Title

Towards Tile Based Distribution Simulation in Immersive Video Streaming

Authors

He, D.
Westphal, C.
Jiang, J.
et al.

Publication Date

2019

Peer reviewed

Towards Tile Based Distribution Simulation in Immersive Video Streaming

Dongbiao He*, Cedric Westphal†, Jinlei Jiang*‡, Guangwen Yang*, J.J. Garcia-Luna-Aceves†
*Beijing National Research Center for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
†Computer Science and Engineering Department, University of California, Santa Cruz, CA 95064, USA
‡Research Institute of Tsinghua University in Shenzhen, Shenzhen 518057, China
hdb13@mails.tsinghua.edu.cn, cedric@soe.ucsc.edu, jjlei@tsinghua.edu.cn, ygw@tsinghua.edu.cn, jj@soe.ucsc.edu

***Abstract*—There has been increasing attention to virtual reality applications in recent years, especially to immersive or 360-degree videos that typically consume much more bandwidth than traditional ones. Though all produced data is transferred, only a small part (denoted as Field of View or viewport) is watched by users due to the nature of immersive videos. Obviously, this causes a large waste of network resources. Hence, it is important to define a viewport-dependent streaming transmission strategy by detecting where the user is gazing and the movement of the user's head. Unfortunately, there are few datasets providing this information. In this paper, we propose a tile-based simulation approach to generate the distribution of the user's behavior and to provide information that can be used to optimize future view-dependent streaming protocols. We first characterize the users' viewport pattern from datasets gathered from real users by decomposing the 360-degree stream into tiles and analyzing the frequency and time-interval distribution for each tile. Then, we devise a hierarchical Markov model that incorporates the beta distribution of each tile time interval to predict tile transition. The results show that the simulation tool characterizes the tile sequences of users accurately, performing close to the empirical results.**

***Index Terms*—AR/VR, 360-degree video, Markov Model, User behavior, Simulation**

I. Introduction

Video streaming is increasingly moving towards immersive experiences. In an immersive stream, the user is placed in the middle of the video and can observe the action from multiple points of view by rotating the device or moving the eyes. Since the immersive stream contains views from multiple directions, it involves far more data than a regular video stream with the same resolution. Some studies [4] [11] show that the increase in the amount of data with immersive video is $6\times$ compared with a regular video.

It is estimated that immersive videos (or 360-degree videos) will grow to a market of 8.2 billion by 2020. Users can now experience 360-degree videos on portable devices from several manufacturers. These head-mounted displays (HMD) are able to freely adjust head orientations (i.e., changing the pitch, yaw, and roll) to watch panoramic views. After that, based on the field of view (FoV) and the orientation of the user, the 360-degree video player displays the visible area on a screen. For the current devices, the FoV is about $90°$ vertically and slightly wider horizontally.

ISBN 978-3-903176-16-4 © 2019 IFIP

The increasing market of immersive video requires the underlying network infrastructure to evolve to support the distribution of 360-degree video streaming seamlessly. Major challenges in 360-degree video streaming are its high bandwidth requirement and low delay tolerance. Recently, most 360-degree videos have been encoded at 4K resolution. According to Netflix [2], it is recommended that the Internet download speed reach at least 25 Mbps for streaming Ultra HD quality video (i.e., 4K quality), whereas the average download speed of the mobile Internet is only 19.27 Mbps recently.

In virtual-reality (VR) streaming, the user is immersed in a virtual environment and can dynamically and freely decide the preferred viewing position, called *viewport* [9], [10], [19], [21]. As Fig. 1 illustrates, the 360-degree video uses motion sensors and gyroscopes to detect the position of the HMD, allowing the device to continually update a scene according to head movement and rotation.

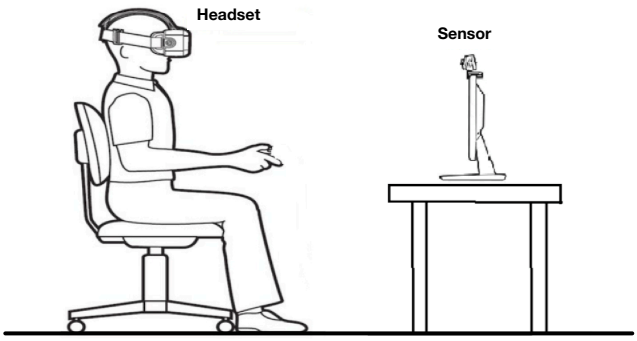


Fig. 1: Positioning of the sensor and headset, adapted from [1]

In order to reduce the bandwidth consumption over the network, viewport-dependent solutions have often been proposed for VR streaming. Namely, the client attempts to fetch only the views that the user will actually observe, and tries not to download directions that the user will not watch. Winnowing strategies have been devised to identify which FoVs will be watched. Similarly, pre-fetching strategies have been designed to carefully balance two contrasting objectives,

namely maximizing quality and avoiding stalls in the played stream and pre-fetching viewports only in the direction that the user will actually watch. In-network caching also would benefit from predicting what views the users will watch.

Obviously, these techniques reduce the bandwidth required to stream the 360-degree video. However, it is difficult to design new algorithms without a prediction model that is easy to simulate. Currently, there are some small-scale datasets [8] [20] [15] [23]; however, to the best of our knowledge, no convenient simulation model exists that describes the motion of the user's field of view as a function of time. We believe this is necessary for proper network evaluation of prefetching and caching algorithms.

