EAI Endorsed Transactions
on Industrial Networks and Intelligent Systems
Research Article EAI.EU

An Accurate Viewport Estimation Method for 360 Video Streaming using Deep Learning

Nguyen Viet Hung[1,3], Dao Thu Ngan[1], Pham Ngoc Son[3], Dang Tran Long[2], Nguyen Trung Dung[3], Truong Thu Huong[3,*]

[1]East Asia University of Technology, Vietnam
[2]University of Greenwich, VietNam
[3]School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Vietnam

Abstract

Nowadays, Virtual Reality is becoming more and more popular, and 360 video is a very important part of the system. 360 video transmission over the Internet faces many difficulties due to its large size. Therefore, to reduce the network bandwidth requirement of 360-degree video, Viewport Adaptive Streaming (VAS) was proposed. An important issue in VAS is how to estimate future user viewing direction. In this paper, we propose an algorithm called GLVP (GRU-LSTM-based-Viewport-Prediction) to estimate the typical view for the VAS system. The results show that our method can improve viewport estimation from 9.5% to near 20% compared with other methods.

Received on 28 July 2022; accepted on 18 September 2022; published on 21 September 2022

Keywords: Video Streaming, 360-degree Video, Quality of Experience, VR, Deep Learning.

doi:10.4108/eetinis.v9i4.2218

1. Introduction

According to various surveys, 360-degree video has become increasingly popular in recent years, which drew our attention [1]. Because the network capacity required for a 360-degree video is much more than that of an ordinary video, streaming 360-degree video is a major challenge today. Furthermore, the latency in video streaming necessitates extremely low requirements in order to match real-time applications. To reduce video capacity while maintaining good user experience quality, numerous 360-degree video streaming methods are recommended. Among of which, Adaptive viewport streaming is one of the most used ways nowadays. In this method, 360-degree videos will be broken into tiles with varying weights to be delivered in a transmission medium. As illustrated in [2], the VAS system's quality will suffer significantly as a result of the incorrect view prediction in [3], [4], [5], [6]. As a result, viewport prediction is an essential need of the VAS system.

Because estimating the viewport helps to divide the weights among the most likely tiles. We can thus reduce the capacity of tiles that users ignore while retaining the quality of tiles that they do pay attention to. To complete this viewport prediction with great accuracy is required for a VAS system.

Recently, there have been numerous suggested viewport prediction algorithms. However, because these approaches used different input data sets and parameters, comparing their quality is difficult. Some metrics used to evaluate the quality of viewport prediction systems include the root-mean-square error (RMSE) [4], the number of tiles lost [5], and the percentage of viewport black area [6]. The preceding methods have an unsolved difficulty in that the authors only measured for a short period of time, primarily near the beginning of the video because users prefer to vary the angle of the entire video, it is not possible to predict the viewport for an entire video. We use a Figure 1 to visually represent the change in the user's perspective and the difference between the predicted viewport and the viewport seen by the user.

In this paper, we assess and compare common viewport prediction algorithms for 360 video viewport

