LSTM-based Viewport Prediction for Immersive Video Systems

Gioacchino Manfredi*, Vito Andrea Racanelli*, Luca De Cicco*, and Saverio Mascolo*

*Politecnico di Bari, Department of Electronic and Information Engineering, Bari, Italy
E-mails: {name.surname}@poliba.it

2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet) | 979-8-3503-3884-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/MedComNet58619.2023.10168847

***Abstract*—Immersive multimedia content delivery is becoming increasingly popular due to the spread of Head Mounted Displays. In particular, omnidirectional video streaming is gaining ground among video delivery platforms. Delivering 360° video content over the Internet requires much larger bandwidth compared to classic 2D videos. Therefore, for the purpose of reducing bandwidth consumption, the tiling technique breaks down the video into smaller portions so that those falling outside the user's viewport are encoded at a low resolution whereas those in the viewport are encoded at a higher resolution. This operation can be performed only when the user's future viewports are known in advance. Thus, it is necessary to provide a trustworthy prediction of future viewports. In this work, we show that users have a tendency to explore the environment at the beginning of the video and then to focus on one of the regions attracting more attention (Points of Interest). This insight is helpful when it comes to designing viewport-adaptive streaming techniques. On this basis, we propose a viewport prediction approach that combines Long Short-Term Memory (LSTM) networks and the classic naive technique. Preliminary simulative tests show promising results.**

***Index Terms*—Omnidirectional Videos, Head Mounted Display, Points of Interest, Machine Learning, LSTM**

I. Introduction

Immersive videos are considerably spreading in several fields such as entertainment, gaming, and precision agriculture [1], [2], [3]. The streaming of Omnidirectional Videos (OVs) already represents a substantial feature for leading platforms such as YouTube or Facebook. Compared to classical 2D videos, OVs are more difficult to handle since they require higher bandwidth to provide the same level of visual quality to users. Streaming a 360° video would require a network bandwidth of 400 Mbps to deliver a video quality similar to that of a 4k resolution 2D video, which would need about 25 Mbps instead [4], [5]. As a consequence, choosing a suitable compression process is crucial when OVs are transmitted.

Immersive videos are produced by capturing scenes from different angles using special cameras. Generally, immersive content is rendered through a suitable device called Head Mounted Display (HMD), which allows users to freely explore the environment and choose which portion of it to watch. Therefore, it is important to point out that in each instant the HMD shows only a part of the captured scene to the user. Such a frame, called viewport, is roughly a sixth of the whole omnidirectional scene [6]. As a result, sending the whole scene with a high resolution would unavoidably imply a waste of bandwidth, which could be employed to improve the quality of the viewport. Hence, the viewport-adaptive streaming technique has been proposed for 360° video streaming over the Internet. With such an approach, only the regions currently falling into the user's viewport are encoded at high resolution, whereas the other regions are encoded at a lower quality (or not delivered at all in the extreme case). This way, bandwidth is optimally utilised provided that the video provider knows in advance the future viewports the user is going to watch. However, it is not trivial to get a trustworthy estimation of future viewports.

The first step towards saving bandwidth is devising a good encoding technique for OV delivery. In particular, 3D spherical scenes must be broken down in order to be encoded and transmitted. The state of art provides several approaches, such as the Facebook pyramidal projection [7], which projects the 3D spherical scene onto the different sides of a pyramidal 3D object. The base of the pyramidal 3D object, presenting less distortion, keeps the portion of the video with the most interesting content. The other video portions are mapped on the sides. Then, the pyramid is unfolded and mapped onto a 2D plane. However, this technique presents several encoding inefficiencies that impact the resulting quality and the achievable bitrate reduction [8]. Other techniques could be the barrel layout and the offset projection mapping [9]. The most widely adopted strategy is the EquiRectangular Projection (ERP), which simply consists of projecting 3D scenes onto a 2D plane. This way, it is possible to treat the video as a common 2D video when it is delivered over the network and then, once it reaches the user, it is projected back onto a sphere so as to be correctly visualised with the HMD.