Based on the observation of empirical data, we present such a model here. Namely, we show that the user's FoV can be mapped to a tile view (which is commonly used to distribute only the tiles of the 360-degree stream that the user is watching) and that the tiles that the user is watching at any point of time can be modeled by a combination of a Markov model for the transitions from tile to tile, and a beta distribution for the time spent on a specific tile. We then evaluate the properties of our model when compared with the actual datasets to show that our model presents a sample path for the user's perspective that is similar to the empirical data.

The contributions of our paper are the following:

- We present a study of the empirical data for the user's viewport evolution in immersive video streams.
- We derive a model from this study that describes the evolution of the tile at which the user is looking over time. This model is simple enough to be used and is built upon a hierarchical Markov model for the tile transitions, combined with a beta distribution for the time spent on a tile.
- We show how to extract the input parameters into our model to generate some traces that correspond to several basic types of videos.
- We evaluate this model to validate that its output process is similar to that of the empirical data.
- We make our model available for others to use and improve at [3].

One benefit of our model is that it allows for mathematical analysis of the processes that take as input the user's FoV. For instance, an algorithm that predicts the user's future views could mathematically verify its accuracy by using our model as it is easily tractable.

The rest of this paper is organized as follows: We present related work in Section II. Then we observe the properties of empirical data in Section III. We present our model in Section IV and evaluate how close the output is from the empirical data in Section V. Finally, we offer concluding remarks in Section VI.

II. Background and Related Work

With the rapid increase in the number of immersive video streaming applications, the network for delivering the content can potentially become a bottleneck. [22] and [13] survey the relationship between immersive video streaming and networking.

A. User behavior Analysis for Immersive Videos

Many prediction algorithms have been designed to anticipate the content that the user will gaze at. Such prediction helps by transmitting only the views that the user will watch.

Machine learning techniques, including *neural networks*, have been adopted for better feature extraction and prediction accuracy in fixation detection [5], [7], [18]. Mavlankar et al [17] perform fixation prediction in videos using features like motion vectors and navigation trajectories. Chaabouni et al [7] build a *Convolutional Neural Network* architecture and use residual motion as the features for predicting saliency in videos. Alshawi et al [5] observe the correlation between the eye-fixation maps and the spatial/temporal neighbors. This provides another way to quantify viewer attention on videos. Nguyen et al [18] propose to adopt the information of static saliency in images and then take camera motions into consideration for prediction of dynamic saliency in videos. Fan et al [11] tackle the problem of fixation prediction for 360-degree video streaming to HMDs using two neural networks. It uses content-related (an image-saliency map and a motion map) and sensor-related (viewer orientation) features as inputs to predict future viewing probability of each tile.

Using available big data and analytical tools, Liu et al [14] employ an approach guided with Field-of-View (FoV) that fetches only the portions of a scene the users will look at. Xie et al [24] illustrate a cross-user behavior analysis for predicting the users preference on content. They find that the different users are sharing the region-of-interest (ROI) when watching immersive videos. Bao et al [6] predict head movement in 360-degree video delivery. They collect motion data for some subjects watching 360-degree videos and observe a strong short-term auto-correlation in viewer motions, which indicates that viewer motion can be well predicted based on motion history. Other works use prediction mechanism to perform rate adaption for network distribution. For instance, He et al [12] propose a joint rate and FoV adaptation that varies the anticipation window based upon the network response.

As we could conclude from these studies, user behaviors (e.g. head movement, viewer motion) follow a similar pattern which could be predicted according to the historical recordings. Therefore, it is possible and necessary for developing a tool to simulate the user behaviors for immersive video streaming.

B. Available Datasets

There are some public content and datasets for 360-degree video. Some researchers have built 360-degree video testbeds for collecting traces from real viewers watching 360-degree videos using HMDs. The collected datasets can be used to (i) empirically find out some key problems and properties of immersive video, and (ii) validate and evaluate new systems and algorithms.

Wu et al [20] present a new dataset of 60 omnidirectional images with the associated head and eye movement data recorded for 63 viewers. A subjective experiment was conducted in which users are asked to explore the images for 25 seconds as naturally as possible. In addition, an image/observer agnostic analysis of the results from this experiment is also performed, which considers both head and eye tracking data. Furthermore, they also provide guidelines and tools to evaluate and compare saliency maps in such omnidirectional scenarios.

Lo et al [15] present the dataset collected from ten YouTube 360-degree videos and 50 subjects. Their dataset has two parts, consisting of both content data, such as image saliency maps and motion maps, and sensor data, such as positions and orientations.

Wu et al [23] present a head tracking dataset composed of 48 users watching 18 sphere videos from five categories. In order to better assess the users' behavior, they record how users watch the videos, how their heads move in each session, what directions they focus, and what content they can remember after each session.

Corbillon et al [8] present a dataset that includes the head positions of 59 users recorded while they are watching five 70s-long 360-degree videos using the Razer OSVR HDK2 HMD. They have published the dataset alongside the used videos and the open-source software that they develop to collect the dataset. Finally, they also introduced examples of statistics that can be extracted from the dataset to provide an overview of the users behavior and the videos characteristics, focusing on the viewport adaptive streaming scenario.

III. Empirical Dataset Study for Modeling Head Movement

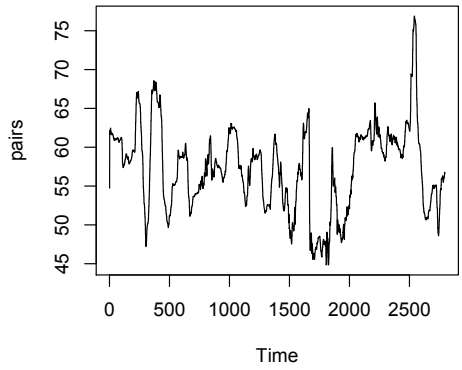
We consider Corbillon's dataset [8] mentioned above in order to investigate the users' behavior in this paper. The dataset has the following properties: (1) It collects the user head movement data in watching 360-degree videos, which directly matches our requirements; (2) the dataset contains 59 users, which is almost sufficient for analysis and modeling; and (3) each video is classified according to a set of categories that could be used to describe the different viewing behaviors.