*Corresponding author's email: huong.truongthu@hust.edu.vn

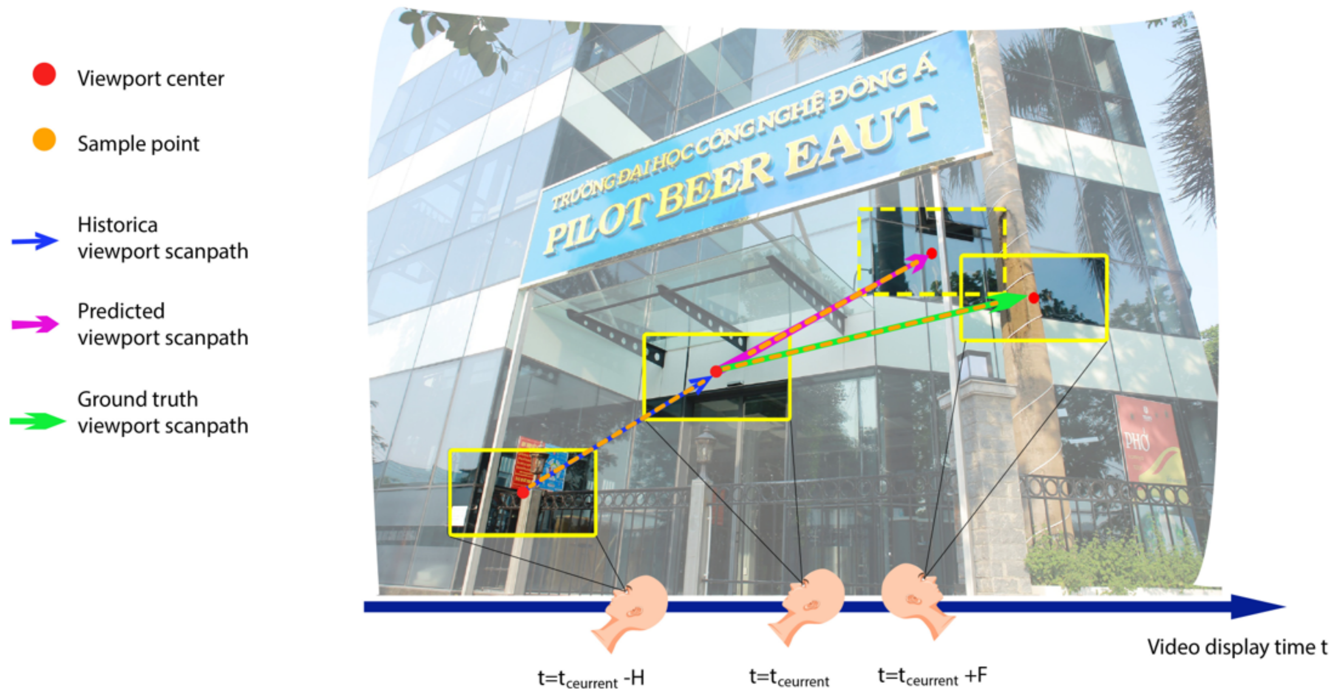


Figure 1. The GLVP model compares the viewport sweep in the past H-second to predict the viewport sweep in the future F seconds.

adaptive streaming. Prediction performance is examined not only in terms of accuracy but also in terms of redundancy to shed light on the actual performance of present models such as Last [3], Linear [6], LSTM [5] and GRU[7].

The rest of the paper is described as follows: Section 2 discusses related work. Section 3 elaborates the proposed method named GLVP. Section 4 contains the performance evaluation. Finally, section 5 concludes our findings and future work.

2. Related work

Viewport Adaptive Streaming has been proposed in [8], [9], [10], [11] to cope with the high bitrate difficulty of 360 video. The concept of VAS is to transmit high-quality video segments visible to users (i.e., viewport) and lower-quality video parts to the rest of the video [8]. The majority of previous research [1] used a tiling-based VAS method, in which the entire 360 video is spatially separated into tiny portions called tiles, each of which is encoded into numerous versions of varying quality. High-quality versions of the tiles that overlap the user viewport are chosen. Low-quality versions, on the other hand, are picked for tiles beyond the user viewport [11].

The so-called viewport predictor, which anticipates where the user will gaze in the future [11], is a critical component of Viewport Adaptive Streaming. Because of its simplicity, early publications [9],[3], [11], [12], [13] used linear regression and its modifications (e.g., Weighted Linear Regression) to predict viewport location. Recent researches have used neural networks to forecast viewports. Specially, Long-Short Term Memory (LSTM) [5], [14], Gated Recurrent Network (GRU) [7], and other Recurrent Neural Networks (RNN) has gotten a lot of attention. Furthermore, probabilistic models like Gaussian Mixture [15] and Reinforcement learning algorithms [16] like Contextual Bandit were applied. In [15], the authors proposed a hybrid user and video evidenced viewport prediction method to reduce bandwidth consumption in live mobile VR streaming, the article [15] differs from our original goal. In [16], the authors proposed a viewport prediction algorithm and ran it on a testbed for streaming video, but the data in that article was based on experiments when showing users 360-degree videos, so the dataset in [16] was different from our dataset. However, all of the above solutions still have limited performance in terms of accuracy, in this paper, we propose a new model to improve the viewport prediction accuracy.

Furthermore, several other approaches for viewport prediction are mentioned in [17], [18], and [19]. The authors in [17] proposed employing FoV prediction and caching to create a live streaming system for 360-degree videos. Besides, in [18], the authors developed a clustering-based viewport prediction algorithm that uses viewport pattern data from prior video streaming sessions. Nevertheless, this method [18] is very dependent on the video content. In [19], the authors extracted video semantic information, in which deep learning-based video analysis requires powerful processing resources and vast memory space. Whereas, most client devices, such as small mobile devices or Head Mounted Displays (HMD) have limited computing and memory resource. In general, the 3 aforementioned studies are based on fixed context and consume a lot of memory. Meanwhile, our solution automatically adapts to the head movement, self-learning through the training process, and is capable of removing unnecessary memory areas, thus consuming less memory.

