https://doi.org/10.2352/ISSN.2470-1173.2021.9.IQSP-255

A Content-Based Viewport Prediction Model

Dario D. R. Morais[1], Lucas S. Althoff[1], Ravi Prakash[2], Marcelo M. Carvalho[1], and Mylène C.Q. Farias[1]

[1]Department of Electrical Engineering, University of Brasília, Brazil.
[2]University Texas at Dallas, US.

Abstract

Viewport prediction technologies are often used by most popular adaptive 360-degree video streaming solutions. These solutions stream only the content considered as being more likely to be watched by the final user, with the goal of reducing the volume of network traffic without compromising the user's Quality of Experience (QoE). In this paper, we propose the Most Viewed Cluster algorithm (MVC), which is a hybrid viewport prediction method. It estimates the user viewport using two types of information: (i) the path of moving objects in the scene and (ii) the viewing behavior of previous users. Preliminary results show that MVC yields good results for long-term predictions.

Introduction

During the last decade, the interest for Augmented Reality (AR) and Virtual Reality (VR) has greatly increased. Among the most popular VR applications are the 360-degree videos, also known as spherical or omnidirectional videos. This type of video provides a 360-degree view of the scene captured from a single point. Users often use head-mounted displays (HMDs) to view these videos. In such case, the 360-degree video content encloses the viewer completely, providing an immersive experience, i.e., the sensation of being in the scene. 360-degree videos are captured using special camera systems which are generally composed of an array of cameras. These systems generate significantly more data than what is common in rectangular (regular) video scenarios.

It is worth noting that the additional information provides the user the ability to "look around" the scene by moving his/her head. To allow the user to explore the scene, 360-degree video technologies use motion-tracking sensors to provide visuals that react to the user's movements. In HMDs, head tracking is used to show the right camera angle and the perspective, using a combination of "inside-out", "outside-in", and inertia tracking. Sensors for magnetic field tracking and inertia tracking are embedded in the HMDs (inside-out) and this information is complemented by optical tracking cameras (outside-in). To control the position and orientation of the scene displayed to the user, the tracked head movements are represented using a 3-dimensional Cartesian coordinate system, with the origin point $(x,y,z) = (0,0,0)$ being the user's head. Generally, these head movements are modeled by intrinsic rotations and represented using Euler angles, which are referred to as yaw, pitch, and roll. Yaw represents the rotation around the y-axis, pitch represents the rotation around the x-axis, and roll represents the rotation around the z-axis.

At each instant of time, the HMD view is limited to only a portion of the video as seen from the center of the sphere, known as the viewport. As illustrated in Figure 1, the total area covered by the viewport is limited by the HMD's Field of View (FoV) and its coordinates depend on the orientation of the user's head [1]. For example, a typical $120^\circ \times 90^\circ$ FOV has a viewing area that corresponds to a sixth of the full spherical frame. Therefore, to provide a good quality of experience (QoE) and to avoid any discomfort, the displayed areas must be represented with high spatial and temporal resolutions. This contributes to the high amount of data necessary to display 360-degree videos, which represents a great challenge for both compression and transmission algorithms.

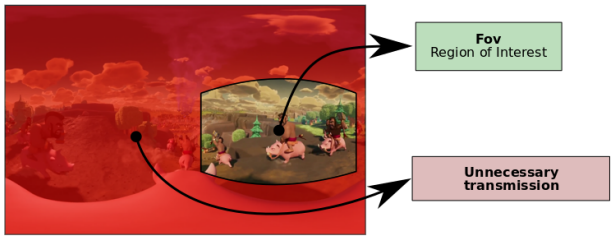


Figure 1: Illustration of the areas of a 360-degree video, a given viewport area, and the unnecessary transmission.

HMD tracking information can be used to reduce the volume of network traffic to support 360-degree video streaming. By analyzing the user's head-orientation, the viewing angle can be estimated and the part of the video that is not being viewed by the user can be discarded or streamed at a lower resolution. In other words, content that falls in the user's peripheral view can be rendered with less quality than the content in the focus area [2, 3, 4]. In Figure 1, the red area represents the user's peripheral view, while the colored area shows the region of interest. In this context, the concept of viewport prediction arises. Assuming that a transmission channel is used to send 360-degree videos, being able to predict where people will look at can help identifying the region of interest for the next frames (or next few seconds) and transmit this information with the highest priority or with more bits. Then, instead of sending the complete 360-degree frame, we send only the portion that contains the areas that users have a higher probability of watching. This way, we can reduce the amount of information transmitted over the channel without compromising the user QoE [5, 6]. Therefore, the goal of our work is to introduce a hybrid prediction algorithm that focus on where people tend to look at in future frames, by taking into account data from previous viewers and semantic information. This helps optimal adaptive streaming systems to send 360-degree videos through a transmission channel.

