Predictive View Generation to Enable Mobile 360-degree and VR Experiences

Xueshi Hou[1], Sujit Dey[1], Jianzhong Zhang[2], Madhukar Budagavi[2]
[1]University of California, San Diego [2]Samsung Research America
x7hou@ucsd.edu,dey@ece.ucsd.edu,{jianzhong.z,m.budagavi}@samsung.com

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 [22], 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 view generation approach, where only the predicted view is extracted (for 360-degree video) or rendered (in case of VR) and transmitted in advance, leading to a simultaneous reduction in bandwidth and latency. The view generation 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 viewpoint prediction model and predictive view generation method to offer very high accuracy while simultaneously significantly reducing bandwidth.

CCS CONCEPTS

•**Computing methodologies** → *Virtual reality; Neural networks;*

KEYWORDS

Virtual reality, video streaming, 360-degree video

ACM Reference format:
Xueshi Hou[1], Sujit Dey[1], Jianzhong Zhang[2], Madhukar Budagavi[2]. 2018. Predictive View Generation to Enable Mobile 360-degree and VR Experiences. In *Proceedings of VR/AR Network'18, Budapest, Hungary, August 24, 2018,* 7 pages.
DOI: 10.1145/3229625.3229629

1 INTRODUCTION

Over the last few years, significant interest has emerged in the adoption of Virtual Reality (VR) and Augmented Reality (AR) in various fields, including entertainment, enterprise, education, manufacturing, transportation, etc. According to market research like [4], VR and 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 and AR technology [40], like cheaper price and compelling content, and most importantly a truly mobile VR/AR 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/AR experience on the mobile HMDs using bandwidth constrained mobile networks, while satisfying the ultra-low latency requirements.

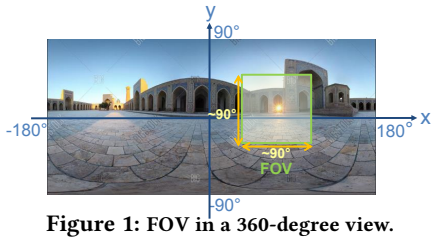


Figure 1: FOV in a 360-degree view.

Current widely used HMDs approximately include three types [1]: PC VR, console VR, mobile VR. Specifically, PC VR is tethered with PC [24, 34]; console VR is tethered with a game console [38]; mobile VR is untethered with PC/console but with a smartphone inside [18, 36]. 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 heavy computational tasks on the cloud/edge server and thus enable mobile VR with lightweight VR glasses. The most challenging part of this solution is 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) [22].

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°×90° for popular HMDs while the 360-degree view is 360°×180° 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 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 do predictive rendering (in case of VR) or extracting FOV (in case of 360-degree video) on the edge device, and then stream predicted

VR/AR Network'18, Budapest, Hungary

DOI: 10.1145/3229625.3229629

FOV to the HMD in advance. Thus, latency needed will be significantly reduced since the view is delivered and pre-buffered on the HMD when the prediction is successful; latency remains the same with the traditional streaming method when prediction is inaccurate. Moreover, bandwidth consumption can be considerably reduced since predicted FOV is streamed instead of the whole 360-degree view. 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 view generation 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.
- Based on viewpoint prediction, we develop optimal FOV generation method which ensures the desired tradeoff between bandwidth savings and prediction accuracy.
- We come up with this predictive view generation idea and show good results on a large-scale real head motion trace dataset from over 36,000 viewers for nineteen 360-degree/VR videos. We demonstrate significant bandwidth savings while ensuring very high accuracy with FOV prediction.

The rest of the paper is organized as follows. In §2, we review related work. §3 describes the system overview and problem definition. §4 describes our dataset and its characteristics. The methodology for viewpoint prediction is described in §5. We present our experimental results in §6, and conclude our work in §7.

2 RELATED WORK

In this section, we focus our review of related work on the following two fields.

FOV-guided streaming: Current FOV-guided 360-degree video streaming studies mainly consist of two types to address bandwidth challenge: *tiling* and *versioning* [30]. 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 [16, 25, 31]. 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 [13]. 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.