3. Proposed viewport estimation method

3.1. Problem Formulation

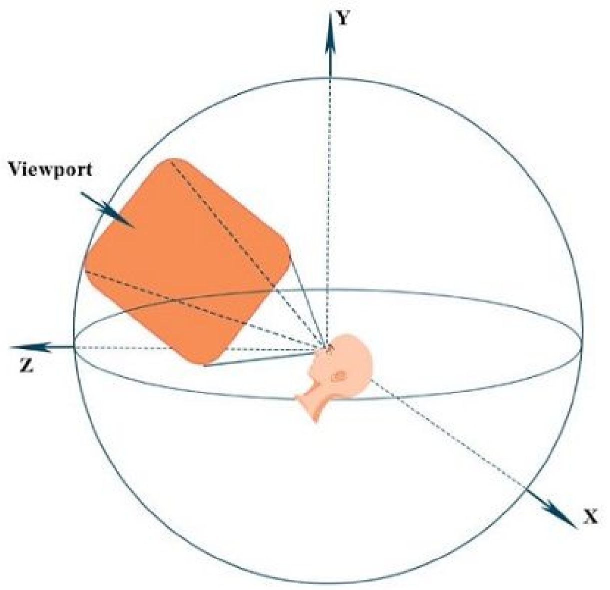


Figure 2. The viewport of a user at a time

Let $P(t_0)$ be the position of the viewport at the time t_0. The longitude and latitude values of a viewport can be used to specify the position of the viewport's center point [1]. In Figure 2, we can see that spherical video captures a 360-degree view of a scene. It is the main content type in Virtual Reality, providing an "immersive" viewing experience. Viewport is the video area that a user can see at a given instant because of the human Field of View.

The viewport predictor's job is to predict the location of the viewport $P(t_0 + m)$ in the future t_0. The forecast horizon is denoted by the letter m. Because 360-video streaming is typically done on a segment/adaptation interval basis [20], the predictor must provide a prediction for the interval $[t_0 + m, t_0 + m + s]$, where s denotes the segment duration, as shown in Figure 3.

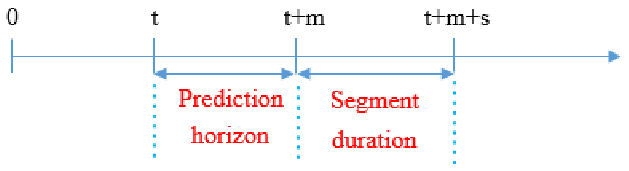


Figure 3. Problem formulation of Viewport Prediction

3.2. Design of viewport prediction and selection

Our proposed position prediction is based on a hybrid of LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) algorithms, which is called GLVP standing for **G**RU-**L**STM-based-**V**iewport-**P**rediction. LSTM can estimate the long-term correlation in the data, thereby being able to model a longer term trend. As the result, LSTM can potentially provide more accurate viewport prediction. However, LSTM requires a rather long initial processing time. Therefore, we design a GRU block in front of LSTM to speed up input data processing and, as a result, improving accuracy in the initial seconds when compared to an algorithm that solely uses LSTM. Figure 4 shows the architecture of a cell in our proposed technique.

The GLVP model includes n inputs corresponding to n frames/images of a video: $\{x_1, x_2, ...x_t, x_n\}$. M_{t-1}, a_{t-1} are the cell state and hidden cell state at time $t-1$. While, M_t, a_t are cell state and hidden cell state at time t. In our design, a_t represent for the selection of the predicted viewport, and M_t is the data that will be used as input for the next cell. The gates used in the cell are defined as follows:

- **Forget gate** - q_t: to remove unnecessary information from the current cell.
- **Input gate** - v_t, $\tilde{M}_t$: to select important information to be used in the current cell.
- **Reset gate** - r_t, n_t: to control how much of the previous state is retained.
- **Output gate** - M_t, d_t, a_t: to determine what information from the current cell is used as output data.