Currently, there are several viewport prediction methods available in the literature. Baek *et al.* [7] designed a subjective

experiment to obtain a dataset, and used the data to propose a VR adaptive streaming method based on saliency maps, compression, and pyramidal projection. Sun *et al.* [8] proposed a live streaming system for 360-degree videos using FoV prediction and caching. Nasrabadi *et al.* proposed a viewport prediction model [9] that classifies traces into clusters by analysing periods of one second [10]. Samples less than 30 degrees apart from one another are clustered together, and the algorithm uses the center of the cluster to predict the viewport for the following 1-10 seconds. Their results show that the algorithm's performance depends on the video content and the prediction window.

Petrangeli *et al.* [11] proposed a trajectory-based viewport prediction algorithm composed of two steps. First, the algorithm clusters the trajectories based on previous user data and it creates trend trajectories. Then, it predicts a trajectory by comparing all trend trajectories to the trajectory of the new user. The shortest path will be the predicted trajectory. The prediction time windows is from 5 to 10 seconds. Qian *et al.* [12] designed a viewport prediction system that uses linear regression. They showed that, for short time intervals (e.g., 0.5 seconds), the system is able to accurately predict the viewport. But, for longer intervals, linear regression is not very effective.

Park *et al.* [14] designed a view prediction system using semantic information, which is based on the Semantic Flow Descriptor (SFD) and View-object State Machine (VOSM) techniques. SFD is responsible for finding objects in each segment, while VOSM is responsible for predicting where the objects are going to be in future segments.

To the best of our knowledge, there is no work in the literature that uses object tracking for long-term (e.g., over 5 seconds) viewport predictions. Therefore, the main contribution of this work is to design a viewport prediction model that is able to predict viewports over 1-10 seconds taking into account object tracking. To accomplish this, we propose a hybrid system that takes into account the viewing traces of previous users and the video content.

This paper is organized as follows. First, we describe the proposed viewport prediction algorithm. Then, we present a descriptive statistical analysis of the adopted head movement dataset. Finally, we present the simulation results of the proposed hybrid prediction algorithm, along with the conclusions and future works.

Proposed Methodology

Human behavior while watching 360° videos can be quite variable, especially in terms of head movements. For some contents, most users move their heads and explore the scenes freely, while for other contents users explore very little of the scene. Therefore, the proposed methodology consists of a hybrid approach that combines (1) head movements from previous viewers and (2) tracking moving objects in the video content that attract the viewer's attention. More specifically, the proposed hybrid solution uses *three* types of prediction approaches: a completely passive approach, a user cluster center approach, and an object tracking approach. Next, we describe these three approaches and the final prediction algorithm.

In this work, we use the 360-degree dataset of in Nasrabadi *et al.* [13]. This dataset contains head movements collected from 30 participants while watching 28 high-definition 360-degree videos. The authors classified the videos in this dataset into 15 categories, according to camera motion and the number of moving objects in the scenes. In total, there are five camera motion categories: fixed, horizontal, vertical, rotational, mixture of the previous camera motion. Also, there are three moving object categories: no moving objects, single moving object, and multiple moving objects. Table 1 shows the classification of the 30 videos in this dataset, along with their spatial and temporal resolutions.

Completely Passive Approach

The completely passive approach is based on the fact that some people watch 360-degree videos passively, without moving their heads. This means that some users basically keep their heads fixed in the central point (yaw = 0° and pitch = 0°) or move their heads just a little. In this work, the completely passive approach considers the central point as the center of the viewport. This approach provides a good viewing prediction for 360-degree videos where the camera is moving and there is only one main subject (or attention focus) in the scene.

Cluster Center Approach

For specific 360-degree contents, certain objects or areas are universally salient, while for other contents the saliency of specific areas or objects vary from person to person. Nevertheless, it can be observed that, on average, people can be grouped while watching 360-degree videos, with people that share similar viewing behaviors [15]. In other words, it is possible to identify groups of individuals with similar head movements (behavior) and classify them as belonging to specific user clusters. More specifically, the head movements belonging to several users can be clustered according to their movement characteristics, with each cluster indicating a head movement behavior of a specific group of users. In this work, we use a simple k-means algorithm to cluster user head movement data, limiting the maximum number of clusters to three for simplicity.

The coordinates of the center of mass of each cluster define the reference point that represents where, on average, users in a particular cluster look when watching a specific scene of the 360-degree video. Let us define α_i and ϕ_i as the yaw and pitch angles, respectively, for the i-the viewer. Assuming each viewer is equally important, the center of a specific cluster composed of n viewers can be computed simply by taking the average of the angles for all viewers in the cluster, with $\overline{\alpha}$ and $\overline{\phi}$ representing the average of the yaw and pitch angles, respectively. In this approach, a viewport prediction is given by the area around the user cluster center that is most similar to the current user. This most similar area is computed by taking the euclidean distance between the reference points and the view of the current user. Further details will be given in the Prediction Algorithm section.

Figure 2 depicts an example of possible clustering scenarios. The left image shows a frame in which the user movement data can be grouped into a single cluster. The blue points, which are the center of the viewports of each viewer from the training set, represent the points in this cluster. The red point represents the cluster center of the blue points. The right image shows a scenario that the training set is scattered and the samples of this training set are grouped in multiple clusters. The red, yellow, and pink points represent points in the first, second, and third clusters, respectively, while the light blue points are the three cluster cen-

Table 1: Spatial and temporal resolution of taxonomy dataset [13]. Analysed videos are highlighted.