To actually implement viewport-based adaptive streaming techniques with ERP, it must be possible to separate the portion of the scene watched by the user from the rest, to be sent at a lower resolution. To this purpose, every frame of the projected 2D video is divided into smaller portions so as to easily identify the part of the scene the user is interested in. The most widely adopted technique is called *tiling* [10], and consists of breaking down frames into smaller frames. As already explained, viewport-based adaptive streaming can be employed only when future viewports are known wrt a user. However, in practice, it is only possible to make predictions about future viewports. Obviously, the more accurate such predictions, the higher the Quality of Experience (QoE), i.e., the degree of satisfaction of the user. Several techniques have been proposed in the literature. In [11], the authors focus on long-term predictions of the user viewport. After clustering users' similar viewing behaviours associated with the roll, pitch and yaw angles, pre-computed functions are built for each cluster. Then, such functions are employed online to predict a user's future viewports. Another approach, presented in [12], implements machine learning methods to predict future head rotations on the basis of past head and eye motions and other users' motions. In this case, good accuracy is maintained up to a one-second horizon, which means only considerably short-term predictions are

considered. In [13], RAPT360 is proposed, i.e., a strategy that employs reinforcement learning and optimisation techniques to perform viewport prediction and rate adaptation. Notice that all of the aforementioned works are based on the precise position of the user's head in order to predict the exact position of their eyes or viewport in the future. In addition, most of them do not consider video streaming standards, which are necessary to actually implement any viewport-adaptive streaming technique. A standard-compliant approach can be found in [14], where the *Mosaic* technique is introduced. Mosaic is a neural network-based viewport prediction technique that assigns a specific bitrate to the different tiles in such a way that QoE is optimised. In this case, too, only users' viewports are considered.

In this work, we show that, in most videos, users may present a predictable behaviour based on the content of the video. In particular, we show that users, after an initial exploring phase, tend to follow a Point of Interest (PoI). PoIs are regions in the video representing the most interesting part of the scene, which therefore attract the user's attention. After this analysis, we propose to map, for each instant of time, the gaze trajectory of a user to one of the PoIs or to the exploring status. Then, machine learning techniques and traditional approaches are combined to get an improvement in the viewport prediction.

II. Users' behaviour

In this section, we are going to analyse some datasets containing users' viewport trajectories for a number of selected OVs. The goal of such an analysis is to prove that users, after an initial phase in which they explore the scenario around them, tend to follow one of the PoIs contained in the scene. This information is crucial to develop a viewport prediction algorithm that is not based on the exact position of the user's head but on the position of the PoIs.

To this end, 6 existing datasets have been selected [15], [16], [17], [18], [19], [20]. Among these, 31 videos have been chosen with the condition that at least one PoI is present. As a consequence, videos with too many PoIs or with no PoIs at all (e.g., documentaries) have been neglected.

Before analysing data, some necessary processing is needed. All videos have been converted to the same format and then an ERP has been performed. At this point, for each video and user, the dataset contains the tuple *(time, x, y)*, where *time* indicates the time samples in seconds of the video while x and y are the coordinates on the ERP of the fixation point, i.e., the point the user's eyes are watching. Then, for each time sample, the PoIs are manually selected, i.e., a rectangle around the PoI is considered and the corresponding width, height, and coordinates of a reference point in the rectangle are saved. Notice that this procedure could be performed automatically by using already existing machine learning algorithms. Once the PoIs identifying regions of the frames are generated, it is possible to overlap such regions with the fixation points of the users to check which of them fall in a PoI. Let us point out that, for each user, we do not consider the single fixation point but a rectangle that models the Field of View (FoV) of the user, i.e., the viewport. After this check, for each video, user, and time sample, we have

