SINGLE AND SEQUENTIAL VIEWPORTS PREDICTION FOR 360-DEGREE VIDEO STREAMING

Qin Yang, Junni Zou, Kexin Tang, Chenglin Li, Hongkai Xiong
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China
Email: {yangqin, zoujunni, tkx1994-china, lcl1985, xionghongkai}@sjtu.edu.cn

***Abstract*—Sending only the viewport of interest provides a solution for 360-degree video streaming under the current bandwidth-constrained infrastructure. To this end, the user viewport requires to be prefetched in advance by conducting viewport prediction. To more accurately capture the nonlinear and long-term dependent relation between the future and past viewports, we develop a single viewport prediction model using convolutional neural network (CNN), in which the pooling layers are dropped and more convolutional layers are added for stronger nonlinear fitting ability. Further, we design a viewport trajectory prediction model based on recurrent neural network (RNN) which learns long-term dependency in sequential viewports. Specially, it is capable to estimate future viewport trajectory and support variable-size prediction window with low complexity. Finally, a correlation filter-based viewport tracker (CFVT) is proposed to perform content-aware viewport prediction. The combination of the RNN and the CFVT through a fusion model enables them to complement each other which is validated by significant improvement in prediction accuracy.**

I. INTRODUCTION

In recent years, watching 360-degree videos with interactive displaying systems, such as head-mounted display (HMD) has become increasingly popular. Compared to traditional videos, 360-degree videos provide users with a panoramic scene captured by an omnidirectional camera. When watching 360-degree videos, the user can obtain immersive experience by freely adjusting his viewing orientation. However, 360-degree videos have huge file size and ultra high resolution, and the delivery of a 360-degree video consumes up to six times the bandwidth of a traditional video. For current bandwidth-constrained infrastructure, in particular for mobile networks, it is hard to send the entire 360-degree video to the users. Therefore, how to efficiently utilize the network bandwidth and provide immersive experience guarantees is challenging for 360-degree video streaming.

In practice, constrained by the field of view (FoV) of the HMD, the user at any time can view a small portion of the video content, also called the *viewport* of the user. Thus, streaming only the viewport of interest and switching it in terms of the head motion of the user will be a more efficient bandwidth utilization. As the motion-to-photon latency required for VR applications is 15-20 ms for a better experience, which is much smaller than the round trip time (RTT) between a viewing device and its nearby server, usually 30-50 ms [1]. Therefore, the user viewport requires to be prefetched in advance by conducting viewport prediction. Specifically, viewport prediction refers to predicting the user's future viewport, e.g. the viewport at time $t+t_w$, on the basis of his current and past viewports. Here, t_w represents the size of the prediction window.

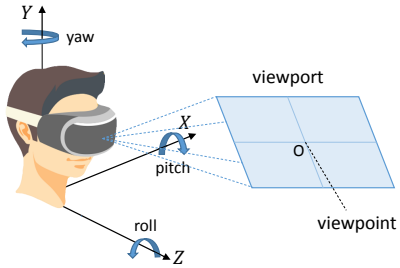


Fig. 1. Head rotation angles

To conduct viewport prediction, a general approach is to extrapolate future viewport using historical viewports. For instance, the Naive model proposed in [2][3] directly utilized the current viewport to replace the future viewport. Y. Bao et al. [3] proposed linear regression (LR) and neural network (NN) to fit the variation of the user viewport. R. Azuma et al. [4] characterized the head motion as position, velocity and acceleration, and proposed a predictor to derive the future head position. A. D. Aladagli et al. [5] and V. Sitzmann et al. [6]took content-related features into account, and predicted the viewport based on saliency algorithm. As a variety of factors, such as the preference, occupation, gender, age, etc., would influence the viewport of interest, the relationship between the future and historical viewports can be characterized as the nonlinearity and long-term dependency, The existing methods fail to well capture these two main properties, resulting in undesirable performance with respect to the prediction accuracy.

