+n}$ denote the predicted viewing probability of tile t within frame samples $f+1$ to $f+n$. That is, if the tile is predicted to be viewed for x times within these frames, the predicted viewing probability of this tile is $\frac{x}{n}$. We collectively write the probabilities of all tiles within frame samples $f+1$ to $f+n$ as $\mathbf{P}_{f+1,f+n}$.

C. Tile-Based Network

The tile-based network is presented in Fig. 4(b). Compared to the orientation-based network, the tile-based network replaces the viewer orientation with the viewing probability of each tile. More specifically, the probabilities of tiles that are viewed by the viewer are 1's and others are 0's. Similar to the orientation-based network, the saliency maps, motion maps, and the viewing probability of tiles from past m frames are concatenated as features and fed into the network to predict the viewing probability of tiles within next n frames.

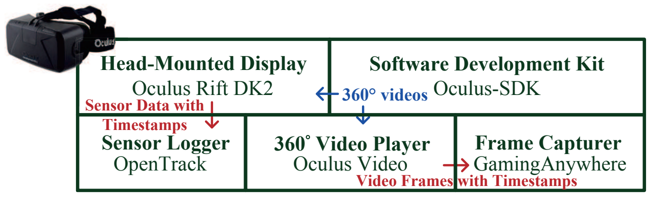
D. Future-Aware Network

Fig. 4(c) presents the proposed future-aware network. It is extended from the orientation-based network due to its better performance compared to the tile-based network [15]. In particular, it also takes the future content features into account. That is, the future-aware network takes the features from F_{f-m} to F_{f+n} as inputs to predict the viewing probabilities $\mathbf{P_{f+1,f+n}}$. The unknown future viewer orientation for F_{f+1} to F_{f+n} are approximated with the last received viewer orientation, if not otherwise specified, while more sophisticated extrapolation [46] can also be used.

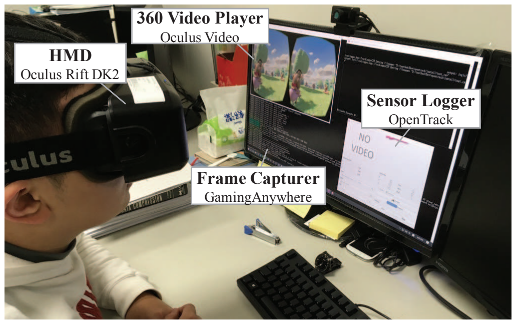
V. Datasets and Network Implementations

We collected a 360° video dataset, which contains 50 viewers, each watches ten 360° videos [56]. In this section, we first summarize the dataset. We next compare the performance of the proposed fixation prediction networks using the collected dataset.

A. Dataset



(a)



(b)

Fig. 5. Our testbed for collecting our dataset: (a) testbed architecture and (b) a photo of a subject performing the experiments.

Fig. 5(a) presents the design of our testbed [15], which consists of: (i) an Oculus Rift HMD, (ii) the Oculus Software Development Kit (SDK), (iii) the 360° video player (rendering 360° videos in HMD and on a mirrored screen), (iv) the sensor logger based on OpenTrack, and (v) the frame capturer based on GamingAnywhere [57].

When a viewer watches a 360° video as shown in Fig. 5(b), the rendered video is captured by the frame capturer and stored to the disk. The viewer's head movements, including position and orientation, are recorded by the sensor logger. Both of them are timestamped on the same computer. By aligning sensor data and 360° videos, we know where the viewer is watching at any moment.

We download ten 360° videos from YouTube, which are in 4K resolution with a frame rate of 30 fps. The videos have diverse characteristics, e.g., computer-generated versus natural images, and slow- versus fast-paced. We recruit 50 viewers for dataset collection. We play all ten videos to each viewer, which result in 500 *traces* in our dataset. By trace, we refer to a combination of a viewer and a video, in the rest of this article. For more details about the compositions of the viewers and the format of the datasets, readers are referred to Lo et al. [56].

B. Network Implementations

We consider the fixation prediction problem on tiles as a multi-label classification problem and have implemented the neural networks using Scikit-Learn and Keras. The ground truth of the fixation prediction networks for each tile is the fraction of frames containing the viewed tiles. This fraction represents the *importance* of each tile. Using the datasets, we sample the points within the viewport by projecting the

TABLE I
THE PERFORMANCE OF THE PROPOSED MODELS WITH 1-SEC SLIDING WINDOW

The Orientation-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*F*	86.93%	0.703	85.79%	0.678
512	2	*T*	88.41%	0.741	87.03%	0.711
1024	**2**	***T***	**89.09%**	**0.760**	**87.05%**	**0.732**
2048	2	*T*	88.11%	0.733	86.67%	0.702
The Tile-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*T*	84.80%	0.636	83.65%	0.610
512	2	*F*	84.68%	0.632	0.147	83.43%
1024	2	*F*	84.96%	0.636	83.75%	0.608
2048	**2**	***T***	**85.15%**	**0.643**	**83.90%**	**0.614**
The Future-Aware Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*T*	88.09%	0.733	86.77%	0.706
512	2	*T*	88.99%	0.759	87.62%	0.732
1024	**2**	***T***	**89.27%**	**0.767**	**87.77%**	**0.737**
2048	2	*F*	85.65%	0.663	84.43%	0.635

TABLE II
THE PERFORMANCE OF THE PROPOSED MODELS WITH 4-SEC SLIDING WINDOW

The Orientation-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*F*	86.37%	0.615	83.27%	0.588
512	**2**	***F***	**87.91%**	**0.729**	**86.43%**	**0.699**
1024	2	*F*	86.90%	0.696	85.27%	0.658
2048	2	*F*	84.73%	0.634	83.61%	0.606
The Tile-Based Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*T*	84.62%	0.629	83.49%	0.604
512	2	*F*	84.69%	0.624	83.44%	0.593
1024	**2**	***T***	**85.00%**	**0.634**	**83.73%**	**0.603**
2048	2	*F*	84.57%	0.623	83.38%	0.594
The Future-Aware Network						
Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*F*	84.44%	0.612	83.45%	0.589
512	**2**	***T***	**88.15%**	**0.733**	**86.70%**	**0.701**
1024	2	*F*	85.27%	0.648	84.18%	0.624
2048	2	*F*	84.80%	0.636	83.71%	0.610

orientation on the sphere to the equirectangular model. Then, the viewed tiles are those that contain some projected samples. For a single video frame, each tile is either watched or not, i.e., it has a boolean viewing probability.

We use the traces from 50 viewers to train the proposed three networks. We randomly divide the 500 traces [56] into two subsets: 80% for training and 20% for testing. We reserve 20% of the training set for validation purpose. The networks are trained to minimize the *logarithmic loss*, also known as *cross-entropy loss*, using Stochastic Gradient Descent [58] with a learning rate of 10^{-1}. An early-stop mechanism, which stops the training once the logarithmic loss is smaller than a given value, is adopted to speed up the network training and avoid over-fitting. We consider the sliding window size of 1 and 4 secs to predict the frames in the upcoming second. To obtain the optimal parameters, we consider the number of neurons in {256, 512, 1024, 2048}, the number of LSTM layers in {1, 2, 3}, and the dropout in {True, False}, where the dropout rate is 0.2.

We note that the predicted probability is a real number between 0 and 1, and we use a *threshold* ρ to round it to a boolean decision. We refer to $p_f^t \geq \rho$ as *predicted tiles*, and the actually viewed tiles as *viewed tiles*. We let $\rho = 0.5$ if not otherwise specified. To select the optimal parameters of the three neural networks, we consider two metrics: (i) *accuracy*, which is the ratio of correctly classified tiles to the union of predicted and viewed tiles and (ii) *F-score*, which is the harmonic mean of the precision and recall, where the precision and recall are the ratios of correctly predicted tiles to the predicted and viewed tiles, respectively.