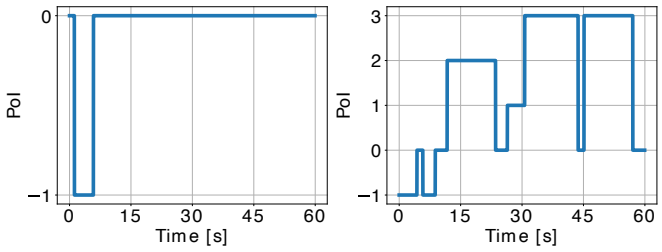

Fig. 1: Examples of PoIs tracked by a user in two different videos

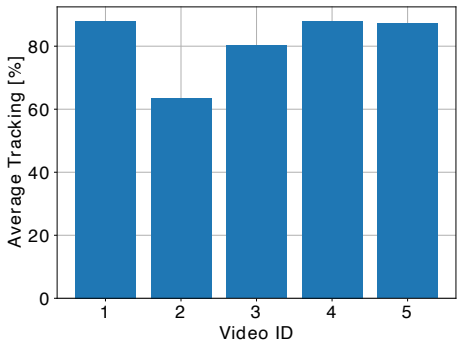

Fig. 2: Average tracking percentage for each video

generated a file containing the PoI watched. Notice that if no PoI overlaps the FoV, then the user is supposed to be in exploration mode. As a preliminary analysis, 5 videos and 30 users have been selected. Then, for each user, we can depict the profile of the PoIs watched during the video. An important assumption made throughout this work is that the number and id of PoIs present in the videos do not change over time.

As an example, Fig. 1 shows the PoIs followed by two users wrt to two different videos. In the first case, the video has only one PoI, which has been identified by 0 in the figure, while -1 indicates the case in which the user is in the exploration phase. The video in the second figure shown has 4 PoIs instead. Notice that such profiles may be very different according to a user's behaviour. For this reason, for each user, a *PoI tracking percentage* has been computed. This way, it is possible to measure the portion of video playback time spent watching at least one PoI. The PoI tracking percentage T has been computed as follows:

$$T = 100 \cdot \frac{\Delta t_{PoI}}{T_{total}} = 100 \cdot \frac{N_{S_{PoI}}}{N_{total}} \quad (1)$$

where Δt_{PoI} is the time spent by a user watching at least one PoI while T_{total} is the total duration of the video. Since in the list we built the video playback time is sampled every Δt seconds ($\Delta t = 0.03$s), the same percentage is obtained by dividing $N_{S_{PoI}}$, i.e., the number of samples in which at least one PoI is watched, by N_{total}, i.e., the total number of samples. Fig. 2 shows the average PoI tracking percentage for each of the 5 videos considered. Overall, it is possible to state that, on average, users spend 81.44% of the video playback time watching a PoI.

As a matter of fact, it is also important to understand when, at least on average, users explore the scene and when they follow a PoI. To this purpose, in Fig. 3 we show the average

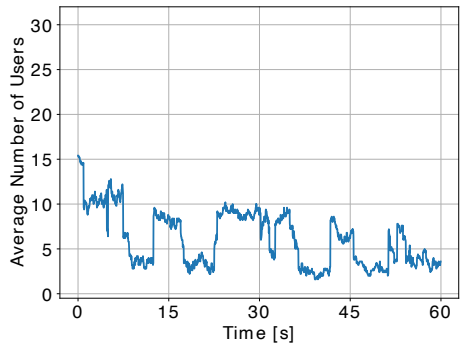

Fig. 3: Average number of users in exploration phase

number of users who do not watch any PoI, i.e., those who are in the exploration phase, for each time sample. From the figure, it is immediate to conclude that the trend of users in the exploration phase is in general decreasing over time. Hence, we can deduce that a user generally tends to explore the scene at the beginning of the video until they find a PoI that attracts their attention. From then on, the user is supposed to always follow a PoI up to some scarce moments of new exploration.

III. PoI PREDICTION

In this Section, we provide some insights that are useful to define an effective viewport prediction strategy. The algorithm used in our tests employs a specific kind of neural network called Long Short-Term Memory (LSTM) [21].

LSTMs are Recurrent Neural Networks (RNNs) that are able to process entire sequences of data, thus being ideal for data prediction. In our case, on the basis of a certain number of behaviours associated with users, the LSTM predicts which PoI–not the exact point–the current user is likely to watch in the following n seconds. Obviously, as n increases, the prediction accuracy is expected to plummet. It is therefore important to tune n in order to get the best trade-off between accuracy and prediction.

