Viewport Prediction for 360° Videos: A Clustering Approach

Afshin Taghavi Nasrabadi, Aliehsan Samiei and Ravi Prakash
The University of Texas at Dallas
Richardson, Texas
{afshin,aliehsan.samiei,ravip}@utdallas.edu

ABSTRACT

An important component for viewport-adaptive streaming of 360° videos is viewport prediction. Increasing viewport prediction horizon enables the client to prefetch more chunks into the playback buffer. Having longer buffer results in less rebuffering under fluctuating network conditions. We analyzed the recorded viewport traces of viewers who watched various 360° videos. We propose a clustering-based viewport prediction method that incorporates viewport pattern information from previous video streaming sessions. For several videos, specifically those with well-defined region of interest, the proposed approach increases the viewport prediction horizon and/or prediction accuracy.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

360° video, viewport prediction, clustering, adaptive streaming

ACM Reference Format:
Afshin Taghavi Nasrabadi, Aliehsan Samiei and Ravi Prakash. 2020. Viewport Prediction for 360° Videos: A Clustering Approach. In *30th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'20), June 10–11, 2020, Istanbul, Turkey.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386290.3396934

1 INTRODUCTION

In a 360° video, the scene is captured from a specific point, covering a 360° view around it. During playback, the captured view is mapped to a sphere with the user placed at its center. 360° videos have high-resolutions (over 4K), require more bandwidth and lower latency. A 360° video is divided into multiple chunks, each of several seconds duration. A chunk is composed of multiple video frames by temporally dividing the video into segments of fixed duration. Each frame can be spatially divided into several tiles.

Unlike traditional video streaming, in a 360° video a user watches just a subset of the whole scene at a time, known as the user's viewport. In fact, sending high quality video for the parts out of user's viewport is waste of bandwidth. An emerging consensus for reducing the bandwidth consumption in streaming such videos is to predict the user's viewport [10][1] and have the client only prefetch that [2][8]. For other parts of the video, they may be streamed at lower quality or not be streamed at all.

The client prefetches a duration of the video to absorb variations in download time due to network throughput fluctuations. Existing methods are able to predict and prefetch the future viewport for up to two seconds, and the prediction accuracy decreases drastically for longer prediction windows. However, the video client needs a longer buffer to continue video playback smoothly. Therefore, a viewport prediction method that can predict viewport accurately for longer intervals will decrease the probability of rebufferings, leading to higher Quality of Experience (QoE) for users.

In some of existing methods, past viewport samples of the current user are being used for viewport prediction [12]. In some other methods, the viewport history of previous viewers has been used as the input to machine learning algorithms for prediction [3]. In this work, we present the video scene with a sphere and model viewport as a spherical cap on it. Then, we propose a clustering-based viewport prediction method which uses the history of viewport changes of the current user in addition to viewport patterns of previous viewers. In this method, at regular intervals we (i) cluster previous users based on their viewport patterns, and (ii) use viewport samples of the current user to match the user to one of the clusters. Using the pattern of viewport changes of the past viewers in the achieved matching cluster, we might be able to predict the future viewport of the current user and increase the viewport prediction horizon to more than two seconds without extreme reduction in prediction accuracy. We have shown through experiments that using aforementioned method, we are able to predict the viewport of the client for up to 10 seconds without drastic depletion in accuracy.

The remainder of the paper is organized as follows. In Section II, we discuss related work. The proposed viewport prediction algorithm is presented in Section III. The performance of the proposed method is evaluated in Section IV. Finally, we conclude this paper in Section V.

2 RELATED WORK

Viewport prediction can be based on: 1) current viewer samples, 2) previous viewers' history, and 3) content-specific features, such as objects in the scene and their motion. Time series methods, such as linear regression and machine learning, are used for viewport prediction.

