Shooting a Moving Target: Motion-Prediction-Based Transmission for 360-Degree Videos

Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli and Xin Liu
Department of Computer Science, University of California, Davis, CA 95616, USA
{ynbao, hswu, txzhang, arramli, xinliu}@ucdavis.edu

***Abstract*—Enabled by the rapid development of virtual reality hardware and software, 360-degree video content has proliferated. From the network perspective, 360-degree video transmission imposes significant challenges because it consumes 4∼6x the bandwidth of a regular video with the same resolution. To address these challenges, in this paper, we propose a motion-prediction-based transmission mechanism that matches network video transmission to viewer needs. Ideally, if viewer motion is perfectly known in advance, we could reduce bandwidth consumption by 80%. Practically, however, to guarantee the quality of viewing experience, we have to address the random nature of viewer motion. Based on our experimental study of viewer motion (comprising 16 video clips and over 150 subjects), we found the viewer motion can be well predicted in 100∼500ms. We propose a machine learning mechanism that predicts not only viewer motion but also prediction deviation itself. The latter is important because it provides valuable input on the amount of redundancy to be transmitted. Based on such predictions, we propose a targeted transmission mechanism that minimizes overall bandwidth consumption while providing probabilistic performance guarantees. Real-data-based evaluations show that the proposed scheme significantly reduces bandwidth consumption while minimizing performance degradation, typically a 45% bandwidth reduction with less than 0.1% failure ratio.**

I. Introduction

Since 2015, virtual reality (VR) has become increasingly popular, propelled by big developments in emerging VR hardware and software, including head-mounted displays (HMDs) such as Facebook's Oculus and HTC's Hive. One major VR application is watching 360-degree videos, also called spherical videos, immersive videos, or 360 videos. From a network perspective, providing 360-degree videos is challenging. First, the transmission of a 360-degree video consumes 4∼6x the bandwidth of a regular one with the same resolution [1], [2]. Second, because HMDs are close to the eyes, they demand higher video resolution; typically, a good viewing experience requires resolution of 6K instead of 4K. Third, an HMD cannot be shared with other viewers, and therefore even in a small room, there could be more than one 360-degree video streaming. Because of these factors, the increasing popularity of 360-degree videos will impose significant bandwidth demands on networks, and satisfying these demands is challenging, in particular for wireless networks.

However, not all data is utilized equally. HMD-based 360-degree video viewing has a unique attribute: when watching a 360-degree video, at any given time, the viewer is facing a certain direction. Therefore, only the content in this viewing direction needs to be rendered and displayed on the HMD, which is typically 20% of the whole sphere. Therefore, if the viewer's viewpoint (defined in Sec. II-A) can be predicted well, he/she needs to receive only a portion of the content, greatly reducing the bandwidth consumption. This observation motivates us to consider the following two questions:

- Is it feasible to predict viewer motion and what is the time scale for relatively accurate predictions?
- Due to its random nature, motion prediction is prone to error. Based on such information, how shall we design transmission schemes that minimize network bandwidth consumption while providing viewing experience guarantees?

To answer these questions, we first built a testbed to collect 3D motion data using HMDs [4]. Wearing an HMD, a viewer's motion has three degrees of freedom (pitch, yaw, and roll), as illustrated in Fig. 1. Our experiment collected motion data in the three dimensions, based on 16 clips of 360-degree video and 153 viewers. In analyzing the collected data, we found that viewer motion has strong short-term auto-correlations in all three dimensions. Using regression techniques, one can predict viewer motions with reasonable accuracy on a time scale of 100∼500 ms, as shown in Sec. IV.

However, due to the random nature of viewer motion, motion prediction is prone to error. Given such error-prone prediction, how can we provide viewing experience guarantees? Our answer is to provide an appropriate level of redundancy in the amount of video content transmitted. Specifically, we not only predict viewer motion, but also estimate the accuracy of the prediction. Estimating accuracy provides an additional indicator of the amount of redundancy to be transmitted. The intuition is as follows: if we know the prediction accuracy is high, we need to transmit only the content in a relatively small area corresponding to the actual viewing direction. On

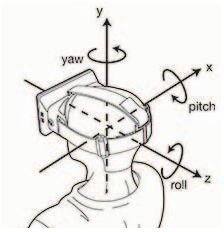


Fig. 1. The three angles [3]

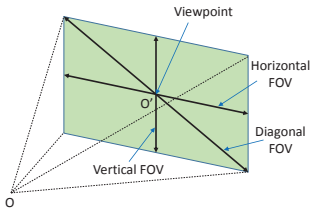


Fig. 2. Field of view

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

the other hand, if the prediction accuracy is low, we enlarge the transmitted area to maintain the quality of viewing experience. In other words, the prediction of motion and the transmission of content are closely related.

Based on the motion prediction and its accuracy, we design a set of transmission schemes to decide which portion and how much of the content we should transmit to the viewer. The objective is to minimize the required bandwidth while upholding viewing quality guarantees. Our transmission schemes consider transmission redundancy size as well as partial or full frame transmission threshold, as discussed in detail in Sec. V. Our result shows that in predicting the next 0.2s, given a failure ratio of 0.1%, our scheme reduces bandwidth by more than 45%.

In summary, our work makes the following contributions:

- We collect motion data for 153 subjects watching 360-degree videos. From the collected data, we observe a strong short-term auto-correlation in viewer motions, which indicates that viewer motion can be well predicted based on motion history.
- We develop regression models to predict viewer's viewpoint and the prediction's deviation. The deviation provides additional information that can be used to decide the amount of redundancy needed.
- We design algorithms that utilize the predicted result for efficient 360-degree video transmission. Evaluations based on our collected data show that the proposed algorithms can reduce bandwidth consumption by more than 45%, given a 0.2s prediction window and a failure ratio of less than 0.1%.

II. Problem Statement

A. Basics of 360-Degree Videos

In this work, we consider 360-degree video viewing (or playback) using HMDs, such as Facebook's Oculus and HTC's Hive. A 360-degree video is typically created by recording a real-world panorama, where the view in every direction is recorded simultaneously using an omnidirectional camera or a collection of cameras. The images of different directions are stitched together using software. Each stitched image is called a frame. Such frames are transmitted to viewers as regular video transmission [5]. During playback (or viewing), the HMD senses the viewing direction of the viewer. Based on the viewing direction, the corresponding portion of the image, defined by the Field Of View (FOV), is rendered and then displayed on the HMD.

The FOV determines the extent of the observable world that can be seen [6], as shown in Fig. 2. Vertical FOV is the range of angle from the top to the bottom, horizontal FOV is from the farthest left to the farthest right, and diagonal FOV is from the top-left corner to the bottom-right corner. For Oculus DK2, its vertical, horizontal, and diagonal FOVs are $90°$, $110°$, and $120°$, respectively [7]. We call the center of the image that the viewer is watching the **viewpoint**, i.e., point O' in Fig. 2. The pitch and yaw angles in Fig. 1 determine the viewpoint. For a given device, the distance between the viewer's eyes and the display is fixed, and so are the vertical, horizontal and diagonal FOVs. Thus the viewpoint and the roll angle (the Z axis rotation as shown in Fig. 1) decide the area of the image seen by the viewer. Note that the Euler's angles (pitch, yaw, roll) are corresponding to the (X, Y, Z) axes that the head rotates around. Therefore, we refer to them as X, Y and Z angles, respectively, for simplicity in the following.