Among the different types of 360-degree videos, we use the following four different video types to analyze the features of user behaviors: Exploration (Paris), Static Focus (Rhino), Rides (Rollercoaster), and Moving Focus (Timelapse).

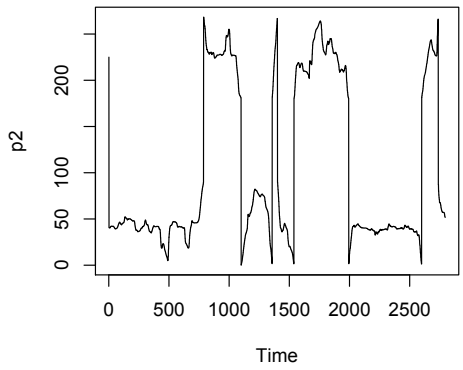
We ask the following question: *can we find a simple common model to describe the properties of these videos?*

A. Raw Data analysis

Fig. 2 displays the distribution of user head movement measured in a spherical coordinate system. θ (the polar angle, showed in Fig. 2(b)) is measured from a fixed zenith direction, and γ (the azimuth angle, showed in Fig. 2(a)) is of its orthogonal projection on a reference plane that passes through the origin and is orthogonal to the zenith. The sample video selected from the dataset is "Paris". From this pair of coordinates (and the corresponding Fig. 2), we can observe that the variations of the angles θ and γ are quite random. It indicates that it is hard to apply the time series model for predicting the change of the user behaviors in a global view.



(a) The distribution of γ



(b) The distribution of θ

Fig. 2: Distribution of the users' head movement

TABLE I: Distance variance with increasing of time

	100ms	250ms	500ms	750ms	1000ms
95%	0.147	0.433	3.012	3.093	3.107
90%	0.096	0.255	0.567	1.11	2.983
85%	0.073	0.19	0.401	0.645	0.956

Corbillon et al [8] measure the position of the user's gaze direction projected on a unit sphere as a function of time. Because the distance is measured on the unit sphere, the most the user's gaze can move within a period of time is π (where it would find itself at the antipode from the initial position). The considered distance is the orthodromic (or Haversine or great-circle) distance.

Using their dataset, we are able to generate Table I. Table I shows the radius r that the user's FoV will stay within with probability p during a period of time ξ. Namely, if x_t is the

center of the FoV at time t, Table I shows the relationship for various probabilities p, time interval ξ and the radius r, that satisfies:

$$\text{argmin}_r P(||x_\xi - x_0|| < r) > p \quad (1)$$

For instance, we can see that there is 90% probability that the user will be viewing an FoV that is within the distance of less than 0.096 after 100ms from the initial FoV. When we enlarge the time interval to $750ms$, the distance of the user's motion is within 1.11 with 90% probability. Even for a $1s$ delay, there is 85% chances that the user stays within the distance of 0.956 to the initial position. Hence, though the user's behavior trend is quite random from the original datasets, the user's head movement would not vary dramatically in most cases when they are watching a video.

B. Tile Distribution Analysis

While spherical coordinates perfectly describe the direction of the user's gaze within the video stream, most systems use a projection of the sphere onto a finite set of tiles for transmission over the network. Equirectangular and cubic projections are two common approaches used to project the video sphere to a 2D plane. In this paper, we choose equirectangular projection while any other projection could also be used. The idea is to transmit only the tiles that are being viewed by the users, so as to reduce the bandwidth. For instance, in the case of a cubic projection (with six tiles, "top", "left", "front", "right", "bottom", "back"), the "back" tile should not be transferred as it is unlikely to be watched.

For the remainder of this paper, we shift the presentation of the motion of the user's gaze from spherical coordinates to tiles. Tiles are proposed as a unit of transmission for immersive video streams in order to reduce the bandwidth usage. Therefore, we are attempting to model the patterns of user behavior at the tile level.[1]

By mapping the spherical coordinates to the tile view, we can leverage the previous data sets.

Tiles have different levels of importance in a 360-degree video depending on the instant viewport of the user. A video view sphere would be split into $\mathcal{K}$ part: $\{\tau_i | i = 1, 2, ..., \mathcal{K}\}$, and each part would have the same size. In practice, the collection of constructed (smaller) video frames for each tile would be regarded as separate videos after the video has been partitioned into spatial tiles. The tiles are then separately encoded and streamed to the user, according to various optimized strategies.

Tile frequency: it is the frequency, in counts, of the tiles being watched in center of users' FoV, measured from all the users of the same video.

We should mention that we only calculate the tile in the center of a total viewport, because it reduces the analysis complexity and computational burden while incurring only marginal information losses in the data.

[1]Note that it would converge back to a spherical view for a large number of smaller and smaller tiles.

Fig. 3 displays the frequency of tiles of all the users when they watched four videos with a setting of 4×6 tiles. Observing the results, we observe that the users spend most of the time watching a small percentage of tiles. Tile 2 accounts for more than 40% of the frequency in the Pairs, Rollercoaster and Timelapse videos. In the video Rhino, tiles 2, 6, 14, and 17 comprise 90% of the user's time. Further, only 46% of the tiles are used in this scenario, leaving half of the tiles unwatched by the users. Statistical tests show that the statistics of the data gotten from various sampling methods contain no significant differences.