The operation of the whole model is described step by step as follows:

Step 1:
In the first step, q_t will determine which information from Input x_t and Hidden Cell State a_{t-1} should be removed and eliminated.

$$q_t = \sigma(U_q \otimes x_t + W_q \otimes a_{t-1} + b_q) \tag{1}$$

where:

q_t: data filter port from the output value of time step

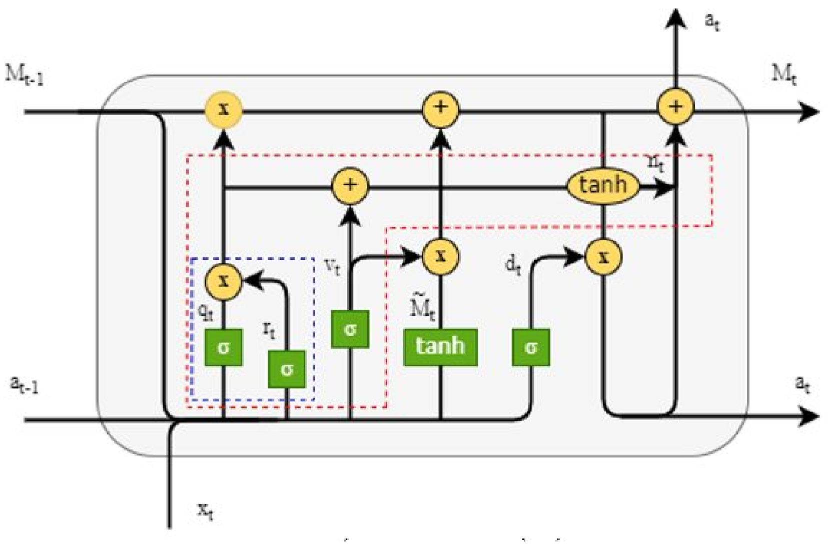
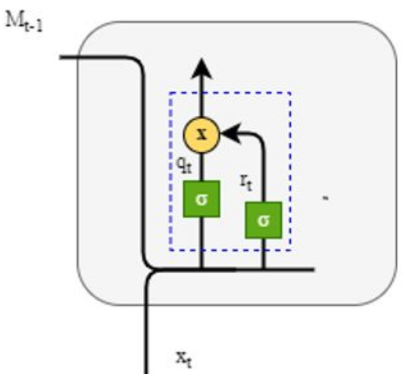
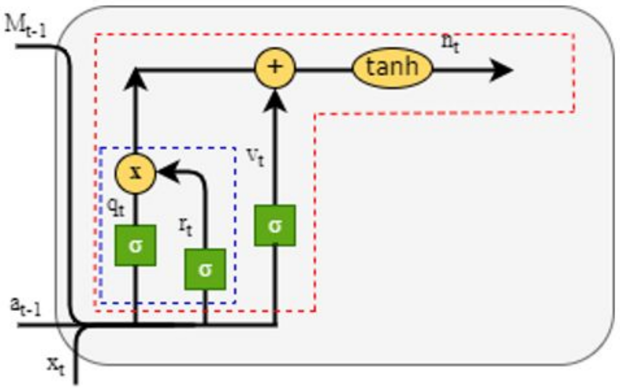


Figure 4. GLVP model for viewport estimation



(a) Step 01: Add r_t port



(b) Step 02: Add n_t port

Figure 5. Operation of the Reset port

$(t-1)$

U_q, W_q: corresponding weight matrix of the forget gate.

b_q: vector bias corresponding to the forget gate.

x_t: vector input at each step time t.

a_{t-1}: output of the cell at the previous time step $(t-1)$.

σ: The sigmoid function to transform the value q_t to the range of $(0, 1)$.

$$q_t = \begin{cases} 1 & \text{Completely remembered} \\ 0 & \text{Completely forgotten} \end{cases}$$

Step 2:
In the second step, input data will not be taken entirely from the input vector x_t and output from the previous cell a_{t-1}, so it is necessary to select what information should be used to calculate M_t. In this step, we create an input gate $(\tilde{M}_t, v_t)$ to select important information to be used in the current cell.

$$\tilde{M}_t = \tanh(U_M * x_t + W_M * a_{t-1} + b_M) \tag{2}$$

$$v_t = \sigma(U_v * x_t + W_v * a_{t-1} + b_v) \tag{3}$$

The parameters U_M, W_M, b_M, U_v, W_v, b_v are similar to the formula (1). The activation function is the tanh function used to change the value to the range of (-1,1).

