Predictive Adaptive Streaming to Enable Mobile 360-degree and VR Experiences

Xueshi Hou, *Student Member, IEEE*, Sujit Dey, *Fellow, IEEE*, Jianzhong Zhang, *Fellow, IEEE*, and Madhukar Budagavi, *Senior Member, IEEE*

***Abstract*—As 360-degree videos and virtual reality (VR) applications become popular for consumer and enterprise use cases, the desire to enable truly mobile experiences also increases. Delivering 360-degree videos and cloud/edge-based VR applications require ultra-high bandwidth and ultra-low latency [1], challenging to achieve with mobile networks. A common approach to reduce bandwidth is streaming only the field of view (FOV). However, extracting and transmitting the FOV in response to user head motion can add high latency, adversely affecting user experience. In this paper, we propose a predictive adaptive streaming approach, where the predicted view with high predictive probability is adaptively encoded in relatively high quality according to bandwidth conditions and transmitted in advance, leading to a simultaneous reduction in bandwidth and latency. The predictive adaptive streaming method is based on a deep-learning-based viewpoint prediction model we develop, which uses past head motions to predict where a user will be looking in the 360-degree view. Using a very large dataset consisting of head motion traces from over 36,000 viewers for nineteen 360-degree/VR videos, we validate the ability of our predictive adaptive streaming method to offer high-quality view while simultaneously significantly reducing bandwidth.**

***Index Terms*—Virtual reality, video streaming, 360-degree video.**

I. Introduction

RECENTLY, 360-degree videos and virtual reality (VR) applications have attracted significant interest in various fields, including entertainment, education, manufacturing, transportation, healthcare, and other consumer-facing services. These applications exhibit enormous potential as the next generation of multimedia content to be adopted by enterprises and consumers via providing richer, more engaging and more immersive experiences. According to market research [2], VR and augmented reality (AR) ecosystem is predicted to be an $80 billion market by 2025, roughly the size of the desktop PC market today. However, several key hurdles need to be overcome for businesses and consumers to get fully on board with VR technology [3], such as cheaper price and compelling content, and most importantly a truly mobile VR experience, in line with the expectation and adoption of mobile experiences in almost all consumer and enterprise verticals today. Of particular interest is how to develop mobile (wireless and lightweight) head-mounted displays (HMDs), and how to enable VR experience on the mobile HMDs using bandwidth constrained mobile networks, while satisfying the ultra-low latency requirements.

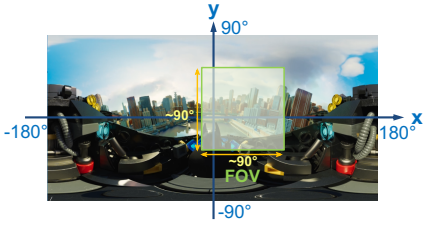


Fig. 1. FOV in a 360-degree view.

Current widely used HMDs approximately include three types [4]: PC VR, console VR, mobile VR. Specifically, PC VR is tethered with PC [5], [6]; console VR is tethered with a game console [7]; mobile VR is untethered with PC/console but with a smartphone inside [8], [9]. Since all the above HMDs perform rendering locally either on a smartphone tethered with the HMD, or on a computer/console tethered to the HMD, today's user experience lacks portability (when using a heavy HMD tethered to a smartphone) or mobility (when tethered to computer/console). To enable lighter mobile VR experience, we propose a cloud/edge-based solution. By performing the rendering on cloud/edge servers and streaming videos to users, we can complete the computation-intensive tasks on the cloud/edge server and thus enable mobile VR with lightweight VR glasses. The most challenging part of this solution is the ultra-high bandwidth and ultra-low latency requirements, since streaming 360-degree video causes tremendous bandwidth consumption and good user experiences require ultra-low latency (<20ms) [1], [10]. Various techniques have been developed for video content delivery such as adaptive streaming algorithms [11], [12], mobile edge caching placement algorithms [13] and hybrid multicast-unicast schemes [14], [15], but these approaches are designed for ordinary videos, and thus have not considered the scenario of 360-degree video streaming.

Motivated by this challenge, in this paper, we propose a novel approach to enable mobile VR with prediction for head motions. Our basic idea comes from the following observations: the field of view (FOV) is $90°\times90°$ for popular HMDs while the 360-degree view is $360°\times180°$ in size (as is shown in Fig. 1). A common approach to reduce bandwidth is streaming only the FOV. However, extracting and transmitting the FOV in response to user head motion can add high latency, adversely

Manuscript received March 15, 2019; revised January 9, 2020; accepted March 24, 2020. Date of publication xxxxxx, 2020; date of current version xxxxxx, 2020. This work was supported in part by the Center for Wireless Communications at University of California, San Diego. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Xiaokang Yang. *(Corresponding author: Xueshi Hou.)*

X. Hou and S. Dey are with Mobile Systems Design Lab, Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mails: x7hou@ucsd.edu; dey@ece.ucsd.edu).

J. Zhang and M. Budagavi are with Standards and Mobility Innovation Lab, Samsung Research America, 6625 Excellence Way, Plano, TX 75023 USA (e-mails: jianzhong.z@samsung.com; m.budagavi@samsung.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2020.xxxxxxx

affecting user experience. This motivates us to predict head motions. With prediction for head motion, our approach can address both bandwidth and latency challenges.

If we can predict head motion of users in the near future, we can achieve predictive rendering (in case of VR) and encoding on the edge device, and then stream the predicted view (i.e., a 360-degree view with more bits for the FOV tiles and less for non-FOV tiles) to the HMD in advance. Thus, latency needed will be significantly reduced since the view is delivered and pre-buffered on the HMD. Moreover, in order to address the challenge of dynamically varying network bandwidth conditions, we use the viewpoint prediction to guide the bitrate adaptation of the stream (i.e., allocate more bits for the FOV tiles and less for non-FOV tiles to compose the predicted view). Thus, good user experience can be achieved under the dynamically varying network bandwidth conditions. Note that since *viewpoint* is defined as the center of FOV, prediction for head motions is equivalent to viewpoint prediction in this case. The main contributions of this paper can be summarized as follows:

- We propose a new approach to enable truly mobile VR using wireless HMDs, where the rendering is performed on edge devices, and the ultra-low latency and high bandwidth requirements are addressed through a novel predictive adaptive streaming approach involving viewpoint prediction.
- We develop a viewpoint prediction method using deep learning to predict where a user will be looking into in the 360-degree view based on their past behavior. Using a very large dataset of real head motion traces from VR applications, we show the feasibility of our long short-term memory (LSTM) model with high accuracy.
- To address fluctuating and constrained wireless bandwidth available, we propose a novel predictive adaptive streaming algorithm. Given the available bandwidth constraint, it selects proper video encoding settings for each tile based on the viewpoint prediction such that user experience is maximized, i.e., PSNR in user FOV is maximized, where user FOV is defined as the actual user field of view in size of $90°\times90°$.
- While adaptive streaming of 360-degree and VR videos has been proposed before, to the best of our knowledge, this is the first predictive adaptive streaming method proposed in the literature. Using a large-scale real head motion trace dataset, we demonstrate significant bandwidth savings while ensuring very high PSNR in the user FOV. We also demonstrate significant quality, bandwidth and network capacity benefits compared to streaming without bitrate adaptation, and a recent adaptive streaming method which does not utilize prediction like our method does.

Note that a preliminary version of our work was published recently at a workshop [16], where we report on the predictive LSTM model and some preliminary results. In this article, we extend our approach by proposing a smart real-time predictive adaptive streaming algorithm, improving our proposed view generation strategy and conducting experiments on various 360-degree/VR videos.

The remainder of the paper is organized as follows. In Section II, we review related work. Section III introduces the system overview and problem definition. Section IV describes our dataset and its characteristics. Section V and Section VI describe our proposed predictive LSTM model and predictive adaptive streaming algorithms. We present our experimental results in Section VII and conclude our work in Section VIII.

II. Related Work

In this section, we review current work in the following topics related to our research.

FOV-guided streaming: Current FOV-guided 360-degree video streaming studies mainly consist of two types to address bandwidth challenge: *tiling* and *versioning* [17]. As for *tiling*, 360-degree video is spatially divided into *tiles* and only tiles within FOV are streamed at high quality while remaining tiles are streamed at lower qualities or not delivered at all [18]–[20]. In terms of *versioning*, the 360-degree video is encoded into multiple versions which have a different high-quality region, and viewers receive the appropriate version based on their own viewing direction [21]. The above methods are based on knowing the actual viewpoint of the user as it happens. Hence, while they can reduce bandwidth requirement of streaming 360-degree video, they cannot reduce the latency as rendering and encoding still need to be done in real-time after user FOV is determined. In contrast, our method aims to predict the user viewpoint and deliver the predicted FOV in advance, thus eliminating the need for rendering (in case of VR) or extracting FOV (in case of 360-degree videos) and transmitting from servers over mobile networks after the user has changed viewpoint, and hence addressing the ultra-low latency requirement besides significantly reducing bandwidth.

Streaming with novel schemes: Some studies [22], [23] recognized ultra-low latency and ultra-high bandwidth challenges in the transmission of 360-degree videos and VR applications. [22] proposed a multipath cooperative routing scheme with software-defined networking architecture to reduce the delay and energy consumption of VR wireless transmissions in 5G small cell networks. [23] also studied a new link scheduling and adaptation scheme to reduce system latency and energy consumption in VR video streaming. By contrast, we are addressing the latency and bandwidth challenges with our proposed adaptive streaming approach, which can be applied on any existing wireless network without any special modification or provisioning of the network required by [22], [23].

Sequence prediction: Viewpoint prediction and related mobility prediction (since viewpoint prediction is equivalent to prediction for viewpoint mobility) both belong to the problem of *sequence prediction*, which is defined as predicting the next value(s) given a historical sequence [24]. We roughly summarize the approaches for sequence prediction as two types: traditional machine learning and deep learning methods. On one hand, traditional machine learning approaches such as randomized decision trees and forest [25], [26] have proven fast and effective performance for many sequence prediction tasks [27], [28]. Bootstrap-aggregated decision trees (BT) [25]

is one of the most efficient methods among them. On the other hand, deep learning methods such as recurrent neural networks (RNN) and their variants including LSTM networks [29] and gated recurrent units (GRU) [30] have proven to be successful for sequence prediction tasks [31], [32]. Apart from RNN and their variants, there are also some studies [33], [34] using deep neural networks including deep belief networks (DBN) [35] and stacked sparse autoencoders (SAE) [36] to achieve sequence prediction. Among these deep learning methods, LSTM recurrent neural networks show a good potential to capture the transition regularities of human movements since they have memory to learn the temporal dependence between observations (i.e., training data) [31], [32]. Inspired by this advantage, we design an LSTM model which can learn general head motion pattern and predict the future viewpoint position based on the past traces. Our prediction model shows promising results on a large-scale real head motion trace dataset.

