Transformer-based Long-Term Viewport Prediction in 360° Video: Scanpath is All You Need

Fang-Yi Chao[†], Cagri Ozcinar[‡], Aljosa Smolic[†]
[†]V-SENSE, School of Computer Science and Statistics, Trinity College Dublin
{fang-yi.chao, aljosa.smolic}@tcd.ie
[‡]Samsung Research UK

***Abstract*—Virtual Reality (VR) multimedia technology has dramatically advanced in recent years. Its immersive and interactive natures enable users to view any direction in 360° content freely. Users do not see the entire 360° content at a glance, but only a portion in the viewport. Viewport-based adaptive streaming, which streams only the user's viewport of interest with high quality, has emerged as the primary technique to save bandwidth over the best-effort Internet. Thus, users' viewport prediction in the forthcoming seconds becomes an essential task for informing the streaming decisions in the VR system. Various viewport prediction methods based on deep neural networks have been proposed. However, typically they are composed of complex Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) that require heavy computation. To achieve high prediction accuracy in limited computation time in a streaming system, we propose a new transformer-based architecture, named 360° Viewport Prediction Transformer (VPT360), that only leverages the past viewport scanpath to predict a user's future viewport scanpath. We evaluate VPT360 over three widely-used datasets and compare the computation complexity with the state-of-the-art methods. The experiments show that our VPT360 provides the highest accuracy for short-term and long-term prediction and achieves the lowest computation complexity. The code is publicly available at https://github.com/FannyChao/VPT360 to further contribute to the community.**

***Index Terms*—360° video, viewport prediction, head motion prediction.**

I. Introduction

The rapidly advanced immersive and interactive technologies in Virtual Reality (VR) bring about the surging popularity of 360° video on commercial streaming platforms. Due to its omnidirectional nature, which captures spherical spatial information in all directions, 360° video requires a massive amount of data to be streamed, *e.g.*, eight times more than traditional videos for the same perceived quality [7]. As users only see a portion of a given video in the viewport of a Head-Mounted Display (HMD), to guarantee high Quality of Experience (QoE) in the viewport seen by users, the entire 360° video has to be very high resolution. Recent works have developed viewport-adaptive streaming solutions [8] where the portion of the video in the user's viewport is streamed at the highest possible quality while the remaining part of the content is streamed at a lower quality for bandwidth saving. An essential element of viewport-adaptive streaming solutions is the viewport prediction algorithm. It predicts the position of the user viewport in forthcoming time to inform the downloading strategy in which part of the content should be streamed in high quality.

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number15/RP/27760.

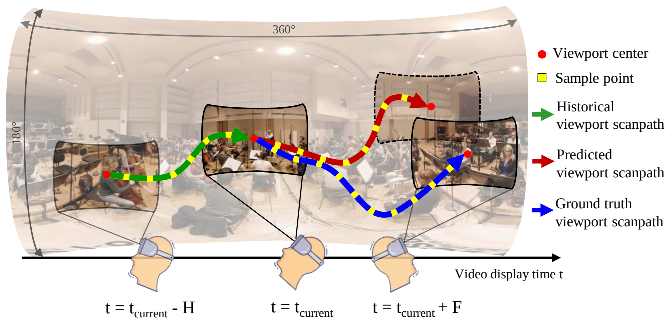

Fig. 1: Illustration of viewport prediction in 360° video. Our model aims to predict the viewport scanpath in the forthcoming F seconds given the past H-second viewport scanpath.

Numerous viewport prediction methods were proposed to benefit 360° streaming system. Most of them predict the viewport center position (*i.e.*, head movement) [1]–[6]. Our work focuses on viewport center position prediction to optimize the QoE in the viewport. Existing viewport prediction methods can be classified into clustering-based [1], [2], and deep-learning-based [3]–[6] depending on the used techniques.