Step 3:
The operation of Reset gate is illustrated in Figure 5, in which, instead of just creating one q_t like in GRU, we need an additional port r_t to ensure that the effect of the previous hidden state is best reduced. Also, to reduce the effect of the previous hidden state, we added a new potential hidden state n_t (as seen in Figure 5b).

Algorithm 1: *Viewport Estimation*

```
   Input: $q_t$, $v_t$, $r_t$, $d_t$, $\tilde{M}_t$
   Output: $\{M_t, a_t\}$
1  for $t = 1$ to $N$ do
2      Calculate v$_t$ $=\sigma(v_t)$
3      Calculate q$_t$ $=\sigma(q_t)$
4      Calculate $\tilde{M}_t = \tanh(\tilde{M}_t)$
5      Calculate d$_t$ $=\sigma(d_t)$
6      Calculate r$_t$ $=\sigma(v_t + M_{t-1})$
7      Calculate n$_t$ $=\tanh(v_t + (r_t \otimes q_t))$
8      $M_t = ((r_t * q_t) \otimes M_{t-1}) + (v_t \otimes M_t)$
9      $a_t = n_t + d_t * \tanh(M_t)$
10 end
11 return $M_t$, $a_t$
```

By adding these two new ports, we can improve the strength of both the GRU and LSTM algorithms. The Reset gate (r_t, n_t) is designed with the following formulae:

$$r_t = \sigma(U_r * x_t + W_r * M_{t-1} + b_r) \tag{4}$$

$$n_t = \tanh(U_n * x_t + W_n * (r_t * q_t) + b_n) \tag{5}$$

Step 4:
In the fourth step, the cell state at current time t is calculated based on the results obtained from (1), (2), (3), and (4), as follows:

$$M_t = ((r_t * q_t) \otimes M_{t-1}) + (v_t \otimes \tilde{M}_t) \tag{6}$$

Step 5:
In the final step, output value a_t of the proposed cell is calculated as follows:

$$a_t = n_t + d_t * \tanh(M_t) \tag{7}$$

Where:

$$d_t = \sigma(U_d * x_t + W_d * a_{t-1} + b_d) \tag{8}$$

Variable d_t decides how much information to get from memory port M_t, combining with port n_t to calculate the hidden state at time t - a_t.

The whole process of estimating viewports can be summarized in short in the Pseudo code illustrated in Algorithm 1

4. Performance Evaluation

4.1. Experimental Settings

In our experiment, we used 360 videos, specifically the diving videos, and head motion traces for our experiment. The head movement traces and videos were obtained from the dataset [21] because the viewport positions are given as a quaternary. The viewport positions are transformed into longitude and latitude values in our implementation to reduce computing complexity. For each of the evaluated approaches, longitude and latitude are computed independently, then blended for the final evaluation. We use the input data set of viewport locations shown in the Figure 6.

Figure 6 shows that viewport position #1 differs from viewport position #2. Viewport position #1's longitude and latitude are more volatile than viewport position #2's longitude and latitude. Because of the variations between the two viewport positions, we will evaluate algorithms with numerous distinct perspective adjustments that will be good. In our evaluation, we compare GLVP with the other approaches such as LAST [3], LINEAR [6], LSTM [5] and GRU[7] under the context of tiling-based VAS [9] in the first 6 seconds. Because it has been found in our experiment that in the following seconds, the accuracy of all methods is almost the same; the difference only appears in the first 6 seconds.

Let *total* denote the current set of visible tiles at time *t*. $total^e$ stands for the estimated visible tile set. In our experiment, the Accuracy metric is used in our performance evaluation.

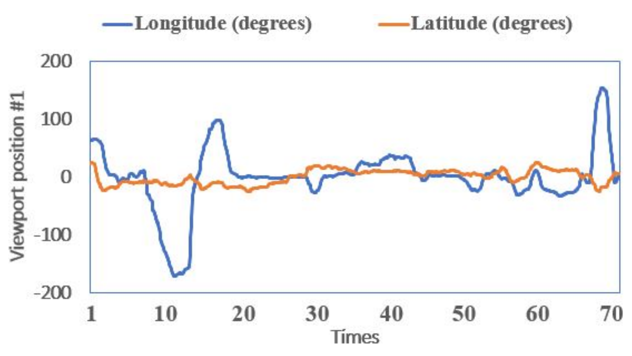
Accuracy [22]: The rate of accurately-calculated visible tiles to the total number of visible tiles.