Head motion prediction: Some studies [37]–[39] explore the feasibility of head motion prediction. Most of them used relatively simple models with euler angles or angular velocity as input without a tile-based perspective. Our proposed multi-layer LSTM model benefits from the design of our tile-based representation and the large-scale dataset, and thus performs better. Some studies [40], [41] also investigate more complicated prediction models to benefit 360-degree video experience. [40] achieves gaze prediction using the saliency maps and gaze trajectories (collected by an extra eye tracker), while [41] studies fixation prediction employing the saliency maps, motion maps, and head motion. However, the gaze prediction technique [40] cannot be implemented directly since most of current HMDs cannot track gaze, and the prediction models in [40], [41] are more time-consuming (i.e., 47ms and 50ms respectively) than our proposed prediction model (i.e., <2ms) because it needs more processing time of extracting image saliency maps and motion maps from videos. Our proposed prediction method achieves high accuracy in real time by using only head motion information, and thus are more efficient and concise for our current 360-degree video streaming scenarios to address the ultra-low latency challenge. Furthermore, these current studies [37]–[41] do not further consider the possibility of doing adaptive streaming using head motion prediction.

Adaptive streaming: Several techniques have been proposed for adaptive streaming for 360-degree videos [21], [42], [43]. [42] proposed to stream the 360-degree video based on average navigation likelihood, while [43] considered optimization based on expected quality distortion, spatial quality variance and temporal quality variance to do the 360-degree video streaming. [21] proposed to stream one of multiple representations of the same 360-degree video, where each representation has a different quality emphasized region in the 360-degree view, such that bitrate fits the available throughput and a full quality region matches user's viewing. However, the above techniques [21], [42], [43] do not consider the problem of adaptive streaming in advance using prediction of user head motion to minimize latency.

To the best of our knowledge, we are the first to consider the problem of streaming predictively in advance of the actual user view so as to ensure ultra-low latency requirement of 360-degree video/VR, and using viewpoint prediction to guide the bitrate adaptation of the stream so that the highest user experience can be achieved under the dynamically varying network bandwidth conditions.

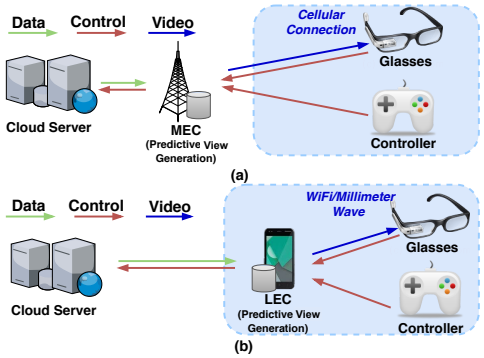


Fig. 2. System overview.

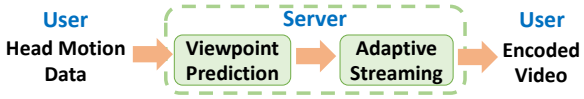


Fig. 3. Proposed predictive adaptive streaming procedure.

III. System Overview

In this section, we present an overview of our system. Note that our predictive adaptive streaming approach works for both 360-degree videos and cloud/edge-based VR applications, since it refers to (i) adaptively selecting encoded tiles (in case of 360-degree videos), and (ii) rendering the view and adaptively encoding tiles (in case of cloud/edge-based VR) depending on the predicted probability of each tile to belong to the user's actual FOV and bandwidth conditions. User's head motion as well as other controlling commands will be sent to the edge device, which performs viewpoint prediction and predictive rendering. The edge device can be either a Mobile Edge Computing node (MEC) in the mobile radio access or core network (Fig. 2(a)), or a Local Edge Computing node (LEC) located in the user premises or even his/her mobile device (Fig. 2(b)). Note that each of the above choices has tradeoffs. Use of MEC will allow for greater mobility of the VR user as compared to LEC, unless LEC is the user's mobile device, in which case the additional (computing) challenge of having to do predictive view generation in the mobile device will need to be addressed. On the other hand, use of MEC will add to more transmission delay of the rendered video than the use of LEC. Use of cloud servers can also be considered to perform predictive view generation; this will allow complete mobility of VR users but will be more challenging in decreasing latency than the use of either MEC or LEC. This paper will not specifically address the above tradeoffs and select either MEC or LEC. Instead, the predictive adaptive streaming technique we propose will apply to either of the edge device options. Note that the primary novelty of our approach in addressing the ultra-low latency requirement is in accurately predicting the user's view in advance and pre-delivering the predictive view so additional rendering (in case

TABLE I
VR DATASET STATISTICS.

Categories	#Video	Video Instances
Movie Trailer	6	Kong VR, Batman Movie
Documentary	6	Fashion Show, Life on Mars
Scenery	4	Whale Encounter, Floating Markets
Entertainment	3	Roller Coaster, Bungee Jump

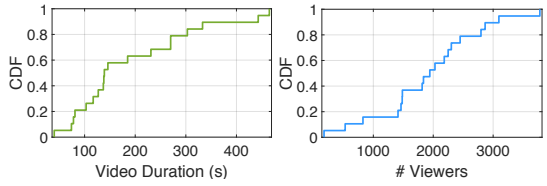


Fig. 4. Statistics of dataset.

of VR) or extracting FOV (in case of 360-degree videos) and transmission time is avoided; our approach does not make any special use of edge devices, except that use of edge devices is recommended as opposed to cloud computing devices so as to reduce additional round-trip transmission latency between the computing device and the HMD. Based on past few seconds of head motion and control data received from the user and using the viewpoint prediction model developed, the edge device will perform predictive adaptive streaming algorithm, and stream the *predicted view* (i.e., a 360-degree view with more bits for the FOV tiles and less for non-FOV tiles) to the user HMD in advance. Later, the predicted view will be displayed on HMD and latency needed will be significantly reduced since the view is delivered and pre-buffered on the HMD before it is needed. The key to achieving efficient predictive adaptive streaming is to first solve the problem stated below.

Problem Statement: A viewpoint can occur in up to K different tiles in each time point (e.g., every 200ms). We decompose the whole predictive adaptive streaming method into two subtasks: *viewpoint prediction* and *adaptive streaming*, shown in Fig. 3. In *viewpoint prediction*, given previous and current viewpoint locations, our goal is to predict one or multiple tiles that the viewpoint will be in for the next time point. In *adaptive streaming*, according to the prediction results obtained from *viewpoint prediction*, we maximize user experience by smartly selecting proper video encoding (quantization step) settings of the video for each tile under dynamically varying network bandwidth conditions and do the streaming. After that the predicted view will be delivered to users.

IV. DATASET AND ITS CHARACTERISTICS

In this section, we first describe the dataset we use, and then show characteristics of the dataset using certain metrics we define.

To investigate viewpoint prediction in 360-degree videos, we conduct our study on a real head motion trace dataset that was collected by Samsung Electronics Company. The trace consists of head motion data from over 36,000 viewers during the week of November 2 – November 8, 2017, for 19 VR videos. Specifically, the frequency of head pose data was every 200ms on each HMD. The information reported includes the content ID, session timestamp, content timestamp, user ID and euler angles of HMD. The session timestamp and content timestamp refer to the time counted since application launches and the location in the video being played respectively, in

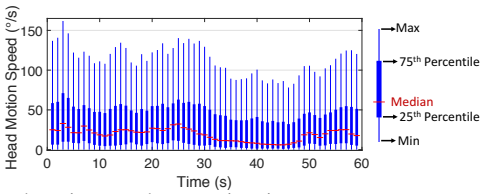


Fig. 5. Head motion speed versus time in *Kong VR*.

milliseconds. Basic statistics of our head motion trace data are shown in Table I. This dataset contains head pose data for 19 online VR videos, which are available on the Samsung VR website [44] and watched by a large number of viewers worldwide using their own HMD. We aggregate these videos by categories, i.e., movie trailer, documentary, scenery and entertainment. In Fig. 4, we plot the cumulative distribution function (CDF) of video duration and the number of viewers for each video. We can observe that over 80% of videos have more than 100s for duration and around 85% of videos have more than 1000 viewers. The large diversity and number of VR videos in the dataset, and the large number of viewers for each video, makes the dataset very suitable for developing and validating our viewpoint prediction method.

To depict key characteristics of the head motion and viewpoint changes in the dataset quantitatively, we offer the following definitions.

Definition 1— Head Motion Vector: Consider a viewer watching a video in certain time-points t_1 and t_2, where $t_1 < t_2$. We have corresponding head poses, which are denoted by $(x(t_1), y(t_1))$ and $(x(t_2), y(t_2))$ respectively. Then the head motion vector $(\triangle x, \triangle y)$ can be represented as $(x(t_2) - x(t_1), y(t_2) - y(t_1))$.

Definition 2— Head Motion Speed: The head motion speed v is defined as the distance the head moved divided by time.

$$v = \frac{\sqrt{\triangle x^2 + \triangle y^2}}{t_2 - t_1} \tag{1}$$

For *Kong VR* video in our dataset, we draw a boxplot in Fig. 5 to analyze head motion speed versus time. Fig. 5 shows head motion speed distribution for over 1500 viewers during 60s with this boxplot. Every dark blue strip represents the head motion speed distribution with an x-axis width of 1 (i.e., a width 1s in video time), whereas the height of a blue strip in the y-axis indicates the interquartile range of the head motion speed, reflecting the variability of the head motion speed. Additionally, each light blue line represents the corresponding maximum and minimum values and red symbols indicate the median head motion speed. From this boxplot, we observe that the distribution exhibits different properties when time changes. For instance, at the time point of 3s, the median head motion speed is as high as 35°/s, while 25 percent of viewers have a head motion speed larger than 75°/s and 75 percent of viewers have a head motion speed larger than 10°/s approximately. At another time point as 45s, median head motion is around 10°/s, while 25 percent of viewers have a head motion speed larger than 47°/s. The whole boxplot presents the challenging situation of predicting head motion since viewers may change viewing direction fast as well as frequently. Moreover, we can see interquartile range of head motion speed during 30-40s is around 5°/s-40°/s while during

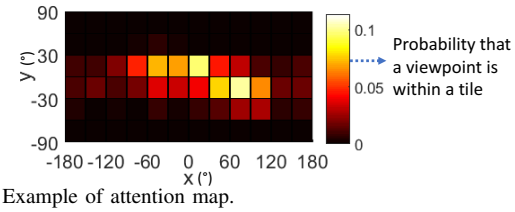
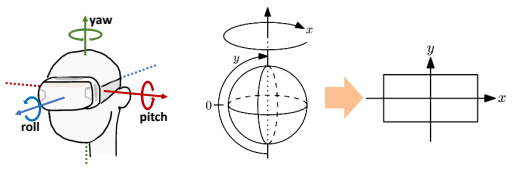


Fig. 6. Example of attention map.



(a) Euler Coordinates (b) Projection

Fig. 7. The viewpoint representation, projected into coordinates in equirectangular map.

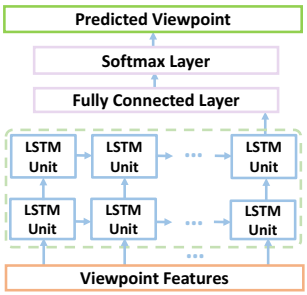


Fig. 8. LSTM model used for viewpoint prediction.