Comparing clustering-based methods and deep-learning-based methods, we discover that clustering-based methods always require collecting viewport trajectories of multiple users for every 360° video in advance since the clusters in every video are content-dependent, while deep-learning-based methods can process any 360° video once the methods are pre-trained. However, deep-learning-based methods are typically composed of complex architectures with numerous Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) layers, including elements such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) that require heavy computation. Recently, we have witnessed the ground-breaking impact of transformers [9], [10] on natural language translations. Instead of using CNN or RNN to extract temporal dependencies in a sequence, a transformer adopts a multi-head self-attention mechanism to directly capture the temporal dependencies within elements at any distance in the sequence.

TABLE I: Taxonomy of existing viewport prediction methods

Cat.	Method	Input	Output	Algorithm	# of Model Parameters	Prediction Window Length
Clustering	Petrangeli_AIVR18 [1]	Past scanpath	Head Coordinates	Spectral Clustering Algorithm	/	10s
	Taghavi_NOSSDAV20 [2]	Past scanpath	Head Coordinates	Clustering	/	10s
Deep-learning	Xu_PAMI18 [3]	Past scanpath + Video Frame	Head Coordinates	CNN, LSTM, RL	34.00M	30ms (1 frame)
	Nguyen_MM18 [4]	Past scanpath + Saliency map	Viewport Map	LSTM	58.48M (saliency map) + 0.36M (scanpath)	2.5s
	Wu_AAAI20 [5]	Past scanpath + Past Viewport Frames + Future Video Frames	Viewport Map	Spherical CNN, RNN	128.87M	8s
	Romero_PAMI21 [6]	Past scanpath + Saliency Map	Head Coordinates	LSTM	172.57M	5s
	Ours	Past scanpath	Head Coordinates	Transformer	6.3M	5s

Our contributions: Considering that the efficiency of 360° streaming system strongly relies on the accuracy of viewport prediction in limited computation time, we propose a viewport prediction transformer for 360° video, called 360° Viewport Prediction Transformer (VPT360). Our proposed method processes only the past viewport scanpath to achieve accurate long-term viewport prediction with low computation complexity. As shown in Fig. 1, our model processes the evolution of users' viewport positions over time as scanpath and captures temporal dependencies between any two positions in given historical scanpath. Unlike other deep-learning-based methods, which incorporate various video content features (*e.g.*, video frames, viewport frames, saliency maps), our approach only takes advantage of viewport scanpath to simplify the computation. We evaluate our model over three widely-used datasets, namely Wu_MMSys17 [11], Xu_PAMI18 [3], David_MMSys18 [12], and compare the computation complexity with other state-of-the-art methods in terms of the number of parameters used in the model. The experiments demonstrate that our VPT360 which uses the least number of parameters outperforms all the other methods on the three datasets.

II. Related Work

Table I lists state-of-the-art viewport prediction methods with the taxonomy of several key characteristics, including required input data (*e.g.*, scanpath, video frames), output data type (*e.g.*, head coordinates, viewport map), the applied algorithmic principle, the number of parameters, and the prediction window length. In the following, we classify these methods into two categories: clustering-based and deep-learning-based techniques.

1) Clustering-based methods: Inspired by the vehicle trajectory prediction in [13], Petrangeli *et al.* [1] leveraged the spectral clustering algorithm to group similar trajectories by measuring the distance between any two trajectories. A single trend trajectory is then computed to represent each cluster's average trajectory and predict the viewer's following viewport positions. Taghavi *et al.* [2] proposed to represent the viewport center with a quaternion. They clustered a group of viewport centers by computing the distance of each viewport trajectory and generated a single trend of viewport center for each cluster.