B. Motivations

Currently, frames, comprising panoramic images of 360-degree videos, are transmitted from a video server to a client. However, at any given time, because a viewer can watch only a portion of the whole sphere defined by the viewpoint and the roll angle, the viewer consumes about only 20% of the transmitted data.

This unique attribute of 360-degree video viewing provides an opportunity to reduce bandwidth consumption. Ideally, if we know the viewpoint and the roll angle in advance, we can transmit only the corresponding portion of the image instead of the whole panoramic image. In other words, we can transmit a portion of the frame instead of the whole frame, and thus reduce the bandwidth consumption significantly.

Practically, to achieve this goal, we need to predict viewer motion with high accuracy and utilize such information for intelligent transmissions while providing viewing experience guarantees. Due to the random nature of viewer motion, both the predicted viewpoint and the predicted roll angle could be inaccurate. Thus, we also need an indicator for these unpredictable situations, along with corresponding methods to address this issue, in order to guarantee viewing experience. To the best of our knowledge, this work is the first attempt to study the feasibility of motion prediction, and to propose viable transmission mechanisms based on motion prediction for 360-degree videos.

C. Proposed Framework

Fig. 3 illustrates the key components of the proposed framework. At a given time t, based on the collected motion traces from the viewer, the video server or its proxy predicts the viewer motion for $t + T_r$, and decides the viewpoint and the size of the partial frame (a full frame is considered as a special case of the partial frame.). T_r is the prediction window. The partial frame is then transmitted to the viewer. At time $t + T_r$ of the playback, depending on the viewpoint and the roll angle of the viewer, the device renders the area in the viewer's FOV. If all the pixels in the FOV are included in the partial frame, the viewing is considered a success. However, if the pixels are not completely included in the partial frame, the viewing is called a failure.

We propose a performance metric, **failure ratio**, to quantify viewer experience. Specifically, the failure ratio is defined as the percentage of frames in which the viewer's FOV is not completely transmitted. A failure ratio is prescribed to provide probabilistic guarantees for viewing experience (for example,

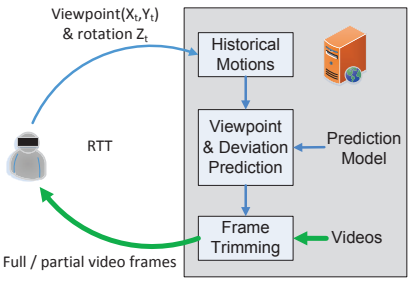


Fig. 3. System Framework

from 1% to 0.01%). Our objective is to minimize the overall required bandwidth given the failure ratio constraint.

We note that our definition of failure is rather conservative. After all, when a failure happens, it is highly possible that only a small portion of the FOV is not included in the transmitted partial frame (shown in Sec. VI). Therefore to complement this performance metric, we further define a **Ratio of Missing Pixels (RMP)**, as the ratio of the missing pixels to the total pixels viewed. We use the failure ratio as the viewing experience constraint in the algorithm development and complement it with the RMP in the numerical evaluation. This approach provides a more comprehensive understanding of user experience.

D. Time Scale

In the proposed framework, we need to select an appropriate prediction window, T_r. Clearly, there is a tradeoff in deciding T_r. The shorter the T_r, the more accurate the motion prediction, but the less time available to transmit the predicted frame. Specifically, T_r needs to be large enough to include the time for a viewer's viewpoint signal to transmit from the client to the 360-degree video server/proxy, the time for a frame to be transmitted from the server to the client, the processing time of the sensors on the HMDs, and the video frame processing and rendering time.

Therefore, an appropriate value of T_r depends on the network setting. We assume that the viewer and the server are relatively close. This is a reasonable assumption in today's networks because CDNs are widely deployed for content delivery that pushes contents closer to viewers for better viewing experience. Cellular network providers use similar techniques to push content close to the edge of networks (base stations or eNodeBs). Alternatively, we could also assume a proxy server that is located near the base station, so that the content transmission occurs between the viewer and the proxy.

Next, we consider the round trip time (RTT) between a viewing device and its nearby server. For different networks (e.g., wired, WiFi, LTE, etc.), different settings, and realtime traffic, this time could be different. However, given the server is nearby, the RTT is normally limited to 20∼50ms. For example, in LTE, the transmission frames are 1ms each. The RTT, including uplink and downlink transmission time, buffering time, retransmission time, UE and eNodeB processing delay, resource request and grant time, is below 15ms (for pre-allocated resources) and 21ms (for scheduled resources) [8], [9], [10]. Therefore, if the server is near the eNodeB, 20ms is a reasonable estimate for RTT.

In this work, we hope to focus on failures caused by user behavior, prediction errors and our transmission schemes. Therefore, we assume the requested partial frame can be successfully transmitted to the viewer within the T_r time frame. This assumption requires that we choose T_r several times of the RTT to be conservative.

Combining all these factors, we conclude that, for our purpose, 100∼500ms is a reasonable range of T_r. We use these values in our trace-driven evaluations.

E. Prediction and Transmission

We design motion algorithms for predicting future motions. As a first attempt, we consider the features of motion prediction to be the past and current motions of the viewer. Other potential features, such as the viewer's motion pattern, other viewers' motion history, and video content, will be considered in the future to further improve the prediction accuracy.

One essential and unique requirement of our prediction model is that we need to estimate not only the viewpoint, but also the accuracy of our predictions. This is critical because the accuracy can help us determine how much redundancy we must transmit to accommodate the random nature of viewer motion. At one extreme, if we know that the prediction is perfect, each partial frame needs to contain only the area of viewer's FOV. At the other extreme, if we know that the prediction is highly inaccurate, we may need to transmit the entire frame. In other words, the prediction and the transmission are closely related. The relationship requires that we predict not only the viewpoint, but also its accuracy. Based on such predictions, we then propose transmission schemes. This joint prediction-transmission design allows us to significantly reduce the bandwidth consumption while guaranteeing the quality of the 360-degree video viewing experience.

We note that the VR hardware and software are fast evolving. Existing challenges include uneven projection from 3D to 2D, as well as nonlinearity pixel sampling. Therefore, in this work, we do not limit to the specific implementation of a particular VR hardware/software. We assume an ideal scenario where for a given viewpoint and a roll angle, a perfect 2D area defined by the FOV can be decoded from the sphere. Furthermore, we assume a round portion of the frame can be trimmed and transmitted. It is reasonable as modern video compression/transmission is done block by block, with each block of 8×8 or 16×16 pixels. Last, we ignore the specifics of video coding, which is our future work. These assumptions allow us to focus on the feasibility of viewer motion prediction and its utilization in video content transmissions.