Table II provides the tile usage in different partitions of the sphere when users watch different type of videos. For example, the usage of the tiles decreases from 0.35 to 0.32 when the tiles number increases from 48 to 144 in the video of Paris. One exception for the trend is Rollercoaster, the percentage of tiles used remains 0.31 when increasing the number of tiles.

TABLE II: The tile usage in different partitions

Tiles	Paris	Rhino	Rollercoaster	Timelapse
6×8	0.35	0.35	0.31	0.40
9×12	0.34	0.32	0.31	0.36
12×12	0.32	0.28	0.31	0.33

C. Time Interval Distribution

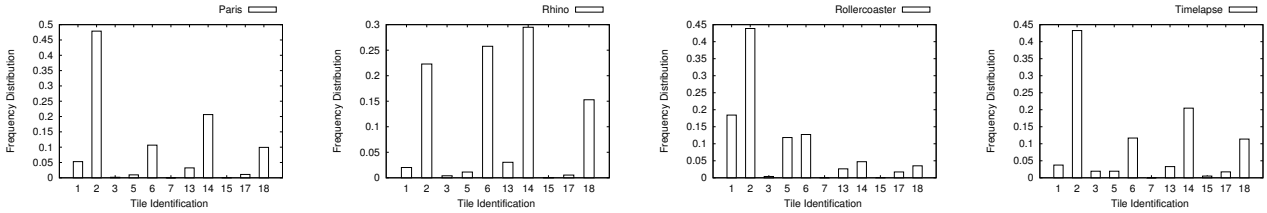
Once the transition from one tile to the next is obtained, we need to know how long the user's gaze will stay within the same tile.

In order to model the time interval distribution of users in a tile, we draw Fig. 4 to illustrate this problem. We select the tile that has the highest frequency (shown in Fig. 4(a)) and the tile with relative lowest frequency (shown in Fig. 4(b)) as the example to show the difference between tiles. The histograms in the figures present the distribution of time interval of the tile with the black line. The red (respectively blue) line is a normal distribution (respectively beta) generated from the mean value and standard deviation values of the dataset. From the figure, we find that the blue line fits the black line, while the red line does not. Hence, in this paper, we choose the beta distribution to model the time spent in each tile.

The general form of the beta distribution has four parameters, including two shape parameters and the upper and lower bounds. The standard beta distribution when bounded between zero and one has only two parameters, and is expressed as:

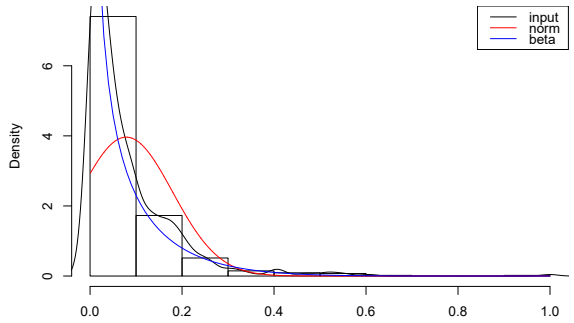
$$f_b(s) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} s^{\alpha-1}(1-s)^{(\beta-1)} & 0 \le s \le 1 \\ 0 & \text{otherwise} \end{cases}$$

where $f_b(s)$ is the Beta distribution function of s, and s is the random variable of relative time interval. Here α and β are the shape parameters of $f_b(s)$, which are calculated using the mean (μ) and standard deviation (σ) from the dataset of each tile.

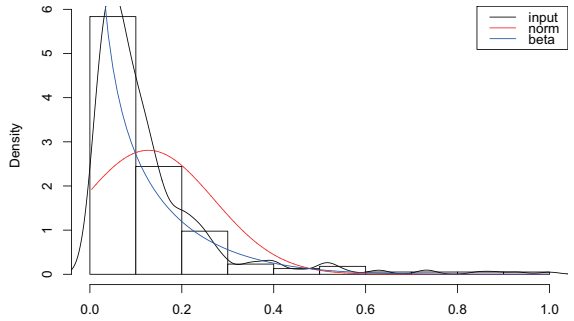


(a) Paris (b) Rhino (c) Rollercoaster (d) Timelapse

Fig. 3: Tile Histogram for Each Video



(a) High frequency: Tile 7



(b) Low frequency: Tile 6

Fig. 4: Distribution of the time spent in a tile

Then, we can induce α and β as follows:

$$\beta = (1-\mu)(\frac{\mu(1+\mu)}{\sigma^2} - 1) \quad \alpha = \frac{\mu \times \beta}{1-\mu} \tag{2}$$

The parameters α and β of the beta distribution could be obtained based on the empirical dataset, through the method of real user experiments or the maximum likelihood estimation method. In other words, commonly-used statistical methods are used to obtain continuous distributions from the observed data.

Suppose $max\{\tau\}$ is the maximum time interval of tile τ, the generated time interval distribution Φ of τ is:

$$\Phi = dBeta(\alpha_\tau, \beta_\tau) \times max\{\tau\} \tag{3}$$

D. Summary

From the above study, we can see that: (1) User behavior changes randomly with the time distribution; (2) users focus only on half of the views when they watch the entire 360-degree video, which might help us reduce the extra state of watching 360-degree videos when modelling the user behavior; (3) the time interval of each tile can be described as a beta distribution from the study of the empirical dataset; and (4) the behavior of different types of video follows the same distribution with some variation in different popularity of tiles. The tiles used and the time interval in each tile essentially determine the distribution of user behavior in watching a 360-degree video.

Therefore, to accurately measure user behaviors of 360-degree videos in terms of tiles transition and time interval in each tile, we need to consider all the features discussed in this section. Next, we would present our modeling approach and the simulation approach based on a Markov model.