2) Deep-learning-based methods: Xu *et al.* [3] proposed to apply Reinforcement Learning (RL) to predict the user's next head movement by considering the user's past viewport scanpath and spatial-temporal visual features in viewport images extracted by CNN and LSTM. Nguyen *et al.* [4] proposed a deep CNN model to predict saliency maps for 360° video and incorporate saliency maps and head orientation map with LSTM to predict future viewport positions. To reduce the geometric distortion inherent in spherical projection (*i.e.*, equirectangular projection), Wu *et al.* [5] proposed to adopt a spherical CNN to extract 360° spatial features in the past viewport frames and future video frames, and map user's preference in the past viewport frames into future video frames. The preference embedded visual features are then incorporated with the user's past head movement to predict future viewport positions with GRU. Romero *et al.* [6] proposed to leverage saliency maps computed from future video frames to guide the prediction model by integrating the user's past scanpath and future saliency maps with LSTM.

Comparing clustering-based methods and deep-learning-based methods, we discover that clustering-based methods have relatively less computation. However, the clusters of every video are content-dependant, which indicates that it requires collecting viewing trajectories from multiple users for any 360° video. On the other hand, deep-learning-based methods can directly be applied to any 360° video using the trained model. Nevertheless, their complex architectures, which have many learnable parameters in the models, require heavy computation and lead to high latency in the streaming system.

Inspired by the success of transformer architectures, we propose a novel viewport prediction model to address viewport prediction as a time series forecasting problem. Unlike RNN, which processes sequential data in order, transformers leverage the powerful self-attention mechanism to simultaneously take account of multiple elements in the input sequence and attribute different weights to model the impacts between each element. This architecture achieves better long-term dependency modeling and larger-batch parallel training [9] compared to RNNs. By leveraging the self-attention mechanism, our transformer-based model uses only the viewport scanpath without requiring any other content information (*e.g.*, video frames, saliency maps, etc.) to reduce the computational cost and attain superior results compared to existing methods.

III. Methodology

A. Problem Description

In viewport prediction, our goal is to predict a viewer's viewport center trajectory (*i.e.*, scanpath) in the following F seconds given the user's historical viewport center trajectory

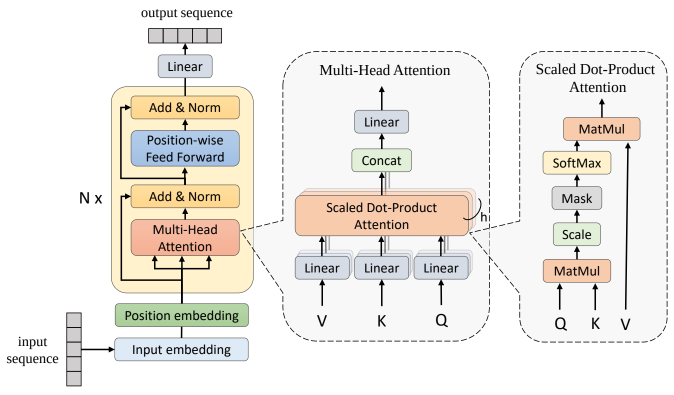

Fig. 2: (a) Architecture of our transformer-based VPT360 model, (b) multi-head attention module (c), scaled dot-product attention.

in the previous H seconds. We define $\{P_t\}_{t=0}^{T}$ as a viewport center trajectory of a viewer consuming a 360° video in duration T. It can be represented in polar coordinates $\{P_t = [\theta_t, \phi_t]\}_{t=0}^{T}$ where $[-\pi < \theta \leq \pi, -\pi/2 < \phi \leq \pi/2]$ or Cartesian coordinates $\{P_t = [x_t, y_t, z_t]\}_{t=0}^{T}$ where $[-1 < x \leq 1, -1 < y \leq 1, -1 < z \leq 1]$. Let F denote output prediction window length and H denote input historical window length. In every time stamp t, the model predicts the future viewport center position, $\hat{P}_{t+s}$, for all prediction steps $s \in [1, F]$ with the given historical information P_{t-h} for all past steps $h \in [0, H]$. We can formalize the problem as finding the best model f_F^*:

$$f_F^* = \arg\min E_t[D(f_F(\{P_t\}_{t=t-H}^{t}), \{P_t\}_{t=t+1}^{t+F})] \tag{1}$$

where $D(\cdot)$ measures the geometric distance between the predicted viewport center positions and corresponding ground truth in each time step s, and E_t computes the average distance of every prediction step in interval $t \in [t+1, t+F]$.