$$\Phi = \frac{|total \cap total^e|}{|total|} \tag{9}$$

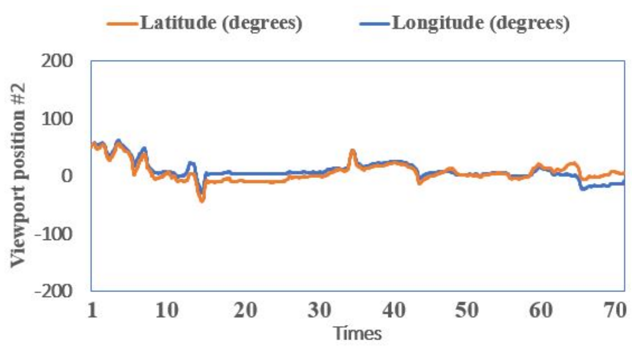
Since the purpose of this forecast is to drastically reduce the capacity rather than consume it as usual, the accuracy is the important metric because it is a measure to dramatically reduce the total number of visible tiles, bringing better results for future users.

4.2. Estimation performance evaluation

As can be seen in Figure 7, GLVP is always the most accurate among all solutions in all 6 seconds. From the third second to the sixth second, the accuracy of all algorithms is larger than 80% and the redundancy of all algorithms is larger than 80%. In all 6 seconds, only the GLVP algorithm has the accuracy over 80%. The LSTM algorithm yields a low accuracy of only approximately 20% at the first second because LSTM takes an early amount of time to process the data, therefore it is not highly accurate in the first second. From the third second, LSTM has higher accuracy at

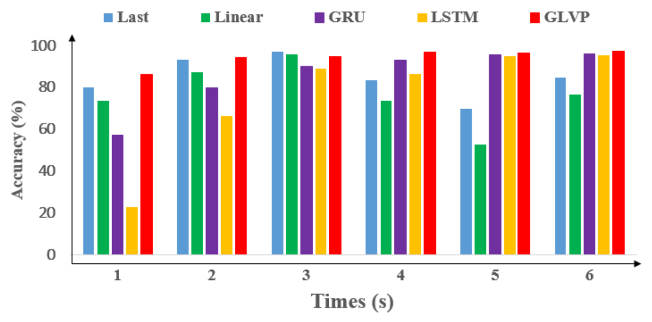


(a) Viewport position #1 over the time

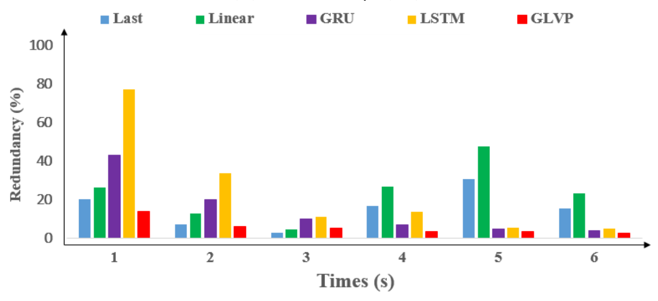


(b) Viewport position #2 over the time

Figure 6. Viewport position #1, and #2 over the time



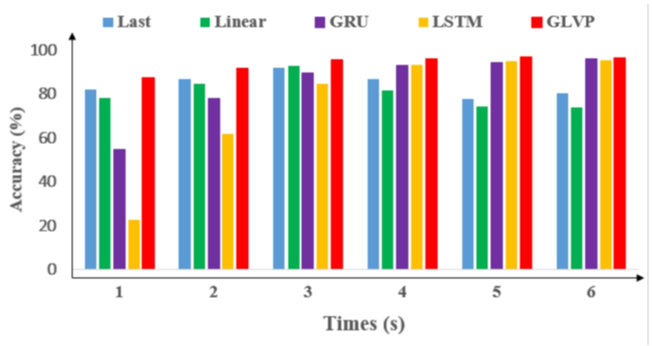
(a) Accuracy (%)



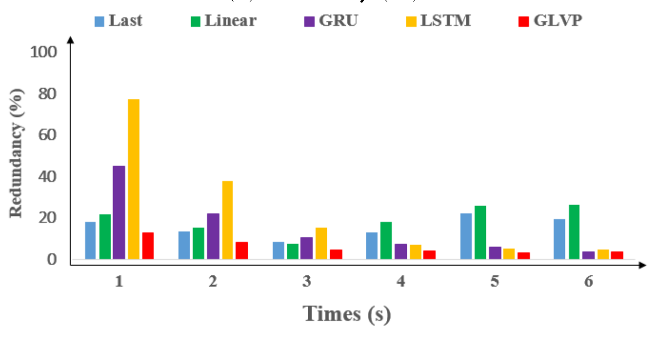
(b) Redundancy (%)