III. Experiment Setup and Data Collection

To study the viewer motion with HMDs, we setup an experimental environment with Oculus DK2. We developed a software to play 16 videos automatically to viewers. After each view of a video clip, the subject was required to provide

a subjective score in the range of 1 to 5. This score is used to detect whether the video content is interesting to the viewer or not. On the other hand, it helps the subject concentrate on the whole video clip. When a subject is watching a video, his/her motion is recorded and logged. As usual, the motion includes 3 degrees of freedom, pitch, yaw and roll (i.e., X, Y, and Z angles). When wearing an HMD, the viewer's initial position defines the zero degree for pitch, yaw and roll. Each dimension is denoted by an angle (-180° to 180°).

A. Hardware and Software

We use the Oculus DK2 as the hardware, Oculus Runtime 7.0 as the hardware driver and *Color Eyes* as the video player. Subjects sit on a chair that can rotate horizontally 360°. The HMD of Oculus DK2 is connected to a PC with a cable.

B. 360-Degree Video Content and Motion Measurement

We downloaded 16 clips of 360-degree video from Youtube and cut each of them into 30s segments. Among the 16 videos, 14 are at 4K resolution, one at 2K resolution, and the other one at 1080P. All the videos run at 30 frames per second. The 16 selected videos cover a variety of popular 360-degree video content available from the Internet: 7 videos have sports content, e.g., basketball, boxing, soccer, skiing, etc., 4 videos contain landscape, e.g., Grand Canyon and tropical rain forest, and the other 5 videos contain entertainment activities, such as thrill rides and kite flying. In terms of camera movement, half of the videos have the camera fixed when shooting the video, while cameras of the other half videos move throughout the playback time. The video clips and sample motion data can be found at http://360videoexp.com/. These videos have typically a bitrate of 20-40Mbps.

For our first attempt, we removed the audio of the video clips, so that subjects motion was related only to visual content. Each subject was asked to watch the video clips in order, while his/her viewing motion was recorded. We took 7-9 measurements per second, each measurement including X, Y and Z angles.

C. Subjects

In total, 153 volunteers joined the experiment: 35 of them watched all 16 video clips, and 118 of them watched 3$\sim$5 randomly selected video clips. Most of the volunteers are people on campus, and most of them have their first VR experience in our experiment. The age distribution is as follows: 10$\sim$20 (36%), 20$\sim$30 (54%), 30$\sim$40 (4%), 40$\sim$50 (4%), 50$\sim$60 (2%). 38% of the volunteers were female, and 34% of the volunteers wore glasses.

D. Data Preprocessing

The overall collected data include 985 views from the subjects, totalling 8.2 recorded hours of viewer motion. Our software collected 7$\sim$9 samples per second, and the intervals between two samples were slightly random due to the processing load on hardware. To facilitate the following study, we generate uniformly 10 samples per second from the raw data using linear interpolation. Linear interpolation determines a sample based on the line segment connecting two neighboring raw data points. After interpolation, we have in total about 295500 samples.

E. Sample Distribution

We first plot the estimated cumulative distribution function (CDF) of motion samples in Fig. 4(a). It shows that viewers focus on the front center much more often than the other directions. For instance, in 90% of time, X, Y and Z angles are within $-33^\circ\sim33^\circ$, $-101^\circ\sim107^\circ$, and $-11^\circ\sim9^\circ$, respectively; in 99% of time, the X, Y and Z angles are within $-66^\circ\sim61^\circ$, $-171^\circ\sim169^\circ$, and $-28^\circ\sim27^\circ$, respectively. It is reasonable because most 360-degree videos have the best viewpoint in the front direction, such as roller coaster, skiing and boxing. Viewers may occasionally turn their heads, but they may soon turn back, due to the video content. The distribution of the Y angle, the horizontal angle, spreads out the most, while the Z angle, the roll angle, has a very small range.

To calculate the difference between two angles, whose values are both in $[-180, 180)$, we use the following equation

$$\Delta(\theta_1, \theta_2) = \text{mod}(\theta_1 - \theta_2 + 180^\circ,\ 360^\circ) - 180^\circ. \quad (1)$$

Fig. 4(b) shows the CDF for the change of angles within 0.2s. From this result, we can see viewers' movement is limited within 0.2s. For instance, in 90% of time, the X, Y and Z angles are within $-15.31^\circ\sim15.06^\circ$, $-34.73^\circ\sim35.54^\circ$, and $-8.61^\circ\sim8.62^\circ$, respectively; in 99.9% of time, the X, Y and Z angles are within $-25.64^\circ\sim25.85^\circ$, $-60.11^\circ\sim64.60^\circ$, and $-25.01^\circ\sim24.03^\circ$, respectively.

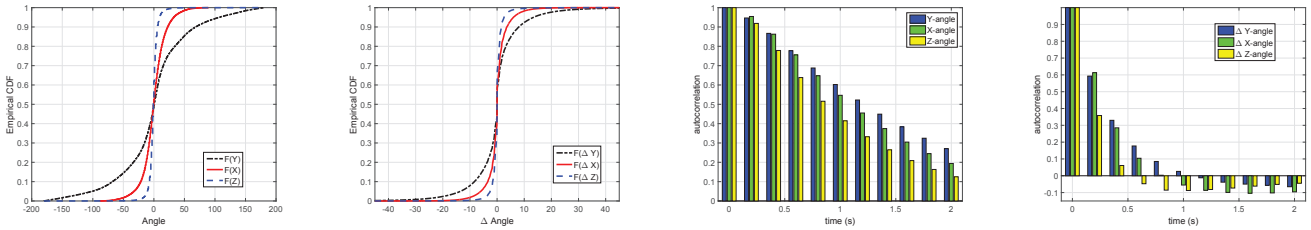
F. Temporal Auto-Correlation Analysis

In Fig. 4(c), we show the auto-correlation of the three angles for lags ranging from 0 to 2s. The figures show that, in a short period, e.g., 0.1s to 0.5s, the auto-correlation is very strong (larger than 0.8 for the X and Y angles, and larger than 0.7 for the Z angle). In Fig. 4(d), we show that the moving velocity also has a strong short-term temporal auto-correlation. For both X angle and Y angle, the correlation is about 0.6 within 0.2s, and for the Z angle, the correlation is about 0.36 for 0.2s. This auto-correlations are also reflected in frequency domain, as shown in Fig. 5. From the figure, we can see that the majority of the signal has spectrum less than 1 Hz, and there is very little high spectrum signal.

Based on these results, we can see that in a short period, viewer motion is predictable. However, in our application, we need to keep the large prediction error in a very small percentage, e.g., 0.1%, where a naive prediction model cannot satisfy the requirement. In the next part, we introduce machine learning based regression models to achieve this goal.

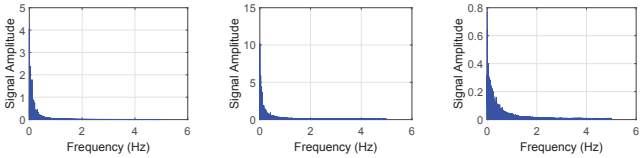
IV. Motion Prediction

In the previous section, we have seen that viewer motion shows strong temporal auto-correlation in small time scales. In this section, we develop regression models to predict viewer motions.