For viewport prediction based on past viewport samples of a user, different extrapolation methods are applied. The most common technique in the literature is linear regression. In [12], linear regression and its weighted variant are applied on viewport samples during past 1 second. It yields an average prediction accuracy of 96.6% within 10 degrees deviation from the actual viewport, when it predicts viewport for 1 second in future. Increasing the prediction

NOSSDAV'20, June 10–11, 2020, Istanbul, Turkey
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7945-8/20/06…$15.00
https://doi.org/10.1145/3386290.3396934

interval to 2 seconds causes the accuracy to drop down to 71.2%. However, these methods consider viewport in Euler or Cartesian coordinate system, and extrapolation is applied on each component of the viewport sample independently. For example yaw angle is modeled according to:

$$yaw(t) = at + b$$

where a and b are determined using linear regression over past samples of yaw. However, viewport changes are rotations on spherical domain. Applying prediction method independently on each component does not result in accurate extrapolation.

Long-Short Term Memory (LSTM) is used in [10][3][6][4]. This machine learning technique is a class of recurrent neural networks that is suitable for time series prediction. The main feature in LSTM is that the network can have a long memory of previous samples and exploit patterns. In [4], viewport is predicted based on past viewport samples. The advantage of this work is that the neural network is trained using a large dataset containing 36,000 viewport traces. The equirectangular projection of the video is divided into tiles, and the probability of each tile being in the predicted viewport is calculated. A similar approach is used in [10]. However, it uses saliency maps as input to LSTM. Salient areas of video can attract viewers' attention, and including this information in prediction can improve prediction accuracy. In addition to viewport samples and saliency maps, optical flows of video frames were used as input to the neural network to improve prediction [3]. All these methods predict viewport for short prediction horizons. In [6], more sophisticated LSTM architecture that incorporates history of other viewers for the purpose of long-term viewport prediction is employed. Results show that for prediction horizon of 5 seconds and longer the accuracy is higher than base-line methods, such as linear regression.

Users watch a limited subset of tiles most of the time. However, they do not watch the same group of tiles [15]. Depending on the video content, viewers can be divided into several clusters based on their region of interest, with each region corresponding to a subset of tiles. The authors use Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to cluster previous viewers in each chunk, and probability of viewing tiles in each cluster is calculated. Then, current users are classified into pre-determined clusters, and tile viewing probability of those clusters is used for viewport prediction. Accuracy of prediction is improved for viewport prediction horizon of 3 seconds and longer, compared to not using history of viewers. However, viewers are clustered on equirectangular frame and use Euclidean distance as closeness metric, while the viewport centers are in spherical domain. So, the clustering is not accurate. In [1], first the viewport is predicted using linear regression, then K nearest viewports of previous viewers are found, and their future viewports are used to correct the output of linear regression. A trajectory-based clustering is proposed in [11]. Based on the idea of vehicle trajectory prediction, trajectory of viewport changes is determined, and viewports with similar trajectories are clustered. Viewport of each user is predicted by finding the closest cluster to its trajectory, and extrapolation of the cluster is used for prediction. However, they find trajectories in three different domains of yaw, roll and pitch. Although their results show that viewport prediction accuracy increases for longer horizons, their method predicts viewport in three independent domains of yaw, roll and pitch.

When a 360° video is processed and encoded, it is projected onto a two dimensional plane. Representing and analysis of the coordinates of the viewport using Eulerian (Cartesian) coordinate system can cause inaccuracy in viewport prediction as (i) prediction on three independent domains reduces the efficiency of the viewport prediction mechanism, and (ii) for higher absolute pitch (y) values, the actual amount of movement is reduced by a factor of $cos(pitch)$ ($cos(y)$). For example, in Eulerian system and at one extreme, yaw rotation on pitch angle of 90° results in no movement. Therefore, all components of the viewport should be considered together for movement extrapolation.

In [13], the authors show that it is important to do clustering in spherical space. The clustering algorithm divides a video into 3-second-long chunks, and viewports which are less than $\pi/10$ apart for 60% of the duration of a chunk are considered in one cluster. In a cluster all viewports should be close to each other, so the problem is equivalent to find the maximal clique among neighboring viewports. This method provides clusters with higher viewport overlap.