We find that the networks with two LSTM layers generally give better performance, and thus we report sample 2-layer results with 1- and 4-sec sliding windows in Tables I and II, respectively. The optimal parameters and results for each network are in bold fonts. These tables show that the future-aware network has higher accuracy and F-score for both sliding window sizes. Besides, with the future-aware network, 1-sec sliding window performs slightly better than 4-sec one (accuracy 87.77% $>$ 86.70%), yet runs 6x faster (69 versus 398 minutes). *Hence, we adopt the future-aware network with the 1-sec sliding window as our fixation prediction network in the rest of the article.*

VI. OVERLAPPING VIRTUAL VIEWPORTS

In this section, we first introduce the projection models and discuss how they negatively affect the performance of CV algorithms designed and trained for 2D images/videos. We then propose overlapping virtual viewport (OVV). We apply OVVs on three existing CV algorithms and report their performance boosts. We then apply OVV to our proposed fixation prediction network. Last, to validate the generality of our solution, we evaluate it using additional videos and viewers.

A. Projection Models

There are many projection models proposed for 360° videos [7], [59] in the literature, and the commonly seen ones[1] are: (i) equirectangular, (ii) cubic, and (iii) rhombic dodecahedron. We present the properties of individual projection models in the following.

Equirectangular. As shown in Fig. 6(a), the equirectangular projection projects the sphere to a *cylinder*. It introduces large shape distortion at the areas close to poles (see Fig. 1(b)), which may result in redundant data transmission and inferior performance (for example, accuracy) of existing CV algorithms.

Cubic. The cubic projection model projects a sphere to a circumscribed cube with six square faces as shown in Fig. 6(b).

[1]Note that researchers and companies continue proposing new projection models for better coding efficiency. Most projection models suffer from some shape distortion and/or ill-segmentation, similar to the representative models presented here.

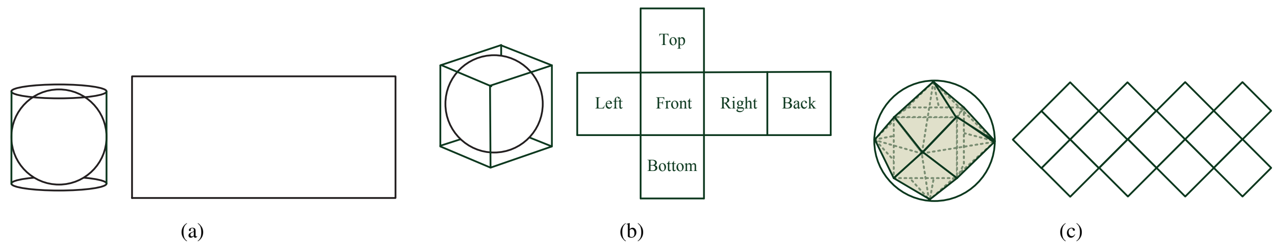


Fig. 6. Several projection models can be used for 360° videos, such as: (a) equirectangular, (b) cubic, and (c) rhombic dodecahedron.

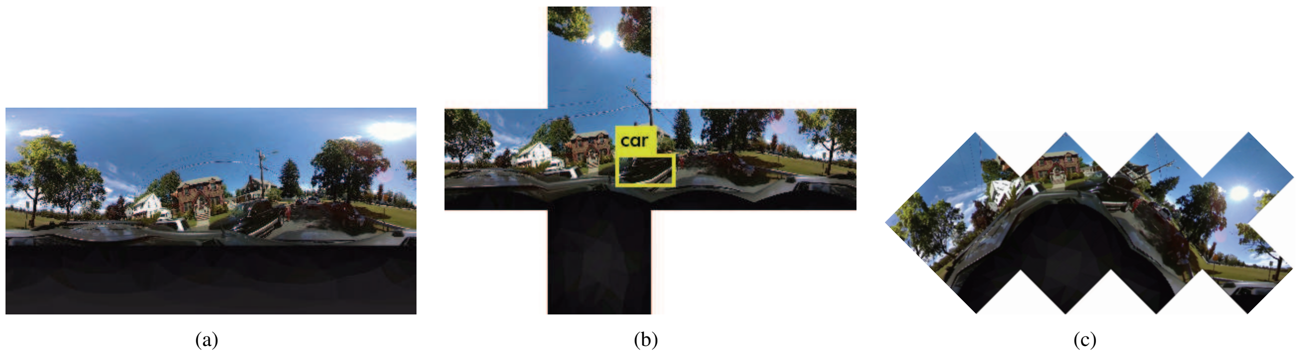


Fig. 7. Object detection on sample image with different projection models: (a) equirectangular, (b) cubic, and (c) rhombic dodecahedron. Only one object is detected.

For each point on the sphere, we first find the closest face as its corresponding face. Each face adopts 90° rectilinear projection, which maps the sphere surface to a tangent plane, where the points are projected along with the line from the sphere center to the plane. In contrast to equirectangular, the cubic projection model preserves the straight lines on each face. Therefore, the cubic projection model results in no pole distortion and reduces about 25% of the data size [60]. However, for lines or objects that span over multiple faces, they are unnaturally, or ill, segmented at the face boundaries.

Rhombic dodecahedron. The rhombic dodecahedron projection model adopts 12 equal-size spherical rhombus. Fig. 6(c) shows the construction of the rhombic dodecahedron, which is an octahedron with a cube embedded in it. Two of the four corners of each rhombus are from the cube, while the other two are from the octahedron. We can project the rhombic dodecahedron to a sphere surface using gnomonic projection, which is a superset of rectilinear projection that does not limit the degree to 90°. One way to project pixels to the rhombic dodecahedron is to use the great circle subdivisions [61]. While the rhombic dodecahedron projection model does not suffer from noticeable distortion, it incurs higher computational overhead.

For the sake of understanding the distortion level, we use YOLO9000 [62] to perform object detection on different projection models. Fig. 7 shows sample object detection results with: (i) equirectangular, (ii) cubic, and (iii) rhombic dodecahedron projection models. This figure shows that only a single object is detected. This can be attributed to shape distortion and ill segmentation.

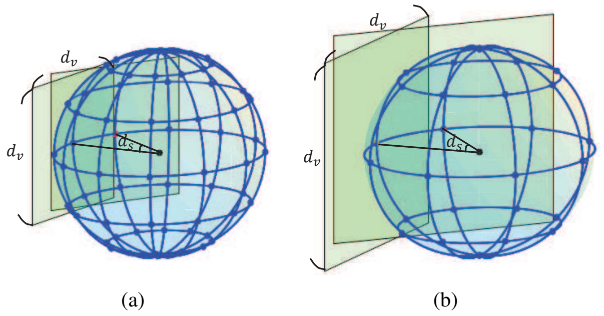


Fig. 8. Examples of the OVV: (a) $(d_s, d_v) = (30°, 60°)$ and (b) $(d_s, d_v) = (45°, 90°)$.

B. Overlapping Virtual Viewport (OVV)

We propose to leverage OVV to cover the whole sphere space so as to turn CV algorithms designed and trained for 2D images/videos applicable to 360° videos. A virtual viewport is a square tangent to a point on the sphere surface. OVV is defined by d_v and d_s, where d_v represents the viewable angle at equator of each virtual viewport, and d_s is the sampling angle of virtual viewports. Both d_v (size) and d_s (density) are in the unit of degrees. Fig. 8 illustrates example OVVs with $(d_s, d_v) = (30°, 60°)$ and $(d_s, d_v) = (45°, 90°)$. In the figure, we only plot two sample virtual viewports for brevity. However, each intersection point on the sphere surface is the center of a virtual viewport, which is tangent to the sphere at that point. Therefore, each 360° sphere has $\frac{2\pi}{d_s}\frac{\pi}{d_s}$ virtual viewports.