To the best of our knowledge, the technique currently adopted to predict future viewports is the *naive* approach, i.e., a technique whose prediction is simply represented by the last viewport observed by the user, thus supposing they do not move for the next n seconds [22]. Such an approach performs more or less well depending on the video content. For this reason, we advocate a hybrid approach that combines both LSTM networks and the naive approach. As a first step, we have considered the following procedure: LSTM networks are used when their predictions on the scenes just watched by the user have an accuracy that is higher than a certain threshold. Otherwise, the naive approach is adopted. Another important issue is that the aforementioned approaches have to be adapted to the standards adopted in the video streaming field, in particular to MPEG-DASH and Adaprive BitRate (ABR) algorithms. MPEG-DASH standard divides the video into fixed-length segments (or chunks) to enhance the transmission. Then, these chunks are encoded at different resolutions, called levels. On the user's side, the ABR algorithm decides, according to the available bandwidth, the most suitable level to download to avoid video playback stalls. Therefore, the proposed approach should indicate to the ABR algorithm which tiles need to have a higher resolution wrt the others. Obviously, the resolution levels of the tiles depend on the available bandwidth, which is unpredictable. However, one could measure the visual quality experienced by the user by means of a full-reference video quality assessment tool such as the Structural SIMilarity (SSIM) or the Video Multi-method Assessment Fusion (VMAF). These tools estimate the visual quality by comparing each frame of the *degraded* video with the reference frames of the non-degraded video.

In the experiments, we have employed a certain number of LSTM layers and a *dense* layer. This architecture receives as an input a temporal series of k elements and outputs the following m, which represent the prediction. Reminding that n is the prediction in terms of seconds, it results that $n = m \cdot \Delta t$. Notice that input data contain the past k PoIs watched by the user. As a consequence, there are as many temporal series as the number of users in the dataset. A part of such series composes the *training set* and is divided into fixed-length vectors, which represent the input to the LSTM. During the training phase, on the basis of each input vector, the network tries to predict the PoI observed by the user in future instants. Such predictions are then compared to the actual PoIs to compute the *loss* function. Through an *adam* optimiser, the weights of the networks are updated at each iteration in order to minimise this function. When the training phase ends, the testing phase starts, during which the remaining part of the temporal series is fed in input to the network. This time the weights do not change since the goal is to evaluate the performance of the trained network.

To carry out the first tests, we have considered the dataset illustrated in Section II. A part of it has been used as a training set, whereas the rest forms the test set. We have considered a chunk duration of 1s and we have fixed the number of input chunks to the LSTM to be equal to 25 with a prediction length set to 3 ($m = 3$). Depending on the video analysed, it is necessary to tune the number of layers, neurons, and hyperparameters in general. It is important that the network is also able to predict the case in which a user will be in the exploration phase. This way, it is possible to download the whole frame at a lower resolution since the user's eyes are moving and thus they do not have enough time to focus on a particular point of the frame.

Let us now describe in more detail the procedure adopted for the download of the chunks. The ABR algorithm running at the user's device is designed to avoid video playback stalls, i.e., situations in which the video playout buffer used to store video chunks gets completely depleted. This means that the buffer must contain at least 1 chunk. Therefore, supposing that we set the number of chunks in the buffer to be exactly equal to one to guarantee a more reactive quality adaptation, for the first 25 seconds, the naive approach is employed because the LSTM framework requires 25 chunks as an input sequence, which are not available yet. After 25 seconds, the LSTM framework receives the first 25 chunks of video and predicts the next 3 chunks, i.e., the 26^{th}, 27^{th}, and 28^{th} chunk. Notice that in this moment the user is watching the 26^{th} chunk and the playout buffer contains the 27^{th}. When the playback of the 26^{th} chunk ends, the viewport trajectory

of this chunk is compared to the trajectory of the first chunk predicted by the LSTM framework. In particular, we observe how many times the prediction failed. If the number of failures does not exceed a certain threshold, the 28^{th} chunk predicted by the LSTM framework is downloaded while the user watches the 27^{th} chunk that was already present in the buffer. Otherwise, it is the naive approach to provide the 28^{th} chunk, which will contain just the last viewport observed by the user for the whole duration of the chunk. At this point, the user watches the 27^{th} chunk while the 28^{th} is in the buffer. The LSTM framework receives input chunks from 2 to 26 and the procedure iterates.

For the first experimental simulations, we have considered a quite complicated video, containing 4 PoIs (plus the additional case of the exploration). We have set up a network with 3 LSTM layers, each of which has 50 neurons, and a final dense layer. Preliminary results show that the proposed hybrid approach presents a prediction accuracy of about 60% whereas the naive approach ensures about 58% accuracy. Despite the slight improvements, we have reason to believe that an enhancement in the design of the proposed LSTM-based framework could lead to relevant outcomes. To this end, a system that optimises the hyperparametes of LSTM networks is needed to improve the prediction accuracy. A framework based on Optuna [23], i.e. a tool that tunes the hyperparameters of a network to maximise an objective function, is currently being investigated.