IV. Markov-process based simulation models

A. Preliminary

In this section, we explain our proposed Markov model for modeling head movement in 360-degree video streaming. The movement dataset of a moving object over several users is obtained by mapping the angles to different tiles. Hence, the dataset is transformed into a series of tiles, which are composed of chronologically ordered two-dimensional tile-spatial points representing the location of the object at each time stamp, where the time interval between each tile is variable. We are looking for a model composed of a number of spatiotemporal rules relating to the noisy and unevenly sampled movement data to their context-related state. These states can represent the activity governing the movement.

Our aim is to model the complete movement track in a way that each state in the model is either a stay-point or the transition path from one stay-point to another, where spatial coordinates have some form of spatiotemporal similarity. We assume that the sequence of tiles requested by the users is a Markov chain process in which the states are the context ruling users activities and the tiles that a person watches are observable two-dimensional spatial points. A possible solution would be to consider each spatial tile as a state and use a Markov model to find the most probable sequence of states that explain the user behavior.

Definition 1 (Tile-State Interval). Let $\mathcal{K} = \{\tau_1, \tau_2, ..., \tau_k\}$ be the set of tiles. Without loss of generality, we define a set of uniformly-spaced time points based on the natural numbers

N. We say the triplet $(\tau_i, s_i, f_i) \in \mathcal{K} \times N \times N$ is a Tile-State Interval, where $\tau_i \in \mathcal{K}, s_i, f_i \in N$ and $s_i < f_i$. The two time points s_i, f_i are called end-time points or the occurring times of state τ_i, where s_i is the starting time and f_i is the finishing time. The set of all tile intervals over $\mathcal{K}$ is denoted by I .

Definition 2 (Tile Interval Distribution). A Tile Interval Distribution is a series of tile state interval triplets $\{(\tau_1, s_1, f_1), (\tau_2, s_2, f_2), ... ,(\tau_n, s_n, f_n)\}$, where $s_i < s_{i+1}$ and $s_i < f_i$. The Interval Distribution has the following properties:

- $\tau_i \neq \tau_{i+1}, \forall i \in \mathcal{K}$
- $s_{i+1} = f_i, \forall i \in \mathcal{K}$

B. Tile Transition Model

We assume the varying process of the tile transition when the users watch a 360-degree video to be a Markov process. Since it is difficult to deal with a continuous Markov process in simulation in angles, we map the state to tiles for building a discrete Markov model. We assume that we have $\mathcal{K}$ (equal to the tile number) possible aggregated states, each of which represents a certain tile of the video.

Given that the time interval is limited to a given tile, the transition probability P_{τ_i,τ_j} from state τ_i to state τ_j at a certain time would approach a steady value $t_{\tau_i,\tau_j} \in (0, 1), \tau_i, \tau_j \in \{\tau_1, ..., \tau_{\mathcal{K}}\}$. It can be calculated as follows:

$$P_{\tau_i,\tau_j} = \frac{\mathcal{H}_{\tau_i,\tau_j}}{\sum_{k\in\mathcal{K}} \mathcal{H}_{\tau_i,\tau_k}} \tag{4}$$

where $\mathcal{H}_{\tau_i,\tau_j}$ refers to the frequency of the transition from the dataset.

We can thus write the one-step transition matrix $\mathbf{T}$ of this Markov process as:

$$\mathbf{T} = \begin{bmatrix} P_{11} & P_{12} & ... & P_{1\mathcal{K}} \\ P_{21} & P_{22} & ... & P_{2\mathcal{K}} \\ ... & ... & ... & ... \\ P_{\mathcal{K}1} & P_{\mathcal{K}2} & ... & P_{\mathcal{K}\mathcal{K}} \end{bmatrix} \tag{5}$$

It should be noted that the transition probability matrix above is stable, given that it is a stochastic matrix that has a maximum eigenvalue equal to one and the sum of each column is 1.

Based on the property of a Markov process with a given transmission matrix, we can directly get the aggregated steady-state tile distribution τ from

$$t^{k+1} = \mathbf{T} \bullet t^k \tag{6}$$

where t^{k+1} is a $\mathcal{K} \times 1$ vector, whose element t_{τ_j} represents the empirical probability that the spacing falls in state τ_j.

We use this model to explain and reproduce the tile distribution with a given input. However, the transition matrix $\mathbf{T}$ is not directly estimated from the tile data since there is some time interval between each tile. Indeed, a combination with the beta distribution to produce the time interval of each tile would be the final output of the tile distribution according to the user's behavior. The complete procedure is displayed in Algorithm 1.

Algorithm 1: Computation of the Tile Distribution

INPUT:
Beta distribution: α_t, β_t;
Transition Matrix $\mathbf{T}$;
Initial State τ_0, s_0, f_0;
OUTPUT: The generated tile Distribution: τ, s, f;
`/* Set the sequence length L */`
$L = l - f_0$;
$k = 0$;
while $L > 0$ **do**
  $s^{k+1} = f^k$;
  $t^{k+1} = \mathbf{T} \bullet t^k$;
  pick the tile $\tau^{k+1} = max\{t^{k+1}\}$;
  `/* Based on equation 3. */`
  $f^{k+1} = dBeta(\alpha_{\tau^{k+1}}, \beta_{\tau^{k+1}}) \times max\{\tau\}$;
  if $f^{k+1} > l$ **then**
    $f^{k+1} = l$;
  end
  $l = l - f^{k+1}$;
  $k = k + 1$;
end
Return τ, s, f;

From the discussion in the prior section, we can see that different kinds of videos maintain different popularity of each tile. Some further improvement of the Markov model would be made via the following characteristics:

- The user would not move his/her head dramatically according to the result shown in Table I, and only the adjacent tiles would be chosen in a short time interval.
- More importantly, if we cut the sphere of 360-degree video into hundreds of tiles, the transition matrix is very sparse and it would not be useful to generate the tile distribution.