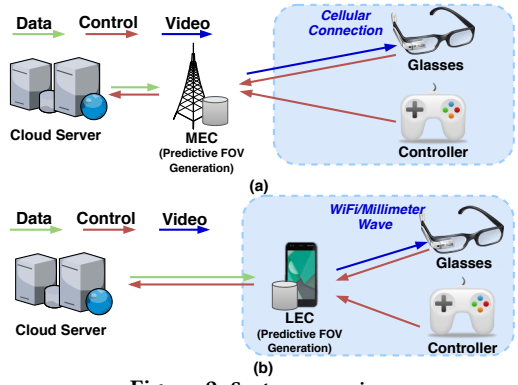


Figure 2: System overview.

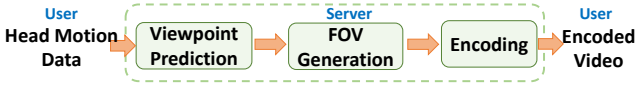


Figure 3: Proposed predictive view generation procedure.

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 [5]. 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 [3, 7] have proven fast and effective performance for many sequence prediction tasks [27, 41]. Bootstrap-aggregated decision trees (BT) [7] is one of efficient methods among them. On the other hand, deep learning methods such as recurrent neural networks (RNN) and their variants including LSTM networks [20] and gated recurrent units (GRU) [12] have proven to be successful for sequence prediction tasks [2, 29]. Apart from RNN and their variants, there are also some studies [8, 28] using deep neural networks including deep belief networks (DBN) [32] and stacked sparse autoencoders (SAE) [19] to achieve sequence prediction. However, research efforts [10, 17, 33, 42] for mobility prediction mostly focus on pattern-based methods. These studies discover pre-defined mobility patterns (e.g. sequential patterns, periodic patterns) from the trajectory traces, give predictions based on these extracted patterns, and thus suffer from the one-sided nature of pre-defined patterns [15]. Compared with other 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) [2, 29]. 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. We propose this predictive view generation idea to reduce both latency and bandwidth and show good results on a large-scale real head motion trace dataset.

3 SYSTEM OVERVIEW

In this section, we describe our system overview. Note that our predictive view generation approach works for both 360-degree videos and cloud/edge-based VR applications, since it refers to (i) extracting tiles (in case of 360-degree videos), and (ii) rendering the view (in case of cloud/edge-based VR). 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

Table 1: 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

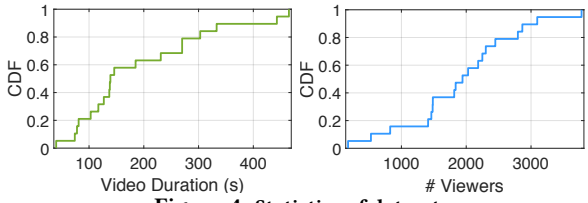


Figure 4: Statistics of dataset.

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 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 view generation techniques we propose will apply to either of the edge device options.

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 view generation, and stream the predicted FOV to the user HMD in advance. Later, if the actual FOV is within predicted FOV, the prediction is successful and the predicted FOV can be displayed on HMD immediately; otherwise, the actual FOV will be computed and transmitted from the edge device. For the former case, latency needed will be significantly reduced since the view is delivered and pre-buffered on the HMD before it is needed; for the latter, latency remains the same with the conventional method of streaming from the edge server. The key to achieving efficient predictive view generation is solving the problem of viewpoint prediction stated below.

Problem Statement: A viewpoint can occur in up to N different tiles in each time point (e.g. every 200ms). We decompose the whole predictive view generation procedure into two subtasks: *viewpoint prediction* and *FOV generation*, 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. *FOV generation* involves taking a list of most possible tiles predicted during viewpoint prediction and mapping each of the predicted tiles to a FOV. After determining the video content or rendering pixels based on predicted FOV, frames can be further encoded to a video and delivered to users.

4 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

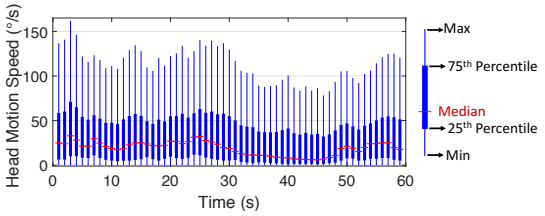


Figure 5: Head motion speed versus time in *Kong VR*.