OVV eliminates the shape distortion and ill segmentation by: (i) extracting virtual viewports, which are the actual views

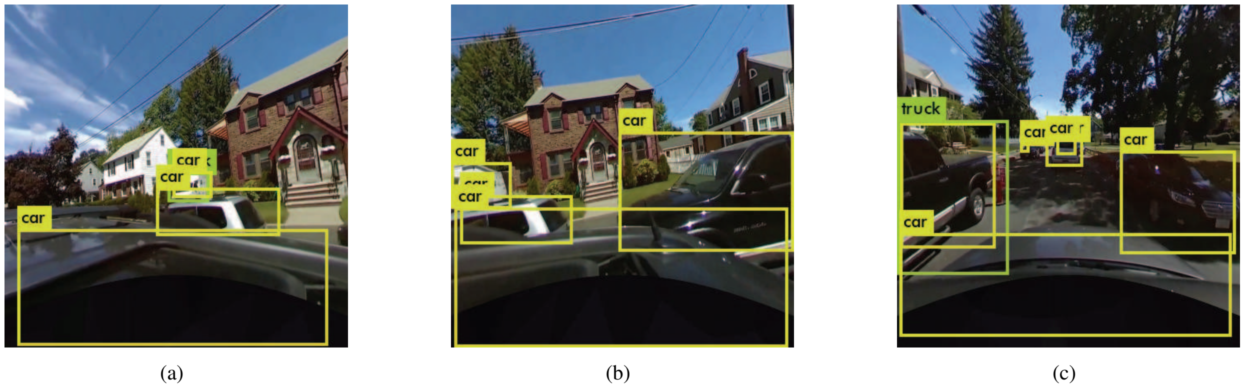


Fig. 9. Object detection on sample virtual viewports of OVV: (a) (yaw, pitch) = (315°, 90°), (b) (yaw, pitch) = (0°, 90°), and (c) (yaw, pitch) = (90°, 90°). More objects are detected with OVV, compared to Fig. 7.

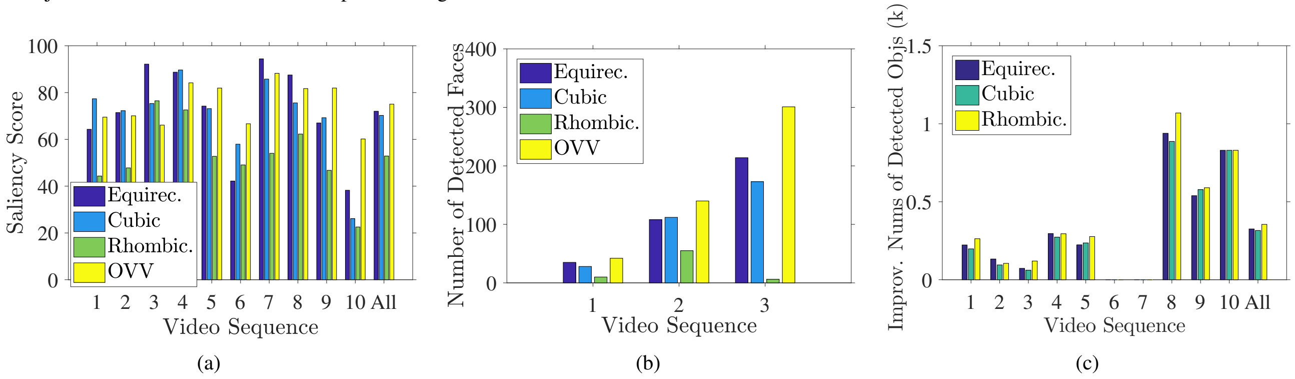


Fig. 10. Our proposed OVV improves the performance of the pre-trained CV algorithms: (i) saliency detection, (ii) face detection, and (iii) object detection.

seen in HMDs, and (ii) oversampling (overlapping) virtual viewports to increase the chance for CV algorithms to identify, for example, objects. Fig. 9 shows the sample results of detecting objects with OVV, which gives more recognized objects than the original approach reported in Fig. 7.

C. Validations with Real Computer Vision Algorithms

We consider three representative CV algorithms: (i) saliency detection, (ii) face detection, and (iii) object detection. In the following, we describe these algorithms and report the benefits of applying OVV with (d_s,d_v)=(45°,90°) on them. The ten videos used for validation are from the dataset mentioned above [56], and the ground truth is tagged by our group.

Saliency detection [9] calculates the attraction level of each pixel of images. This can be done based on analyzing image contrast, detecting objects, and locating outstanding and meaningful parts of images. We adopt a pre-trained deep multi-level network [63] that takes low- to high-level features to perform saliency detection with diverse projection models. To compare the performance of the saliency detecting algorithm on different projection models, we use *true* viewports from our traces to quantify the quality of the resulting saliency maps. For each projection model, we normalize the detected saliency map so that the sum of the saliency values is 1. Then we sum all the detected saliency values within the *actual* viewport at each frame of each projection model, and refer to it as the *saliency scores*. For OVV, we perform saliency detection on individual virtual viewports and normalize the overlapped regions among virtual viewports for fair comparisons. Fig. 10(a) plots the total saliency scores with 95% confidence intervals under different projection models. This figure shows that OVV generally has higher saliency scores than other projection models. Besides, the p-value from one-way analysis of variance (ANOVA) test is only 0.012 (≤ 0.05), which shows the statistic significance of the performance difference.

Face detection [64] overlays rectangles that contain human faces on images. This can be done by first extracting features and then performing cascades detection, which is to check the features with classifiers stage-by-stage. Once the checking process fails at a stage, it terminates. A face is detected only if all stages are passed. We implement face detection based on Haar Cascades classifier using OpenCV [65] with a scaleFactor of 1.3 and a minNeighbors of 5. We perform face detection on 3 (out of 10) videos that contain people. We record the number of correctly detected faces of each projection model and plot the total number of each video in Fig. 10(b). This figure shows that OVV detects up to 300 more faces compared to other projection models.

Object detection [66] highlights the regions that contain real-world objects, such as dogs, bicycles, and chairs. We adopt the YOLO9000 network [62], which is trained by ImageNet [67] and COCO [68] datasets and is able to recognize more than 9000 objects. We apply YOLO9000 on the ten videos from the dataset, and generate the annotated videos

TABLE III
THE PERFORMANCE OF FUTURE-AWARE NETWORK WITH OVV

$(d_s, d_v) = (30°, 60°)$						
Model Parameters			*Training Set*		*Testing Set*	
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*F*	87.31%	0.714	86.03%	0.686
512	2	*F*	82.57%	0.602	81.54%	0.582
1024	2	*T*	89.22%	0.764	87.71%	0.733
2048	**2**	***T***	**89.72%**	**0.778**	**87.93%**	**0.742**

$(d_s, d_v) = (45°, 90°)$						
Model Parameters		*Training Set*		*Testing Set*		
No. Neurons	*LSTM Layers*	*Dropout*	*Accuracy*	*F-Score*	*Accuracy*	*F-Score*
256	2	*F*	86.71%	0.702	85.35%	0.674
512	2	*T*	84.83%	0.647	83.59%	0.620
1024	2	*T*	88.36%	0.745	86.94%	0.716
2048	**2**	***T***	**88.63%**	**0.753**	**87.09%**	**0.722**

with different projection models and OVV. We report the correctly detected results in Fig. 10(c). This figure clearly demonstrates that OVV significantly increases the number of correctly detected objects. Note that all projection models detect no objects in video 6 and 7 since these two videos are of landscapes and 2D games, which have almost no real-world objects.

In sum, the performance of the existing CV algorithms designed and trained for 2D images/videos are improved by our proposed OVV, which is a unified approach to apply them on 360° videos. In the next section, we integrate OVV with our proposed fixation prediction network.