Camera Movement	No Object	**One Object**	Multiple Objects
Fixed	Video 1 - 3840x1920 25fps	**Video 3 - 3840x2048 29fps**	Video 5 - 3840x2048 30fps
	Video 2 - 3840x2160 29fps	Video 4 - 3840x1920 29fps	Video 6 - 3840x2048 29fps
Horizontal	Video 7 - 3840x2160 25fps	**Video 9 - 2560x1440 29fps**	Video 11 - 3840x2160 24fps
	Video 8 - 3840x2048 29fps	**Video 10 - 3840x1920 29fps**	Video 12 - 3840x2160 30fps
Vertical	Video 13 - 3840x1920 29fps	Video 15 - —	Video 17 - 3840x2048 30fps
	Video 14 - 3840x1920 29fps	Video 16 - —	Video 18 - 3840x2160 29fps
Rotational	Video 19 - 3840x1920 29fps	**Video 21 - 3840x2160 29fps**	Video 23 - 3840x1920 29fps
	Video 20 - 3840x1920 29fps	**Video 22 - 3840x1920 29fps**	Video 24 - 3840x1920 29fps
Mixed	Video 25 - 3840x2048 25fps	Video 27 - 3840x1920 30fps	Video 29 - 3840x2160 25fps
	Video 26 - 3840x2160 29fps	**Video 28 - 3840x1920 29fps**	Video 30 - 3840x1906 29fps

Figure 2: Two frames of a sample 360-degree video. The user data in the 1st frame (left) can be grouped in a single cluster, while the user data of the other frame (right) requires multiple clusters.

ters. It is worth mentioning that viewers' attention changes over time, i.e., objects and specific areas that attract attention in one instant of time can change in a future instant of time.

Notice from Figure 2 (left image) that when viewers explore similar areas of a 360-degree video, their movements can be combined into a single cluster. In this very particular case, the cluster center gives a good viewport prediction. On the other hand, when there is a large variation among different users, i.e. users explore the 360-degree video very differently, using a single cluster does not provide a good viewport predictor. In fact, in these cases we may have many user clusters and, therefore, before we can perform viewport prediction, we have to find the best cluster for the current user and use its center as the center of the predicted viewport.

Object Tracking Approach

An object tracking algorithm is able to track object paths over a certain period of time. In this case, objects can include faces, persons, animals, vehicles (such as cars, trucks, or motorcycles), aircrafts, drones, signs, buildings, books, trees, tables, chairs, couches, and so on. Objects like buildings and signs are fixed and do not move around the camera, while objects like people, animals, and vehicles can be mobile and change their position in the scene. The applications of object tracking algorithms are very diverse, including sports analysis, autonomous driving, robotic navigation, and viewport prediction. In this work, we focus on moving objects for viewport prediction, since movement tends to attract viewer attention in 360-degree videos.

In a 360-degree video, objects can be closer or farther away from the camera. Moreover, objects can initially be in the background and, after some time, come to the foreground and vice-versa. Therefore, object size may change over time such that, the closer an object is to the camera, the bigger it appears. Some object tracking algorithms in the literature [16], such as boosting [20], Multiple Instance Learning (MIL) [21], Kernelized Correlation Filters (KCF)[22], Channel and Spatial Reliability Tracking (CSRT) [23], medianflow [24], Tracking Learning Detection (TLD) [25], Minimum Output Sum of Squared Error (MOSSE) [26], and go turn [27], do not take this into consideration. Therefore, in this work, we use a tracking algorithm that takes into account object size variation over time.

Moving objects are tracked in 360-degree videos using a motion tracking algorithm, which is a two-step method. First, we detect the object(s) of interest in a frame F using Yolo [17]. Second, we track this object over the following frames (from frame $F+1$ to frame $F+k-1$) using an object tracking technique. For example, let us say that we have a video with 100 frames. The motion tracking algorithm is executed every k frames, which means the algorithm follows the object size variation over an interval of k frames. If we set $k=10$, the object detection algorithm would be applied to frames $1, 11, 21, \ldots, 91$, while the object tracking algorithm would be applied in the intermediate frames 2-10, 12-20, ..., 92-100. So, if the object detection algorithm fails to find an object in a given frame, the algorithm tries to identity objects in the subsequent frame. Given that a reference object is found in a given frame F in the position (l,c), the object tracking algorithm tries to identify the most similar object by searching the area in each subsequent frame that is close to the reference object. This area can range from $(l-d,c-d)$ to $(l+d,c+d)$, where d is the distance in pixels from the (l,c) position. To measure the sim-