3 VIEWPORT PREDICTION

In this section, we propose a viewport prediction method that utilizes the history of other viewers. Before presenting the prediction method, we discuss the viewport model used and provide a quaternion extrapolation mechanism.

3.1 Viewport Model

Viewport can be modeled as a spherical cap on 360° video, where the scene is represented with a sphere. Although viewport is not completely a circle, this model approximately represents viewport and makes viewport overlap calculation easier. Width of the viewport is determined by the field of view (FoV). In most of the Head Mounted Displays (HMDs) FoV is about 100°. The center of viewport is represented by unit quaternion. So, viewport of a viewer v can be determined by the unit quaternion of its center, denoted by q_v, and the width. q_v has four components $[w, x, y, z]$, and it can be converted to a point in Euler or Cartesian coordinate system.

Using spherical cap modeling, we can find the overlap between two viewports by using the formula for the surface area of the intersection of two spherical caps proposed in [5]. We only need to know FoV of two viewports and the great circle distance between their centers. We show this distance by θ, and Eq. 1 shows how to calculate it for two quaternions. Figure 1 shows the viewport overlap, highlighted with purple color, between two viewports q_a and q_b. By dividing the area of intersection by the area of target viewport we can calculate the fraction of viewport overlap.

$$\theta(q_a, q_b) = 2\ arccos(w) \tag{1}$$

$$[w, x, y, z] = q_a^{-1} * q_b \tag{2}$$

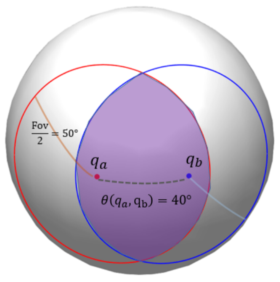


Figure 1: Viewport overlap between viewports q_a and q_b with Fov=$100°$ and distance of $40°$

3.2 Viewport prediction based on quaternion extrapolation

As mentioned earlier, extrapolation using linear regression mechanism may lead to inaccuracy. To address this issue, we propose a method based on quaternion rotation extrapolation. First, consider rotation extrapolation for two sample points represented with unit quaternions q_a and q_b recorded in time t_a and t_b. Assume that we want to extrapolate rotation at time $t_e > t_b$. First, we find the quaternion rotation from q_a to q_b, denoted by q_r:

$$q_r = q_b * q_a^{-1}$$

Then we convert q_r to vector-angle representation of rotation: v_r and ϕ_r. The rotation is around vector v_r and the amount of rotation is ϕ_r. The extrapolation will happen around the same vector, but we need to extrapolate the angle:

$$\phi_e = mod(\frac{t_e - t_a}{t_b - t_a} \times \phi_r, 360°)$$

ϕ_e degree rotation around v_r from q_a results in extrapolated point q_e. When we have more samples, we propose prediction using linear combination of rotation extrapolations. Assume that we have n quaternions samples, and we want to predict quaternion at time t_e. For any two consecutive quaternions q_i and q_{i+1}, we extrapolate the rotation for time t_e. This results in $n-1$ quaternions at time t_e. We find the quaternion that minimizes distance to all extrapolated quaternions. This quaternion can be calculated using quaternion averaging method proposed in [7]. Let q_{avg} show the average quaternion, then it satisfies Eq. 3. We use q_{avg} as the prediction of viewport center at time t_e. We refer to this method as Quaternion Extrapolation (QE).

$$q_{avg} = \arg\min_q \left(\frac{1}{n-1} \sum_{i=1}^{n-1} \theta(q, q_i)^2\right) \tag{3}$$

3.3 Viewport prediction based on history of all viewers

In this section, we investigate the usability of history of viewport traces of other viewers for viewport prediction. As shown in [9], depending on video content, some videos have majority of viewers in few clusters as those viewers watch certain areas of the video, known as Regions of Interest (RoI). So, the question that arises is whether a viewer who is watching the RoI viewed by a majority of previous viewers, *will follow the same viewport path as previous viewers?*