B. Model Architecture

We introduce a new viewport prediction model, called VPT360, which adopts the self-attention layer from transformer to predict the user's long-term viewport positions in the following F seconds. Fig. 2 shows the overall architecture of our transformer-based model. As illustrated in Fig. 2a, the transformer layer comprises two modules, a multi-head self-attention module, and a position-wise feed-forward network. We can repeat it N times to extract complex features in all elements in the sequence.

C. Multi-Head Self-Attention Module

The transformer block shown in Fig. 2a processes a set of scanpath embeddings $\{e_t\}_{t=t-H}^{t}$ as input, and output a set of updated embeddings $\{e_t'\}_{t=t-H}^{t}$ with temporal dependencies. We can create the query, key, and value matrices $Q \in \mathbb{R}^{d_k \times d_{model}}$, $K \in \mathbb{R}^{d_k \times d_{model}}$, $V \in \mathbb{R}^{d_v \times d_{model}}$, respectively, from the given input sequence with the functions:

$$Q = f_Q(\{e_j\}_{j=1}^{t}), K = f_K(\{e_j\}_{j=1}^{t}), V = f_V(\{e_j\}_{j=1}^{t}), \tag{2}$$

where f_Q, f_K and f_V are the corresponding query, key and value functions which linearly project the input sequence. The query matrix Q contains vector representing one element in the sequence, the key matrix K represents all the elements in the sequence with a vector, and the value matrix V is the same as the matrix Q which represents all the elements in the sequence.

The attention weights between each element can be calculated with the scaled dot-product attention shown in Fig. 2c and defined as:

$$\text{Att}(Q, K, V) = \text{softmax}(\frac{Q, K^T}{\sqrt{d_k}})V \tag{3}$$

It can be regarded as the matrix V multiply with the weights calculated by the matrix Q and K. The weights are defined by how each element of the sequence Q is influenced by all the other elements in the sequence K. Additionally, the softmax function normalizes the weights to yield a distribution between 0 and 1. Those weights are then applied to all the elements in the sequence in V. The scale factor $\sqrt{d_k}$ is to avoid overly large values of the inner product, especially when the dimensionality is high.

The attention mechanism, shown in Fig. 2b, can be repeated multiple times with linear projections of Q, K, and V. This multi-head attention benefits the model to learn from different representations of Q, K, and V by jointly attending to information from different representation subspaces at other positions. These linear representations are done by multiplying Q, K, and V by weight matrices W^Q, W^K, W^V that are learned during the training process.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}([\text{head}_j]_{j=1}^{h})W^O \\ \text{where head}_j &= \text{Att}(QW_j^Q, KW_j^K, VW_j^V) \end{aligned} \tag{4}$$

where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

D. Position-wise Feed-Forward Network

After self-attention sub-layers aggregate all input embeddings with adaptive weights, each layer contains a fully connected feed-forward network to consider interactions between different dimensions. It consists of two linear transformations with a ReLU activation in between and is applied to each position separately and identically.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{5}$$

where $W_1 \in \mathbb{R}^{d_{model} \times 4d_{model}}$, $W_2 \in \mathbb{R}^{4d_{model} \times d_{model}}$, $b_1 \in \mathbb{R}^{4d_{model}}$ and $b_2 \in \mathbb{R}^{d_{model}}$ are learnable weights and shared across all positions. Note that while the linear transformations are the same across different positions, they use different weights in different layers.

E. Positional Embedding

Since there is no recurrent unit in transformer layer to capture temporal features, we exploit positional embedding to infuse the relative or absolute position information of the

TABLE II: Main characteristics of three datasets containing scanpaths collected with HTC Vive HMD in 360° video. HM and EF denote Head Movement and Eye Fixation, respectively.