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 milliseconds. Basic statistics of our head motion trace data are shown in Table 1. This dataset contains head pose data for 19 online VR videos, which are available on the Samsung VR website [37] 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 data set, and the large number of viewers for each video, makes the data set 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.

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

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 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 6 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 N tiles in one 360-degree view, we have N 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.

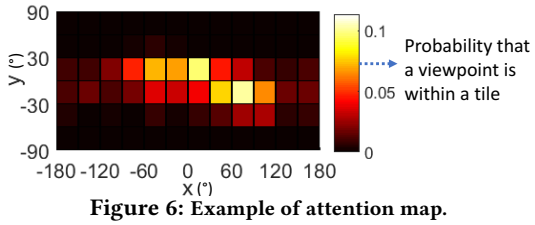
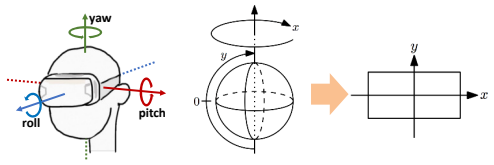


Figure 6: Example of attention map.

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 [35] 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.

5 PREDICTIVE LSTM MODEL

In this section, we describe our methodology including *viewpoint prediction* and *FOV generation*. 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. *FOV generation* will map each of predicted tiles to a FOV based on the results of viewpoint prediction.



(a) Euler Coordinates (b) Projection

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

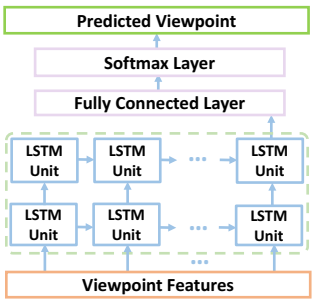


Figure 8: LSTM model used for viewpoint prediction.

5.1 Viewpoint Prediction

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 [39] 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°×30°, 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 [13]. For training the model, we design a one-hot encoding representation [9, 11] 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 1x10 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 [2, 29], 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 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

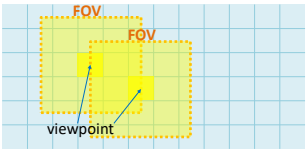


Figure 9: FOV generation.

LSTM units, 72 nodes and 30 as mini-batch size are selected during experiments and proved to be good by empirical results.

5.2 FOV Generation

The objective of FOV generation is maximizing the probability that the actual user view will be within the predicted FOV 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). Note that the FOV prediction accuracy depends on the LSTM model accuracy and FOV generation method, and thus reflects both the performance of our LSTM model and FOV generation method.

There is a tradeoff between FOV size (hence bandwidth consumed) and FOV prediction accuracy. FOV prediction accuracy is 100% (meaning that the user view has a 100% guarantee of being within predicted FOV) if predicted FOV is the whole 360-degree view (all tiles) but it will have very high bandwidth consumption. By selecting more tiles with high viewpoint probability, we can achieve higher FOV prediction accuracy but also higher bandwidth consumption. In this paper, we propose the following FOV generation method:

- 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°×120° 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°×90° in size (i.e. 90°×90° when the viewpoint is at the corner of predicted tile and 120°×120° when the viewpoint is in the center of predicted tile). We can use choice of m to achieve the desired tradeoff between FOV prediction accuracy and bandwidth consumed in transmitting the predicted FOV. Choice of larger m leads to higher bandwidth but better FOV prediction accuracy, while smaller m causes lower bandwidth but higher risk in FOV prediction accuracy. Note that an alternative strategy, in which we also transmit the remaining tiles but in lower quality, maybe used to mitigate the risk of prediction accuracy, and effectuating a different tradeoff with bandwidth needed. We plan to study the alternative approach in our subsequent work.