First, we study a dataset[1] to examine if a group of viewers watch the same RoI and follow it over a period of time. We find the clusters of viewers according to the algorithm proposed in [13]. We set clustering window to one second, and if centers of a group of viewports are less than 30° apart in 90% of that interval they are clustered together. We use 30° threshold for clustering because it guarantees that all viewers in a cluster have viewport overlap of at least 60% with each other. Let the algorithm find a set of clusters every Δt seconds. Then, we measure how dispersed the viewers in the same cluster are in the near future. If they stay close together, it shows that the viewers in the cluster follow the same RoI over time. We measure the closeness of viewports in a cluster using the quaternion averaging method described in Section 3.2. Assume that we find a set of m clusters: $\{V_{t_0,i}, 1 \leq i \leq m\}$ in time window $[t_0, t_0 + \Delta t]$, and $V_{t_0,i}$ contains the viewers in cluster i at time t_0. We show the quaternion of viewport center of a viewer $v \in V_{t_0,i}$ at time t with $q_{v,t}$. Let $\Phi_{t_0,i,t}$ denote the quaternion average of viewport centers of viewers $V_{t_0,i}$ at time $t > t_0 + \Delta t$. Then, we measure closeness of viewers in $V_{t_0,i}$ at time t, denoted by $\Theta(V_{t_0,i}, t)$, as the summation of great circle distance, θ, between each viewport and $\Phi_{t_0,i,t}$ according to Eq. 4. We cluster viewports with $\Delta t = 1s$ and calculate $\Theta(V_{t_0,i}, t)$ for different window sizes, more specifically at time $t = t_0 + \Delta t + \delta$ where $\delta = \{1, 3, 5, 10\}$ seconds. We are interested in seeing if $\Phi_{t_0,i,t}$ is a good predictor for viewports in cluster $V_{t_0,i}$. δ acts like prediction window.

$$\Theta(V_{t_0,i}, t) = \sum_{v \in V_{t_0,i}} \theta(q_{v,t}, \Phi_{t_0,i,t}) \tag{4}$$

Figure 2 shows the closeness of viewports in clusters of size three or more for each video in the dataset. For $\delta = 1$, the average Θ is not greater than 30°, which shows that after one second the viewers are still close to each other. However, for longer δ, viewports are more diffused. The amount of closeness depends on the video content. For some videos, such as video 9, the viewports stay close even for time windows of 10 seconds.

Assume that $\Phi_{t_0,i,t}$ is the center of predicted viewport for all viewers in $V_{t_0,i}$. User's FoV is fixed, but FoV of predicted viewport can be widened to increase probability of viewport coverage. Regardless of the closeness, if we make FoV of predicted viewport large enough, we can increase viewport overlap. Figure 3(a) shows the amount of viewport coverage for different closeness values, $\{10°, 20°, 40°, 60,° 80°\}$, and different predicted viewport FoV size, $\{100°, 120°, 140°, 160°\}$. Figure 3(b) demonstrates viewport overlap between actual viewport q_b and predicted viewport q_a with different FoV size for the case that distance between the two viewports is 40°. It can be seen that when closeness is 20°, the actual viewport can be covered by a predicted viewport with FoV of 140°, and for closeness of 40° the FoV should be 160°. In Figure 2, it can be seen that for some videos the average closeness is less than 40°,

[1]The dataset description is provided in Section 4.1.

even after 10 seconds. Therefore, for those cases, we can design a prediction algorithm that uses $\Phi_{t_0,i,t}$ as prediction.