Thus, in the next section, we will present a hierarchical Markov model to solve the above problems.

C. The Hierarchical Markov Model

Decomposing a target Markov Model into a hierarchy of smaller Markov processes is straightforward in solving a large number of tiling. The main idea is that we group the tiles near into a sub-region, and a set of sub-regions could also group together into larger regions. This process would keep up until the number of states within the threshold (e.g., 24). Therefore, The method of hierarchical Markov model decomposes the original process M into a set of sub-processes arranged over a hierarchical structure. Each sub-process is treated as a region of tiles processing for a high-level process. Specifically, let the decomposed processes be $\{M_1, M_2, ..., M_n\}$, then the M_n would be one of a leaf sub-process such that solving all the sub-processes in the leaf layer gives the final solution of the original Markov process M. Each sub-process M_i is defined as a tuple (T_i, L_i), where

- T_i is a set of tiles or regions within the sub-process M_i. According to the definition, T_i also contains one or multiple Markov processes.
- L_i is the termination indicator for transfer the state to another when combining the states into a total tile distribution.

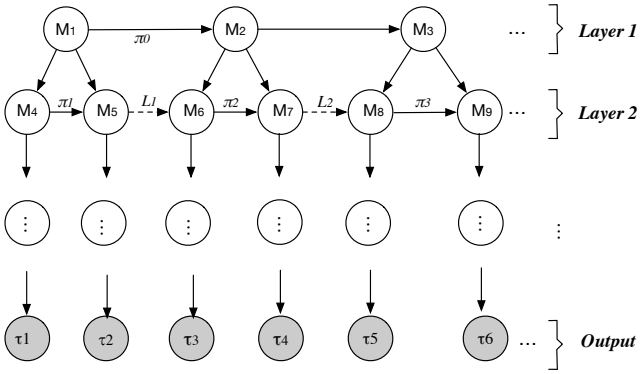


Fig. 5: An example of hierarchical Markov model

This hierarchical structure can be represented as a process graph, shown in Fig.5. In the figure, the first layer has three sub-processes: M_1, M_2, and $M_3 (i.e., Layer_1 = \{M_1, M_2, M_3\})$. Sub-processes M_1, M_2, and M_3 are sharing lower-level primitive actions M_i as their sub-processes. In other words, a sub-process in the task graph is also a (macro)action of its parent. Each sub-process must be fulfilled by a policy, unless it is a primitive action.

Given the hierarchical structure, a set of transmission policies π is defined for each sub-process as $\pi = \{\pi_0, \pi_1, ..., \pi_n\}$, where π_i for sub-process M_i is a mapping from its active states to actions $\pi_i : T_i \rightarrow T_j$. The function of termination indicator $L^{\pi}(i, S)$ is defined as the expected time interval attributed of following a hierarchical policy $\pi = \{\pi_0, \pi_1, ..., \pi_n\}$ starting from upper layer tasks until M_i terminates at one of its terminal states $\tau \in L_i$. From the hierarchical structure of this model, the leaf transmission policies only contain the corresponding τ.

V. Simulation Results and Discussion

In this section, we present simulation results showing that: (1) The proposed model can generate the 360-degree video tile distributions with similar properties as the datasets observed; and (2) the model can be applied to different types of 360-degree videos.

The input of this simulation model consists of the setting of tile partitions for generating the Markov transition matrix from the real world dataset [8]. Each tile would have its own beta distribution parameter pair(α, β). The initial state of our tool is obtained randomly and we run the model 100 times for each video. We have made the code for the simulation model available at [3].

A. Simulation Benchmark

As we know, different users might have different head movement traces when they are watching the same video. The Kolmogorov-Smirnov hypothesis testing (KS test) [16] results for the datasets with the same user illustrate that they are not following the same distribution. However, KS tests is a universal evaluation tool, which is not effective in our scenario. Hence, in this paper, we take advantage of all the data in the datasets to evaluate the effectiveness of the simulation proposal we designed.

We define the *Relative Distance* RD metric on tile distribution by:

$$RD = \sum_{i \in E} (p_i - (\hat{p}_i))^2 \tag{7}$$

where p_i is the popularity for a single user or generated tile distribution of tile i and $\hat{p}_i$ is the average popularity of the total users from the given datasets. The value of this parameter indicates the distance between the tested distribution and the average tile distribution. Ideally, we hope that the simulation model results in a performance for a single user behavior with good accuracy when running the model several times.