6 EXPERIMENTAL RESULTS

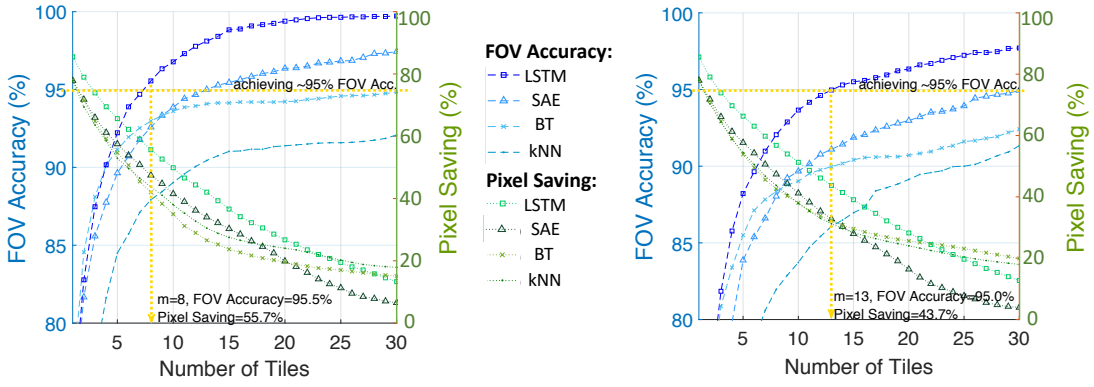
We use 90% of the dataset for training the LSTM viewpoint prediction model, and 10% for testing, ensuring the test data is from viewers which are different than those in training data. Specifically, we have 32400 samples as training data and 3600 samples as test data for both medium motion and high motion sequences in Fig. 10 and Table 2, while we take 45000 samples as training data and 5000 samples as test data for each of three sequences in Table 3. 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 [26]. We compare the performance of our LSTM model with state-of-the-art methods as follows:

- *Stacked sparse autoencoders* ***(SAE):*** We use SAE [6, 19] 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 [7], 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 [14] using 10-timestamp tile information as input and set 100 as the number of nearest neighbors in our training.

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, 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. 10, Table 2 and Table 3, *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. 10 and Table 2. 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 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, gre-en cross markers and green point markers represent corresponding pixel saving for these models.

From Fig. 10, 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. 10(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. 10(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. 10(b) to achieve a comparable FOV prediction accuracy of 95.0%, LSTM



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

Figure 10: (a)(b) show FOV prediction accuracy and pixel saving versus number of tiles selected for FOV (for two sequences in *Kong VR*).

Table 2: 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 3: 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

needs a larger FOV generated by 13 tiles (i.e. $m = 13$) with lower pixel savings of 43.7%. Table 2 summarizes the experimental results shown in Fig. 10. 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 3 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 [21, 23]. Thus, our experimental results also illustrate the tradeoff between FOV prediction accuracy and bandwidth savings.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a predictive view generation 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.

Our planned future work includes further development and evaluation of strategies for FOV generation, considering interpolated viewpoint positions within the interval between time point (i.e. 200ms) to generate view and encode videos in advance, evaluation of the proposed predictive approach from bandwidth and latency perspectives, and performing subjective studies to understand and quantify user experience using our proposed predictive approach. We also plan to study and develop predictive models for body motion, and joint head and body motion, for six Degrees of Freedom (6DoF) immersive experiences.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Wireless Communications at UC San Diego.