50-60s interquartile range of head motion speed is 10°/s-50°/s approximately. Thus, we take the sequence of 30-40s as an example of *medium motion sequence* and the sequence of 50-60s as an instance of *high motion sequence*. As results presented in Section VII show, viewpoint prediction and FOV generation for *high motion sequences* are relatively more challenging than for *medium motion sequences*, resulting in either less FOV prediction accuracy, or larger FOV and hence less bandwidth savings.

Definition 3— Attention Map: For n viewers, content timestamps cts_1, cts_2 ($cts_1 < cts_2$) denote the video clip the viewers are watching. *Attention map* is defined as a series of probability that a viewpoint is within a tile for n viewers during time-period from cts_1 to cts_2. When we have K tiles in one 360-degree view, we have K elements (i.e., probabilities) in the attention map and the total sum of these probabilities is 1. When there are more tiles with relatively high probabilities, viewpoint prediction will be more challenging since different users may have multiple points of interest and require various FOVs.

Fig. 6 shows an example of attention map, demonstrating users' attention distribution (for over 1500 viewers) during 1s within the high motion sequence in *Kong VR* video [45] mentioned above. The value in legend represents the probability that a viewpoint is within a tile for n viewers during the given time-period. According to the legend, we can observe that the yellow tiles attract most attention and viewers are more likely to look at these areas. The yellow and red colors indicate that the probability that a viewpoint is within the corresponding tile is around 0.1 and 0.05 respectively for all n viewers during the given time-period, meaning this tile is of high interest for users. The attention map in Fig. 6 points to the feasibility of performing viewpoint prediction, since there are always areas attracting more attention than remaining areas within a 360-degree view. On the other hand, the attention map in Fig. 6 shows multiple tiles (as high as 11 tiles) have relatively high probabilities (0.05-0.1), indicating the difficulty of predicting viewpoint accurately. By visualizing a series of consecutive attention maps in a given sequence, we can observe the changes of viewpoint (as well as user attention) continuously. With proposed metrics such as head motion speed and attention map, we can characterize the viewpoint as well as user attention from both temporal and spatial perspectives.

V. Viewpoint Prediction

In this section, we describe our methodology of *viewpoint prediction*. Given previous and current viewpoint locations, our goal is to predict one or multiple tiles that the viewpoint will be in for the next time point. In our dataset, head motion files include user information, timestamp (time in video content), euler angles (pitch, yaw, roll), etc. Euler angles are shown in Fig. 7(a)) and timestamps appear each 200ms. We transform euler angles into the variables x, y in the equirectangular map [46] for 360-degree view, which is presented in Fig. 7(b). Variables x and y are within $(-180, 180]$ and $[-90, 90]$ degrees respectively.

We use tile-based format for viewpoint feature representation. With each grid size as $30°\times30°$, the 360-degree view can be divided into 72 tiles. We select 2s as the prediction time window (i.e., predict viewpoint according to viewpoint traces in past 2s), since it achieves better performance than 3s, 4s and 5s based on our experiments. Note our selection of 2s is in line with the observation made by [21]. For training the model, we design a one-hot encoding representation [47], [48] for viewpoint as a 72×10 matrix V. Each element of V is 0 or 1. The dimensions of V correspond to the 72 tiles in a 360-degree view for possible viewpoint positions, and 10 timestamps corresponding to 2s. Thus, the element $v_{i,j}$ of matrix V equals to 1 when the viewpoint is within the i-th tile at the j-th timestamp, and equals to 0 when viewpoint is not within the corresponding tile. Another simple representation for viewpoint is a 1×10 vector, where each element equals to i when viewpoint is in the i-th tile. With the two representations above, we can obtain viewpoint features from previous and current viewpoint locations.

Inspired by the good performance of LSTM to capture transition regularities of human movements since they have memory to learn the temporal dependence between observations [31], [32], we design a multi-layer LSTM model which can learn general head motion patterns and predict the future viewpoint position based on the past traces. Fig. 8 shows the LSTM model we designed and used in our training, where first and second LSTM layers both consist of 128 LSTM units, and the fully connected layer contains 72 nodes. Our LSTM model predicts the next tile within which the viewpoint will be, given the previous sequence of viewpoint tiles. The outputs are the predicted probabilities over the 72 possible

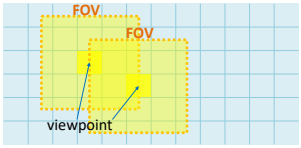


Fig. 9. Example of generating predicted FOV.

tiles. The proposed model learns parameters by minimizing cross-entropy and we train with mini-batches of size 30. Note that the settings including 128 LSTM units, 72 nodes and 30 as mini-batch size are selected during experiments and proved to be good by empirical results.

We can use the viewpoint prediction probabilities of the tiles to generate an FOV, such that the probability that the actual user view in the next time point will be within the predicted FOV is maximized. In that case we will be able to "predictively stream" the generated FOV in advance of the user's actual user view in the next time point, instead of the entire 360-degree/VR video, thus ensuring no additional latency at the next time point, while at the same time minimizing bandwidth consumption of the FOV transmitted (minimizing pixels/bitrate of FOV). We define FOV prediction accuracy as the probability that actual user view will be within the predicted FOV (generated from one or multiple tiles).

In our preliminary work [16], we select m tiles with highest probabilities predicted by the LSTM model, compose the predicted FOV as the combination of FOVs for each selected tile, and transmit the predicted FOV with high quality while leaving the rest of tiles blank. Fig. 9 shows an example of FOV generation when we select the top two highest probability tiles (i.e., $m = 2$) provided by the LSTM model, where yellow area illustrates the predicted FOV consisting of 26 tiles (i.e., the combination of FOVs for two selected tiles). In our current method, we build $120°\times120°$ FOV around the center of the selected tile. By doing this, we can guarantee that when the viewpoint is within the predicted tile, the actual FOV is larger than $90°\times90°$ in size (i.e., $90°\times90°$ when the viewpoint is at the corner of predicted tile and $120°\times120°$ when the viewpoint is in the center of predicted tile). We can use choice of m to achieve the desired trade-off between FOV prediction accuracy and bandwidth consumed in transmitting the predicted FOV.

As we show in Section VII.A, a very high FOV prediction accuracy can be obtained using the above method, while also saving significant bandwidth compared to transmitting the entire 360-degree video. However, the above predictive streaming approach may not work, as the wireless network bandwidth available may not be always sufficient to transmit the predicted FOV with high accuracy (that is, considering high enough number of "m" tiles), thereby necessitating the predictive adaptive streaming approach we propose and describe in Section VI.

VI. Predictive Adaptive Streaming Algorithm

While predictive streaming of FOV generated using viewpoint prediction in advance of the user actually looking at the FOV can address the ultra-low latency requirement of immersive 360-degree or VR experience, as mentioned earlier we also need adaptive streaming to address the challenge of fluctuating network bandwidth conditions. Hence in this section we propose a novel predictive adaptive streaming method. We investigate how viewpoint prediction can also be used to develop an effective adaptive streaming technique for 360-degree and VR videos, such that user experience can be maximized under dynamically varying network bandwidth conditions. Note that our predictive adaptive streaming algorithm will be executed every time slot (e.g., 200ms) using the network bandwidth at the beginning of the time slot. This way, our approach can address dynamically changing network bandwidth conditions.

TABLE II
Notations used.

Notation	*Meaning*
$BW(t)$	network bandwidth limit for time slot t
$p_{fov}(k)$	predicted FOV probability for tile k
K	total number of tiles
qp	quantization parameter (QP)
L	number of levels in quantization parameter
q	quantization step
$I(VE)$	impairment caused by video encoding (VE)
q_k	optimal quantization step for each tile k
$MSE(q)$	mean square error (MSE) between encoded tile video data and corresponding raw video data for quantization step q
$R(q)$	bitrate for quantization step q
a, b, θ, γ	parameters in bitrate and MSE models
p_v	viewpoint probability
i, j, k	index parameters for tiles
q_{min}	minimum boundary of quantization step setting
R_{max}	bitrate when encoding with setting of q_{min}
MSE_{min}	MSE when encoding with setting of q_{min}

We define FOV probability as probability of a given tile to belong to the user's actual FOV, where the actual user FOV refers to the actual user field of view in size of $90°\times90°$. Our proposed algorithm aims to maximize user experience (i.e., maximize the quality of tiles with high FOV probability) by smartly selecting proper video encoding (quantization step) settings of the video for each tile. We can find an optimal solution for maximizing user experience (minimizing impairment I) given a bandwidth limit $BW(t)$ for time slot t. Finally, we give an analysis of the algorithm complexity. The notations used in our approach are described in Table II.

A. Problem Formulation

We formulate the problem as an optimization problem as follows. Since minimizing impairment I caused by video encoding (VE) equals to maximizing user experience, we set our optimization target as minimizing I, where I is defined as $I(VE) = \sum_{k=1}^{K} p_{fov}(k)MSE(q_k)$, $MSE(q)$ represents mean square error between the encoded tile video data and the corresponding raw video data for quantization step q, and q_k denotes the optimal quantization step for each tile k. And qp_1 as well as qp_L are the minimum and maximum boundaries of quantization step (QP) settings used for an application. This optimization problem aims to minimize impairment caused by video encoding under constraint of the bandwidth limit.

Given:

1) Network bandwidth limit $BW(t)$ for time slot t

2) Predicted FOV probability $p_{fov}(k)$ for each tile k, and total number of tiles K
3) Quantization parameter set $\{qp_1, qp_2, \ldots, qp_L\}$
4) Model parameters including a, b, θ, γ

Find:

The optimal quantization step q_k for each tile k to minimize impairment

$$I^{OPT} = \min_{q_k} I(VE) = \min_{q_k} \sum_{k=1}^{K} p_{fov}(k) MSE(q_k) \quad (2)$$

$s.t.$

$$\sum_{k=1}^{K} R(q_k) \leq BW(t) \quad (3)$$

$$q_k = (2^{(1/6)})^{QP_l - 4}, \quad k \in \{1, 2, \ldots, K\} \quad (4)$$

$$QP_l \in \{qp_1, qp_2, \ldots, qp_L\}, \quad l \in \{1, 2, \ldots, L\} \quad (5)$$

$$MSE(q_k) = a \cdot q_k + b \quad (6)$$

$$R(q_k) = \frac{\theta}{q_k^{\gamma}} \quad (7)$$

B. Viewpoint Probability and FOV Probability

From our predictive LSTM model described in Section VI, we can obtain the viewpoint probability for each tile. In this subsection, we study how to calculate the *FOV probability* based on *viewpoint probability*. Note that *FOV probability* p_{fov} is defined as the probability of a given tile to belong to the user's actual FOV, while *viewpoint probability* p_v is defined as whether the user viewpoint is within a given tile. Equation 8 describes the relationship between p_{fov} and p_v. We use Fig. 10, a frame in an example 360-degree video, as an instance to illustrate how to calculate p_{fov} from p_v.

$$p_{fov} = \sum_i p_v(i) + 0.5 * \sum_j p_v(j) + 0.25 * \sum_h p_v(h) \quad (8)$$

where p_{fov} and p_v refer to the *FOV probability* and *viewpoint probability* corresponding to a tile, and i, j, k are index parameters for tiles surrounding this given tile. For example, consider the red dashed tile (tile #1) in Fig. 10; in this case, tile i belongs to tiles #1-#9, tile j belongs to tiles #10-#21, and tile h belongs to tiles #22-#25. This equation describes that the probability of the red dashed tile (tile #1) will belong to the actual user FOV (including both totally- and partially- within cases) equals to the probability of user viewpoint is within the total colored area (including purple, orange and yellow area), since we assume that the user FOV is 3×3 tiles corresponding the size of 90°×90°. Specifically, the former refers to the left

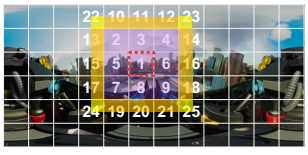


Fig. 10. Illustration for the calculation of FOV probability p_{fov} from viewpoint probability p_v.