(a) The CDF of X, Y, Z angles. (b) The CDF of Δ angles (0.2s). (c) The auto-correlation of angles. (d) The auto-correlation of Δ angles.

Fig. 4. The CDFs and auto-correlations.



(a) X angle (b) Y angle (c) Z angle

Fig. 5. Single-sided amplitude spectrum of the traces.

Specifically, we only need to predict the viewpoint (as shown in Fig. 2) and its deviation. We do not need to predict the rotation around the Z axis (roll angle, as shown in Fig. 1). This is due to the transmission schemes that we develop, as discussed in detail in Sec. V.

A. Viewpoint Prediction

We predict the future viewpoint based on the current and past rotation status, which are treated as features in the prediction model. Note that for a fixed HMD, once the rotation angles around the X and Y directions are given, the viewpoint is determined. Therefore, predicting viewpoint is equivalent to predicting the X and Y angles.

Let X_t and Y_t be the angles of pitch and yaw at time t. Without loss of generality, we consider the initial viewpoint has $X_0 = 0$ and $Y_0 = 0$. Furthermore, let $\boldsymbol{X}_{t_1:t_2}$ and $\boldsymbol{Y}_{t_1:t_2}$ be the measurements of the X and Y angles from time t_1 to time t_2, respectively, i.e., $\boldsymbol{X}_{t_1:t_2} = (X_{t_1}, X_{t_1+1}, \ldots, X_{t_2})$ and $\boldsymbol{Y}_{t_1:t_2} = (Y_{t_1}, Y_{t_1+1}, \ldots, Y_{t_2})$.

At time t, we have measurements $\boldsymbol{X}_{0:t}$ and $\boldsymbol{Y}_{0:t}$ and we hope to predict (X_{t+T_r}, Y_{t+T_r}), which uniquely decides the viewpoint. This is a time-series regression problem. We treat angles in each direction independently and train two separate models for the prediction of X_t and Y_t. The reason is that auto-correlations are much stronger than the correlation between X angle and Y angle. As we have observed in Sec. III-F, the auto-correlations are strong in short period, and therefore we predict based on the motion data collected in a sliding window of size T_w, i.e.,

$$\hat{X}_{t+T_r} = f_{x,T_r}(\boldsymbol{X}_{t:(t-T_w)}); \tag{2}$$

$$\hat{Y}_{t+T_r} = f_{y,T_r}(\boldsymbol{Y}_{t:(t-T_w)}). \tag{3}$$

We consider three regression models: Naive, linear regression (LR), and neural networks (NN). The Naive model is the baseline model where we use the current angle as the value of the future angle, e.g., $\hat{X}_{t+T_r} = X_t$ and $\hat{Y}_{t+T_r} = Y_t$. For the NN model, we use 3 layers and 5 hidden neurons. We use 50% of data for training to minimize the sum of square error, and the other 50% for test to see the prediction accuracy.

We choose the LR and NN models because they are commonly used for regression. However, in our scenario, there is a special feature of angle prediction: because X_t and Y_t are both angles in the range of $[-180°, 180°)$, $-180°$ and $179°$ have a difference of only $1°$. However, if we directly use the values of angles in Eqs. (2) and (3), they are considered (significantly) different. To address this issue, we first project the angles to a circle with unit radius, use the projected physical location on the circle to do prediction, as shown in Eqs. (4) and (5), and then project the location back to the predicted angles using Eq. (6). Specifically, following equations show the relation between an angle θ_t and its projected point $(P_{d_1,t}, P_{d_2,t})$ in a $d_1 \times d_2$ space.

$$P_{d_1,t} = \sin(\theta_t); \tag{4}$$

$$P_{d_2,t} = \cos(\theta_t); \tag{5}$$

$$\theta_t = \arctan(P_{d_1,t}/P_{d_2,t}). \tag{6}$$

This technique reduces the prediction error by about 40% compared to using the angles directly in Eqs. (2) and (3).

Last, we define the prediction error as follows:

$$e^x_{t+Tr} = |\Delta(X_{t+T_r} - \hat{X}_{t+T_r})|; \tag{7}$$

$$e^y_{t+Tr} = |\Delta(Y_{t+T_r} - \hat{Y}_{t+T_r})|. \tag{8}$$

Table I shows the test error for the Y angle prediction. We choose Y angles because, compared to X angles, they are harder to predict. From Table I, we can see that the further we predict, the larger the error, for all methods. The NN models are the most accurate. Comparing the neural-network-based results and the naive prediction results, we can see a large gain of using regression methods. For example, for 0.2s prediction, all four indicators (mean, root-mean-square error (RMSE), 99th percentile and 99.9th percentile) have been improved by about 50%. For 0.3s prediction, all four indicators have been improved by 30%∼45%. The X angle has the similar improvement, which is omitted here.

B. Deviation Prediction

In the previous subsection, we have developed regression models to predict the viewpoint. Ideally, we wish to know

TABLE I
THE ERROR OF VIEWPOINT CENTER PREDICTION (Y ANGLE IN DEGREES)

T_r(s)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Naive (Mean)	2.58	5.09	7.53	9.89	12.16	14.33	16.39	18.39	20.29	22.06
Naive (RMSE)	4.71	9.10	13.23	17.06	20.58	23.84	26.82	29.70	32.33	34.76
Naive (99th)	18.24	35.24	50.85	65.14	77.89	89.22	99.25	108.87	117.75	125.23
Naive (99.9th)	32.59	62.75	88.90	112.90	130.25	143.52	154.57	161.44	166.44	170.43
LR (Mean)	0.95	2.53	4.47	6.56	8.72	10.87	12.97	15.02	17.03	18.92
LR (RMSE)	1.98	4.84	8.28	11.78	15.28	18.54	21.61	24.55	27.32	29.84
LR (99th)	7.00	18.89	33.17	47.18	60.82	72.47	83.20	93.11	102.28	110.14
LR (99.9th)	14.90	38.25	65.41	89.60	112.03	128.86	140.65	151.73	159.40	163.29
NN (Mean)	0.92	2.44	4.33	6.33	8.40	10.43	12.45	14.39	16.34	18.13
NN (RMSE)	1.92	4.52	7.77	11.09	14.48	17.61	20.64	23.51	26.27	28.76
NN (99th)	6.54	17.25	30.54	44.03	57.59	68.88	80.02	90.21	99.88	108.22
NN (99.9th)	14.34	35.16	61.02	84.46	107.14	124.75	137.81	149.68	157.24	162.83

TABLE II
THE ERROR OF VIEWPOINT DEVIATION PREDICTION (Y ANGLE IN DEGREES)