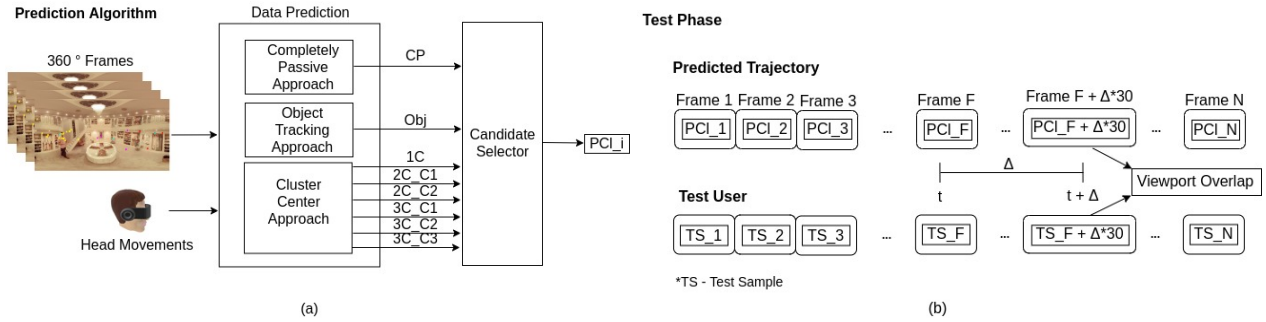


Figure 3: Pipeline of MVC Prediction algorithm in the (a) Training and (b) Test phases.

ilarity of two areas, we use a correlation measure, given by the following equation:

$$r_{xy} = \frac{Cov(x,y)}{\sigma_x \sigma_y} = \frac{\sum_{i=o}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=o}^{N}(x_i - \bar{x})^2 \sum_{i=o}^{N}(y_i - \bar{y})^2}}, \quad (1)$$

where x and y are the two areas (sets) being compared, $\bar{x}$ and $\bar{y}$ are their average values, x_i and y_i are the values in x and y, respectively, and N is the total number of elements (pixels) in both areas. The areas in each subsequent frames with the highest correlation values are the corresponding detected object for each subsequent frame. It is worth pointing out that the center of the predicted moving object is going to be the center of the predicted viewport.

Prediction Algorithm

Figure 3 shows the block diagram of the proposed viewport prediction algorithm, the MVC. The method requires a training stage to obtain the predicted trajectories, using the previously described approaches. The data processing of this phase is done offline. Notice in Figure 3(a) that the method takes two inputs: the user head movement data acquired from previous users and the 360-degree video frames. The first stage of the method is the predictor (indicated in the "data prediction" block), which is responsible for finding candidate areas for the viewport algorithm using the three approaches described in the previous sections. The algorithm computes a central point (CP) using the Completely Passive Approach, identifies the scene main objects (Obj) using the object tracking approach, and computes user clusters using the central cluster approach. In summary, the predictor outputs the 8 following candidates for viewport prediction:

- 1 Central point (CP - Completely Passive Approach);
- 1 object position (Obj);
- 1 cluster center (1C) considering only one k-means cluster ($k = 1$);
- 2 cluster centers (2C_C1, 2C_C2) considering two k-means clusters ($k = 2$);
- 3 cluster centers (3C_C1, 3C_C2, 3C_C3) considering three k-means clusters ($k = 3$).

The next stage of the method, the "Candidate Selector block" stage, is responsible for deciding what is the best prediction based on the computed candidates. The best candidate corresponds to the cluster that is closer to the head position of the current user. In this case, we use the Euclidean distance to identify the closest candidate and, then, we apply this metric to the entire training set. The cluster that is more populated, or has more people looking at, will be the predicted cluster (PCl_i) for the ith-frame. In summary, we compute a predicted candidate PCl_i for each frame i, where $1 \leq i \leq N$ (N is the number of video frames). Therefore, after running the algorithm for N frames, we build a predicted trajectory (as depicted in Figure 3(b)), showing the areas where people are likely to look.

To verify if the predictions generated are accurate, we compare the viewport areas that users (in the dataset) watched to the viewport areas predicted by the proposed method. Figure 3(b) shows the procedure used to test the proposed method. For example, consider that $t = 10$s and the temporal resolution is equal to 30 frames per second (fps), which means that this time instant corresponds to the 300-th frame ($F = 300$) of the video. To predict $\Delta t = 1s$, i.e., an interval of one second into the future, the algorithm computes the predicted viewport for all frames from $F_i = 300$ to $F_f = F_i + \Delta \times 30 = 330$. To verify if our prediction is accurate, we use the viewport overlap as our performance metric [18].

Head Movement Data analysis

In this section, we present a descriptive statistical analysis of the behavior of viewers that watched the dataset videos in the dataset subjective experiment. Our statistical analysis, as well as the prediction simulations, are implemented on a subset of the dataset, which consisted of videos containing *only* one moving object (videos 3, 9, 10, 21, 22, 28). These videos were selected because they were the simplest parametric case with fewer factors impacting viewers' attention that could compromise our analysis.

For the analysis, a single head gaze direction is defined by angles pitch (α) and yaw (ϕ). For a single video, we represent pitch and yaw data as matrices of concatenated participants angle arrays. So, for a video with F frames and N participants there will be two $F \times N$ matrices, one for each head motion angle. Finally, to match video temporal resolution (30fps) and the experiment device sample rate (60Hz), the matrices are sub-sampled by averaging every two sub-sequential angles of the original data.