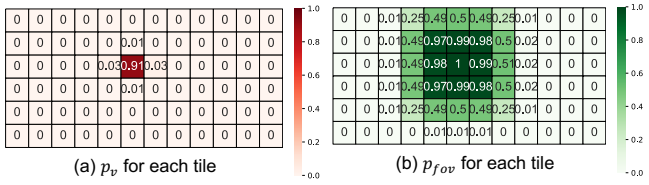


Fig. 11. Instances (a) left, p_v for each tile (b) right, p_{fov} for each tile.

TABLE III
EXPERIMENT SETTING.

Settings	*Experimental Values*
QP	15, 20, 25, 30, 33, 37, 40
Corresponding q	3.5, 6.5, 11, 20, 28, 44, 64
Resolution	4K (3840x2160) for two views, 3840x1080 for each view
GOP and Framerate	GOP: 12, framerate: 60fps

side of the equation, while on the right side, the first, second and third terms represents the probability of the viewpoint is within purple, orange and yellow area respectively. Note that the second and third terms assume that the viewpoint has the average probability distribution within the same tile. Fig. 11 presents an instance for the p_v for each tile and the corresponding p_{fov} for each tile using Equation 8.

C. Bitrate and MSE Models

In this subsection, we study and model the relationship between video encoding setting used for the 360-degree/VR video (quantization step q) and the resulting bitrate (R) as well as mean square error (MSE), terms used in the problem formulation in Section VI.A.

1) Bitrate Model and Validation: Next, we introduce how we perform experiments to validate the model of relationship between the bitrate and quantization step q. Several techniques have been proposed to model the bitrate of the encoded video as a function of the video encoding parameters. [49] proposed models of bitrate R using quantization step q and video frame rate t. Since in this paper, we do not consider the influence of different frame rate to bitrate, we fix the frame rate and thus we can simplify the model in [49] as follows.

$$R(q) = \frac{\theta}{q^{\gamma}} = R_{max}(\frac{q}{q_{min}})^{-\gamma} \quad (9)$$

In order to derive and validate this bitrate model, we encoded videos for 72 tiles respectively with different quantization parameter (QP) settings from Table III. Using the H.265/HEVC standard definition that $q = (2^{(1/6)})^{QP-4}$ [49], the corresponding q values are 3.5, 6.5, 11, 20, 28, 44, and 64. For each video, we encode it by using x265 encoding library and record the bitrate under each q value. We set R_{max} to be the bitrate when encoding with q_{min} and calculate normalized bitrate $R(q)/R_{max}$.

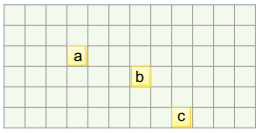


Fig. 12. Tiles selected for model validation of bitrate and MSE.

We randomly pick several tiles shown in Fig. 12 as tiles a, b and c, and present results in Fig. 13(a)(b), where x-axis of the figures is q and y-axis are the bitrate and normalized bitrate in (a) and (b) respectively. The results of videos for each tile are represented by a specific color. Bitrates are shown as circles for the different tiles of videos and the average value of all 72 tiles, we also plot a line for each video tile to represent the model equation. The parameter γ is obtained by minimizing mean square error between the

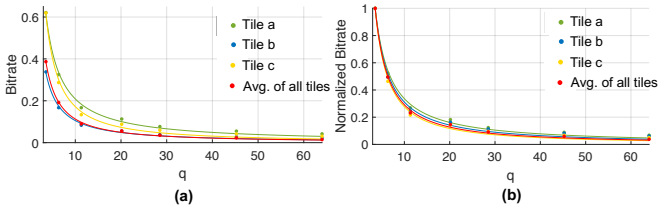


Fig. 13. Validation of model equation for bitrate (a) left, bitrate versus q (b) right, normalized bitrate versus q.

TABLE IV
PARAMETERS FOR RELATIONSHIP BETWEEN q AND BITRATE.

Tile	γ
Tile a	1.056
Tile b	1.131
Tile c	1.244
Avg. of all tiles	1.195

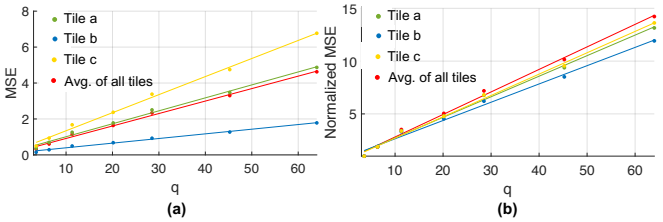


Fig. 14. Validation of model equation for MSE (a) left, MSE versus q (b) right, normalized MSE versus q.

TABLE V
PARAMETERS FOR RELATIONSHIP BETWEEN q AND MSE.

Tile	a_1	b_1
Tile a	0.6939	0.7825
Tile b	0.6169	0.9127
Tile c	0.7207	0.6791
Avg. of all tiles	0.7603	0.6806

model predicted and measured bitrates for each video of tiles. Table IV shows the corresponding parameter γ for the four lines in Fig. 13, and we can also calculate the parameter θ in Equation 9 accordingly. From Fig. 13, we can conclude that Equation 9 can model the bitrate of 360-degree video tiles using H.265/HEVC standard with high accuracy. Moreover, we do the validation of bitrate estimation for the tile. Fig. 15(a) shows the bitrate estimation results comparing the estimated bitrate for tile versus actual bitrate for tile using another 18 3s-tile video clips encoded with different QPs (i.e., QP = 15, 20, 30). The correlation is 0.9979 indicating the high accuracy of the proposed model (i.e., power law relationship between q and bitrate).

2) Model Validation for MSE: We investigate the relationship between the mean square error (MSE) and quantization step q, where the MSE refers to mean square error between the encoded tile video data and the corresponding raw video data for the tile. MSE reflects the average deviation of encoded tile pixels from their raw data counterparts. We adopt the linear model [50] between q and MSE, and investigate modeling how quantization step q influence encoding distortion MSE as follows.

$$MSE(q) = a \cdot q + b = MSE_{min}(a_1 \cdot \frac{q}{q_{min}} + b_1) \quad (10)$$

In order to validate the relationship between MSE and quantization step q for videos of different tiles, we follow the same encoding settings in Section VI.C 1), then calculate the MSE and pick three tiles in Fig. 12 for illustration. In Fig. 14, we use markers to show different data points and plot a line for each video tile to represent the fitted model equation. We set MSE_{min} to be the MSE when encoding with q_{min} and calculate normalized MSE as $MSE(q)/MSE_{min}$. The parameters a_1 and b_1 are also obtained by minimizing the mean square error between the model predicted and measured MSE for each video of tiles. Table V shows the parameters obtained in our experiments, and we can also calculate the parameters a and b in Equation 10 accordingly. From Fig. 14, we can see that Equation 10 can model the MSE of 360-degree video tiles using H.265/HEVC standard with high accuracy. In particular, for higher quantization step q, larger distortion (i.e., higher MSE) can be observed for tiles with more dynamic content (e.g., Tile c), while relatively smaller distortions are obtained for relatively static content (e.g., Tile b). Furthermore, we do the validation of MSE estimation for the tile. Fig. 15(b) shows the MSE estimation results comparing the estimated MSE for tile versus actual MSE for tile using another 18 3s-tile video clips encoded with different QPs (i.e., QP = 15, 20, 30). The correlation is 0.9923 indicating the high accuracy of the proposed model (i.e., linear relationship between q and MSE). Note that in our experiments, we employ the parameters obtained for average of all tiles to calculate the bitrate as well as MSE.

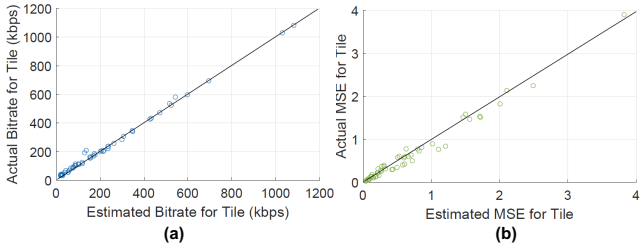


Fig. 15. Validation of bitrate and MSE estimation for tile (a) left, results for bitrate (b) right, results for MSE.

D. Algorithm Description

We first describe the key ideas and insights of how we analyze the problem and develop the algorithm. Then we discuss the detailed steps of the algorithm.

First, notice that the problem we are to solve contains K discrete variables (i.e., one variable for each tile). If quantization step q has L levels, then we have L^K combinations for all q_k variables ($k \in \{1, 2, \ldots, K\}$). In our experiment, we choose $L = 3$ resulting in 3^{72} combinations in total when the total number of tiles as $K = 72$. A brute-force algorithm based on exhaustive enumeration of the exponential number of combinations will be prohibitively expensive and cannot be applicable in real-time to perform adaptive streaming responding to real-time changes in network bandwidth. Hence, we propose a heuristic greedy algorithm to select the q level for each tile, such that impairment I is minimized (Eq. 2) subject to the bandwidth and other constraints (Eqs. 3-7).

- VR-PAS Algorithm

In order to solve the above problem, we first propose a heuristic greedy algorithm, VR Predictive Adaptive Streaming (*VR-PAS*) algorithm, which runs periodically (in this paper

Algorithm 1 VR-PAS Algorithm

Inputs:
1) Network bandwidth limit $BW(t)$ for time slot t
2) Predicted FOV probability $p_{fov}(k)$ for each tile k, and total number of tiles K
3) Quantization parameter set $\{qp_1, qp_2, \ldots, qp_L\}$
4) Model parameters including a, b, θ, γ

Output: The optimal quantization step q_k for each tile k to minimize impairment.