REFERENCES

[1] Adobe. 2016. Capitalizing on Viewers' Hunger for Virtual and Augmented Reality White Paper. (2016). https://www.creativeplanetnetwork.com/videoedge/362797
[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*. 961–971.
[3] Yali Amit and Donald Geman. 1997. Shape quantization and recognition with randomized trees. *Neural computation* 9, 7 (1997), 1545–1588.
[4] Heather Bellini. 2016. The Real Deal with Virtual and Augmented Reality. (2016). http://www.goldmansachs.com/our-thinking/pages/virtual-and-augmented-reality.html
[5] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*. 1171–1179.
[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. Greedy layer-wise training of deep networks. In *NIPS*. 153–160.
[7] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.
[8] Judith Bütepage, Michael J Black, Danica Kragic, and Hedvig Kjellström. 2017. Deep representation learning for human motion prediction and classification. In *CVPR*. 2017.
[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*. ACM, 785–794.
[10] Chen Cheng, Haiqin Yang, Irwin King, and Michael R Lyu. 2012. Fused Matrix Factorization with Geographical and Social Influence in Location-Based Social Networks.. In *Aaai*, Vol. 12. 17–23.
[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & deep learning for recommender systems. In *DLRS*. ACM, 7–10.
[12] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[13] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *ICC*. IEEE, 1–7.
[14] Scott Cost and Steven Salzberg. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine learning* 10, 1 (1993), 57–78.
[15] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *WWW*. ACM.
[16] Vamsidhar Reddy Gaddam, Michael Riegler, Ragnhild Eg, Carsten Griwodz, and Pål Halvorsen. 2016. Tiling in interactive panoramic video: Approaches and evaluation. *IEEE Transactions on Multimedia* 18, 9 (2016), 1819–1831.
[17] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. 2007. Trajectory pattern mining. In *KDD*. ACM, 330–339.
[18] Google. 2018. Google Daydream. (2018). https://vr.google.com/daydream/
[19] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313, 5786 (2006), 504–507.
[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[21] Xueshi Hou and Sujit Dey. 2017. Online appendix. (2017). http://esdat.ucsd.edu/sites/esdat.ucsd.edu/files/publications/Appendix.pdf
[22] Xueshi Hou, Yao Lu, and Sujit Dey. 2017. Wireless VR/AR with Edge/Cloud Computing. In *ICCCN*. IEEE.
[23] Xueshi Hou, Yao Lu, and Sujit Dey. 2018. Novel Hybrid-Cast Approach to Reduce Bandwidth and Latency for Cloud-Based Virtual Reality. *Transactions on Multimedia Computing, Communications, and Applications (TOMM), to appear* (2018).
[24] HTC. 2018. HTC Vive. (2018). https://www.vive.com/us/
[25] Ran Ju, Jun He, Fengxin Sun, Jin Li, Feng Li, Jirong Zhu, and Lei Han. 2017. Ultra Wide View Based Panoramic VR Streaming. In *VR/AR Network*. ACM, 19–23.
[26] Keras. 2018. Keras. (2018). https://keras.io
[27] Taehwan Kim, Yisong Yue, Sarah Taylor, and Iain Matthews. 2015. A decision tree framework for spatiotemporal sequence prediction. In *KDD*. ACM, 577–586.
[28] Martin Längkvist, Lars Karlsson, and Amy Loutfi. 2014. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters* 42 (2014), 11–24.
[29] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. 2016. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*. Springer, 816–833.
[30] Xing Liu, Qingyang Xiao, Vijay Gopalakrishnan, Bo Han, Feng Qian, and Matteo Varvello. 2017. 360 Innovations for Panoramic Video Streaming. In *HotNets*. ACM, 50–56.
[31] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran, and Marco Dias Silva. 2017. VR is on the edge: How to deliver 360 videos in mobile networks. In *VR/AR Network*. ACM, 30–35.
[32] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton. 2012. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (2012), 14–22.
[33] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. Wherenext: a location predictor on trajectory pattern mining. In *KDD*. ACM, 637–646.
[34] Oculus Rift. 2018. Oculus. (2018). https://www.oculus.com
[35] Samsung. 2018. Kong VR. (2018). https://samsungvr.com/view/AJXWuVU5E3Q
[36] Samsung. 2018. Samsung Gear VR. (2018). https://www.samsung.com/us/mobile/virtual-reality/
[37] Samsung. 2018. Samsung VR. (2018). https://samsungvr.com
[38] Sony. 2018. Playstation VR. (2018). https://www.playstation.com/en-us/explore/playstation-vr/
[39] Wikipedia. 2018. Equirectangular map. (2018). https://en.wikipedia.org/wiki/Equirectangular_projection/
[40] Chris Wiltz. 2017. Five Major Challenges for VR to Overcome. (April 2017). https://www.designnews.com/electronics-test/5-major-challenges-vr-overcome/187151205656659/
[41] Chenggang Yan, Yongdong Zhang, Jizheng Xu, Feng Dai, Liang Li, Qionghai Dai, and Feng Wu. 2014. A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors. *IEEE Signal Processing Letters* 21, 5 (2014), 573–576.
[42] Chao Zhang, Keyang Zhang, Quan Yuan, Luming Zhang, Tim Hanratty, and Jiawei Han. 2016. Gmove: Group-level mobility modeling using geo-tagged social media. In *KDD*. ACM, 1305–1314.