T_r(s)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LR (Mean)	0.79	2.12	3.78	5.53	7.26	8.98	10.60	12.13	13.59	14.95
LR (RMSE)	1.68	3.82	6.43	9.10	11.70	14.19	16.41	18.45	20.40	22.10
LR (99th)	5.57	14.93	26.14	37.89	48.93	59.25	67.92	76.13	83.92	90.25
LR (99.9th)	13.43	32.71	56.28	78.30	98.42	114.29	125.67	134.16	140.98	143.81
NN (Mean)	0.68	1.77	3.18	4.71	6.32	7.92	9.40	10.81	12.21	13.48
NN (RMSE)	1.50	3.34	5.73	8.24	10.74	13.13	15.24	17.19	19.08	20.71
NN (99th)	4.71	12.54	22.87	33.87	44.86	54.87	63.11	70.71	78.03	84.39
NN (99.9th)	11.52	28.33	50.07	71.77	91.65	108.35	120.50	128.87	135.95	138.74

the accuracy of such predictions. Intuitively, accuracy provides additional input on the amount of redundancy to be transmitted: if we know the prediction accuracy is high, we need to transmit the content in a relatively smaller area that contains the FOV in the actual viewing direction. On the other hand, if the prediction accuracy is low, we enlarge the area that is transmitted to the viewer.

In other words, knowing e^x_{t+Tr} and e^y_{t+Tr} helps us better decide the amount of content to be transmitted. Since we could not know the value of e^x_{t+Tr} and e^y_{t+Tr}, we propose to use regression methods to estimate them. Specifically, we have

$$\hat{e}^x_{t+T_r} = f_{e^x,T_r}(\hat{X}_{t+T_r}, \boldsymbol{X}_{t:(t-T_w)}); \quad (9)$$

$$\hat{e}^y_{t+T_r} = f_{e^y,T_r}(\hat{Y}_{t+T_r}, \boldsymbol{Y}_{t:(t-T_w)}). \quad (10)$$

Again, we use linear regression and neural networks as the prediction models. Table II shows that neural networks have better accuracy compared with linear regressions in predicting the Y angle error. The X angle has the similar result, and is omitted here.

V. MOTION-PREDICTION-BASED TRANSMISSIONS

In this section, moving beyond prediction, we design motion-prediction-based transmission algorithms. Based on the predicted viewpoint, we can reduce bandwidth transmission by targeting only at the area that the viewer will likely watch. However, to avoid viewing experience deterioration under motion randomness, we need to solve bandwidth minimization problem subject to viewing experience constraint.

Specifically, given a viewpoint prediction, in order to guarantee the viewer experience, we need to decide the additional area to transmit. In particular, since we only transmit partial frames, due to the presence of motion prediction error, the transmitted data may not cover the whole FOV that a viewer actually watches. As discussed in Sec. II-C, we define such an event as a failure. In order to guarantee viewer experience, we limit the failure ratio to be less than a predefined ratio, e.g., 0.1%. In this section, we first calculate the transmitted area, and then design several targeted transmission algorithms to leverage the prediction results for bandwidth optimization.

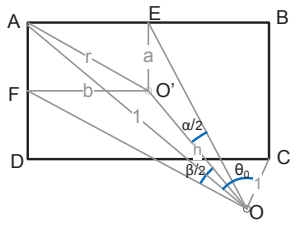


Fig. 6. The calculation of diagonal FOV.

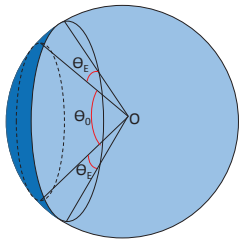


Fig. 7. The transmitted round.

A. Transmitted Area

The actual viewing area is a rectangle determined by the viewpoint and the roll angle. To guarantee the transmitted data covers the actual viewing area with high probability, we transmit more data than necessary. Basically, instead of rectangle, we transmit data for a round shape from the video server to the client. The techniques below are also applicable to other transmitted shapes. In the 3D space, this area corresponds to the angle ratio of a circular cone. In the following, we calculate the transmitted area for the targeted round.

We first calculate the diagonal FOV, which determines the beam angle of the targeted round. As shown in Fig. 6, a viewer is at the center denoted by point O, and $\square ABCD$ defines the viewing area. Let α and β be the vertical and horizontal FOVs, respectively. The following statement presents an equation to

calculate the diagonal FOV.

Statement 1: For a rectangle viewing area with vertical FOV α and horizontal FOV β, its diagonal FOV is given by:

$$\text{diag}(\alpha,\beta) = 2\arccos\left[\frac{1}{\sqrt{1+\tan^2(\frac{\alpha}{2})+\tan^2(\frac{\beta}{2})}}\right]. \quad (11)$$

As a reference point, the Oculus DK2 has 90° vertical FOV and 110° horizontal FOV. Therefore its diagonal FOV is 120°.

Proof: We next show how Eq. (11) is calculated.

As shown in Fig. 6, $\angle AOC$ is the diagonal FOV. The center of the rectangle is the viewpoint, denoted by point O'. Then $\angle EOO'$ is half of the vertical FOV and$\angle FOO'$ is half of the horizontal FOV, i.e., $\angle EOO' = \alpha/2$ and $\angle FOO' = \beta/2$.

We assume the distances from the point O to the vertices of $\square ABCD$ are 1 without loss of generality. Line segment OO' is a perpendicular to the rectangle $\square ABCD$ at point O', whose length is given by h. Then, considering right triangles $\triangle EO'O$ and $\triangle FO'O$, we have

$$a = h\tan(\tfrac{\alpha}{2}), \quad b = h\tan(\tfrac{\beta}{2}). \quad (12)$$

In the rectangle $\square AEO'F$, we have

$$r = \sqrt{a^2+b^2}. \quad (13)$$

Considering the right triangle $\triangle AOO'$, we have

$$h^2 + r^2 = 1. \quad (14)$$

Therefore, from Eqs. (12), (13) and (14), we have

$$h = \frac{1}{\sqrt{1+\tan^2(\alpha/2)+\tan^2(\beta/2)}}. \quad (15)$$

The conclusion then follows as $\angle AOC = 2\arccos(h)$. ■

We typically transmit data for the targeted round plus a margin, where the beam angle of the targeted round is θ_0 and the margin is θ_E, as shown in Fig. 7. Then the transmitted round has a beam angle of

$$\theta_{total} = \theta_0 + 2\theta_E. \quad (16)$$

The following statement reveals a sufficient condition for a successful transmission, i.e., the entire viewing area is covered by the transmitted area.

Statement 2: If the X angle prediction deviation e^x and the Y angle prediction deviation e^y meet the following constraint

$$\text{diag}(e^x, e^y) \le \theta_E, \quad (17)$$

then the transmitted round area covers all possible viewing area for the predicted frame.

Proof: From the X angle prediction deviation and Y angle prediction deviation, we can create a rectangle, whose diagonal FOV equals to $\text{diag}(e^x, e^y)$. This is the maximal angle that the center of the smaller round with FOV θ_0 can move away from the center of the larger round in Fig. 7. Therefore, when $\text{diag}(e^x, e^y) \le \theta_E$, the larger round with FOV of θ_{total} always covers any point in the smaller round. ■

Note that Eq. (17) is a sufficient but not a necessary condition. Because we transmit more data than necessary, it is still possible that actually no failure happens even Eq. (17) is violated.