1: Initilization: for each tile k, set q_k to be q_{max} (i.e., $l(k) \leftarrow 1$, low quality), calculate the current bandwidth needed BW_{cur} and impairment I_{cur}
2: **while** $(BW_{cur} < BW(t))\&\&!(\text{all } q == q_{min})$ **do**
3: $BW_{per} \leftarrow BW_{cur};\ I_{per} \leftarrow I_{cur}$
4: **for** $k = 1 : K$ **do**
5: $l(k) \leftarrow l(k) + 1;\ q_k \leftarrow (2^{(1/6)})^{qp_{l(k)}-4}$
6: calculate BW_{cur} and I_{cur}
7: $\Delta BW_k \leftarrow BW_{cur} - BW_{per}$
8: $\Delta I_k \leftarrow I_{cur} - I_{per}$
9: $l(k) \leftarrow l(k) - 1;\ q_k \leftarrow (2^{(1/6)})^{qp_{l(k)}-4}$
10: **end for**
11: Find k which has the maximum value of $\Delta I_k / \Delta BW_k$
12: Set the change of q_k of tile k as $l(k) \leftarrow l(k) + 1;\ q_k \leftarrow (2^{(1/6)})^{qp_{l(k)}-4}$
13: Calculate the new bandwidth needed BW_{cur} and I_{cur}, check if the new $BW_{cur} > BW(t)$ then revert q_k as previous value
14: **end while**
15: **return** q_k for each tile k

we set the period to be 200ms). At the beginning of each time period, we will obtain the inputs of this algorithm: 1) network bandwidth limit $BW(t)$ for time slot t; 2) predicted FOV probability $p_{fov}(k)$ for each tile k, total number of tiles K, quantization parameter set $\{qp_1, qp_2, \ldots, qp_L\}$, and model parameters (a, b, θ, γ). We use $p_{fov}(k)$ and model parameters with Equation 7 to estimate the bitrate consumption in the next time period. The output of the algorithm will be the optimal quantization step q_k for each tile k.

This problem can be formulated as a variant of discrete *knapsack* problem [51], in which the bitrate consumed by each tile k can be interpreted as its *weight* (i.e., $w(k) = R(q_k)$); and the video encoding impairment caused by each tile k can be regarded as its *price* (i.e., $v(k) = p_{fov}(k)MSE(q_k)$). The problem can be restated as, for total K groups (K tiles), with each group having L objects (L quantization steps), select the optimal object (quantization step) for each group (tile), such that the total weight of these selected objects does not exceed the limit ($BW(t)$ for time slot t), and the total price (impairment) is minimized.

The underlying principle of the *VR-PAS* algorithm is as follows:

Algorithm 1 shows the pseudo-code of the *VR-PAS* algorithm. Initially we set the encoding quality of all tiles to be Low ($l = 1$, quantization step max). Then we keep adjusting the encoding quality of tiles using a while loop, as long as

Algorithm 2 VR-OPT Algorithm

Inputs:
1) Network bandwidth limit $BW(t)$ for time slot t
2) Predicted FOV probability $p_{fov}(k)$ for each tile k, and total number of tiles K
3) Quantization parameter set $\{qp_1, qp_2, \ldots, qp_L\}$
4) Model parameters including a, b, θ, γ

Output: The optimal quantization step q_k for each tile k to minimize impairment.

1: $f \leftarrow zeros(K, BW(t) * 100)$
2: $mark \leftarrow zeros(K, BW(t) * 100)$
3: **for** $k = 1 : K$ **do**
4: **for** $c = 0 : 0.01 : BW(t)$ **do**
5: $temp \leftarrow INT\_MAX$
6: **for** $l = 1 : L$ **do**
7: $q \leftarrow (2^{(1/6)})^{qp_l - 4}$
8: $w[l] \leftarrow R(q)$
9: $v[l] \leftarrow p_{fov}(k)MSE(q)$
10: **if** $c - w[l] \geq 0$ **then**
11: $f_{temp} \leftarrow f[k-1][100 * (c - w[l])] + v[l]$
12: **if** $f_{temp} < temp$ **then**
13: $temp \leftarrow f_{temp}$
14: $mark[k][c] \leftarrow l$
15: **end if**
16: **end if**
17: **end for**
18: $f[k][100 * c] \leftarrow temp$
19: **end for**
20: **end for**
21: Get q_k from $FindQuality()$
22: **return** q_k for each tile k

Procedure $FindQuality()$:

1: $pre \leftarrow BW(t) * 100$
2: **for** $k = K : -1 : 1$ **do**
3: $tile_l \leftarrow mark[k][pre]$
4: Set the change of quantization step setting of tile k as $l(k) \leftarrow tile_l;\ q_k \leftarrow (2^{(1/6)})^{qp_{l(k)}-4}$
5: Denote bandwidth needed for the tile k as bw and calculate the new pre as $pre \leftarrow pre - bw * 100$
6: **end for**
7: **return** q_k for each tile k

the total bitrate does not exceed the bitrate budget ($BW(t)$ for time slot t). During each iteration, the algorithm will first iterate over all the tiles, and for each tile k, the algorithm computes the possible degradation in its encoding impairment (ΔI_k), and the possible increase in the consumed bandwidth (ΔBW_k), if we set its encoding quality to be one level higher (corresponding to Line 5). Among all tiles, the algorithm will choose the one with the highest ratio of $\Delta I_k / \Delta BW_k$. Note that the encoding impairment is calculated based on predicted FOV probability p_{fov} and MSE, thus a higher p_{fov} may lead to a higher $\Delta I_k / \Delta BW_k$ when parameters including MSE and bitrate R are fixed. Therefore, for a tile with higher p_{fov}, our algorithm will be more likely to choose this tile in the while

loop and assign it with high quality. Algorithm will stop when 1) there is no more bandwidth available or 2) all tiles set encoding quality to be High. The proposed *VR-PAS* algorithm has a run time of less than 70ms with a Quad-core i7 processor, and thus can meet the real-time execution requirement of our predictive adaptive streaming method.

- VR-OPT Algorithm

To be able to compare the performance of our proposed real time heuristic algorithm *VR-PAS*, we next present a dynamic programming based algorithm, *VR-OPT*, which can produce an optimal solution for the adaptive streaming problem (Equation 2), though the latter has high runtime and hence cannot be used in practice. The core formula for the dynamic programming is as follow:

$$f[k][c] = min\{f[k-1][c-w[l]]+v[l] \mid object\ l \in group\ \#k\}$$

where $f[k][c]$ denotes the minimum price for the first k groups using cost c (condition: $0 < c \leq BW(t), 1 \leq k \leq K, 1 \leq l \leq L$), and different object l in group k corresponding to different encoding quality for a tile. The minimum $f[K][:]$ corresponds to the final solution. Note that this formula requires $BW(t)$ for time slot t and all $w[i]$ are integrals, assuming in our experiments the $w[i]$ can be as small as 0.06Mbps, then we magnify them 100 times simultaneously to make them become integral value. In this algorithm shown in Algorithm 2, Lines 3-20 achieve the core formula described above, which are solving the problems by breaking it apart into a sequence of smaller decisions. Specifically, we calculate the value of $f[k][c]$ for each combination of variables k and c (condition: $1 \leq k \leq K$, $0 < c \leq BW(t)$), which means solving the problem of the minimum price for first k groups using cost c and obtaining the optimal solutions of all these smaller decisions. Then by considering all these smaller decisions, the algorithm gets the minimum $f[K][:]$ as the final solution, corresponding to the minimum impairment achieved with a total bitrate within the bandwidth limit. Finally, in Line 21, Procedure *FindQuality* is a function tracing back the selected quantization step (i.e., selected quality) for each tile in the optimal solution.

E. Complexity Analysis

Since this problem is a variant of *knapsack problem*, it is also a NP-complete problem. Our proposed heuristic algorithm *VR-PAS* has the worst-case time complexity of $O(K \cdot K(L-1))$. The worst-case happens when the bandwidth limit equals to or larger than the bandwidth needed for highest quality for the whole 360-degree view. In this case, the iteration (i.e., Lines 2-14 of *VR-PAS*) will be conducted for $K(L-1)$ times, where one of the tiles is set one-level quality better during each iteration and finally all tiles are set with highest quality. As for the *VR-OPT* algorithm, the worst-case time complexity is $O(K \cdot 100BW(t) \cdot L)$ due to the three nested loops in Lines 3, 4, and 6 (whose number of iterations are K, $100BW(t)$, and L respectively). This algorithm can obtain optimal results but will be much more time-consuming than *VR-PAS*, which will be further discussed in the next section.

VII. Experimental Results

In this section, we report on experiments conducted to evaluate the performance of our proposed predictive LSTM model, as well as our proposed predictive adaptive streaming algorithms. We also do an end-to-end timeline analysis of our predictive adaptive streaming approach.

A. Predictive LSTM Model

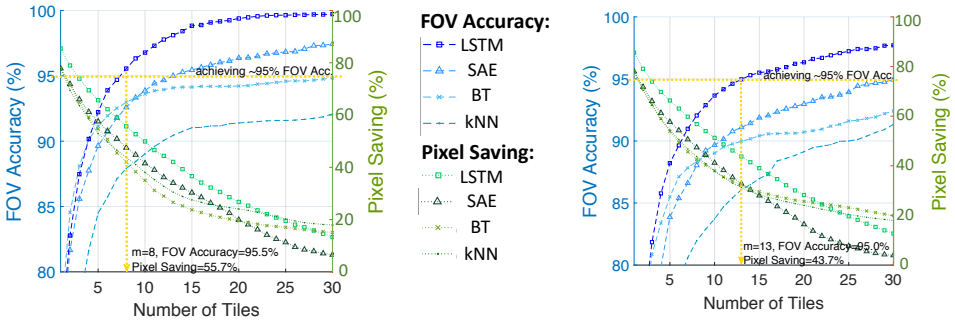
Next we present the experimental results for our predictive LSTM model, including the experimental setup. We use 90% of the dataset for training the LSTM viewpoint prediction model, and 10% for testing, ensuring the test data is from viewers which are different than those in training data. Specifically, we have 32400 samples as training data and 3600 samples as test data for both medium motion and high motion sequences in Fig. 16 and Table VI, while we take 45000 samples as training data and 5000 samples as test data for each of three sequences in Table VII. As for the experimental setup, we use an Intel Core i7 Quad-Core processor with 32GB RAM and implement our approach in Python using Keras [52]. We compare the performance of our LSTM model with state-of-the-art methods as follows:

- *Stacked sparse autoencoders* ***(SAE):*** We use SAE [36], [53] with tile information during 10 timestamps as input to predict the tile where the viewpoint is within for next timestamp. The SAE model contains two fully-connected layers with 100 and 80 nodes respectively for training.
- *Bootstrap-aggregated decision trees* ***(BT):*** Following the work of [25], we also compare against BT using 10-timestamp tile information as input. The BT model ensembles with 30 bagged decision trees, which reduces the effects of overfitting and improves generalization.
- *Weighted k-nearest neighbors* ***(kNN):*** We implement a kNN [54] using 10-timestamp tile information as input and set 100 as the number of nearest neighbors.

Note that while the training time for BT and kNN are less than 20 minutes for the above training set for a 10-second sequence, the training time for the deep learning models including LSTM and SAE are up to one hour.

After training the various models with both two representations described in Section V.A, we decide on using the one-hot encoding representation to train SAE and LSTM models, while using the simple representation for BT and kNN, since the simple representation works better for the latter two approaches in our experiments.