To improve the prediction accuracy, we develop a viewport prediction model using convolutional neural network (CNN). We abandon the pooling layers, then add more convolutional layers so as to obtain stronger nonlinear fitting ability. Experimental results show that our model outperforms the previous works, especially for the large-size prediction window. To the best of our knowledge, the CNN-based viewport prediction has never been studied in the literature.

Further, besides anticipating the viewport at some future

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

point, we extend out study to predict the viewport sequence in a future duration, also called *viewport trajectory*. Although the viewport trajectory prediction can be realized by a series of the proposed CNN models, with each of them predicting one point of the trajectory, such models are only suitable for fixed-size prediction window. When the adjustment of window size is asked to adaptively match the speed of head motion, all the CNN models have to be re-trained. To achieve a low complexity trajectory prediction that supports variable-size prediction window, we design a viewport trajectory prediction model based on recurrent neural network (RNN). It learns long-term dependency information in sequential viewports and outputs the estimation of viewport sequence within a future duration.

Finally, to explore the correlation between the viewport and the video content, a correlation filter-based viewport tracker (CFVT) is proposed to perform content-aware viewport prediction. The combination of the RNN and the CFVT through a fusion model enables them to complement each other, which achieves the prediction accuracy improvement up to 40%.

II. CNN-BASED SINGLE VIEWPORT PREDICTION

When watching a 360-degree video, the user wearing a HMD is supposed to stand at the center of the sphere, with his viewport in the spherical space recorded by the Euler angles, i.e., pitch (θ), yaw (φ), and roll (ψ), corresponding to the head rotation around the X, Y and Z axis, respectively. Knowing θ and φ, we can find the center point of the viewport, namely the *viewpoint*. Further, combining the viewpoint and roll angle ψ, we can determine the spherical region of the viewport, as shown in Fig. 1. Define the initial head rotation as zero degree for these three angles, then each angle rotates in a range of $[-180°, 180°]$. In reality, the ranges of θ and φ are determined by the FoV of the HMD. For example, the FoV of Oculus DK2 is 110-degree horizontally and 90-degree vertically.

The experimental results in [3] show that, compared with the auto-correlations of these three angles, their correlations are much smaller and negligible. Therefore, we assume that the rotations in three directions are independent with each other. It means that we can predict each angle independently and train three separate models to predict θ, φ and ψ respectively. Moreover, when the user moves his head, the yaw angle varies much more drastically than the other two angles, and therefore is more difficult to predict. Thus, in this paper, we focus on the prediction of the yaw angle φ. Note that the proposed approach can be straightforwardly extended to predict the pitch and roll angle.

The design of a CNN-based single viewport prediction model aims at building a CNN model that takes the current and past viewing angles (i.e. the features) as the inputs and outputs the future viewing angle. Let φ_t represent the yaw angle at time t, and $\boldsymbol{\varphi}_{t-t_s:t} = (\varphi_{t-t_s}, ..., \varphi_{t-1}, \varphi_t)$ be the yaw angles from time $t - t_s$ to time t that are collected from the head sensor. The task of the CNN model comes to predict the yaw angle φ_{t+t_w} at some future point $t + t_w$ based on the value of $\boldsymbol{\varphi}_{t-t_s:t}$, as shown in Fig. 2. Here, t_s is the size of the input, and t_w is the size of the prediction window.

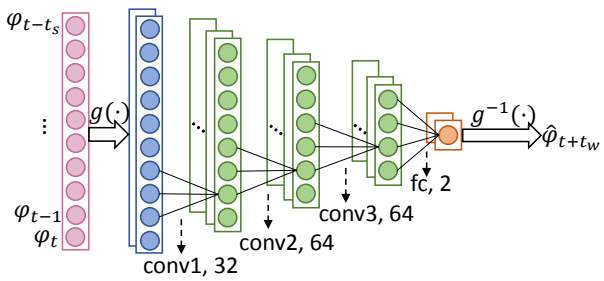


Fig. 2. CNN-based single viewport prediction model

According to our angle definition, $-180°$ and $179°$ just have a difference of $1°$ instead of $359°$. To avoid this issue, we make an angle transformation, and use $\mathbf{v}_t = (v_t^s, v_t^c)$ rather than φ_t as the input. That is

$$g(\varphi_t) = (\sin(\varphi_t), \cos(\varphi_t)) \triangleq (v_t^s, v_t^c) \tag{1}$$

Before outputting the prediction result, we make the inverse transformation and obtain φ_t from $\mathbf{v}_t$. Namely, $\varphi_t = \arctan(v_t^s / v_t^c)$.