3.3.1 *Clustering-Based Viewport Prediction Algorithm.*

Since the number of views for a video can be in the thousands or more, the clustering-based viewport prediction algorithm should be scalable, as we cannot compare the viewport of current viewer to all other previous viewers. In order to predict viewport for the current viewer, two functionalities should be provided: (i) classifying current viewer to a potential cluster, (ii) predicting viewport based on the cluster. Assume that a server stores the viewport trace of all previous viewing of a video. For each viewer v, the server stores unit quaternion of viewport center at time t, denoted by $q_{v,t}$, with sampling rate of $\frac{1}{30}$ second. The server runs the clustering algorithm offline and periodically as new views are received. For each video, the clustering is done at integer time steps t_0 and within $\Delta t = 1s$ intervals. So, for each time interval $\{[t_0, t_0 + \Delta t], t_0 = 1, 2, 3, ...\}$ a set of clusters is created. In order to classify a viewer according to the clusters, we examine if the viewer viewport center $q_{v,t}$ follows the path of a specific cluster. In other words, if $q_{v,t}$ is within 30° of a cluster center for the duration of clustering interval. So, server stores the center of each cluster at the trace sampling rate, i.e., every $\frac{1}{30}$ second. The center is the quaternion average of all viewers in the cluster. So for each cluster, we have quaternion of cluster center: $\Phi_{t_0,i,t}$ for $t_0 \leq t \leq t_0 + \Delta t$. Formally viewer v is put inside cluster i in time interval $[t_0, t_0 + \Delta t]$ if:

$$\theta(q_{v,t}, \Phi_{t_0,i,t}) < 30^\circ \quad \forall t \in [t_0, t_0 + \Delta t]$$

After classification, two outcomes are possible. If no cluster was found for the current viewer, we use a default prediction algorithm. The default is to use the most recent sample of the viewport as prediction. If a cluster is found for the viewer, the prediction is done based on $\Phi_{t_0,i,t}$ where $t = t_0 + \Delta t + \delta$ and δ is prediction window. In addition to $\Phi_{t_0,i,t}$, the number of viewers in the cluster, and average distance of viewports in the cluster to the center, i.e., $\Theta(V_{t_0,i}, t)$, is provided as meta-data in the clustering that can be used to determine the potential error in prediction. Next we discuss the effect of the number of viewers in the cluster and closeness on prediction accuracy.

4 PERFORMANCE EVALUATION AND RESULTS

In order to evaluate the performance of the proposed clustering method for viewport prediction, we needed a dataset of viewport traces. In addition, we wanted to investigate the importance of moving objects in the video and their effect on the pattern of viewport changes.

4.1 Viewport dataset

We use the dataset presented in [9]. This dataset contains 28 videos categorized into 15 classes based on camera motion and number of moving objects. Camera can be fixed or have horizontal, vertical, rotational, or mixed motion. In terms of number of moving objects, there can be no, single, or multiple moving objects. Usually, for videos with moving objects, viewports are more focused in certain RoI . Each video is one minute long and viewed by 30 persons.

4.2 Results

Assume that we have n viewport samples in time window $[t_0, t_0 + \Delta t]$, and we predict viewport at time $t_p = t_0 + \Delta t + \delta$, where δ is the prediction time horizon. We run the clustering-based viewport prediction for all videos in the dataset. We run prediction every second, with $\Delta t = 1s$, and for $\delta = 0.5$ to 10 seconds. In each run, we consider 80% of the viewers for clustering and run prediction for the remaining 20% ones. We repeat the same procedure until the average prediction accuracy converges to a stable value. Since viewers are not always classified into a cluster, we get two sets of results using the proposed method:

(1) prediction from clustering: when a cluster is found for the viewer we use the cluster-center to predict the viewport. We refer to this group as Cluster-Results (CR).
(2) when a user cannot be mapped to a cluster we use the last sample of user location to predict its viewport

The result of the method described above is referred to as Overall-Cluster-Results (OC). For cases where prediction from clustering is possible, we also try prediction from last sample and refer to it as Last-sample-on-Cluster-Results (LC). We compare performance of OC with other baseline methods:

- Linear regression (LR): In each time window $[t_0, t_0 + \Delta t]$, we apply linear regression on yaw and pitch samples separately and predict viewport using the obtained model.
- Last sample (LS): We use the current viewport center as the prediction at time t_p.
- Quaternion extrapolation (QE): This method extrapolates quaternion rotation of points in the time window using the method explained in Section 3.2.