D. Fixation Prediction with OVV

We perform saliency detection on virtual viewports and stitch the saliency maps of virtual viewports into one image for each frame. The stitched saliency maps then replace the original equirectangular saliency maps as the inputs of the fixation prediction networks. We consider two different configurations of OVV: (i) (d_s, d_v)=(30°, 60°) and (ii) (d_s, d_v)=(45°, 90°). We then train the proposed neural network using these two OVV setups and report sample results in Table III. The optimal parameters and results are in bold fonts. This table shows that the performance of the future-aware network with OVV is better when (d_s,d_v)=(30°,60°). Moreover, it outperforms the equirectangular projection model in terms of accuracy and F-score reported in Table I. Thus, we adopt OVV with (d_s,d_v)=(30°,60°) as the fixation prediction network in the rest of this article.

E. Validation with Additional Videos/Viewers

To understand the generality of our proposed prediction model, we validate our trained model with additional videos and viewers.

Setup. We perform prediction on five new 360° videos downloaded from YouTube at 3840x1920 resolution and 30 fps. We cut them into the same length of 30 seconds. Table IV lists the considered videos. We recruit 30 viewers between 19 and 28 years old. Among them, about 2/3 are males. All viewers are asked to freely watch the five videos in random order. The viewer orientation are logged when they are watching the videos. We consider Ban et al. [31], which is introduced in Sec. II-A, as our baseline and call it CUB360[2]. Because CUB360 employs KNN for prediction, we perform 3-fold validations on CUB360, where 10 viewers are used as the testing set, and 20 viewers are used as the KNN inputs. For our prediction network, we use the model derived in the previous section to predict the viewing probabilities of each tile on all the traces from the 30 viewers viewing the five videos. We note that the comparisons give CUB360 a slight edge of peeking into the new videos/viewers.

TABLE IV
VALIDATION VIDEOS

Video	Time Interval	Link
Snow Boarding	01:30–02:00	https://goo.gl/2H4RxM
Elephants	00:50–01:20	https://goo.gl/aejCVM
Sharks	00:05–00:35	https://goo.gl/BUEBG9
CERN	01:50–02:20	https://goo.gl/68gDAZ
London	00:40–01:10	https://goo.gl/YRN1kN

Results. Table V reports the performance in accuracy and F-score for our proposed prediction network and CUB360 under different K-nearest viewpoints selection. To be conservative, we grid-search on the thresholds to round the probabilities to boolean decisions and report the absolutely optimal solution of CUB360 for each K value. This table shows that although CUB360 references the K-nearest viewpoints from other viewers on the same video, our prediction network still achieves at least 8.7% higher accuracy without peeking into these traces.

The results demonstrate the generality of our proposed fixation prediction network. In terms of CUB360, increasing K do not always improve the performance. This is because other viewers' fixation may be misleading, since different viewers may have quite diverse viewing behavior. In contrast, our proposed fixation prediction network is trained with other 360° videos and also takes the current viewer's past orientation into account. This makes our prediction algorithm more robust. In addition to CUB360, Nguyen et al. [33] also propose a head movement prediction network using LSTM, which employs the same network architecture as our preliminary study [15], while introducing three improvements as we detail in Sec. II-A. While we are not able to compare with their work, applying their enhancements in our systems will likely boost our performance further.

VII. EVALUATIONS

In this section, we evaluate the performance of our 360° video streaming system with the future-aware network and

[2]While we also want to consider Nguyen et al. [33] as another baseline, we couldn't do so because some of their parameters are not made public yet at the time of writing.

TABLE V
THE PERFORMANCE OF DIFFERENT PREDICTION ALGORITHMS

Prediction Algorithm		Accuracy	F-Score
Our		81.8%	63.1%
CUB360	K**=0**	73.1%	31.0%
	K**=2**	73.0%	53.4%
	K**=5**	73.0%	54.3%
	K**=10**	72.2%	54.6%

OVV. We conduct the evaluations through simulations, because: (i) the viewer behavior can be repeated for fair comparisons among different algorithms, and (ii) more traces/subjects can be leveraged at a relatively lower cost.

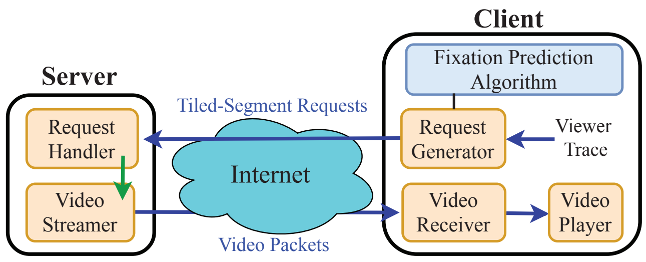


Fig. 11. The streaming server and client in our simulator.

A. Implementations

We have implemented two other prediction algorithms using Python: (i) Cur, which uses the current orientation as the prediction in the next segment and (ii) Dead Reckoning (DR)[3] [46], which computes a weighted moving average of the viewer orientation velocity for prediction. We have implemented a simulator using C++ based on NS-3 [69] and DASH simulator [70]. We modify the simulator to read the real traces of viewing 360° videos, which contains sizes of the transmitted tiles. We implement our proposed fixation prediction network and the other two baseline algorithms in the fixation prediction algorithm as shown in Fig. 11. In addition, there are five more components: (i) the request generator, (ii) the request handler, (iii) the video streamer, (iv) the video receiver, and (v) the video player. The request generator reads the viewer traces and invokes one of the prediction algorithms to generate requests. The request handler parses the received requests. The video streamer encapsulates the tiles into packets and sends them to the video receiver. After the video player reaches an initial buffering time and receives enough number of segments, the video is played until there is no segment available in the receiving buffer. If a segment is not received in time, a rebuffering event is logged and the player pauses the video until the next segment is received.

B. Setup

We use all the traces from the testing set (see Sec. V), 98 traces in total, to drive our simulations. We encode these videos into 20x10 tiles, where each tile has 192x192 pixels, with a QP of 28 using Kvazaar [71] and divide them into 1-sec segments using MP4Box. We assume that the network bottleneck is at the client side, therefore, we repeat the simulations with three access networks: (i) (fixed) Broadband, (ii) WiFi, and (iii) 4G cellular network. We consider the average bandwidth (latency) of the above networks in our simulator as 43.2 (3 ms), 37.1 (10 ms), and 12.7 (40 ms) Mbps, respectively, following recent white papers [72], [73]. To accommodate the latency caused by networks and protocols, we run fixation prediction algorithms a couple of seconds[4] ahead of the current playout time.

We consider the following performance metrics:

- **Missing ratio.** The fraction of unavailable tiles at the client over all tiles that are watched by the viewer. Higher missing ratio leads to more *holes* (missing tiles) in the 360° videos.
- **Unseen ratio.** The fraction of the tiles at the client that are not watched by the viewer over all transmitted tiles. Higher unseen ratio indicates more wasted network resources.
- **Bandwidth consumption.** The consumed bandwidth used to stream the predicted tiles.
- **Peak bandwidth.** The peak bandwidth consumption due to streaming the predicted tiles.
- **Video quality.** We employ the objective quality metric V-PSNR, which is proposed for 360° videos [74] and adopted by JVET [75]. V-PSNR is essentially the Peak Signal-to-Noise Ratio (PSNR) value of a viewer's viewport. We use the ground truth of viewports in the datasets to calculate V-PSNR values.
- **Total rebuffering time.** The total rebuffering time throughout each 1-min playout.

Some pilot simulations reveal that the missing ratio are non-trivial for our and baseline solutions: more than 15% missing ratio are observed. To be practical, we augment our solution to ensure sub-τ missing ratio by adjusting ρ, where *target missing ratio*: $\tau \in \{1\%, 5\%, 10\%\}$. The default τ is 10% if not otherwise specified. For Cur and DR, we iteratively add new tiles at the edge of predicted tiles for δ_{Cur} and δ_{DR} times to accommodate the inferior missing ratio, respectively. Besides, the missed tiles are assumed to be concealed by replaying the last received tiles during video playback. In the next section, we report the simulation results with 95% confidence intervals whenever applicable.