Note in Fig. 16, Table VI and Table VII, *FOV accuracy* refers to *FOV prediction accuracy*. We first show results of experiments with the medium and high motion sequences of *Kong VR* in Fig. 16 and Table VI. We show the FOV prediction accuracy and pixel savings obtained when selecting different number of tiles (i.e., the choice of m) to generate FOV. The blue plots show the FOV prediction accuracy achieved by each of the models for specific number of tiles (i.e., the choice of m) selected to generate the FOV, while the green plots show the corresponding pixel saving of the generated FOV compared to the whole 360-degree view. Lines with blue triangle markers, blue square markers, blue cross markers



(a) Medium Motion Sequence (b) High Motion Sequence

Fig. 16. (a)(b) show FOV prediction accuracy and pixel saving versus number of tiles selected for FOV (for two sequences in *Kong VR*).

TABLE VI
EXPERIMENTAL RESULTS FOR TWO SEQUENCES IN *Kong VR*.

Model	*Medium Motion Sequence*		*High Motion Sequence*	
	FOV Accuracy(%)	*Pixel Saving(%)*	*FOV Accuracy(%)*	*Pixel Saving(%)*
SAE	95.0	34.0	95.0	3.9
LSTM	95.5	**55.7**	95.0	**43.7**
BT	95.0	14.8	95.2	14.4
kNN	94.8	12.0	95.3	12.0

TABLE VII
EXPERIMENTAL RESULTS FOR THREE VIDEO SEQUENCES.

Model	*Fashion Show*		*Whale Encounter*		*Roller Coaster*	
	FOV Accuracy(%)	*Pixel Saving(%)*	*FOV Accuracy(%)*	*Pixel Saving(%)*	*FOV Accuracy(%)*	*Pixel Saving(%)*
SAE	95.4	52.7	95.1	46.8	95.3	29.9
LSTM	95.2	**69.7**	95.5	**66.8**	95.2	**71.0**
BT	95.3	19.1	95.0	18.6	95.2	48.9
kNN	94.9	12.0	95.2	10.3	95.1	21.2

and blue point markers represent FOV prediction accuracy for SAE, LSTM, BT and kNN models respectively while lines with green triangle markers, green square markers, green cross markers and green point markers represent corresponding pixel saving for these models.

From Fig. 16, we observe the following. As number of tiles m increases, the FOV prediction accuracy continuously increases and pixel saving simultaneously decreases. This shows the tradeoff between FOV prediction accuracy and pixel saving. Furthermore, we can see that our proposed LSTM model outperforms the other three methods. For instance, in both Fig. 16(a) and (b), the line with blue square markers (denoting FOV prediction accuracy achieved by LSTM) is significantly higher than the other three blue lines when the number of selected tiles (i.e., the choice of m) is larger than 5. We also observe that high FOV prediction accuracy can be achieved by LSTM (and other models) with smaller FOV and hence higher pixel savings for medium motion sequences compared to high motion sequences. For example, in Fig. 16(a), LSTM achieves a high FOV prediction accuracy of 95.5% when selects 8 tiles (i.e., $m = 8$) to generate FOV, leading to pixel savings of 55.7%, while in Fig. 16(b) to achieve a comparable FOV prediction accuracy of 95.0%, LSTM needs a larger FOV generated by 13 tiles (i.e., $m = 13$) with lower pixel savings of 43.7%. Table VI summarizes the experimental results shown in Fig. 16. When we set FOV prediction accuracy as around 95%, we can observe that our LSTM model achieves significantly larger pixel savings than the other three models, achieving 55.7% and 43.7% pixel savings for medium and high motion sequences respectively. In our experiments, the inference time for all the models including LSTM is less than 2ms.

We further perform more experiments on three relatively low motion video sequences including *Fashion Show, Whale Encounter* and *Roller Coaster* in our dataset to evaluate our LSTM model. It corresponds to the fact that for instance in *Fashion Show* sequence, viewers have similar area of interest (e.g., the stage) and seldom change viewpoint out of this area to other tiles. Similarly, in *Roller Coaster* sequence, viewers tend to look towards front more time than other directions when roller coaster keeps up high speed. Moreover, note that we select 10s-duration for each sequence to keep consistency with experiments done with *Kong VR*. The inference time for all the models including LSTM is still less than 2ms. Table VII exhibits the experimental results for the three video sequences. Our LSTM model can achieve a very high FOV prediction accuracy of approximately 95% with selecting 4 tiles (i.e., $m = 4$) to generate FOV and corresponding pixel savings of around 70% for *Fashion Show* and *Roller Coaster*, and choosing 5 tiles (i.e., $m = 5$) to generate FOV and corresponding pixel savings of 66.8% for *Whale Encounter*. Note that the above savings are significantly higher than achieved by the other three models. Therefore, our experimental results above demonstrate that our LSTM model and FOV generation approach can achieve very high FOV prediction accuracy while significantly reducing pixels needed. In a separate work involving different VR applications, we have shown empirically that there is a high correlation between pixel and bitrate savings [55]. Thus, our experimental results also illustrate the tradeoff between FOV prediction accuracy and bandwidth savings.

While the above results demonstrate the accuracy and band-

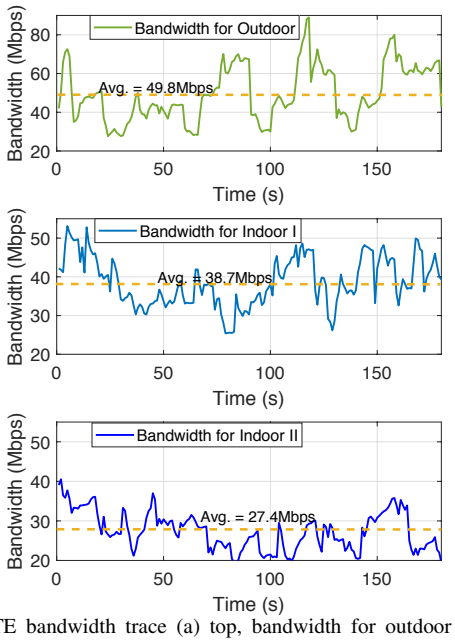


Fig. 17. LTE bandwidth trace (a) top, bandwidth for outdoor (b) middle, bandwidth for indoor location I (c) bottom, bandwidth for indoor location II.

TABLE VIII
QP AND TOTAL BANDWIDTH NEEDED FOR DIFFERENT QUALITY.

Quality	*QP*	*Total Bandwidth Needed*
Low	30	8.64Mbps
Medium	20	28.2Mbps
High	15	57.6Mbps

width savings potential of our viewpoint prediction and FOV generation approach, the network bandwidth available may not be sufficient to transmit the predicted FOV, thereby necessitating the predictive adaptive streaming approach we described in Section VI. We next describe experiments we conducted to evaluate our proposed predictive adaptive streaming algorithm *VR-PAS*, including comparison with the optimal algorithm *VR-OPT* as well as a recent related work on adaptive 360-degree video streaming.

B. Predictive Adaptive Streaming

We first collect 4G-LTE network traces by using network bandwidth testing software Speedtest [56] to record the bandwidth. Fig. 17 shows the bandwidth measured during 180s in one outdoor location and two indoor locations respectively (for indoor environment, we select two indoor locations under different bandwidth conditions). We can see the average bandwidth for outdoor location, indoor location I, and indoor location II are 49.8Mbps, 38.7Mbps and 27.4Mbps respectively. The bandwidth in outdoor location is relatively largest among three locations while the bandwidth in indoor location II is more limited than in indoor location I. Table VIII presents the bandwidth needed for the video sequence called *Fashion Show* if all tiles are encoded in different quality (QP high, medium and low). We take it as an example to do our experiments and similar results can be obtained for other videos. From Table VIII, we can see that all tiles can be encoded in high quality when bandwidth limit is larger than or equal to 57.6Mbps. In our experiments, for each time slot (e.g., 200ms), we consider the bandwidth value at the beginning of the time slot in the network traces in Fig. 17(a)(b)(c) as the ongoing bandwidth limit for the current time slot, and run our proposed algorithm with head motion traces.

In addition, for comparison reason, we also implemented two more algorithms as follows:

1) *Adaptive streaming*: We implement the *adaptive streaming* method, which is called viewport-driven rate-distortion optimized streaming from [42]. This method enables the adaptive video streaming according to the heatmaps that capture the likelihood of navigation of different spatial segments of a 360-degree video over time. Specifically, during one GOP (e.g., 1s), corresponding to 30 frames, for each tile k they count the number of times that the tile is occupied by user FOV, denoted as w_k. This technique computes the likelihoods of navigating different tiles during the given GOP as $\frac{w_k}{\sum_k w_k}$ and then do the bitrate allocation (assign video quality) for tiles based on these likelihoods.
2) *Non-adaptive streaming*: We also compare against *non-adaptive streaming* method, which is the normal approach to stream the whole 360-degree view in the same quality (e.g., low/medium/high) according to the bandwidth condition.

Fig. 18, Fig. 19, and Fig. 20 show the results for the three different network bandwidth traces in outdoor location, indoor location I, and indoor location II respectively. We employ following metrics to evaluate the performance of our algorithm: $P(Q_{High})$ is defined as percentage of tiles in the actual user FOV which are encoded with High quality (i.e., low QP), while $P(Q_{Medium})$ and $P(Q_{Low})$ are defined as percentage of tiles in the actual user FOV which are encoded with Medium quality (i.e., medium QP) and Low quality (i.e., high QP) respectively. Using 1000 head motion traces for the video sequence called *Fashion Show*, we calculate the $P(Q_{High})$, $P(Q_{Medium})$, $P(Q_{Low})$ and PSNR (Equations 11 and 12) in actual user FOV under different bandwidth conditions. In Equations 11 and 12 [57], MSE_{avg} represents the average MSE of the actual user FOV while MSE_H, MSE_M, and MSE_L denotes the average MSE of high, medium, low quality tile respectively.

$$PSNR = 10log_{10}\frac{255^2}{MSE_{avg}} \tag{11}$$

$$\begin{aligned} MSE_{avg} =& P(Q_{High})MSE_H + \\ & P(Q_{Medium})MSE_M + P(Q_{Low})MSE_L \end{aligned} \tag{12}$$

Fig. 18(a), Fig. 19(a), and Fig. 20(a) plot the average $P(Q_{High})$ for all these traces while Fig. 18(b), Fig. 19(b), and Fig. 20(b) plot the average $P(Q_{Medium})$ and Fig. 18(c), Fig. 19(c), and Fig. 20(c) plot the average $P(Q_{Low})$. The corresponding PSNR in the actual FOV is also illustrated in Fig. 21 for these three locations. In each figure, we compare the performance of the three algorithms (our proposed *VR-PAS* as the predictive adaptive streaming method, adaptive streaming method [42] and non-adaptive streaming method). From the figures, we can make the following observations:

1) From Fig. 18 and Fig. 19, we can observe that our predictive adaptive streaming achieves high average $P(Q_{High})$

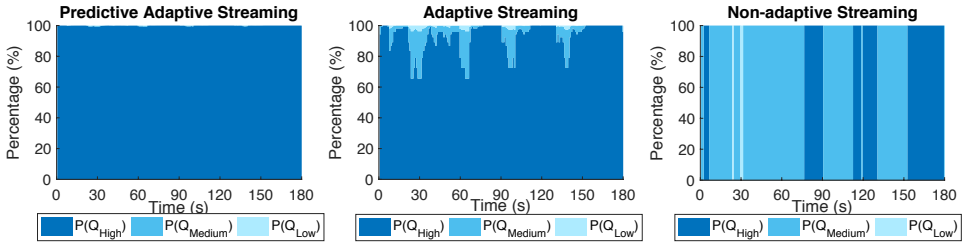


Fig. 18. In outdoor location, average percentage of different quality view in actual FOV (a) left, predictive adaptive streaming (b) middle, adaptive streaming (c) right, no adaption streaming.