The accuracy of prediction was measured as the overlap of predicted and actual viewport. It is the fraction of the actual viewport area covered by the predicted viewport in spherical domain. We assume FoV of viewport equal to 100°.

For each video, we also report the ratio of number of CR to OC under each figure inside parenthesis. This number shows that how many times the prediction was obtained using only clustering-based prediction method.

Depending on the content of the video, we get different results for the accuracy of prediction. In all cases, LR and QE have the lowest accuracy, especially for longer δ. These methods consider a linear movement and extrapolate it over time, while the actual viewport changes are different. Viewers move their head in a certain direction, and after a while they stop and change the direction of movement. So, modeling viewport changes as a continuous linear movement does not result in a good prediction. However, for longer δ, QE performance is better as it extrapolates movement in quaternion space and considers movement on the surface of sphere, unlike LR that models movement on yaw and pitch separately.

Interestingly, LS provides highest accuracy for shorter δ. Based on our observation in viewport traces, when viewers fixate in a certain direction, or have consecutive movements in opposite direction, using last sample provides better accuracy. Especially, for $\delta = 0.5$ to 1 second, the viewport is not very different from last sample. For longer δ values, for some videos, the accuracy of clustering-based prediction is better than LS. We show the detailed results for videos for which the performance of clustering-based prediction is worse

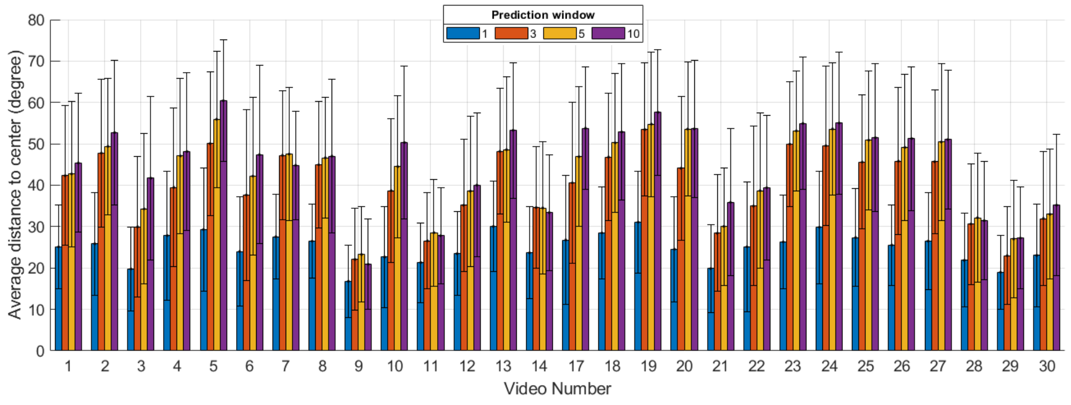
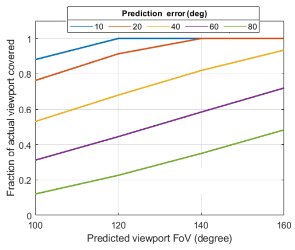
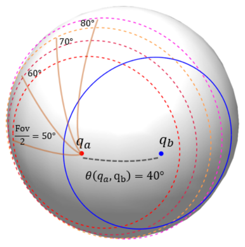


Figure 2: Average closeness of viewports in a cluster ± one Std. Dev.