C. Results

Our fixation prediction network consumes less network bandwidth at any target missing ratio. To meet $\tau \in \{1\%, 5\%, 10\%\}$, δ_{Cur} and δ_{DR} are set to 1, which lead to $\rho = \{0.008, 0.027, 0.053\}$, respectively. We plot the bandwidth consumption under different networks with different τ values in Fig. 12. In this figure, Cur and DR show high bandwidth consumption. On the other hand, with properly selected ρ, our fixation prediction network can reduce about 2, 6, and 8 Mbps in bandwidth consumption compared to Cur and DR in Broadband and WiFi networks. Note that the available bandwidth of the 4G cellular network is too scarce, and is used up no matter which algorithm is adopted.

Fig. 13 plots the unseen ratio under different target missing ratios τ. This figure shows that our fixation prediction network reduces the unseen ratio by 5%. The reduction becomes 10% and 15% when $\tau = 5\%$ and $\tau = 10\%$, respectively. In

[3]There are two variants of DR prediction: based on the past velocity or on both past velocity and acceleration. We implement the DR algorithm based on the past velocity in our simulations, following a recent work [46].

[4]Increasing it to 4 seconds or reducing it to 1 second results in similar results.

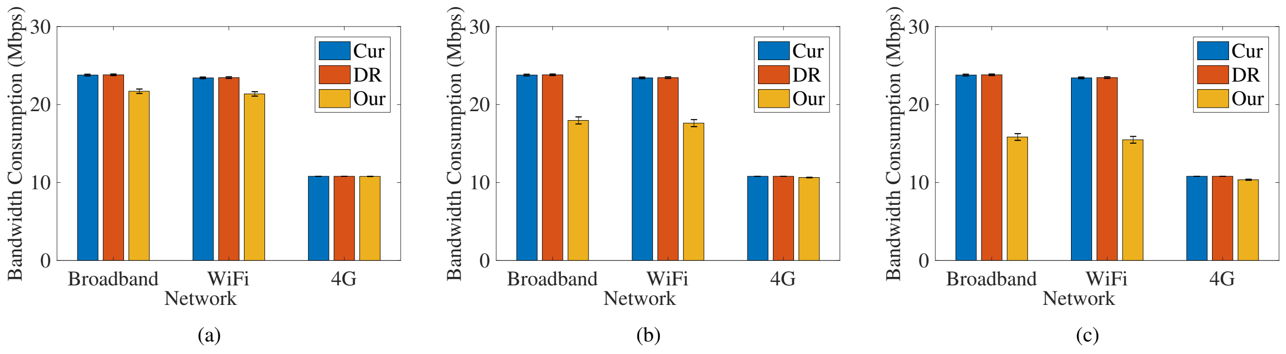


Fig. 12. The bandwidth consumption in different networks with different target missing ratio: (a) $\tau = 1\%$. (b) $\tau = 5\%$, and (c) $\tau = 10\%$.

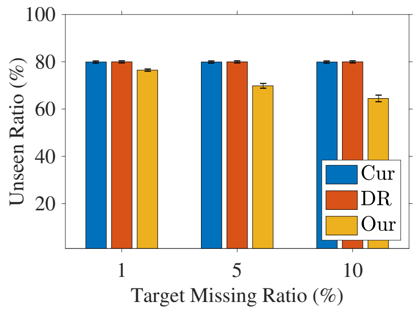


Fig. 13. The unseen ratio under different target missing ratios.

TABLE VI
VIDEO QUALITY UNDER DIFFERENT PREDICTION ALGORITHMS
(V-PSNR IN dB)

Missing Ratio	1%			5%			10%		
Prediction Algorithm	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**
Cur	38.62	42.03	47.58	38.63	42.03	47.58	38.62	42.03	47.58
DR	38.70	42.05	47.64	38.70	42.05	47.64	38.70	42.05	47.64
Our	38.39	42.08	47.89	37.24	41.63	47.27	36.11	41.11	46.66

summary, Figs. 12 and 13 show that our fixation prediction network consumes less bandwidth due to fewer unseen tiles.

Our fixation prediction network leads to shorter rebuffering time. We plot the total rebuffering time under different networks in Fig. 14(a). This figure shows that all considered prediction algorithms lead to no rebuffering event in Broadband and WiFi networks. However, the rebuffering events occur in all the considered prediction algorithms under bandwidth-limited 4G cellular networks. It is worth to note that our fixation prediction network has shorter rebuffering time than other algorithms in 4G cellular networks: up to 40 second reduction. Fig. 14(b) presents the total rebuffering time under different initial buffering time in 4G networks. This figure shows that the rebuffering time is reduced as the initial buffering time increases. However, the total rebuffering time is still non-negligible to the 60-sec streaming session. This indicates the importance of *rate adaptation* for streaming system. In our future work, we may choose lower bitrates for the selected tiles that have lower predicted viewing probability to further reduce the bandwidth consumption.

TABLE VII
CONSUMED BANDWIDTH UNDER DIFFERENT PREDICTION ALGORITHMS
(MBPS)

Missing Ratio	1%			5%			10%		
Prediction Algorithm	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**
Cur	21.11	23.99	24.42	21.11	23.99	24.42	21.11	23.99	24.42
DR	21.13	24.02	24.23	21.13	24.02	24.23	21.13	24.02	24.23
Our	17.13	21.88	24.10	14.08	18.04	22.23	12.26	15.85	20.48

Our fixation prediction network requires less available bandwidth to avoid rebuffering events. To understand the minimum available bandwidth required to avoid rebuffering events without rate adaptation, we conduct simulations under different available bandwidth and plot the results in Fig. 15. This figure shows that our fixation prediction network gets rid of rebuffering events while the available bandwidth is higher than 25 Mbps, which is approximately 10 Mbps less than other algorithms. We note that recent survey [73] reports that the mean available bandwidth of 4G cellular networks is less than 15 Mbps in North America. Hence, we do not consider 4G cellular networks in the rest of this article. The limitation of the current 4G cellular networks are likely to be lifted in the future, as 4G/5G cellular networks continue to advance.

Our fixation prediction network achieves comparable video quality at lower bandwidth consumption. We dig a bit deeper and report the minimum, average, and maximum of video quality of the considered prediction algorithms in Table VI. This table shows that our fixation prediction network has comparable video quality compared to other algorithms. The minimum, average, and maximum of the consumed bandwidth are given in Table VII, which shows that our algorithm consumes less bandwidth. Combining these two tables, we observe that, with $\tau = 1\%$, our prediction network saves about 9% of the bandwidth, while achieving similar video quality. In addition, it reduces about 8 Mbps in bandwidth consumption while only sacrifices less than 1 dB video quality on average when $\tau = 10\%$. Fig. 16 plots the video quality of individual traces from all viewers in WiFi networks. This figure shows that our solution leads to the points located at the left to the points from other solutions, which confirms the above observations.

Our fixation prediction network has short training and prediction time. We run the training and prediction process on an Intel 32-core Xeon server with a Nvidia 1080Ti GPU. We report the training time of each parameter setting in Table VIII. We note that some training sessions take less than 1 hour. This

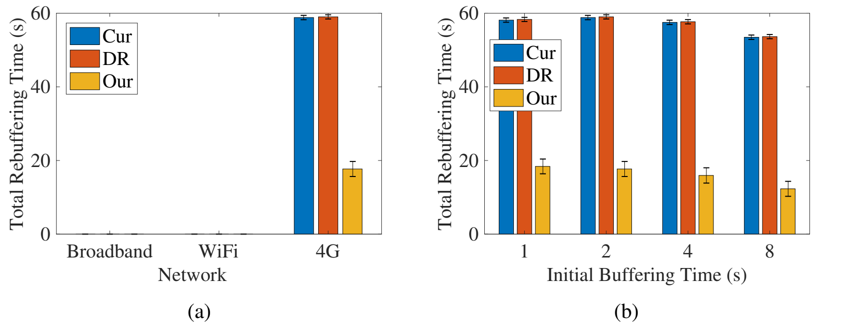


Fig. 14. The total rebuffering time under: (a) different networks and (ii) different initial buffering time.