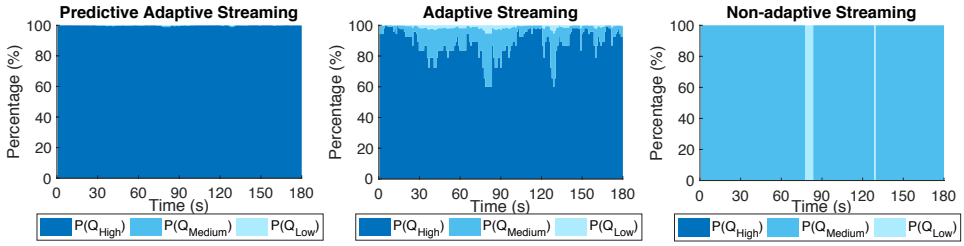


Fig. 19. In indoor location I, average percentage of different quality view in actual FOV (a) left, predictive adaptive streaming (b) middle, adaptive streaming (c) right, no adaption streaming.

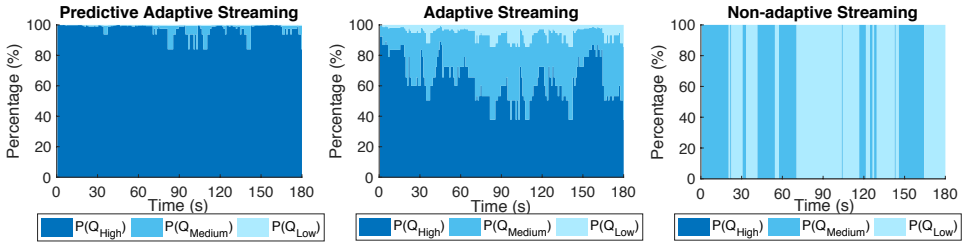


Fig. 20. In indoor location II, average percentage of different quality view in actual FOV (a) left, predictive adaptive streaming (b) middle, adaptive streaming (c) right, no adaption streaming.

(i.e., larger than 99%) in outdoor location and indoor location I. In Fig. 20, our predictive adaptive streaming also achieves high average $P(Q_{High})$ (i.e., larger than 80%) in indoor location II, while both adaptive and non-adaptive streaming produces significantly lower user experience with $P(Q_{High})$ less than 65% and 0 respectively.

2) In all three locations, our predictive adaptive streaming algorithm *VR-PAS* performs significantly better (result in higher $P(Q_{High})$ and PSNR) compared to adaptive streaming [33] and non-adaptive streaming. For example, in Fig. 21(c), our predictive adaptive streaming achieves an average PSNR more than 1dB and 4dB larger compared to adaptive streaming and non-adaptive streaming respectively.

To evaluate the bandwidth savings and thereby wireless network capacity gain that can be achieved by *VR-PAS*, we measure $P(Q_{High})$, $P(Q_{Medium})$, $P(Q_{Low})$ and PSNR of streaming using *VR-PAS*, and compare with adaptive and non-adaptive streaming. Fig. 22 and Fig. 23 illustrate the results for different bandwidth limits (x-axes) using three algorithms. Fig. 22 presents the $P(Q_{High})$, $P(Q_{Medium})$, $P(Q_{Low})$ under different bandwidth limits while Fig. 23 shows the PSNR of actual user FOV under different bandwidth limits. We provide a summary of key observations as follows:

1) Fig. 22 shows that our predictive adaptive streaming method has an average $P(Q_{High})$ of around 98% using 25Mbps while 57.6Mbps is the total bandwidth needed

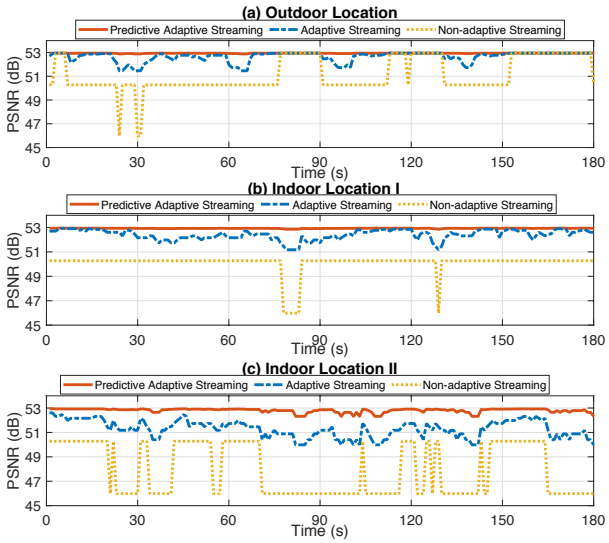


Fig. 21. PSNR results (a) top, bandwidth for outdoor (b) middle, bandwidth for indoor location I (c) bottom, bandwidth for indoor location II.

for the whole 360-degree view in High quality (Table VIII), meaning that the user can have around 98% view in High quality with 56.6% bandwidth savings. This shows that *VR-PAS* can lead to significant bandwidth savings while ensuring very high user experience and satisfying ultra-low latency requirement due to accurate viewpoint prediction and advanced streaming.

2) In Fig. 23, we can observe that significantly lower band-

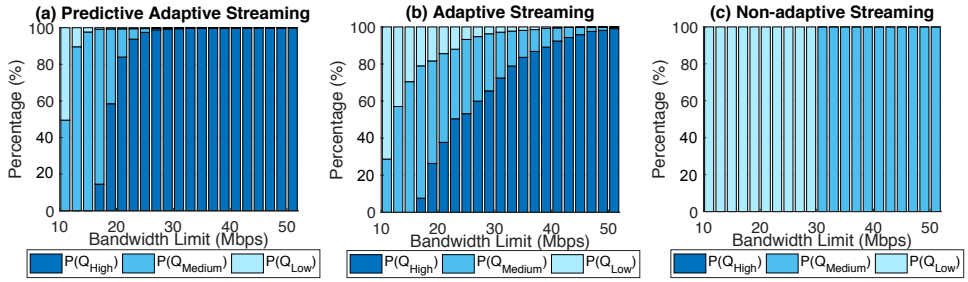


Fig. 22. Percentages of different quality in actual user FOV under different fixed bandwidth limits (a) left, predictive adaptive streaming (b) middle, adaptive streaming (c) right, no adaption streaming.

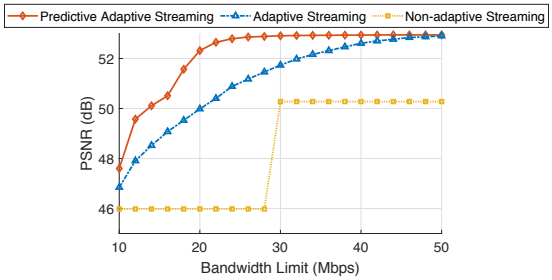


Fig. 23. PSNR results versus the different fixed bandwidth limits.

width is needed by *VR-PAS* to produce the same high-quality experience compared to adaptive and non-adaptive streaming. For example, to achieve PSNR of 50dB in actual user FOV, from Fig. 23, *VR-PAS* requires around 14Mbps while the adaptive and non-adaptive streaming methods need around 20Mbps and 30Mbps respectively. Thus, our proposed predictive adaptive streaming method can provide more than 53.3% bandwidth saving compared to normal streaming (i.e., non-adaptive streaming) to offer the user FOV with PSNR of 50dB.

The bandwidth savings achieved by the predictive adaptive streaming method can enable network operators to significantly improve capacity of their network, being able to serve significantly more users their 360-degree/VR video applications. For example, 2X more users can be served a high-quality experience of 50dB PSNR, compared to adaptive and non-adaptive techniques. This in turn can significantly improve the economics of delivering wirelessly 360-degree/VR experiences by service providers and/or network operators, while ensuring high user experience including visual quality and ultra-low latency requirements.

Next, we do a comparison for predictive adaptive streaming algorithms *VR-PAS* and *VR-OPT* in Table IX. Similar impairment $I(VE)$ is achieved by both algorithms while our *VR-PAS* algorithm can be completed within around 70ms and *VR-OPT* cannot be finished in real time (e.g., runtime is 8.3s approximately when bandwidth limit is 40Mbps). This shows that our *VR-PAS* algorithm is a real time algorithm and can generate similar results compared to the optimal results obtained by *VR-OPT* algorithm.

C. Timeline Analysis

In this subsection, we give an analysis of the timeline for our proposed predictive adaptive streaming method in Table X. Specifically, we can see that the latency for transmission from

TABLE IX
COMPARISON FOR PREDICTIVE ADAPTIVE STREAMING ALGORITHMS *VR-PAS* AND *VR-OPT*.

Bandwidth Limit	*Algorithm*	*I(VE)*	*Runtime*
$10Mbps$	*VR-PAS*	21.807	13.9ms
	VR-OPT	21.807	2051ms
$20Mbps$	*VR-PAS*	7.929	27.8ms
	VR-OPT	7.877	4116ms
$30Mbps$	*VR-PAS*	5.501	39.4ms
	VR-OPT	5.496	6159ms
$40Mbps$	*VR-PAS*	5.224	51.3ms
	VR-OPT	5.223	8294ms
$50Mbps$	*VR-PAS*	5.186	63.4ms
	VR-OPT	5.186	10398ms

TABLE X
TIME NEEDED FOR DIFFERENT PROCEDURES.

Procedure	*Time Needed*
Transmission from HMD to edge	*Depends on distance*
Viewpoint Prediction	$< 2ms$
Predictive Adaptive Streaming Alg.	$< 70ms$
Transmission from edge to HMD	*Depends on distance*
Decoding	$\approx 3ms$

HMD to edge and from edge to HMD depend on distance between them. Thus, our proposed predictive adaptive streaming algorithm *VR-PAS* can run in real-time, with less than 70ms runtime as shown in Table IX; with added round-trip transmission latency of under 25ms and the other delays due to viewpoint prediction and decoding (Table X), our algorithm can be executed in real-time about every 100ms. Since we predict the user view 200ms in advance, we have adequate time to send the predicted view in advance and display the needed view for users on the HMD with no additional latency, hence satisfying the ultra-low latency requirement of 360-degree video and VR immersive experiences. Note that Table X does not include the time of encoding tiles since video tiles are known and have been encoded before streaming in the current 360-degree video streaming scenario; the time consumption of encoding tiles will have to be added if real-time tile encoding is needed.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a predictive adaptive streaming approach in order to reduce the latency and bandwidth needed to deliver 360-degree videos and cloud/edge-based VR applications, leading to better mobile VR experiences. We present a multi-layer LSTM model which can learn general head motion pattern and predict the future viewpoint based on past traces.