Figure 7. Viewport estimated the performance of the methods reviewed at each early motion trace of Viewport position #1

over 80%. While LAST has a high accuracy range from 70% to 90%. However, LAST is not steady for the entire 6 seconds because LAST only relies on the previous viewport for prediction. Therefore, the solution will not be highly accurate in case the viewer's viewport changes a lot. In addition, in all 6 seconds, LINEAR gets the accuracy of between 60% and 85%. It is interesting to see that LINEAR is always less accurate than LAST in the whole 6 seconds. Because the LAST technique merely estimates the viewport location based on the last viewport position. Besides, the LINEAR approach, previous viewport position data is fitted to a linear function in order to reduce the root mean squared error. The accuracy of the GRU approach ranges from 50% to 95%. GRU is less accurate than LAST and LINEAR from the first second and greater than of LAST and LINEAR in the next three seconds. It can be explained that because the GRU algorithm takes the first amount of time to process the data, there is no high accuracy in the first seconds. But from the next seconds, the GRU gives good results.



(a) Accuracy (%)



(b) Redundancy (%)

Figure 8. Viewport estimated the performance of the methods reviewed at each early motion trace of Viewport position #2

In another case study viewport position #2 as presented in Figure 8, GLVP also outperforms LAST, LINEAR, LSTM, and GRU. That proves that GLVP works well in different cases of viewport positions.

Table 1. Performance of the GLVP and reference methods under viewport positions #1 and #2.

Metrics		GLVP	Last	Linear	GRU	LSTM
Viewport Position #1	Accuracy (%)	94.28	84.64	76.58	85.33	75.76
	Redundancy (%)	5.72	15.36	23.42	14.67	24.24
Viewport Position #2	Accuracy (%)	94.04	84.38	80.93	84.38	75.51
	Redundancy (%)	5.96	15.62	19.07	15.62	24.49

To have a better overview on quantitative evaluation, TABLE 1 summarizes the performance of our proposed GLVP and the other algorithms such as LAST, LINEAR, LSTM and GRU. For viewport position #1, the accuracy of GLVP is higher than the accuracy of LAST, LINEAR, GRU, and LSTM by 10.23%, 18.78%, 9.50%, and 19.65%, respectively. For viewport position #2, GLVP outperforms LAST, LINEAR, LSTM and GRU in terms of accuracy by 10.27%, 13.93%, 10.27%, and 19.70%, respectively.

From another aspect, we also show the redundancy landscape of those solutions. The results show that GLVP has a smaller redundancy than any of the other solutions. Redundancy has the purpose of reducing the impact of viewport prediction errors on user experience.

4.3. Training time evaluation

Besides investigating the estimation performance of GLVP, we also study the training time of the GLVP model in comparision with the other existing solutions. To measure this parameter, a Python-written experiment was developed on a machine running 64-bit Windows 10 with 16384 MB of RAM and an Intel(R) Core(TM) i7-6500U CPU running at 2.50GHz (4 CPUs), 2.6GHz processor. As Figure 9 shows the training time

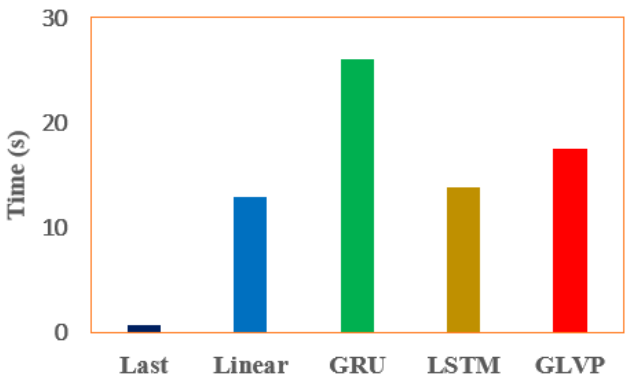


Figure 9. Training time overview

in the second time scale. GRU, LSTM, LINEAR, and our proposed GLVP take the training time of more than 10s, whilst LAST takes the least time. However, this is the time for training. Once the training process is finished, and the learning model is exported, the inference time of this estimation process will be very fast in real time. It is assumed that we just need to update and re-trainign the learning model periodically since data pattern of such an application is not changing quicky over time.