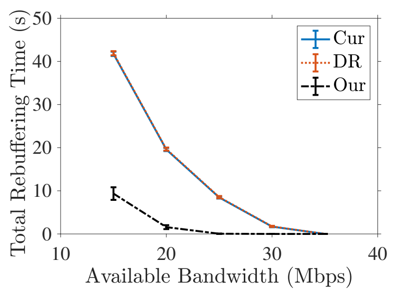


Fig. 15. The total rebuffering time under different available bandwidth. The curves from Cur and DR overlap.

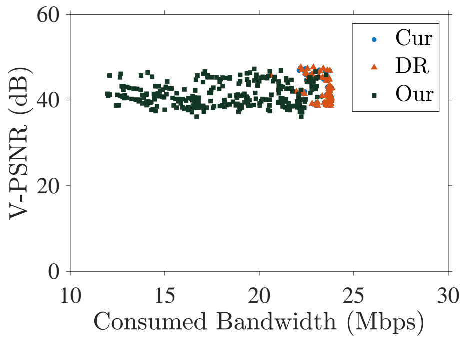


Fig. 16. The relation between video quality and bandwidth consumption observed under individual traces in WiFi networks.

TABLE VIII
THE TRAINING TIME IN MINUTE

No. Neurons		256		512		1024		2048	
Dropout		T	F	T	F	T	F	T	F
No. Layers	1	105	109	78	108	120	125	67	103
	2	47	50	95	59	50	71	89	83
	3	83	105	47	77	119	65	123	69

may be attributed to the early stop (see Sec. V) adopted by the training process. Fig. 17 plots the prediction time from a sample trace. This figure shows that the prediction time of the next segment is always less than 90 ms for this trace. The average and maximum running time for each prediction across all testing traces are 88.74 ms and 124 ms, respectively, which

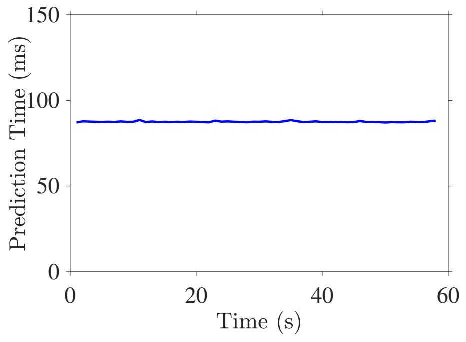


Fig. 17. The prediction time from a sample trace.

are relatively short compared to, for example, 2-sec segments.

TABLE IX
THE MOS AND CONSUMED BANDWIDTH FROM THREE SAMPLE TRACES

Trace	MOS			Bandwidth (Mbps)		
	Cur	DR	Our	Cur	DR	Our
Roller Coaster	3.14	2.86	2.86	24.35	24.33	15.32
Hog Rider	3.43	3.43	3.43	24.18	24.21	13.32
SFR Sport	3.14	3.00	3.29	24.19	24.25	13.71
Average	*3.24*	*3.10*	*3.20*	*24.24*	*24.26*	*14.12*

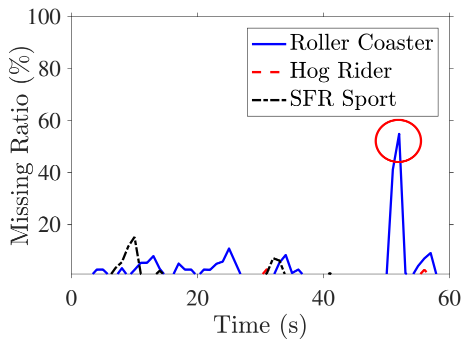


Fig. 18. The missing ratio of our prediction algorithm on the considered random traces. Roller Coaster, in general, suffers from higher missing ratio.

D. A Small-Scale User Study

We conduct a user study to understand the correlation between V-PSNR and user experience. We randomly select three sample user traces (one for each video category) from the testing set (see Sec. V). The considered videos are: (i) Roller Coaster, (ii) Hog Rider, and (iii) SFR Sport. We use our prediction algorithm and baseline algorithms to predict the fixations with a missing ratio of $\tau < 10\%$. No error concealment is performed. For each prediction algorithm, we generate viewport videos from the predicted tiles according to the sensor data (yaw, roll, and pitch) from the trace, where the viewports are in 1066×1066 resolution (equivalent to $100^\circ\times100^\circ$). In total, nine (three traces with three prediction algorithms) viewport videos are generated. We play the viewport videos to seven subjects, who provide user experience scores in the 1 (worst) - 5 (best) scale. We disable the inertial

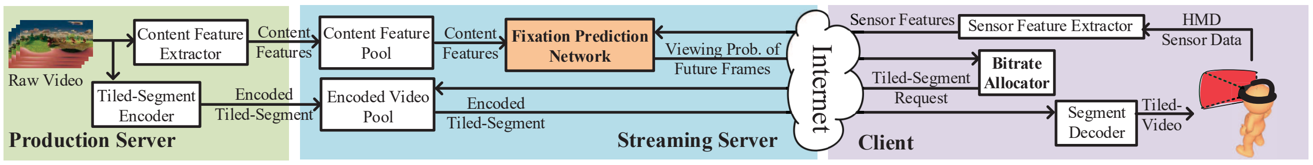


Fig. 19. A reference framework of the 360° video streaming systems with the proposed fixation prediction network. A pull-based system is shown.

sensors on the HMDs, so that all subjects watch exactly the same viewport videos following the head movements of the sample user trace. For each pair of the trace and prediction algorithm, we compute the Mean Opinion Score (MOS) across the user experience scores from all subjects.

We report the MOS and consumed bandwidth in Table IX. This table shows that, compared to the baseline algorithms, our solution achieves very similar average MOS for all sample videos, yet saves about 41% bandwidth on average. This table also reveals that our prediction algorithm suffers from an inferior MOS score, with Roller Coaster, than the Cur algorithm. We plot the missing ratio of our algorithm on the considered three traces over time in Fig. 18. This figure shows that our algorithm leads to higher missing ratio on Roller Coaster compared to the other two traces. A deeper investigation shows that the higher missing ratio is due to the higher maximum head rotation speed in Roller Coaster trace. More precisely, it is about 35.37 degree/s in yaw direction, which is higher than that of other two traces by up to 14 degree/s. In particular, the maximum rotation speed occurs at 50 s, which is inline with the peak (highlighted with the circle) in Fig. 18. We cross check the V-PSNR values of Our and Cur algorithms with Roller Coaster: which are 34.37 and 38.35 dB, respectively. The V-PSNR values are consistent with our MOS results.

In summary, our preliminary user study results (about 40% bandwidth saving) are inline with our earlier experiments using V-PSNR. Our user study also reveals that improving the fixation prediction accuracy for even lower missing ratio is important. It is, however, possible to cope with the imperfect fixation prediction using some innovative networking tools. For example, recent studies [76], [77] leverage emerging network protocols, such as HTTP/2 and QUIC [78], to stream critical tiles over high-priority concurrent streams, in order to avoid missing tiles (holes).

VIII. Conclusion and Future Work

360° video streaming has become increasingly popular. However, their extremely large file sizes impose high loads on networks. In this article, we propose to leverage both sensor and content features to predict the viewing probability of each tile in future frames, in order to reduce the network loads and improve the video quality. Several novel enhancements are proposed to improve the prediction performance, including generating virtual viewports, considering future content, reducing the feature sampling rate, and training with larger datasets. We conduct extensive simulations using real traces to quantify the performance of our proposed solution. The evaluation results show that compared to other algorithms, our proposed fixation prediction network achieves comparable video quality while: (i) saving about 8 Mbps in bandwidth consumption and (ii) cutting the rebuffering time by 40-sec. Besides, our proposed prediction network estimates tile viewing probability in almost real-time.