In order to describe head movement data, we first explored the overall head navigation behavior of the users. Table 2 shows a summary of the statistics of the pitch and yaw values captured from the head movements of viewers for the considered subset of videos. From this table, we notice that videos 9 and 21 have low deviations in pitch, but the highest deviations in yaw. This illustrates the significant challenges in predicting head movement, since navigation variability depends on multiple factors, including

Table 2: Overall statistics of analysed videos.

Video	Yaw std. dev.	Pitch std. dev.	Yaw Mean	Pitch Mean
3	125°	13°	−18°	−0.57°
9	150°	7°	−34°	0.29°
10	135°	11°	20°	−0.31°
21	143°	5°	18°	0.70°
22	97°	9°	20°	−0.45°
28	135°	13°	−45°	−0.96°

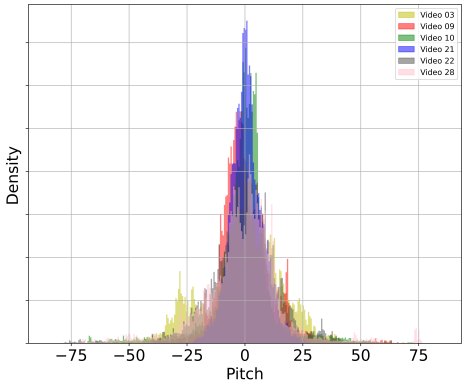


Figure 4: Histogram showing the overall pitch distribution for each video. Better seen with colors.

the video content and individual user viewing preferences.

For a closer look on pitch values, Figure 4 shows a histogram of the captured pitch values. Notice that, for all videos, the pitch values are concentrated between $-25°$ and $25°$. Therefore, in terms of pitch head movement direction, viewers tended to gaze mostly around the equator (pitch = $0°$). This feature is known in the literature as the equator bias [19]. However, for video 03, we observe a second pronounced peak around $-25°$. This could be due to the narrator's position, which is below the equator in the majority of the frames in this video. This effect shows that, depending on the content, there can be significant deviations from equator bias. However, the majority of the distributions shows the same equator bias tendency. This tendency is especially pronounced for videos 10 and 21. It is worth pointing out that, even though video 10 has relevant visual contents at different positions, the narrator, who is the most prominent visual cue, stays at the equator line for the entire video. Figure 5 shows a rain cloud of pitch data gathered from a subset of 15 participants while watching video 10. The graph shows the distribution of pitch head motion across the different viewers (experiment participants) coincidentally in terms of maximums (box plot), dispersion (scatter plot) and means (probability density function). Notice that, from this inter-viewer visualization, that pitch distribution is very different across viewers. For example, users 4 and 5 have low pitch variability (around $20°$), while users 10 and 11 have high pitch variability (more than $80°$). Considering dispersion patterns, plato-like and peak-like exploration distinguish viewers, for example viewer 13 the most concentrated distribution while 11 had the most diffuse one. However, highly concentrated navigation do not determine exploration limits, for example, participant 4 explored less angles but had more diffuse navigation than participant 13.

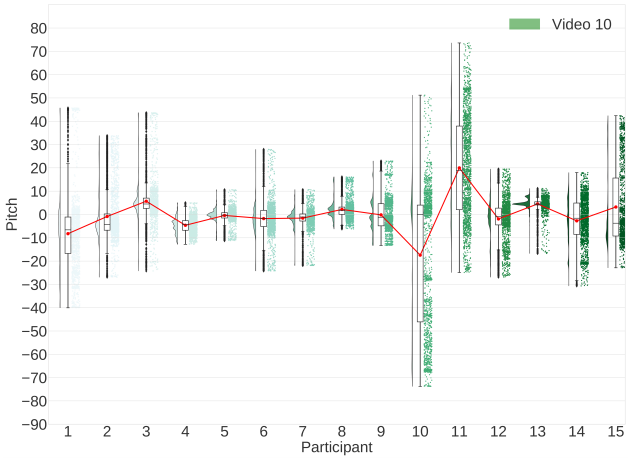


Figure 5: Rain cloud plot of pitch values gathered from a subset of 15 participants while watching video 10. Average pitch values are connected by a red line.

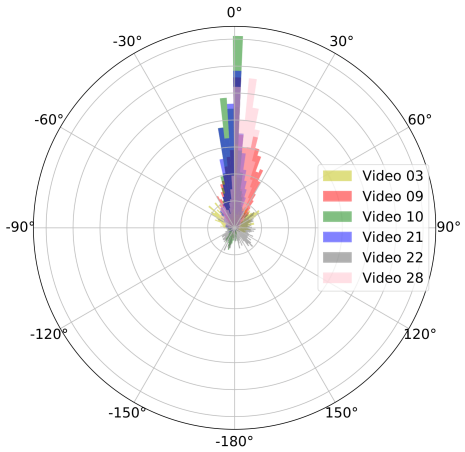


Figure 6: Polar histogram showing yaw distribution for each video.