Since the value of φ_{t+t_w} exhibits strong nonlinear correlation with $\boldsymbol{\varphi}_{t-t_s:t}$, we abandon the pooling layers to construct more convolutional layers so as to obtain stronger nonlinear fitting ability. We set all the kernel size to 3 and stride size to 1 without padding. Therefore, the depth of the network only depends on the input size. To find the optimal input size, we try the input size to be 5, 7, 10, 12, 15, and the convolutional layers to 2, 3, 3, 5, 7, respectively. We find that all the CNN models with different input sizes perform better than baseline models, and the prediction accuracy is highest when the input size is set to 10.

The values of v_t^s and v_t^c range from -1 to 1. Therefore, we choose $tanh$ as the activation function of the fully connected layer since its output ranges from -1 to 1. For all the convolutional layers, we set the activation function to be Recetified Linear Unit (ReLU), i.e., $f(x) = \max(0, x)$. In this work, we use mean squared error (MSE) as the loss function.

III. RNN-BASED CONTENT-AWARE VIEWPORT TRAJECTORY PREDICTION

In this section, we investigate the viewport trajectory prediction, i.e., simultaneously predict multiple yaw angles $\boldsymbol{\varphi}_{t+1:t+t_w} = (\varphi_{t+1}, \varphi_{t+2}, ..., \varphi_{t+t_w})$ in a duration of t_w. If the user's head moves slightly, we can have a large prediction window t_w. In contrast, if the head moves at a very fast speed, the window size would set to be small. Therefore, the viewport trajectory prediction with a variable-size prediction window t_w is critical. RNN is an ideal tool for variable-length sequence learning. Moreover, since the future viewports not only depend on the past viewports, but have a close relation with the video content, we propose a RNN-based content-aware trajectory prediction model. It consists of three modules, i.e., RNN

module, CFVT module and fusion module, as shown in Fig. 3. The RNN module learns the trajectory information from the past viewport. The CFVT module captures the relationship between the video content and the trajectory. The fusion module combines the prediction results of the RNN and CFVT module.

A. RNN-based Trajectory Prediction

The RNN module takes the current viewport φ_t as the input, and outputs the predicted trajectory $\hat{\boldsymbol{\varphi}}^r_{t+1:t+t_w}$. During training process, at time step i, we first encode φ_i value into the 128-dimensional vector, then take the embedding vector $\mathbf{x}_i$ as the input, computes hidden states $\mathbf{h}_i$ and then outputs φ^r_i through the following equations:

$$\mathbf{x}_i = \sigma_1(\mathbf{W}_{xv} g(\varphi_i) + \mathbf{b}_x) \tag{2}$$

$$\mathbf{h}_i = \sigma_2(\mathbf{W}_{hx}\mathbf{x}_i + \mathbf{W}_{hh}\mathbf{h}_{i-1} + \mathbf{b}_h) \tag{3}$$

$$\mathbf{y}_i = \mathbf{W}_{oh}\mathbf{h}_i + \mathbf{b}_o \tag{4}$$

$$\hat{\varphi}^r_{i+1} = g^{-1}(\mathbf{y}_i) \tag{5}$$

where $\mathbf{W}_{xv}$, $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{oh}$ are the learnable weights, $\mathbf{b}_x$, $\mathbf{b}_h$ and $\mathbf{b}_o$ are the learnable biases. During testing process, we provide the ground-truth of φ_i at the first iteration. For the following iterations, the inputs are the predicted results obtained from the latest iteration, i.e., $\varphi_i = \hat{\varphi}^r_i$. Here, $\sigma_1(\cdot)$ and $\sigma_2(\cdot)$ are the activation functions. Specifically, $\sigma_1(\cdot)$ is ReLU function and $\sigma_2(\cdot)$ is $tanh$ function.