We note that the proposed fixation prediction network is only *a* component of 360° video streaming systems. Integrating it with existing RTP or DASH video streaming systems require additional efforts beyond what have been done in Sec. VII. Fig. 19 shows a reference framework of 360° video streaming systems. It contains three entities: (i) the production server, which compresses and analyzes raw 360° videos into encoded videos and content features, (ii) the streaming server, which helps the client to predict the viewer fixation using the fixation prediction network, and (iii) the client, which sends the HMD sensor readings to the streaming server and renders 360° videos to the HMD viewer.

There are quite a few open challenges for researchers and developers to turn this reference framework into a reality, including:

- *Bitrate allocator:* It is a component in the HMD client, which distributes the available bitrate among tiles following the predicted viewing probability and network conditions. Designing an optimal bitrate allocator is one of our current tasks (see Sec. VII-C).
- *DASH enhancements:* Additional control messages are exchanged between the streaming server and the client. For example, the tile viewing probability is predicted at the fixation prediction network on the streaming server, which needs to be sent to the client. Another example is the sensor features, which needs to be sent from the client to the streaming server. Designing a DASH-compatible way for the exchanging of control messages is another future task.
- *Streaming protocols:* The pull-based protocol, e.g., HTTP/1.1, adopted by conventional DASH leads to higher latency because of the HTTP requests generated from client. Thus, a push-based protocol, e.g., RTP and HTTP/2.0, may be leveraged to reduce the latency by getting rid of some extra control messages. Besides, both HTTP/2 and QUIC [78] support multiplexed and prioritized streams that can be adopted to avoid the negative impacts of inaccurate fixation prediction.
- *User study procedure:* High resolution and low latency are becoming crucial in 360° video streaming systems to provide immersive experience. However, a trade-off between them exists due to the limitations on networking and computing resources. Therefore, a comprehensive user study procedure is needed to better quantify the

impacts of different streaming parameters (e.g., resolution, frame rate, latency, and content types) on viewer experience. Some preliminary work has been done, e.g., in Yao et al. [79].

REFERENCES

[1] "Facebook Oculus Rift," 2016, https://www.oculus.com.

[2] "HTC Vive," 2016, https://www.htcvive.com.

[3] "Samsung Gear VR," 2016, http://www.samsung.com/global/galaxy/gear-vr.

[4] "Worldwide augmented and virtual reality headset market expected to grow at a compound annual rate of 58%, reaching 99.4 million units in 2021, according to IDC," 2017, http://www.idc.com/getdoc.jsp?containerId=prUS42371517.

[5] "Introducing Facebook 360 for Gear VR," 2017, https://newsroom.fb.com/news/2017/03/introducing-facebook-360-for-gear-vr/.

[6] "360-degree video case study," 2017, https://www.magnifyre.com/360-degree-video-case-study/.

[7] T. El-Ganainy and M. Hefeeda, "Streaming virtual reality content," *arXiv preprint arXiv:1612.08350*, 2016.

[8] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 969–977, 2013.

[9] A. Borji, M. Cheng, H. Jiang, and J. Li, "Salient object detection: A survey," *arXiv preprint arXiv:1411.5878*, 2014.

[10] Y. Kavak, E. Erdem, and A. Erdem, "A comparative study for feature integration strategies in dynamic saliency estimation," *Signal Processing: Image Communication*, vol. 51, pp. 13–25, 2017.

[11] M. Assens, X. Giro-i Nieto, K. McGuinness, and N. O'Connor, "Saltinet: Scan-path prediction on 360 degree images using saliency volumes," in *Proc. of IEEE International Conference on Computer Vision (ICCV'17)*, 2017, pp. 2331–2338.

[12] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, "Salnet360: Saliency maps for omni-directional images with CNN," *Signal Processing: Image Communication*, vol. 69, pp. 26–34, 2018.

[13] J. Kopf, "360 video stabilization," *ACM Transactions on Graphics*, vol. 35, no. 6, 2016.

[14] K. Lin, S. Liu, L. Cheong, and B. Zeng, "Seamless video stitching from hand-held camera inputs," in *Computer Graphics Forum*, vol. 35, no. 2. Wiley Online Library, 2016, pp. 479–487.

[15] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu, "Fixation prediction for 360° video streaming in head-mounted virtual reality," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*, 2017, pp. 67–72.

[16] C. Fan, W. Lo, Y. Bai, and C. Hsu, "A survey on 360 video streaming: Acquisition, transmission, and display," *accepted to appear at ACM Computing Surveys*, 2019.

[17] Y. Ma and H. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proc. of the ACM International Conference on Multimedia (MM'03)*, 2003, pp. 374–381.

[18] F. Liu and M. Gleicher, "Region enhanced scale-invariant saliency detection," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'06)*, 2006, pp. 1477–1480.

[19] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[20] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[21] K. Wang, L. Lin, J. Lu, C. Li, and K. Shi, "PISA: Pixelwise image saliency by aggregating complementary appearance contrast measures with edge-preserving coherence," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3019–3033, 2015.

[22] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum, "Learning to detect a salient object," *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 33, no. 2, pp. 353–367, 2011.

[23] G. Li and Y. Yu, "Visual saliency based on multiscale deep features," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, 2015, pp. 5455–5463.

[24] A. Mavlankar and B. Girod, "Video streaming with interactive pan/tilt/zoom," in *Signals and Communication Technology*, 2010, pp. 431–455.

[25] T. Nguyen, M. Xu, G. Gao, M. Kankanhalli, Q. Tian, and S. Yan, "Static saliency vs. dynamic saliency: a comparative study," in *Proc. of ACM International Conference on Multimedia (MM'13)*, 2013, pp. 987–996.

[26] S. Chaabouni, J. Benois-Pineau, and C. Amar, "Transfer learning with deep networks for saliency prediction in natural video," in *Proc. of IEEE International Conference on Image Processing (ICIP'16)*, 2016, pp. 1604–1608.

[27] T. Alshawi, Z. Long, and G. AlRegib, "Understanding spatial correlation in eye-fixation maps for visual attention in videos," in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'16)*, 2016, pp. 1–6.

[28] Y. Rai, J. Gutiérrez, and P. Le Callet, "A dataset of head and eye movements for 360 degree images," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, 2017, pp. 205–210.

[29] H. Cheng, C. Chao, J. Dong, H. Wen, T. Liu, and M. Sun, "Cube padding for weakly-supervised saliency prediction in 360 videos," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*, 2018, pp. 1420–1429.

[30] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360 videos," in *Proc. of the European Conference on Computer Vision (ECCV'18)*, 2018, pp. 488–503.

[31] 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. of IEEE International Conference on Multimedia and Expo (ICME'18)*, 2018, pp. 1–6.

[32] 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 Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[33] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction," in *Proc. of ACM International Conference on Multimedia (MM'18)*, 2018, pp. 1190–1198.

[34] V. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen, "Tiling in interactive panoramic video: Approaches and evaluation," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1819–1831, 2016.

[35] O. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Lim, "MPEG-DASH SRD: spatial relationship description," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, 2016, pp. 5:1–5:8.

[36] J. Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, 2016, pp. 41:1–41:3.

[37] L. D'Acunto, J. Berg, E. Thomas, and O. Niamut, "Using MPEG-DASH SRD for zoomable and navigable video," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'16)*, 2016, pp. 34:1–34:4.

[38] C. Zhou, Z. Li, and Y. Liu, "A measurement study of oculus 360 degree video streaming," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, 2017, pp. 27–37.