As discussed earlier, we do not know (e^x, e^y) and thus we obtain their estimates $(\hat{e}^x, \hat{e}^y)$ by using the prediction models in Sec. IV. Then, we can estimate the diagonal deviation as

$$\hat{d} = \text{diag}(\hat{e}^x, \hat{e}^y). \quad (18)$$

The diagonal deviation $\hat{d}$ can be used to decide the transmission margin θ_E as discussed later, and the transmitted round has a beam angle of $\theta_0 + 2\theta_E$. For such a transmitted round, according to [11], its area ratio (angle ratio) can be calculated as follows

$$\text{area}(\theta_0+2\theta_E) = \begin{cases} \frac{1}{2} - \frac{1}{2}\cos(\frac{\theta_0}{2}+\theta_E), & \text{if } \theta_0+2\theta_E \le 360^\circ; \\ 1, & \text{otherwise.} \end{cases} \quad (19)$$

B. Transmission Algorithms

In the following, we propose three targeted transmission algorithms. Unlike the traditional machine learning paradigm, in addition to training data and test data, here we also use part of the data as *decision data*, i.e., the data that are used for transmission-decision making. Assume there are N samples in the decision data. For each frame $i = 1, 2, ..., N$, denote its real deviations by e_i^x and e_i^y, predicted deviations by $\hat{e}_i^x$ and $\hat{e}_i^y$, and diagonal deviation $\hat{d}_i = \text{diag}(\hat{e}_i^x, \hat{e}_i^y)$.

To measure the viewer experience, we introduce an indicator I_i^f to denote whether the frame i is a failure or not, where $I_i^f = 1$ if frame i suffers a failure and $I_i^f = 0$ otherwise. The indicator I_i^f is decided by the following parameters: the real prediction deviations e_i^x and e_i^y, the Z angle Z_i, and the beam angle of the transmitted round $\theta_0+2\theta_E$. Note that Statement 2 provides a sufficient condition for $I_i^f = 0$. In the remaining (small percentage) cases, to calculate I_i^f numerically, one can perform an exhaustive search in a grid of points with fine granularity on the FOV. Combining the two cases, we denote

$$I_i^f = f_f(e_i^x, e_i^y, Z_i, \theta_0 + 2\theta_E). \quad (20)$$

The following transmission algorithms decide transmitted area based on the prediction results, with the objective to minimize the required bandwidth subject to the failure ratio constraint r_f.

1) TT-A (Targeted Transmission All the time): TT-A transmits the data in the predicted targeted area for all frames. As discussed in Sec. V-A, due to the motion randomness, a round with beam angle $\theta_0+2\theta_E$ is transmitted in each frame. In TT-A, we use a constant margin θ_E, which is chosen by solving the following optimization problem **(P-0)** numerically.

$$\textbf{(P-0)} \quad \arg\min_{\theta_E} \quad \text{area}(\theta_0+2\theta_E); \quad (21)$$

$$s.t. \quad \tfrac{1}{N}\textstyle\sum_{i=1}^N I_i^f \le r_f; \quad (22)$$

$$I_i^f = f_f(e_i^x, e_i^y, Z_i, \theta_0+2\theta_E). \quad (23)$$

Since the larger θ_E is, the more the bandwidth it is consumed, and the less the failure occurs, binary search can be used to obtain the optimal θ_E effectively.

2) TT-C (Targeted Transmission when Confident): TT-C transmits the data for the targeted area if the prediction accuracy is good, and the whole frame otherwise. In addition to the viewpoint prediction results, TT-C also utilizes the deviation prediction results to judge whether the viewpoint prediction is accurate or not. This algorithm needs to decide two parameters: the margin θ_E and the threshold d_{th}. When the predicted diagonal deviation has $\hat{d}_i \leq d_{\text{th}}$, partial transmission is applied; otherwise, the whole frame is transmitted. During partial transmission, similar to TT-A, a round with beam angle $\theta_0 + 2\theta_E$ of data is transmitted. Given a failure ratio r_f, the optimal θ_E and d_{th} can be obtained by solving **(P-1)**.

$$\textbf{(P-1)} \quad \arg\min_{\theta_E, d_{\text{th}}} \quad r_p \text{area}(\theta_0 + 2\theta_E) + (1 - r_p); \tag{24}$$

$$s.t. \quad \frac{1}{N}\sum_{i=1}^{N} I_i^f \leq r_f; \tag{25}$$

$$r_p = \frac{1}{N}\sum_{i=1}^{N} I_i^p; \tag{26}$$

$$I_i^p = \begin{cases} 1, & \text{if } \hat{d}_i \leq d_{\text{th}}, \\ 0, & \text{otherwise}; \end{cases} \tag{27}$$

$$I_i^f = f_f(e_i^x, e_i^y, Z_i, \theta_0 + 2\theta_E) \wedge I_i^p. \tag{28}$$

In Eq. (28), $\wedge$ means the operation of logic *and*.

The consumed bandwidth is a non-increasing function of d_{th} and a non-decreasing function of θ_E. The failure ratio is a non-decreasing function of d_{th}, and a non-increasing function of θ_E. Therefore, we use exhaustive search for one parameter while binary search for the other to obtain the global optimum.

3) TT-A (HR) (TT-A with Hybrid Resolution): Similar to TT-A, TT-A (HR) transmits only the targeted area all the time. However, TT-A (HR) explores an additional degree of freedom: resolution. Each transmitted area consists of two resolution portions: high resolution in the center portion and low resolution in the outside area. Assume high resolution area has a margin θ_h and low resolution has a margin θ_l.

We use the **Ratio of Low Resolution (RLR)** as a new performance metric for this hybrid resolution algorithm. RLR denotes the percentage of frames that have partially low resolution pixel or missing pixels, and it can be obtained in the following equation

$$\frac{1}{N}\sum_{i=1}^{N} f_f(e_i^x, e_i^y, Z_i, \theta_0 + 2\theta_h). \tag{29}$$

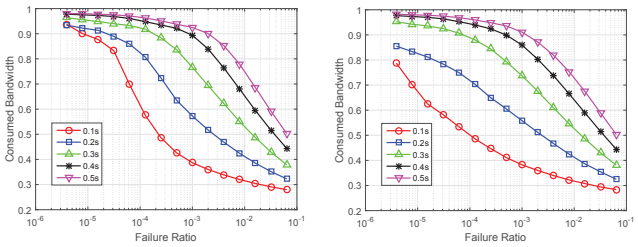
In this case, the failure ratio is

$$\frac{1}{N}\sum_{i=1}^{N} f_f(e_i^x, e_i^y, Z_i, \theta_0 + 2\theta_l). \tag{30}$$

Assume the bitrate requirement for high resolution and low resolution transmission are r_h and r_l, respectively. Then the consumed bandwidth is