B. Content-aware Trajectory Prediction

Besides the past viewport, the video content also influences the head motion. To model the correlation between the viewport and the video content, we can design a correlation filter which has peak response to the viewport area in the whole spherical image. Following the correlation filters with weighted convolution responses (CFWCR) algorithm [7], we propose a correlation filter-based viewport tracker (CFVT) to track the viewports in the future frames.

Correlation filter-based trackers such as CFWCR are designed for tracking the specific target in a normal video. The objective to be tracked in this paper is the viewport that is more abstract than a target. Thus, we first project the spherical image into a planar image using the equirectangular projection (ERP), then find the viewport region in the planar image. For the ERP, the content near the poles will be expanded horizontally, and the viewport is not rectangular in the tangent planes, so the region of the viewport needs to be redefined. To do that, we set a bounding box around the center point of the viewport, representing the viewport region in the planar image.

C. Model Fusion of RNN and CFVT

The fusion module is to combine together the prediction results of the RNN and CFVT module and output the final viewport trajectory. That is

$$\hat{\boldsymbol{\varphi}}_{t+1:t+t_w} = \boldsymbol{w}_1 \odot \hat{\boldsymbol{\varphi}}^r_{t+1:t+t_w} + \boldsymbol{w}_2 \odot \hat{\boldsymbol{\varphi}}^c_{t+1:t+t_w} \tag{6}$$

where $\hat{\boldsymbol{\varphi}}_{t+1:t+t_w}$ is the final result, $\hat{\boldsymbol{\varphi}}^r_{t+1:t+t_w}$ and $\hat{\boldsymbol{\varphi}}^c_{t+1:t+t_w}$ are the outputs of the RNN and CFVT module respectively, $\odot$ is the element-wise multiplication, and $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are the weights satisfying $\boldsymbol{w}_1 + \boldsymbol{w}_2 = \mathbf{1}$.

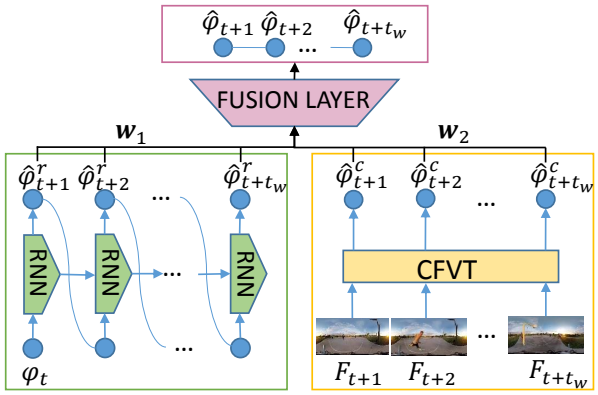


Fig. 3. RNN-based content-aware viewport trajectory prediction model, where F_i represents the frame at time step i.

For the CFVT module, as we do not update the filter, the gap between the estimated result and the real value would gradually enlarge arising from the accumulated prediction errors. It means that the prediction errors of the tracker increases with the prediction window size. Therefore, for the large prediction window j, we would have a small weight $w_{2,j}$ for the CFVT module, and vice versa.