Figure 6 depicts the distribution of the captured yaw values in a polar histogram. This visualization of yaw distribution is adequate due to the head navigability of the spherical grid continuity in yaw direction, in contrast with pitch direction where users cannot rotate their heads. Notice that, in general, yaw fixations peak in the central position (yaw $= 0°$), gazing around the equator, and stabilizes mostly in the viewer frontal regions ($-30°$ <yaw $< 30°$). Another observed pattern is that videos with higher pitch also have a higher standard deviation in yaw angles. Specifically, video 22 has a higher yaw variability, even for back-regions (with yaw around $-180°$)), where the content has low movement.

In summary, in terms of viewers' behavior, there is some variability in users head movements. The yaw standard deviation is greater than $97°$ for all videos, reaching $150°$ for some videos. Despite the fact that the pitch standard deviation is low for all videos, when we examine the pitch angles user-by-user, we find a high variability. As mentioned earlier, these variations in head movements are probably due to the video content and the individual viewer preferences.

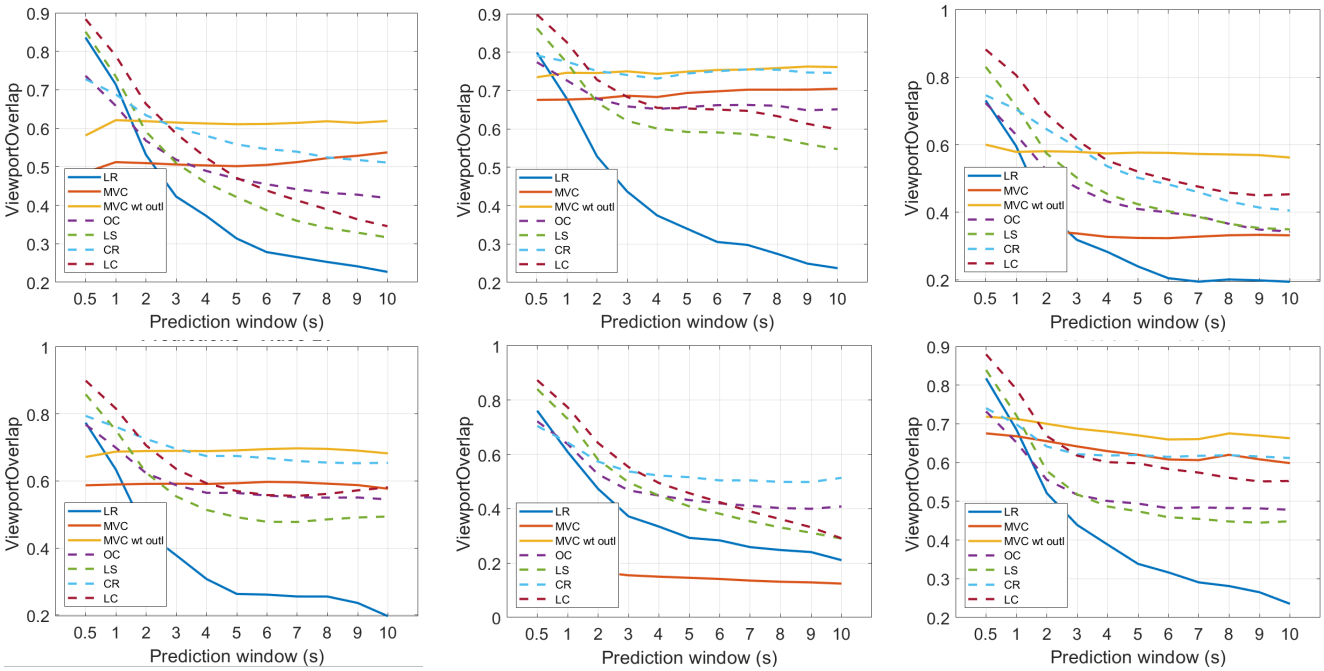


Figure 7: Comparison between MVC Performance and different prediction methods.

Viewport Prediction Results

As mentioned earlier, to assess the performance of the MVC algorithm, we use a subset of six videos from Nasrabadi's dataset [13]: videos 3, 9, 10,21, 22, and 28. These videos had a single moving object, which made the object tracking approach more feasible. In our tests, we examine two versions of the MVC algorithm. The first one uses an outlier threshold, which excludes samples where the angular distance between test and predicted sample are greater than 90 degrees. The second version considers all samples, without taking into account an outlier threshold. We compared the performance of the MVC algorithm with five viewport prediction algorithms [9]: Linear Regression (LR), Last Sample (LS), Cluster-Results (CR), Overall-Cluster-Results (OC) and Last-sample-on-Cluster-Results (LC).

Figure 7 presents the viewport predictions obtained for each of the 6 videos. The graphs show the viewport overlap [18] versus the prediction window (in seconds). The viewport overlap metric computes how close 2 viewports are to each other. If, for example, 2 viewports completely overlap, that would be the case of a perfect viewport prediction and the computed viewport overlap value would be 1. If there is no overlap area between two viewports, the viewport overlap value is 0. From the graphs in Figure 7, we notice that the MVC without an outlier threshold generally outperforms the MVC with an outlier thresold.