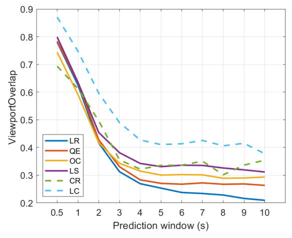


(a) Coverage of actual viewport for different prediction error and FoV size

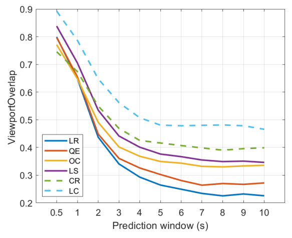


(b) Illustration of viewport coverage for distance of 40° and different prediction FoVs

Figure 3: Viewport overlap for different FoV size

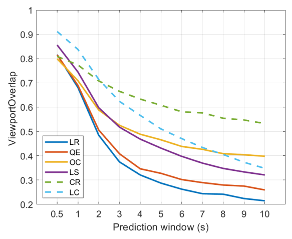


(a) Video 2 (11%)

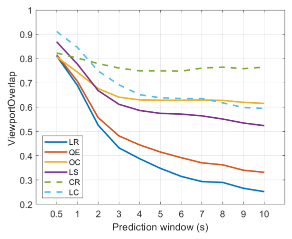


(b) Video 7 (19%)

Figure 4: Viewport overlap for different prediction methods and windows where CR is worse than LC



(a) Video 3 (35%)



(b) Video 9 (42%)

Figure 5: Viewport overlap for different prediction methods and windows where CR is better than LC

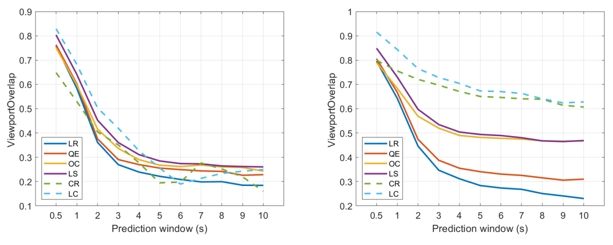
than, better than, or comparable to LS in Figures 4, 5, and 6, respectively. In these figures, LC and CR are shown with dashed-line, as they show accuracy for a subset of results, i.e., those that are classified in a cluster.

In Figure 4, performance of LS is better. These videos, 2 and 7, have no specific RoI, and they show a scenery of mountains with no moving target. In this case using the center of cluster for prediction results in worse accuracy compared to LS. For these two videos 11% and 19% of the times, respectively, the prediction is done using clustering.

Figure 5 shows the results for video 3 and video 9. These two videos have one moving target. It can be seen that for $\delta > 3$ seconds, clustering provides better accuracy. For example in video 9, 42% of the prediction are made with clustering. For prediction at $\delta = 10s$, the viewport overlap is 17% better than using LS. However, we do not see the same pattern for video 10 which is in the same category as video 9. It seems that clustering prediction works better for the cases with strong RoI, where most of the action happens in a specific part of the video and viewer is inclined to watch a certain RoI.

Viewport overlap results for videos 19 and 30 are shown in Figure 6. For these videos the performance of clustering and LS are comparable. In video 19, which does not have specific RoI, only 6% of predictions are made with clustering. The measurements for angular head speed in [9] also show that there is more head motion in this video. For this type of video, both clustering and LS cannot provide good results. On the other hand, for video 30, 29% of the predictions are based on clustering. However, this video has low head motion, and clustering does not improve performance.

Next, we investigate the relationship between accuracy of clustering prediction and the number of viewers in the prediction cluster, and also the closeness of points at prediction time. Table 1 shows the Pearson correlation coefficient between accuracy, absolute error, and those variables. There is weak correlation between absolute



(a) Video 19 (6%) (b) Video 30 (29%)

Figure 6: Viewport overlap for different prediction methods and windows where CR is comparable to LC

error and closeness. Lower closeness means that the viewports in the same cluster stay closer to each other in future. However, for cluster size there is not significant correlation.