IV. Experiment and Result

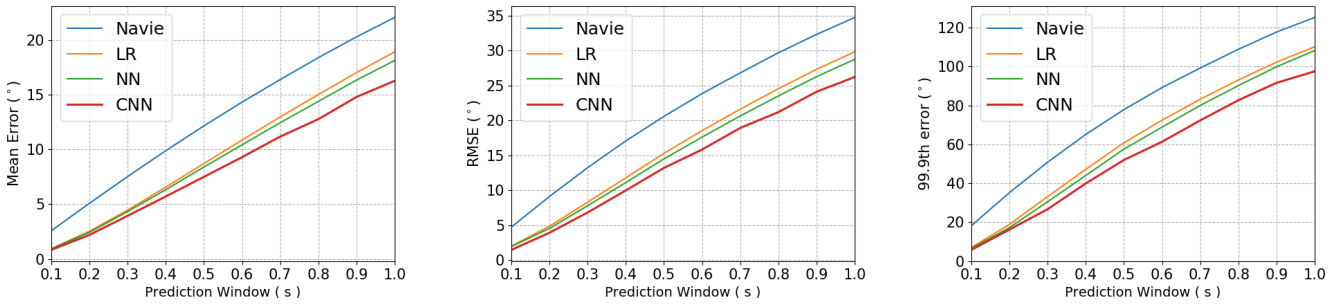
A. Experiment Setting

1) Datasets: In this paper, we evaluate our method on the dataset presented in [3]. This dataset collected the head motion data (recorded by Euler angles) of 153 volunteers when they was watching 16 clips of 360-degree videos covering a variety of scenes, such as sports activities and landscape. Most of the volunteers only viewed part of the 16 clips, and there are 985 views in total. After preprocessing, the head movements were sampled 10 times per second, and each view recorded 289 samples. Thus, the dataset includes 285665 samples of the head motion in total. For the CNN and RNN model, we use 80% of the data for training, 20% of the data for testing.

2) Metric: To evaluate the accuracy of viewport prediction, we adopt three metrics, namely, mean error, root-mean-square error (RMSE) and 99.9th percentile.

3) Implementation details: For the CNN-based model, the first convolutional layer has 32 channels, all the other convolutional layers have 64 channels. We use the Adam optimization[8]. The momentum and weight-decay are set to 0.8 and 0.999 respectively. We set the batch size to be 128 and train the model for 200 epochs. The learning rate is linearly decayed from $1e-3$ to $1e-4$ in the first 100 epochs.

For the RNN model, we train RNN with hidden state size being 256. The optimization method is the same with the CNN-based model. We set the batch size to be 128 and train the model for 500 epochs. The learning rate is linearly decayed from $1e-3$ to $1e-4$ in the first 250 epochs.



(a) Mean Error (b) RMSE (c) 99.9th

Fig. 4. The result of Naive, linear regression (LR), neural network (NN) and CNN (proposed model) in terms of mean error, rmse and 99.9th.

For the CFVT model, we resize all images to 1800×900, and set the bounding box of the viewpoint to be 10×10. Besides, we use the feature maps of the first and last convolution layer of VGG-M network[9].

For the fusion model, we investigate the prediction performance by assigning different values for the weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, and select the weights with best prediction results for each prediction window.

B. Results

In terms of three metrics, we compare the prediction accuracy of our proposed CNN-based viewport prediction model with three exisiting prediction models, that is, Naive, LR and NN in the aforementioned dataset. As shown in Fig. 4, our CNN model significantly improves the prediction accuracy, especially when the prediction window being large. Compared to NN model which performs best among the prior works, our proposed CNN-based model improves the prediction accuracy by $9\% \sim 27\%$ in all the three metrics, and has up to 69% improvement in contrast to the baseline model, i.e., the Naive model. This indicates that the CNN model has stronger nonlinearity fitting ability and performs better for the large prediction window.

Furthermore, we compare the performance of RNN, CFVT and FM models for predicting viewport trajectory with respect to mean error and RMSE. In Table 1, the result shows that the CFVT model performs better than RNN model in the small prediction window ($0.1s \sim 0.4s$). By analyzing the viewport trajectory generated by RNN and CFVT model, we find that the CFVT's output changes more slowly than RNN. In other words, the auto-correlation of CFVT's output is stronger than RNN's. Therefore, CFVT performs better for short prediction window. The final viewport trajectory is closer to CFVT's output for small prediction windows ($0.1s \sim 0.9s$), and closer to RNN's output for large prediction windows ($1.0s \sim 2.0s$). Compared to RNN, the prediction accuracy of fusion model has been improved by up to 40%.