Results in Figure 7 are in accordance with previous results [9] that show that when the time window increases, the prediction accuracy decreases for most viewport prediction algorithms. This happens because the predictions for these 5 techniques listed are based solely on information available at the client side. So, future viewport predictions are based only on the last samples watched by the viewer. In the case of the proposed MVC algorithm, the prediction is based on information gathered from previous viewers. For this reason, the predictions remain steady over the prediction window.

It is worth pointing out that the results for the LR algorithm are very similar for all videos. LR only presents good results for prediction windows with sizes ranging from 0.5 to 1 second. Comparing the results obtained with the LS, CR, OC, and LC prediction techniques, we observe that they all have similar results, with LC performing the best and OC performing the worst. The CR prediction technique has the best long-term performance, while the LS prediction technique has the worst performance. If we compare the MVC and these techniques, we notice that, in general, its performance is worse for prediction windows smaller than 2 or 3 seconds. For videos 9 and 28, MVC has a better performance beyond the 2-seconds prediction window. For the videos 3, 10 and 21, MVC performs better than the other techniques for prediction windows beyond 3-seconds. However, MVC does not perform well for the video 22. We believe this happens because the 6 test viewers had a higher variability of head movements, as can be seen in Figure 5.

Conclusions

In this paper, we presented the Most Viewed Cluster (MVC) viewport prediction algorithm. The proposed algorithm uses a hybrid approach to predict the viewport of 360-degree videos using two types of information: (i) the path of moving objects in the scene and (ii) the viewing behavior of previous users. In this paper, we conducted a statistical analysis of head movement data of the 360-degree dataset introduced by Nasrabadi *et al.* [13]. The analysis showed that there is a great inter-user head movement variability, which reinforces our assumption that predicting head motion is a great challenge. The proposed MVC algorithm was tested on Nasrabadi *et al.*'s dataset and the prediction results were compared to 5 other viewport prediction algorithms. On the one hand, MVC achieves superior performance for long-term prediction windows (e.g., 2-10 seconds or 3-10 seconds). On the other hand, for shorter time windows (e.g., 0-2 seconds or 0-3 seconds),

the other viewport prediction algorithms perform better. For future work, we intend to examine multiple moving objects and explore different behavior patterns so that we can make predictions for the whole dataset used in this work.

Acknowledgments

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001, by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), and by the Decanato de Pesquisa e Inovação, University of Brasília Brazil.

References

[1] El-Ganainy, T.; Hefeeda, M. Streaming virtual reality content, CoRR, vol. Abs/1612.08350, 2016.

[2] Yaqoob, A.; Bi, T.; Muntean, G. A survey on adaptive 360 video streaming: Solutions, challenges and opportunities. In: . [S.l.: s.n.], 2020. v. 22, p. 2801–2838.

[3] Ozcinar, C.; Cabrera, J.; Smolic, A. Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality. In: . [S.l.: s.n.], 2019. v. 9, p. 217–230.

[4] Garcia, H. D.; Farias, M. C.; Prakash, R.; Carvalho, M. M. Statistical characterization of tile decoding time of hevc-encoded 360 video. Electronic Imaging, Society for Imaging Science and Technology, v. 2020, n. 9, p. 285–1, 2020.

[5] Cater, K.; Chalmers, A.; Ledda, P. Selective quality rendering by exploiting human inattentional blindness: looking but not seeing, in Proceedings of the ACM symposium on Virtual reality software and technology. ACM, 2002, pp. 17–24.

[6] Farias, M. C.; Akamine, W. Y. On performance of image quality metrics enhanced with visual attention computational models, Electronics letters, vol. 48, no. 11, pp. 631–633, 2012

[7] Baek, D.; Kang, H.; Ryoo, J. Sali360: design and implementation of saliency based video compression for 360 video streaming. In: Proceedings of the 11th ACM Multimedia Systems Conference. [S.l.: s.n.], 2020. p. 141–152.

[8] Sun, L.; Mao, Y.; Zong, T.; Liu, Y.; Wang, Y. Flocking-based live streaming of 360-degree video. In: Proceedings of the 11th ACM Multimedia Systems Conference. [S.l.: s.n.], 2020. p. 26–37.

[9] Nasrabadi, A. T.; Samiei, A.; Prakash, R. Viewport prediction for 360 videos: a clustering approach. In: Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. [S.l.: s.n.], 2020. p. 34–39.

[10] Rossi, S.; De Simone, F.; Frossard, P.; Toni, L. Spherical clustering of users navigating 360 content. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). [S.l.: s.n.], 2019. p. 4020–4024.

[11] Petrangeli, S.; Simon, G.; Swaminathan, V. Trajectory-based viewport prediction for 360-degree virtual reality videos. In: IEEE. 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR). [S.l.], 2018. p. 157–160.

[12] Qian, F.; Ji, L.; Han, B.; Gopalakrishnan, V. Optimizing 360 video delivery over cellular networks. In: Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges. [S.l.: s.n.], 2016. p. 1–6.

[13] Nasrabadi, A. T.; Samiei, A.; Mahzari, A.; Mcmahan, R. P.; Prakash, R.; Farias, M. C.; Carvalho, M. M. A taxonomy and dataset for 360 videos. In: Proceedings of the 10th ACM Multimedia Systems Conference. [S.l.: s.n.], 2019. p. 273–278.