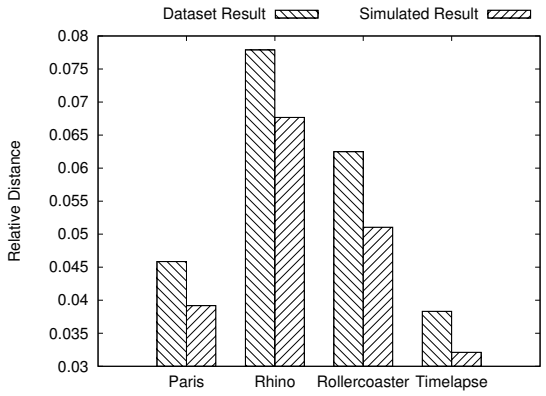
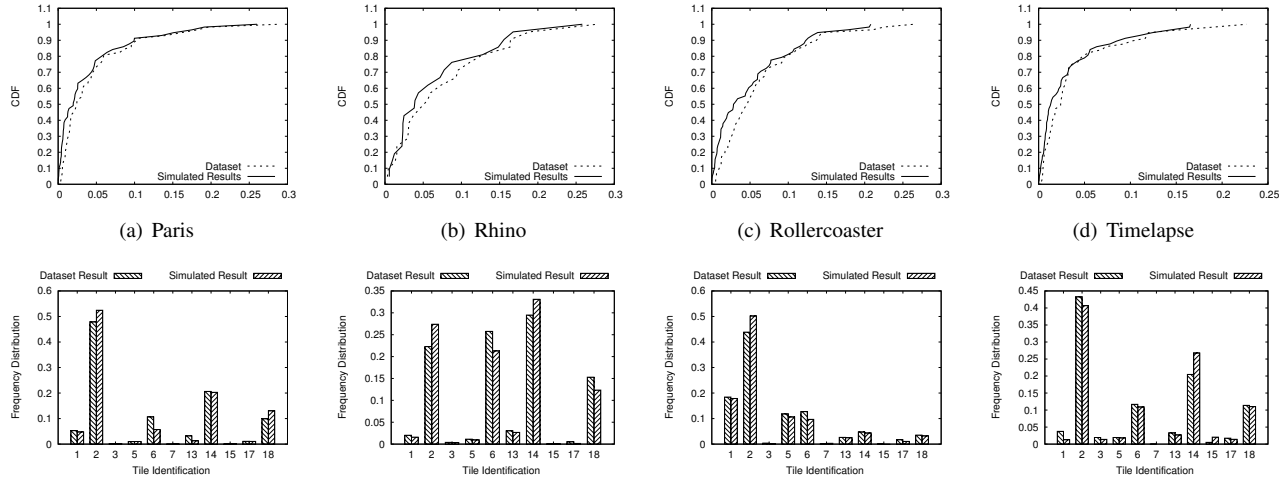


Fig. 6: Average relative distance

B. Distribution Test for Small Number of Tiles

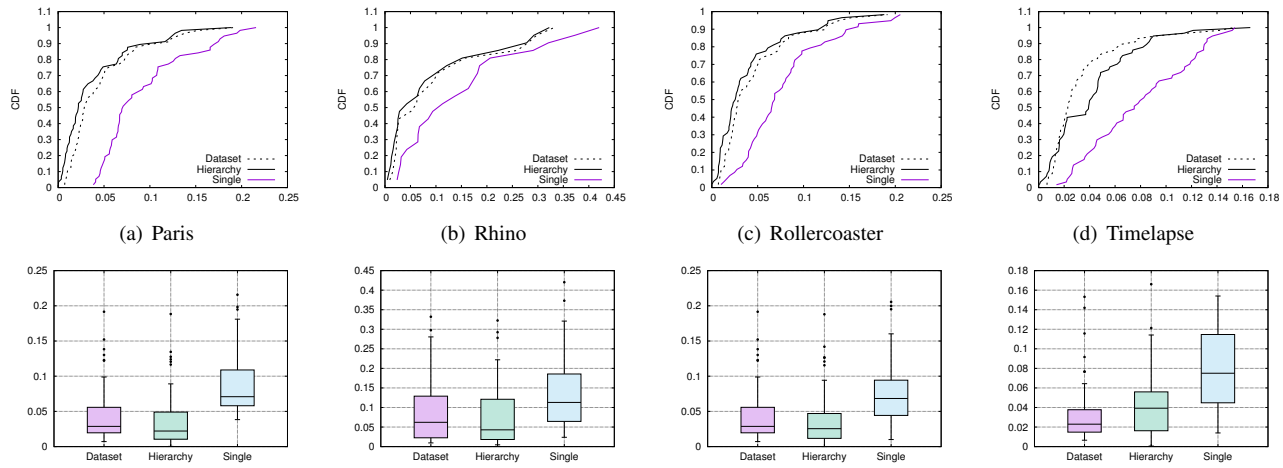
Fig. 6 shows the average relative distance for the four kinds of videos we consider. We observe from this figure that Rhino has a larger gap between the empirical datasets, which reaches 0.078 of datasets and 0.068 of the generated data. The video of Timelapse has the lowest average distance to the total data, which are both below 0.04. The gap of the dataset and generated distribution are not large, which is around 0.006 to 0.01. From Fig. 6, we observe that the generated data is closer to the combination of all the user datasets.

Fig. 7 further displays the results of the comparison between the user datasets and the distribution generated by giving the CDF trends of each scenario and the comparison of a tile frequency distribution. From all these figures, we find that the generated distribution has a smaller value in relative error, which indicates that it is closer to the datasets of the combination of all the users. We build the Markov model of the transition matrix with the total user datasets because of this result, and it is normal for the transition state to follow more accurately the datasets with the data of all the users. However, we generate the time interval with beta distribution



(a) Paris (b) Rhino (c) Rollercoaster (d) Timelapse

Fig. 7: Comparison of tiles in 6×4 tessellation



(a) Paris (b) Rhino (c) Rollercoaster (d) Timelapse

Fig. 8: Comparison of tiles in 12×12 tessellation

independent with each run, which makes the distribution much more like every single user. Hence, the trend of the CDF is also very close to the user dataset. The CDF line in this set of figures also illustrates that our model is very robust in generating the distribution with a high matching frequency with the dataset. Apart from the CDF figures, the rest four figures represent the difference between the tile frequency distribution of empirical datasets and the generated results. As the figures display, the generated tile frequency of the videos almost matches the distribution of the datasets. In most cases, the high frequency tiles in the generated sequences often have a relatively higher ratio compared with the original data. The reason for this is that the popular tiles could be more easy to be reached when the model is built with a moderate size dataset. We can estimate that the gap would be further reduced when the size of the dataset goes to infinity.