5. Conclusion

In this research, we have proposed a new method called GLVP to predict viewport position, thereby selecting the most appropriate viewports. The solution has been shown to outperform 4 current existing estimation methods in different scenarios. In the future, we will concentrate on improving the proposed method by incorporating more content-based data.

Acknowledgment

Thanks Ms. Truong Thi Diu - East Asia University of Technology and Mr. Nguyen Tai Minh - Amsterdam High School for the Gifted for supporting us in emulating the VR transmission system.

References

[1] D. V. Nguyen, H. T. T. Tran, and T. C. Thang, "An evaluation of tile selection methods for viewport-adaptive streaming of 360-degree video," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 16, no. 1, mar 2020. [Online]. Available: https://doi.org/10.1145/3373359

[2] D. V. Nguyen, H. T. Tran, and T. C. Thang, "Impact of delays on 360-degree video communications," in *2017 TRON Symposium (TRONSHOW)*. IEEE, 2017, pp. 1–6.

[3] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ser. ATC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1–6. [Online]. Available: https://doi.org/10.1145/2980055.2980056

[4] 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)*, 2016, pp. 1161–1170.

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

[6] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang, ": Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[7] C. Wu, R. Zhang, Z. Wang, and L. Sun, "A spherical convolution approach for learning long term viewport prediction in 360 immersive video," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 14 003–14 040.

[8] M. Hosseini and V. Swaminathan, "Adaptive 360 vr video streaming: Divide and conquer," in *2016 IEEE International Symposium on Multimedia (ISM)*, 2016, pp. 107–110.

[9] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "A new adaptation approach for viewport-adaptive 360-degree video streaming," in *2017 IEEE International Symposium on Multimedia (ISM)*, 2017, pp. 38–44.

[10] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proceedings of the 24th ACM International Conference on Multimedia*, ser. MM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 601–605. [Online]. Available: https://doi.org/10.1145/2964284.2967292

[11] D. V. Nguyen, H. T. T. Tran, A. T. Pham, and T. C. Thang, "An optimal tile-based approach for viewport-adaptive 360-degree video streaming," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 29–42, 2019.

[12] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An http/2-based adaptive streaming framework for 360° virtual reality videos," in *Proceedings of the 25th ACM International Conference on Multimedia*, ser. MM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 306–314. [Online]. Available: https://doi.org/10.1145/3123266.3123453

[13] Z. Xu, X. Zhang, K. Zhang, and Z. Guo, "Probabilistic viewport adaptive streaming for 360-degree videos," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[14] C. Li, W. Zhang, Y. Liu, and Y. Wang, "Very long term field of view prediction for 360-degree video streaming," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2019, pp. 297–302.

[15] X. Feng, V. Swaminathan, and S. Wei, "Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 2, jun 2019. [Online]. Available: https://doi.org/10.1145/3328914

[16] J. Heyse, M. T. Vega, F. de Backere, and F. de Turck, "Contextual bandit learning-based viewport prediction for 360 video," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 972–973.

[17] L. Sun, Y. Mao, T. Zong, Y. Liu, and Y. Wang, "Flocking-based live streaming of 360-degree video," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 26–37. [Online]. Available: https://doi.org/10.1145/3339825.3391856

[18] A. T. Nasrabadi, A. Samiei, and R. Prakash, "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*, ser. NOSSDAV '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 34–39. [Online]. Available: https://doi.org/10.1145/3386290.3396934

[19] J. Park, M. Wu, K.-Y. Lee, B. Chen, K. Nahrstedt, M. Zink, and R. Sitaraman, "Seaware: Semantic aware view prediction system for 360-degree video streaming," in *2020 IEEE International Symposium on Multimedia (ISM)*, 2020, pp. 57–64.

[20] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro, "An evaluation of bitrate adaptation methods for http live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 693–705, April 2014.

[21] X. Corbillon, F. De Simone, and G. Simon, "360-degree video head movement dataset," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 199–204. [Online]. Available: https://doi.org/10.1145/3083187.3083215

[22] D. Nguyen, "An evaluation of viewport estimation methods in 360-degree video streaming," in *2022 7th International Conference on Business and Industrial Research (ICBIR)*, 2022, pp. 161–166.