[14] Park, J.; Wu, M.; Lee, E.; Chen, B.; Nahrstedt, K.; Zink, M.; Sitaraman, R. Seaware: Semantic aware view prediction system for 360-degree video streaming. In: . [S.l.: s.n.], 2020.

[15] Nguyen, A.; Yan, Z. A saliency dataset for 360-degree videos. In: Proceedings of the 10th ACM Multimedia Systems Conference. [S.l.: s.n.], 2019. p. 279–284.

[16] www.pyimagesearch.com/2018/07/30/opencv-object-tracking

[17] Redmon, J.; Farhadi, A. "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[18] Lee, Y.; Kim, W. C. Concise formulas for the surface area of the intersection of two hyperspherical caps. KAIST Technical Report, 2014.

[19] Sitzmann, V.; Serrano, A.; Pavel, A.; Agrawala, M.; Gutierrez, D.; Masiá, B.; Werzstein, G. How do people explore virtual environments. arXiv: Computer Vision and Pattern Recognition. 2016.

[20] Grabner, H.; Grabner, M.; Bischof, H. Real-time tracking via on-line boosting. In:Citeseer. Bmvc. [S.l.], 2006. v. 1, n. 5, p. 6.

[21] Babenko, B.; Yang, M.-H.; Belongie, S. Visual tracking with online multiple instance learning. In: IEEE. 2009 IEEE Conference on computer vision and Pattern Recognition. [S.l.], 2009. p. 983–990.

[22] Henriques,J. F.; , Caseiro, R. ; Martins, P. ; Batista,J. Exploiting the circulant structure of tracking-by-detection with kernels. In proceedings of the European Conference on Computer Vision, 2012.

[23] Lukezic, A.; Voj'ir, T.; Zajc, L.C.; Matas, J.; Kristan, M. Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision, 2018.

[24] Kalal, Z., Mikolajczyk, K., Matas, J. (2010, August). Forward-backward error: Automatic detection of tracking failures. In 2010 20th international conference on pattern recognition (pp. 2756-2759). IEEE.

[25] Kalal, Z.; Mikolajczyk, K.; Matas, J. Tracking-learning-detection. IEEE Computer Society, USA, v. 34, n. 7, p. 1409–1422, jul. 2012. ISSN 0162-8828.

[26] Bolme D.S.; Beveridge J.R.; Draper B.A.; Yui M.L. Visual object tracking using adaptive correlation filters. In Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[27] Herakleous, K; Poullis, C. 3DUNDERWORLD-SLS: An Open-Source Structured-Light Scanning System for Rapid Geometry Acquisition. arXiv preprint arXiv:1406.6595, 2014.

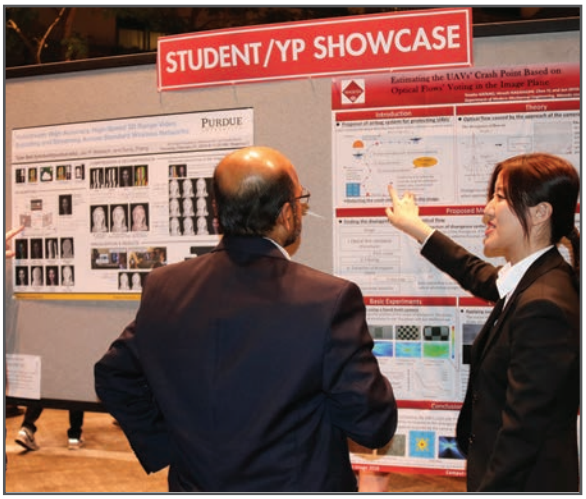
JOIN US AT THE NEXT EI!

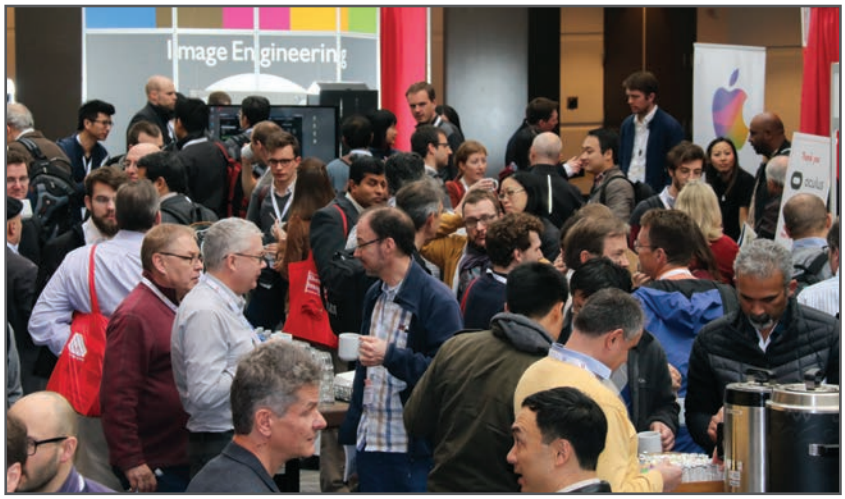
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org