Our method outperforms state-of-the-art methods on a real head motion trace dataset and shows great potential to reduce bandwidth while keeping a good user experience (i.e., high PSNR).

Our planned future work includes performing subjective studies to understand and quantify user experience using our proposed predictive approach. For viewpoint prediction, we also plan to study and develop more comprehensive models considering (i) the type of video content to improve prediction accuracy further, and (ii) joint head and body motion to enable six Degrees of Freedom (6DoF) immersive experiences. Moreover, we plan to explore in the future various tile as well as projection options, develop further refined formulation and algorithms for adaptive bitrate streaming, and perform more detailed timing analysis considering real-time tile encoding.

REFERENCES

[1] X. Hou, Y. Lu, and S. Dey, "Wireless vr/ar with edge/cloud computing," in *Proc. ICCCN*, 2017.

[2] H. Bellini, "The real deal with virtual and augmented reality," 2016. [Online]. Available: http://www.goldmansachs.com/our-thinking/pages/virtual-and-augmented-reality.html

[3] C. Wiltz, "Five major challenges for vr to overcome," April 2017. [Online]. Available: https://www.designnews.com/electronics-test/5-major-challenges-vr-overcome/187151205656659/

[4] Adobe, "Capitalizing on viewers' hunger for virtual and augmented reality white paper," 2016. [Online]. Available: https://www.creativeplanetnetwork.com/videoedge/362797

[5] O. Rift, "Oculus," 2018. [Online]. Available: https://www.oculus.com

[6] HTC, "Htc vive," 2018. [Online]. Available: https://www.vive.com/us/

[7] Sony, "Playstation vr," 2018. [Online]. Available: https://www.playstation.com/en-us/explore/playstation-vr/

[8] Samsung, "Samsung gear vr," 2018. [Online]. Available: https://www.samsung.com/us/mobile/virtual-reality/

[9] Google, "Google daydream," 2018. [Online]. Available: https://vr.google.com/daydream/

[10] Qualcomm, "Whitepaper: Making immersive virtual reality possible in mobile," 2016. [Online]. Available: https://www.qualcomm.com/media/documents/files/whitepaper-making-immersive-virtual-reality-possible-in-mobile.pdf

[11] Z. Wang, L. Sun, C. Wu, W. Zhu, Q. Zhuang, and S. Yang, "A joint online transcoding and delivery approach for dynamic adaptive streaming," *IEEE Trans. Multimedia*, vol. 17, no. 6, pp. 867–879, 2015.

[12] G. Gao, H. Zhang, H. Hu, Y. Wen, J. Cai, C. Luo, and W. Zeng, "Optimizing quality of experience for adaptive bitrate streaming via viewer interest inference," *IEEE Trans. Multimedia*, vol. 20, no. 12, pp. 3399–3413, 2018.

[13] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 965–984, 2018.

[14] S. Almowuena, M. M. Rahman, C.-H. Hsu, A. A. Hassan, and M. Hefeeda, "Energy-aware and bandwidth-efficient hybrid video streaming over mobile networks," *IEEE Trans. Multimedia*, vol. 18, no. 1, pp. 102–115, 2016.

[15] J. Yang, E. Yang, Y. Ran, Y. Bi, and J. Wang, "Controllable multicast for adaptive scalable video streaming in software-defined networks," *IEEE Trans. Multimedia*, vol. 20, no. 5, pp. 1260–1274, 2018.

[16] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive view generation to enable mobile 360-degree and vr experiences," in *Proc. VR/AR Network*, 2018, pp. 20–26.

[17] X. Liu, Q. Xiao, V. Gopalakrishnan, B. Han, F. Qian, and M. Varvello, "360 innovations for panoramic video streaming," in *Proc. HotNets*, 2017, pp. 50–56.

[18] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "Vr is on the edge: How to deliver 360 videos in mobile networks," in *Proc. VR/AR Network*, 2017, pp. 30–35.

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

[20] R. Ju, J. He, F. Sun, J. Li, F. Li, J. Zhu, and L. Han, "Ultra wide view based panoramic vr streaming," in *Proc. VR/AR Network*, 2017, pp. 19–23.

[21] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. ICC*, 2017, pp. 1–7.

[22] X. Ge, L. Pan, Q. Li, G. Mao, and S. Tu, "Multipath cooperative communications networks for augmented and virtual reality transmission," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2345–2358, 2017.

[23] Y. Liu, J. Liu, A. Argyriou, and S. Ci, "Mec-assisted panoramic vr video streaming over millimeter wave mobile networks," *IEEE Trans. Multimedia*, 2018.

[24] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. NIPS*, 2015, pp. 1171–1179.

[25] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[26] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Comput.*, vol. 9, no. 7, pp. 1545–1588, 1997.

[27] T. Kim, Y. Yue, S. Taylor, and I. Matthews, "A decision tree framework for spatiotemporal sequence prediction," in *Proc. KDD*, 2015, pp. 577–586.

[28] C. Yan, Y. Zhang, J. Xu, F. Dai, L. Li, Q. Dai, and F. Wu, "A highly parallel framework for hevc coding unit partitioning tree decision on many-core processors," *IEEE Signal Process. Lett.*, vol. 21, no. 5, pp. 573–576, 2014.

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

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[31] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proc. CVPR*, 2016, pp. 961–971.

[32] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *Proc. ECCV*, 2016, pp. 816–833.

[33] J. Bütepage, M. J. Black, D. Kragic, and H. Kjellström, "Deep representation learning for human motion prediction and classification," in *Proc. CVPR*, 2017.

[34] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognit. Lett.*, vol. 42, pp. 11–24, 2014.

[35] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 14–22, 2012.

[36] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[37] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proc. AllThingsCellular*, 2016, pp. 1–6.

[38] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos." in *Proc. BigData*, 2016, pp. 1161–1170.

[39] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov, "Head tracking for the oculus rift," in *Proc. ICRA*, 2014, pp. 187–194.

[40] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze prediction in dynamic 360 immersive videos," in *Proc. CVPR*, 2018, pp. 5333–5342.

[41] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *Proc. NOSSDAV*, 2017, pp. 67–72.

[42] J. Chakareski, R. Aksu, X. Corbillon, G. Simon, and V. Swaminathan, "Viewport-driven rate-distortion optimized 360º video streaming," in *Proc. ICC*, 2018, pp. 1–7.

[43] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Hu, "An optimal spatial-temporal smoothness approach for tile-based 360-degree video streaming," in *Proc. VCIP*, 2017, pp. 1–4.

[44] Samsung, "Samsung vr," 2018. [Online]. Available: https://samsungvr.com

[45] W. Bros., "Kong vr," 2018. [Online]. Available: https://samsungvr.com/view/AJXWuVU5E3Q

[46] Wikipedia, "Equirectangular map," 2018. [Online]. Available: https://www.wikiwand.com/en/Equirectangular_projection

[47] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. KDD*, 2016, pp. 785–794.
[48] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proc. DLRS*, 2016, pp. 7–10.
[49] Z. Ma, M. Xu, Y.-F. Ou, and Y. Wang, "Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 5, pp. 671–682, 2012.
[50] W. Ding and B. Liu, "Rate control of mpeg video coding and recording by rate-quantization modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, pp. 12–20, 1996.
[51] Wikipedia, "Knapsack problem," 2018. [Online]. Available: https://www.wikiwand.com/en/Knapsack_problem
[52] Keras, "Keras," 2018. [Online]. Available: https://keras.io
[53] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. NIPS*, 2007, pp. 153–160.
[54] S. Cost and S. Salzberg, "A weighted nearest neighbor algorithm for learning with symbolic features," *Machine Learning*, vol. 10, no. 1, pp. 57–78, 1993.
[55] X. Hou, Y. Lu, and S. Dey, "Novel hybrid-cast approach to reduce bandwidth and latency for cloud-based virtual space," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 14, no. 3s, p. 58, 2018.
[56] Speedtest, "Speedtest.net," 2018. [Online]. Available: http://www.speedtest.net/
[57] S. Winkler, *Digital video quality: vision models and metrics*, 2005.

Xueshi Hou (SM'17) received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 2015. She is currently working toward the Ph.D. degree with the University of California, San Diego, La Jolla, CA, USA.

Her research interests include multimedia, virtual reality, mobile computing, and wireless communications.

Sujit Dey (SM'03-F'14) is a Professor in the Department of Electrical and Computer Engineering, the Director of the Center for Wireless Communications, and the Director of the Institute for the Global Entrepreneur at University of California, San Diego. He heads the Mobile Systems Design Laboratory, developing innovative and sustainable edge computing, networking and communications, multi-modal sensor fusion, and deep learning algorithms and architectures to enable predictive personalized health, immersive multimedia, and smart transportation applications. He has created inter-disciplinary programs involving multiple UCSD schools as well as community, city and industry partners; notably the Connected Health Program in 2016 and the Smart Transportation Innovation Program in 2018. In 2017, he was appointed as an Adjunct Professor, Rady School of Management, and the Jacobs Family Endowed Chair in Engineering Management Leadership.

Dr. Dey served as the Faculty Director of the von Liebig Entrepreneurism Center from 2013-2015, and as the Chief Scientist, Mobile Networks, at Allot Communications from 2012-2013. In 2015, he co-founded igrenEnergi, providing intelligent battery technology and solutions for EV mobility services. He founded Ortiva Wireless in 2004, where he served as its founding CEO and later as CTO and Chief Technologist till its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless till its acquisition by Broadcom in 2004, and as an advisor to multiple companies including ST Microelectronics and NEC. Prior to joining UCSD in 1997, he was a Senior Research Staff Member at NEC C&C Research Laboratories in Princeton, NJ. He received his Ph.D. in Computer Science from Duke University in 1991.

Dr. Dey has co-authored more than 250 publications, and a book on low-power design. He holds 18 U.S. and 2 international patents, resulting in multiple technology licensing and commercialization. He has been a recipient of nine IEEE/ACM Best Paper Awards, and has chaired multiple IEEE conferences and workshops. Dr. Dey is a Fellow of the IEEE. For Dr. Dey's list of publications, please see http://esdat.ucsd.edu/publications.

Jianzhong Charlie Zhang (F'16) is SVP and head of the Standards and Mobility Innovation Lab of Samsung Research America, where he leads research, prototyping, and standards for 5G and future multimedia networks. From 2009 to 2013, he served as the Vice Chairman of the 3GPP RAN1 working group and led development of LTE and LTE-Advanced technologies such as 3D channel modeling, UL-MIMO, CoMP, Carrier Aggregation for TD-LTE. He received his Ph.D. degree from the University of Wisconsin, Madison.

Madhukar Budagavi (SM'05) is a Senior Director R&D in the Standards and Mobility Innovation Lab at Samsung Research America, where he leads R&D and standards activities in 5G multimedia processing, coding and networking. He currently represents Samsung in multimedia standards activities in MPEG and 3GPP and has been active in the past in UHD Alliance and SMPTE. He was a co-editor of the Springer book on High Efficiency Video Coding (HEVC): Algorithms and Architectures published in 2014. He received the Ph.D. degree in Electrical Engineering from Texas A & M University.