4.3 Comparison with machine learning methods

In this section, we compare the performance of our proposed method with a LSTM-based method proposed in [10]. We ran our proposed method and LS method on the dataset that they have used. They have used nine videos from [14], where each video was viewed by 48 subjects. We obtain prediction accuracy of the LSTM-based method as reported in their paper. They measured accuracy in terms of the ratio of overlapping tiles between predicted and ground truth viewports. This metric results in higher accuracy compared to our method which measures the overlapped area covered by the two viewports. Table 2 shows the accuracy results for our proposed method (OC), LS, and LSTM-based method for prediction window length of 0.5, 1, and 2 seconds, using the dataset that they have used. The results are the average over all nine videos. It can be perceived that LSTM-based method underperforms in all cases. Similar to the previous results, OC and LS provide better accuracy for short prediction window.

Table 1: Correlation between prediction accuracy and closeness/cluster size

	Prediction window			
	1s	3s	5s	10s
Corr. Closeness & accuracy	0.29	0.35	0.33	0.35
Corr. Cluster size & accuracy	-0.21	-0.18	-0.16	-0.16

Table 2: Viewport prediction accuracy for videos in [14]

	Prediction Window(s)		
	0.5	1	2
OC	0.8828	0.8098	0.7172
LS	0.8993	0.8223	0.7247
LSTM-Based	(0.75,0.8)	(0.6,0.65)	(0.5,0.55)

5 CONCLUSION AND FUTURE WORK

In this paper, we presented a viewport prediction method based on clustering. Depending on the content of the video, this method achieves better accuracy for prediction window of 5 seconds and longer in more than 40% of the videos. However, for short prediction window, less than 3 seconds, using current viewport as prediction provides better accuracy. The viewport prediction results show that the accuracy highly depends on the video content. For videos with strong RoI, viewport can be predicted for longer horizon. Additionally, the accuracy can be higher for users that lie in clusters which cover a RoI. Therefore, as future work, a dynamic prefetching algorithm could be proposed for viewport-adaptive 360° video streaming. For the subset of viewers who focus on a RoI, the algorithm could prefetch video chunks for longer duration, as the accuracy of prediction is higher.

REFERENCES

[1] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[2] Xavier Corbillon, Gwendal Simon, Alisa Devlic, and Jacob Chakareski. 2017. Viewport-adaptive navigable 360-degree video delivery. In *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 1–7.

[3] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. 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*. ACM, 67–72.

[4] Xueshi Hou, Sujit Dey, Jianzhong Zhang, and Madhukar Budagavi. 2018. Predictive View Generation to Enable Mobile 360-degree and VR Experiences. In *Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network*. ACM, 20–26.

[5] Yongjae Lee and Woo Chang Kim. 2014. Concise formulas for the surface area of the intersection of two hyperspherical caps. *KAIST Technical Report* (2014).

[6] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang. 2019. Very Long Term Field of View Prediction for 360-degree Video Streaming. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 297–302.

[7] F Landis Markley, Yang Cheng, John Lucas Crassidis, and Yaakov Oshman. 2007. Averaging quaternions. *Journal of Guidance, Control, and Dynamics* 30, 4 (2007), 1193–1197.

[8] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D Beshay, and Ravi Prakash. 2017. Adaptive 360-degree video streaming using scalable video coding. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 1689–1697.

[9] Afshin Taghavi Nasrabadi, Aliehsan Samiei, Anahita Mahzari, Ryan P McMahan, Ravi Prakash, Mylene CQ Farias, and Marcelo M Carvalho. 2019. A Taxonomy and Dataset for 360◦ Videos. In *Proceedings of the 10th ACM Multimedia Systems Conference*. ACM, 273–278.

[10] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 1190–1198.

[11] Stefano Petrangeli, Gwendal Simon, and Viswanathan Swaminathan. 2018. Trajectory-Based Viewport Prediction for 360-Degree Virtual Reality Videos. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 157–160.

[12] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[13] Silvia Rossi, Francesca De Simone, Pascal Frossard, and Laura Toni. 2019. Spherical Clustering of Users Navigating 360° Content. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4020–4024.

[14] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 193–198.

[15] Lan Xie, Xinggong Zhang, and Zongming Guo. 2018. CLS: A Cross-user Learning based System for Improving QoE in 360-degree Video Adaptive Streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 564–572.