[39] W. Lo, C. Fan, S. Yen, and C. Hsu, "Performance measurements of 360° video streaming to head-mounted displays over live 4G cellular networks," in *Proc. of Asia-Pacific Network Operations and Management Symposium (APNOMS'17)*, 2017, pp. 205–210.

[40] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, 2017, pp. 261–271.

[41] P. Alface, M. Aerts, D. Tytgat, S. Lievens, C. Stevens, N. Verzijp, and J. Macq, "16k cinematic VR streaming," in *Proc. of ACM International Conference on Multimedia (MM'17)*, 2017.

[42] M. Xiao, C. Zhou, Y. Liu, and S. Chen, "Optile: Toward optimal tiling in 360-degree video streaming," in *Proc. of ACM International Conference on Multimedia (MM'17)*, 2017.

[43] F. Duanmui, E. Kurdoglu, S. Hosseini, Y. Liu, and Y. Wang, "Prioritized buffer control in two-tier 360 video streaming," in *Proc. of the Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network'17)*, 2017, pp. 13–18.

[44] Y. Sanchez, R. Skupin, C. Hellge, and T. Schierl, "Spatio-temporal activity based tiling for panorama streaming," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'17)*, 2017, pp. 61–66.

[45] S. Petrangeli, F. Turck, V. Swaminathan, and M. Hosseini, "Improving virtual reality streaming using HTTP/2," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, 2017, pp. 225–228.

[46] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. of ACM Workshop on All

Things Cellular: Operations, Applications and Challenges, 2016, pp. 1–6.

[47] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-Prediction-Based multicast for 360-Degree video transmissions," in *Proc. of IEEE International Conference on Sensing, Communication, and Networking (SECON'17)*, June 2017, pp. 1–9.

[48] W. Lo, C. Huang, and C. Hsu, "Edge-assisted rendering of 360° videos streamed to head-mounted virtual reality," in *Proc. of IEEE International Symposium on Multimedia (ISM'18)*, 2018, pp. 44–51.

[49] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[50] K. Ngo, R. Guntur, and W. Ooi, "Adaptive encoding of zoomable video streams based on user access pattern," in *Proc. of ACM Conference on Multimedia Systems (MMSys'11)*, 2011, pp. 211–222.

[51] H. Wang, M. Chan, and W. Ooi, "Wireless multicast for zoomable video streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 12, no. 1, pp. 5:1–5:23, 2015.

[52] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon, "Transcoding live adaptive video streams at a massive scale in the cloud," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'15)*, 2015, pp. 49–60.

[53] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," in *Proc. of Conference on Virtual Reality Conference (VR'10)*, 2010, pp. 211–218.

[54] T. Mikolov, M. Karafiat, L. Burget, J.Cernocky, and S. Khudanpur, "Recurrent neural network based language model." in *Proc. of Conference on International Speech Communication Association (Interspeech'11)*, 2010, pp. 1045–1048.

[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[56] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C. Hsu, "360° video viewing dataset in head-mounted virtual reality," in *Proc. of ACM International Conference on Multimedia Systems (MMSys'17)*, 2017, pp. 211–216.

[57] C. Huang, K. Chen, D. Chen, H. Hsu, and C. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 10:1–10:24, no. 1, Jan 2014.

[58] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. of International Conference on Computational StatisticsParis France (COMPSTAT'10)*, 2010, pp. 177–186.

[59] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of IEEE International Conference on Communications (ICC'17)*, 2017.

[60] "Next-generation video encoding techniques for 360 video and VR," 2016, https://goo.gl/UNGIwT.

[61] C. Fu, L. Wan, T. Wong, and C. Leung, "The rhombic dodecahedron map: An efficient scheme for encoding panoramic video," *IEEE Transactions on Multimedia*, vol. 11, no. 4, pp. 634–644, 2009.

[62] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[63] M. Cornia, L. Baraldi, G. Serra, and R. Cucchiara, "A deep multi-level network for saliency prediction," in *Proc. of International Conference on Pattern Recognition (ICPR'16)*, 2016.

[64] E. Hjelmås and B. Low, "Face detection: A survey," *Computer vision and image understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[65] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[66] C. Galleguillos and S. Belongie, "Context based object categorization: A critical survey," *Computer Vision and Image Understanding*, vol. 114, no. 6, pp. 712–722, 2010.

[67] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, 2009.

[68] T. Lin, M. Maire, s. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014, pp. 740–755.

[69] "NS-3 network simulator," 2018, http://www.nsnam.org/.

[70] "An MPEG/DASH client-server module for simulating rate adaptation mechanisms over HTTP/TCP," 2018, https://github.com/djvergad/dash.

[71] M. Viitanen, A. Koivula, A. Lemmetti, A. Yla-Outinen, J. Vanne, and T. Hämäläinen, "Kvazaar: Open-source hevc/h.265 encoder," in *Proc. of ACM International Conference on Multimedia (MM'16)*, 2016, pp. 1179–1182.

[72] "The Zettabyte Era: Trends and Analysis," 2017, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html.

[73] "Global State of Mobile Networks (February 2017)," 2017, https://opensignal.com/reports/2017/02/global-state-of-the-mobile-network.

[74] M. Yu, H. Lakshman, and B. Girod, "A framework to evaluate omnidirectional video coding schemes," in *Proc. of IEEE International Symposium on Mixed and Augmented Reality*, 2015, pp. 31–36.

[75] JVET of ITU-T, "Joint video exploration team software, 360Lib," 2017, https://jvet.hhi.fraunhofer.de/svn/svn_360Lib/tags/360Lib-2.0.1/.

[76] S. Yen, C. Fan, and C. Hsu, "Streaming 360° videos to head-mounted virtual reality using dash over quic transport protocol," in *Proc. of Packet Video Workshop (PV'19)*, 2019.

[77] M. B. Yahia, Y. Le Louedec, G. Simon, and L. Nuaymi, "Http/2-based streaming solutions for tiled omnidirectional videos," in *Proc. of IEEE International Symposium on Multimedia (ISM'18)*, 2018, pp. 89–96.

[78] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Y. F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi, "The QUIC transport protocol: Design and internet-scale deployment," in *Proc. of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'17)*, 2017, pp. 183–196.

[79] S. Yao, C. Fan, and C. Hsu, "Towards quality-of-experience models for watching 360° videos in head-mounted virtual reality," in *Proc. of IEEE International Conference on Quality of Multimedia Experience (QoMEX'19)*, 2019, pp. 1–6.

Ching-Ling Fan is a Ph.D. student at National Tsing Hua University in HsinChu, Taiwan. She received her B.S. degree in Computer Science from National Tsing Hua University. Her research interests are in virtual reality, immersive video streaming and multimedia networking.

Shou-Cheng Yen is a Master student at National Tsing Hua University in Hsinchu, Taiwan. He received his B.S. degree in Electrical Engineering and Computer Science from National Tsing Hua University. His research interests are in virtual reality, immersive video streaming and transport protocols.

Chun-Ying Huang is a Professor at Department of Computer Science, National Chiao Tung University. He received his Ph.D. in Electrical Engineering Department from National Taiwan University in 2007. Dr. Huang joined National Chiao Tung University as an Associate Professor in 2016, and is a Professor since 2018. His research interests include system security, multimedia networking, and mobile computing. Dr. Huang is a member of ACM and IEEE. He was a recipient of the 2014 ACM Taipei/Taiwan Chapter K. T. Li Young Researcher Award.

Cheng-Hsin Hsu (S'09–M'10–SM'16) received the B.Sc. degree in Mathematics and M.Sc. degree in Computer Science and Information Engineering from National Chung-Cheng University, Taiwan. He received the M.Eng. degree in Electrical and Computer Engineering from the University of Maryland, College Park and the Ph.D. degree in Computing Science from Simon Fraser University, Burnaby, BC, Canada. He is an Associate Professor at National Tsing Hua University in Hsin Chu, Taiwan.