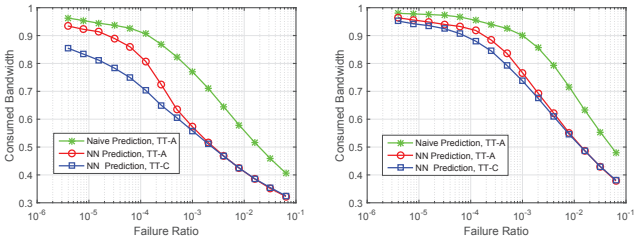
$$\text{area}(\theta_0 + 2\theta_h) r_h + [\text{area}(\theta_0 + 2\theta_l) - \text{area}(\theta_0 + 2\theta_h)]\, r_l. \tag{31}$$

Given both RLR and failure ratio as constraints, the bandwidth minimization problem can be decoupled into two independent **(P-0)**s: with RLR (resp., failure ratio) as the constraint, and area$(\theta_0 + 2\theta_h)$ (resp., area$(\theta_0 + 2\theta_l)$) as the objective function, **(P-0)** provides the optimal θ_h (resp., θ_l).



(a) TT-A (b) TT-C

Fig. 8. The consumed bandwidth v.s. failure ratio



(a) $T_r = 0.2s$ (b) $T_r = 0.3s$

Fig. 9. The comparison of TT-A and TT-C.

VI. Performance Evaluation

In this section, we evaluate the proposed algorithms based on the data that contain 295500 samples. We run 10 iterations to obtain the average values. In each iteration, 50% of the data are selected for training the prediction models; the other 50% are used to see the bandwidth requirements using the above algorithms with optimized parameters. As we have seen in Sec. IV, the neural networks achieve more accurate prediction than linear regressions, and therefore all the algorithms apply the neural networks. With a perfect prediction of the X, Y and Z angles, we need to transmit just a rectangle whose area ratio is 19.66%. Therefore, the bandwidth saving upper bound is 80.34%.

Figs. 8 shows the consumed bandwidth v.s. failure ratio of Algorithms TT-A and TT-C, respectively, for a prediction window ranging from 0.1s to 0.5s. We can see that for a given failure ratio, the required bandwidth increases as the prediction window increases. For instance, given 0.1% as the failure ratio, for both algorithms, additional 15%∼20% bandwidth is consumed when the prediction window grows from 0.1s to 0.2s, or from 0.2s to 0.3s. Meanwhile, if we limit the consumed bandwidth to be 50%, both algorithms have their failure ratio grow by more than 10x, when the prediction window grows from 0.1s to 0.2s, or from 0.2s to 0.3s. This is because as the prediction window T_r gets smaller, we can obtain more accurate predictions that will help us target the viewing area and reduce the required bandwidth. However, a shorter T_r allows less time for transmitting and processing the data. Thus, the prediction window T_r should be carefully adjusted with regards to the practical network and device conditions.

Figs. 9 compares TT-A and TT-C, for a prediction window of 0.2s and 0.3s. These figures include a baseline, which is TT-A with Naive prediction. As discussed in Sec. IV-A,

TABLE III
THE CONSUMED BANDWIDTH OF TT-A (HR) ($T_r = 0.2$S).

RLR \ Failure Ratio	0.1%	0.05%	0.025%	0.0125%
6.4%	0.3783	0.3932	0.4081	0.4217
1.6%	0.4214	0.4363	0.4512	0.4648
0.4%	0.4788	0.4938	0.5087	0.5223
0.1%	0.5525	0.5675	0.5824	0.5960

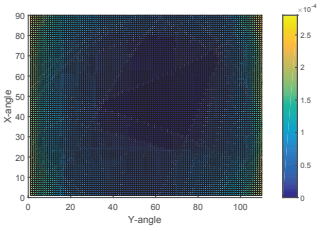


Fig. 10. RMPs in the FOV.

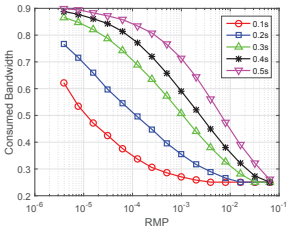


Fig. 11. The bandwidth v.s. RMP

the Naive prediction uses just the current motion as the prediction result for future motion. From these figures, we can see the benefits of viewpoint and deviation prediction. With viewpoint prediction, given consumed bandwidth in the range 0.4∼0.8, roughly 10x failure ratio reduction is achieved by prediction. TT-A, which only uses viewpoint prediction result is suboptimal compared with TT-C that utilizes both the viewpoint and deviation prediction results, especially when the failure ratio is small.

For TT-A (HR), we have Table III to show the consumed bandwidth given both the failure ratio and the ratio of low resolution (RLR). Assume the high resolution is 4K and the low resolution is 1080P. Typically, in this case, a low resolution frame has 1/4 bitrate of the high resolution frame. We normalize the low resolution bitrate by the high resolution bitrate, such that $r_l = 1/4$ and $r_h = 1$. We calculate the consumed bandwidth of the hybrid resolution transmission algorithm in Table. III. If low resolution image is acceptable in certain part, especially in the marginal area, this hybrid resolution algorithm enjoys both low failure ratio and low bandwidth compared with other algorithms. For example, taking 0.2s as the prediction window, from Fig. 8(b), we can see TT-C consumes 55% bandwidth while having a failure ratio of 0.1%. For the hybrid resolution algorithm, on the one hand, given 0.1% as the failure ratio constraint, it further saves the bandwidth by 13% by tolerating 1.6% RLR; on the other, given 55% bandwidth consumption, it achieves less than 0.0125% failure ratio while having 0.4% RLR. Therefore, for a given failure ratio requirement, using the hybrid resolution algorithm can significantly reduce the required bandwidth.

Last, taking TT-C as an example, we calculate the actual missing pixels to show that very few pixels are lost in targeted transmission. We use the ratio of missing pixels (RMP), defined in Sec. II-C as the performance metric. Fig. 10 shows the RMP within the FOV. We can see even though a failure occurs, it is very likely that the viewer only misses the most left and most right marginal areas. Fig. 11 shows the relation between RMP and the bandwidth consumption. Compared with Figs. 8, given the same bandwidth consumption, RMP is typically over 10x smaller than failure ratio. Because of this, the viewer can enjoy very high quality viewing experience, without consuming much bandwidth.

VII. DISCUSSION AND FUTURE WORK

As a first attempt to investigate the feasibility of 360-degree video bandwidth optimization based on motion-prediction, our work has certain limitations and may be improved in the following directions: 1) From the server to the client, we transmit a round area, which reduces the complexity of managing the complicated shapes created by Z dimension rotation. One could consider other more fitting shapes, e.g., rectangles, to further reduce the consumed bandwidth. For simplicity, the current transmission algorithms use constant margins based on decision data. In the future, online learning, e.g. Lyapunov techniques [12] may use adaptive margins to lower failure ratio and bandwidth consumption. 2) Our work demonstrates the feasibility and benefits of targeted transmission for 360-degree video, with the assumption that the video can be partially transmitted in runtime efficiently. The findings under this assumption inspire us to study efficient video-coding and compression schemes to implement targeted transmission in the future. 3) In our experiment, most viewers are using VR or watching 360-degree video for the first time. After they get used to 360-degree video, viewers may change their behaviors. We conjecture that viewer motion will be more related to viewer behavior and video content. If this conjecture is correct, we can expect better performance when a viewer has used HMD for a longer period of time. In the future, we will study user behavior and long term motion prediction with more collected data.