IV. Conclusions

In this short paper, we have shown that users watching omnidirectional videos, after an initial phase in which they explore the environment, tend to follow one of the Points of Interest present in the video, i.e., the regions in the frame that are more interesting. Then, we have proposed to employ LSTM networks to make predictions on the viewport the user is likely to watch in the immediate future, based on the PoIs. The approach described shows promising results and, with further studies and tuning, could significantly improve those currently adopted.

V. Acknowledgements

This work was partially supported by the Pon Miur research contract Agreed (ARS01_00254) and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

References

[1] S. Dambra, G. Samela, L. Sassatelli, R. Pighetti, R. Aparicio-Pardo, and A.-M. Pinna-Déry, "Film editing: New levers to improve vr streaming," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 27–39.

[2] F. Pallavicini, A. Pepe, and M. E. Minissi, "Gaming in virtual reality: What changes in terms of usability, emotional response and sense of presence compared to non-immersive video games?" *Simulation & Gaming*, vol. 50, no. 2, pp. 136–159, 2019.

[3] D. W. Carruth, C. Hudson, A. A. Fox, and S. Deb, "User interface for an immersive virtual reality greenhouse for training precision agriculture," in *Virtual, Augmented and Mixed Reality. Industrial and Everyday Life Applications: 12th International Conference, VAMR 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*. Springer, 2020, pp. 35–46.

[4] M. Zink, R. Sitaraman, and K. Nahrstedt, "Scalable 360 video stream delivery: Challenges, solutions, and opportunities," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 639–650, 2019.

[5] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "Vr is on the edge: How to deliver 360 videos in mobile networks," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, 2017, pp. 30–35.

[6] G. Ribezzo, G. Samela, V. Palmisano, L. De Cicco, and S. Mascolo, "A dash video streaming system for immersive contents," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 525–528.

[7] "Facebook developers, next-generation video encoding techniques for 360 video and vr," in *https://code.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/, online*, 2016.

[8] 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 *2016 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2016, pp. 583–586.

[9] "Facebook developers, enhancing high-resolution 360 streaming with view prediction," in *https://code.fb.com/virtual-reality/enhancing-high-resolution-360-streaming-with-view-prediction/, online*, 2017.

[10] C. Concolato, J. Le Feuvre, F. Denoual, F. Mazé, E. Nassor, N. Ouedraogo, and J. Taquet, "Adaptive streaming of hevc tiled videos using mpeg-dash," *IEEE transactions on circuits and systems for video technology*, vol. 28, no. 8, pp. 1981–1992, 2017.

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

[12] J. Vielhaben, H. Camalan, W. Samek, and M. Wenzel, "Viewport forecasting in 360 virtual reality videos with machine learning," in *2019 IEEE international conference on artificial intelligence and virtual reality (AIVR)*. IEEE, 2019, pp. 74–747.

[13] N. Kan, J. Zou, C. Li, W. Dai, and H. Xiong, "Rapt360: Reinforcement learning-based rate adaptation for 360-degree video streaming with adaptive prediction and tiling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1607–1623, 2021.

[14] S. Park, A. Bhattacharya, Z. Yang, M. Dasari, S. R. Das, and D. Samaras, "Advancing user quality of experience in 360-degree video streaming," in *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 2019, pp. 1–9.

[15] I. Agtzidis, M. Startsev, and M. Dorr, "360-degree video gaze behaviour: A ground-truth data set and a classification algorithm for eye movements," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1007–1015.

[16] S. Fremerey, A. Singla, K. Meseberg, and A. Raake, "Avtrack360: An open dataset and software recording people's head rotations watching 360° videos on an hmd," in *Proceedings of the 9th ACM multimedia systems conference*, 2018, pp. 403–408.

[17] 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*, 2017, pp. 211–216.

[18] A. T. Nasrabadi, A. Samiei, A. Mahzari, R. P. McMahan, R. Prakash, M. C. Farias, and M. M. Carvalho, "A taxonomy and dataset for 360 videos," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 273–278.

[19] S. Rossi, C. Ozcinar, A. Smolic, and L. Toni, "Do users behave similarly in vr? investigation of the user influence on the system design," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2, pp. 1–26, 2020.

[20] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 193–198.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] 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 *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1161–1170.

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.