TABLE I
THE MEAN ERROR AND RMSE OF VIEWPORT PREDICTION FOR THE RNN, CFVT AND FUSION MODEL(YAW ANGLE IN DEGREE)

	Mean Error			RMSE			
$T_r s$	RNN	CFVT	FM	RNN	CFVT	FM	w_1
0.1	4.17	2.72	**2.49**	5.90	**4.64**	6.36	0.1
0.2	6.58	5.35	**4.33**	9.95	9.00	**7.52**	0.1
0.3	8.64	7.93	**6.58**	13.73	13.12	**10.65**	0.1
0.4	10.73	10.46	**8.86**	17.28	17.02	**14.08**	0.1
0.5	12.81	12.97	**11.16**	20.56	20.75	**17.56**	0.1
0.6	14.70	15.13	**13.47**	23.53	23.91	**21.02**	0.1
0.7	16.56	17.36	**15.70**	26.27	27.12	**24.22**	0.1
0.8	18.30	19.50	**17.80**	28.75	30.11	**27.24**	0.1
0.9	19.87	21.52	**19.83**	30.92	32.86	**30.01**	0.2
1.0	21.41	23.43	**21.41**	32.93	35.46	**32.93**	0.7
1.1	22.82	25.64	**22.82**	34.73	38.01	**34.73**	1.0
1.2	24.11	27.43	**24.11**	36.35	40.42	**36.35**	1.0
1.3	25.28	29.10	**25.28**	37.85	42.60	**37.85**	1.0
1.4	26.42	30.74	**26.42**	39.31	44.70	**39.31**	1.0
1.5	27.44	32.33	**27.44**	40.59	46.72	**40.59**	1.0
1.6	28.45	33.82	**28.45**	41.82	48.63	**41.82**	1.0
1.7	29.33	35.28	**29.33**	42.96	50.45	**42.96**	1.0
1.8	30.21	36.54	**30.21**	44.01	52.00	**44.01**	1.0
1.9	31.00	37.74	**31.00**	45.01	53.5	**45.01**	1.0
2.0	31.73	38.89	**31.73**	45.93	54.94	**45.93**	1.0

V. CONCLUSION

In this paper, we designed single viewport prediction model based on CNN. Through exploiting the strong nonlinear fitting ability of CNN, our model outperformed the previous works. Further, a RNN-based model is proposed to predict the viewport trajectory which supported variable prediction window size with low complexity. Finally, we developed a correlation filter-based viewport tracker (CFVT) to perform content-aware viewport prediction. The combination of the RNN and the CFVT by a fusion model significantly improved the prediction accuracy. How to extract features from the spherical image is critical for content-aware viewport prediction. Applying VGG-M network to the projected planar image may cause prediction error because of the projection distortion. Our future work is to explore the spherical CNN which can process spherical image directly.

ACKNOWLEDGMENTS

The work has been partially supported by the NSFC grants No.61622112, and the Program of Shanghai Academic Research Leader (17XD1401900).

REFERENCES

[1] Richard Yao, Tom Heath, Aaron Davies, Tom Forsyth, Nate Mitchell, and Perry Hoberman, "Oculus vr best practices guide," *Oculus VR*, pp. 27–39, 2014.

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

[3] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *IEEE International Conference on Big Data*, 2017.

[4] Ronald Azuma and Gary Bishop, "A frequency-domain analysis of head-motion prediction," in *Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 401–408.

[5] A. Deniz Aladagli, Erhan Ekmekcioglu, Dmitri Jarnikov, and Ahmet Kondoz, "Predicting head trajectories in 360 virtual reality videos," in *International Conference on 3d Immersion*, 2018, pp. 1–6.

[6] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein, "Saliency in vr: How do people explore virtual environments?," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 4, pp. 1633–1642, 2018.

[7] Zhiqun He, Yingruo Fan, Junfei Zhuang, Yuan Dong, and Hong Liang Bai, "Correlation filters with weighted convolution responses," in *IEEE International Conference on Computer Vision Workshop*, 2017, pp. 1992–2000.

[8] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.

[9] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.