C. Distribution Test of Large Number Tiles

In this section, we evaluate the hierarchical Markov model for the cases in which there are more than 100 tiles of the video, for which we cut the video into 12×12 tiles. We divide the sphere of the video view into 6 groups, which represent 6 directions in the space. Each group contains 24 tiles. Thus, it yields a two layer hierarchical Markov model for the 144 tiles. We compare the performances with the single layer of the model and the real user datasets. The results are shown in Fig. 8.

We observe that the trends of real user datasets of each type videos are very similar to the cases of 4×6 tiles. This illustrates that the frequency of each tile is not dependent on the number of tiles in a video, which further supports our result shown in Section III. From the figures, we see that the results for the case using the hierarchical Markov model are much closer to the user generated data. Most of the videos have very similar performance, with the CDF lines of the hierarchical model above the real datasets, except the video

of Timelapse. In addition, the curve of the single model is always below the line of the datasets from a real user. This result can be explained by the fact that the single model has a very sparse transition matrix and a very limited state transformation when running only for a short time. Namely, the model is not able to transfer the tiles with a relatively long distance on the sphere of the 360-degree video. Hence, by using the hierarchical Markov model, the upper layer of the simulation tool could manage the long distance transmission of the tiles, which fits better the user's behavior. The four box figures further demonstrate the different quality of the single and hierarchical model. The box areas of the dataset and hierarchical model are very close in the first three kinds of the videos, and this observation is also very similar to the CDF figures.

VI. Conclusion

In order to facilitate the evaluation of 360-degree video distribution over a network, we have presented an approach to support modeling and simulation of the tile distribution and the tile evolution in immersive video streaming. Our model is built upon a hierarchical Markov model. We first investigate the characteristics of the user's behavior when watching different types of videos. We then formulate the time interval spent on a specific tile as a beta distribution. By mapping each tile to its state, we build a Markov model to reproduce the tile transition distribution. A hierarchical model is proposed to further improve the effectiveness of the simulation. The simulation is computationally simple and adaptable by generating the different state transition matrix and beta distribution parameters. Our model generates sequences of tiles that closely match the user-generated data. Furthermore, it is simple enough to provide a mathematical analysis of existing prefetching algorithms. We hope that this model can be applied to the analysis and evaluation of in-network caching and prefetching mechanisms.

Acknowledgment

This work was sponsored in part by the Natural Science Foundation of China (61572280, 61672312, U1701263) and by the Jack Baskin Chair of Computer Engineering at UC Santa Cruz.

References

[1] Dk2 development kit. https://static.oculus.com/sdk-downloads/documents/Oculus_Rift_DK2_Instruction_Manual.pdf, 2018. Accessed: 2018-05-22.

[2] Netflix help center. https://help.netflix.com/en/node/306, 2018. Accessed: 2018-05-28.

[3] 360 tile motion simulation tool. https://github.com/herbertt12/MarTile.git, 2019.

[4] S. Afzal, J. Chen, and K. Ramakrishnan. Characterization of 360-degree videos. In *Proceedings of the Workshop Viewing Dataset in Head-Mounted on VR/AR Network*, pages 1–6. ACM, 2017.

[5] T. Alshawi, Z. Long, and G. AlRegib. Understanding spatial correlation in eye-fixation maps for visual attention in videos. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[6] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 1161–1170. IEEE, 2016.

[7] S. Chaabouni, J. Benois-Pineau, and C. B. Amar. Transfer learning with deep networks for saliency prediction in natural video. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 1604–1608. IEEE, 2016.

[8] X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 199–204. ACM, 2017.

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

[10] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–7. IEEE, 2017.

[11] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 67–72. ACM, 2017.

[12] D. He, C. Westphal, and J. Garcia-Luna-Aceves. Joint rate and fov adaptation in immersive video streaming. In *ACM Sigcomm workshop on AR/VR Networks*, Aug. 2018.

[13] D. He, C. Westphal, and J. Garcia-Luna-Aceves. Network support for ar/vr and immersive video application: A survey. In *ICETE SIGMAP*, July 2018.

[14] X. Liu, Q. Xiao, V. Gopalakrishnan, B. Han, F. Qian, and M. Varvello. 360 innovations for panoramic video streaming. 2017.

[15] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. 360 video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 211–216. ACM, 2017.

[16] F. J. Massey. The kolmogorov-smirnov test for goodness of fit. *Publications of the American Statistical Association*, 46(253):68–78, 1951.

[17] A. Mavlankar and B. Girod. Video streaming with interactive pan/tilt/zoom. *High-Quality Visual Experience, Signals and Communication Technology*, pages 431–455, 2010.

[18] T. V. Nguyen, M. Xu, G. Gao, M. Kankanhalli, Q. Tian, and S. Yan. Static saliency vs. dynamic saliency: a comparative study. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 987–996. ACM, 2013.

[19] C. Ozcinar, A. De Abreu, and A. Smolic. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. *arXiv preprint arXiv:1711.02386*, 2017.

[20] Y. Rai, J. Gutiérrez, and P. Le Callet. A dataset of head and eye movements for 360 degree images. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 205–210. ACM, 2017.

[21] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj. Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications. In *ISM*, pages 583–586. IEEE Computer Society, 2016.

[22] C. Westphal. Challenges in networking to support augmented reality and virtual reality. In *IEEE ICNC*, Jan. 2017.

[23] C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in VR spherical video streaming. In *MMSys*, pages 193–198. ACM, 2017.

[24] L. Xie, X. Zhang, and Z. Guo. Cls: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 564–572. ACM, 2018.

[1] Simulation tool available at [3]