The proposed framework has potential applications beyond the current scenario of 360-degree video transmissions. For example, in 360-degree video games, we need GPU to render the screen. When future viewpoints can be predicted accurately, GPU computation could be done in advance to improve viewing experience. To achieve this goal, one must first understand user behavior in 360-degree video gaming scenarios, which may be different from behavior in 360-degree video viewing.

VIII. RELATED WORK

In terms of transmitting 360-degree videos, the most related work is a protocol named *Dynamic Streaming* proposed by Facebook [13]. In this transmission protocol, the data around the FOV are transmitted with high resolution and the other areas are transmitted with low resolution. However, without viewer motion prediction, there is a high probability that viewers may watch the lower resolution part when they change their motion fast within one group of pictures (GOP). In our work, short term motion prediction reduces the pixel loss ratio to less than 0.1%, which gives user experience a guarantee.

[5] proposes a region adaptive smoothing technique to save bits at the top and bottom areas in an equirectangular frame, which reduces up to 20% bitrate. However, the technique is limited to equirectangular projection of the 360-degree videos.

For viewpoint motion prediction, [14] considers motion prediction with constant angular velocity or acceleration to predict motion in the scale of 20∼100 ms. However, the method is not data-driven, and the time scale it considers is not proper for video transmission. [15] presents a double exponential smoothing method to predict head position and rotation in the next 50 ms. Different from [15], we study viewpoint prediction in the range of 0.1∼0.5s, which is reasonable for video frame transmission from a server to a client. The longer prediction windows make prediction results more error-prone, which is addressed in our work by adding margins.

There is some work on optimizing the process of projecting 360-degree videos from 3D space to 2D space, e.g., equirectangular projection, cube-based mapping [16], and pyramid-based mapping [13]. Different methods have different 3D reconstruction quality and computing complexity. Compared with equirectangular projection, cube-based mapping and pyramid-based mapping can reduce a certain amount of the bandwidth consumption given the same viewing experience. In [17], a non-uniform spherical ray sampling method is proposed to give more priority to the important regions. In [18], the authors study the mapping between virtual and physical reality to achieve free walking in a room with limited space. These projecting methods could be utilized and adjusted in the implementation of our framework.

Apart from 360-degree videos, there are also free-viewpoint videos (FVV), where viewers are able to interact with the scene by navigating to different viewpoints. [19] studies a FVV streaming system based on DASH (Dynamic Adaptive Streaming over HTTP) and the multi-view-plus-depth (MVD) representation to improve viewer's watching experience and increase the visual quality of rendered views. [20] develops optimized prefetching and buffer management policies to ensure seamless playback for interactive branched videos. [21] considers multiview video streaming systems and improves efficiency in coding and content replication strategies. However, 360-degree videos are different from FVV in many aspects, such as video recording, frame projection, display rendering, which makes these techniques less applicable to 360-degree videos.

IX. Conclusion

The increasing popularity of 360-degree videos imposes a significant challenge on networks. To address this challenge, we study motion-prediction-based transmission schemes that reduce bandwidth consumption while providing viewing experience guarantees. First, based on collected viewer motion data, we show that motion prediction is feasible within a 100∼500 ms timeframe. Based on the viewing motion data, we develop regression schemes that predict not only the viewpoint but also the prediction accuracy. Based on such predictions, we develop a set of partial content transmission schemes to guarantee viewer experience. Our trace-driven simulation results show significant bandwidth reduction with viewing experience guarantees. For example, with a prediction time scale of 0.2s, our proposed scheme can reduce bandwidth consumption by 45% while guaranteeing a failure ratio of 0.1%.

Acknowledgments: The work was partially supported by NSF through grants CNS-1547461; CNS-1457060; CCF-1423542.

References

[1] S. Hollister, "Youtube's ready to blow your mind with 360-degree videos," available at http://gizmodo.com/youtubes-ready-to-blow-your-mind-with-360-degree-videos-1690989402.

[2] K. Leswing, "Virtual reality is coming to youtube; here's how to watch it," available at http://www.ibtimes.com/virtual-reality-coming-youtube-heres-how-watch-it-1949250.

[3] V. Oculus, "Oculus rift," *Available at http://www.oculusvr.com/rift*, 2015.

[4] Y. Bao, H. Wu, A. A. Ramli, B. Wang, and X. Liu, "Viewing 360 degree videos: Motion prediction and bandwidth optimization," in the proceedings of ICNP 2016, poster paper.

[5] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson, "360 degrees video coding using region adaptive smoothing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 750–754.

[6] Wikipedia, "Field of view," available at https://en.wikipedia.org/wiki/Field_of_view.

[7] P. R. Desai, P. N. Desai, K. D. Ajmera, and K. Mehta, "A review paper on oculus rift-a virtual reality headset," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 13, no. 4, 2014.

[8] U. G. C. at Kjeller, "Latency in lte systems," available at http://cwi.unik.no/images/Latency_in_LTE_comments.pdf.

[9] H. Holma and A. Toskala, *LTE for UMTS: Evolution to LTE-advanced*. John Wiley & Sons, 2011.

[10] ——, *LTE for UMTS-OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009.

[11] E. W. Weisstein, ""zone." from mathworld–a wolfram web resource," available at http://mathworld.wolfram.com/Zone.html.

[12] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.

[13] D. P. Evgeny Kuzyakov, "Next-generation video encoding techniques for 360 video and VR," available at https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[14] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov, "Head tracking for the oculus rift," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 187–194.

[15] J. J. LaViola, "Double exponential smoothing: an alternative to kalman filter-based predictive tracking," in *Proceedings of the workshop on Virtual environments 2003*. ACM, 2003, pp. 199–206.

[16] D. P. Evgeny Kuzyakov, "Under the hood: Building 360 video," available at https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/.

[17] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh, "Rich360: optimized spherical representation from structured panoramic camera arrays," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 63, 2016.

[18] Q. Sun, L.-Y. Wei, and A. Kaufman, "Mapping virtual and physical reality," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 64, 2016.

[19] A. Hamza and M. Hefeeda, "Adaptive streaming of interactive free viewpoint videos to heterogeneous clients," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 10.

[20] V. Krishnamoorthi, N. Carlsson, D. Eager, A. Mahanti, and N. Shahmehri, "Quality-adaptive prefetching for interactive branched video using http-based adaptive streaming," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 317–326.

[21] H. Huang, B. Zhang, S.-H. G. Chan, G. Cheung, and P. Frossard, "Coding and replication co-design for interactive multiview video streaming," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2791–2795.