Dataset	Viewer #	Video #	Video Length	Video Size	Ground truth Annotation
Wu_MMSys17 [11]	48	9	2-10 min.	2k	HM
Xu_PAMI18 [3]	58	76	10-80 sec.	3k-8k	HM / EF
David_MMSys18 [12]	57	19	20 sec.	4k	HM / EF

elements in the input sequence. We follow the work in [9] to use the summation of input sequence with sine and cosine functions of different frequencies:

$$\begin{aligned}\mathrm{PE}(pos, 2i) &= \sin(pos/10000^{2i/d_{model}}) \\ \mathrm{PE}(pos, 2i+1) &= \cos(pos/10000^{2i/d_{model}})\end{aligned} \tag{6}$$

where pos denotes the position and i denotes the dimension. This sinusoidal function allows the model to attend by relative positions easily. We also tried the learnable position embedding as used in [10], but found that this led to worse performance in our case. We analyze the effect of the position embedding in our experiments.

F. Combination Loss Function

In the training procedure, we use mean square error (MSE) as a loss function due to its simplicity to measure the distance between the predicted sequence and the ground truth sequence. To improve the prediction accuracy, we compute the velocity of each element in a sequence and measure the distance of velocity between the predicted sequence and the ground truth sequence by MSE. The motion velocity V is the root mean square value of the position in current moment and the position in the last moment, where $V = \sqrt{(P_t - P_{t-1})^2} = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2}$. Our loss function is then defined as the combination of position MSE and motion velocity MSE as:

$$L = \alpha\ \mathrm{MSE}(\hat{P}, P) + \beta\ \mathrm{MSE}(\hat{V}, V) \tag{7}$$

where $\hat{V}$ denotes the motion velocity in predicted position $\hat{P}$, and V denotes the motion velocity in ground truth position P. The hyper-parameters α and β are used to balance the scale of two loss components. We set $(\alpha, \beta) = (0.75, 0.25)$ in this work by experiments.

IV. Experiments

A. Dataset Analysis

Existing viewport prediction methods listed in Table I were developed on different datasets. We selected three widely-used datasets for a fair evaluation, namely Wu_MMSys17 [11], Xu_PAMI18 [3], and David_MMSys18 [12]. They were recorded as Head Movements (HM) or Eye Fixations (EF) of observers watching 360° videos with HTC Vive HMD. The main characteristics of the datasets are shown in Table II.

Different video contents lead to different viewing scanpath. We computed accumulated angular motion in every 5-second scanpath of the three datasets and illustrated the histograms in Fig. 3. Accumulated angular motion is defined as the

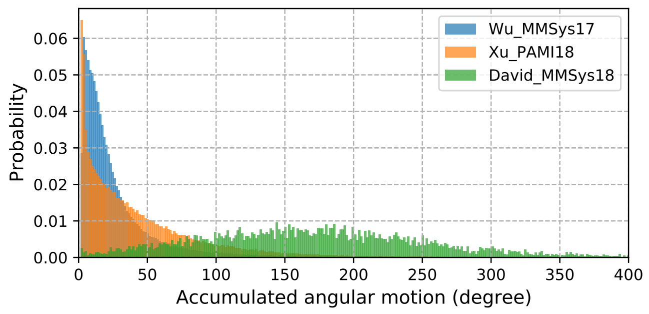

Fig. 3: Histogram of accumulative angular motion in every 5-second scanpath in three datasets. Accumulated angular motion is the summation of absolute head rotation angle in a given scanpath.

summation of absolute head rotation angle in a given scanpath sequence. For instance, if the observer's head rotates $a°$ to the left and then rotates $b°$ to the right, the accumulated angular motion is $(a+b)°$. We can see from Fig. 3 that 85% of scanpaths in the Wu_MMSys17 dataset and 67% of scanpaths in the Xu_PAMI18 dataset have accumulated angular motion smaller than 50°. It indicates that most scanpaths in these two datasets do not have motion more than a half Field of View (FoV) size of the viewport. Note that the FoV size is set to 100° referring to the common HMD (*i.e.*, HTC Vive, Oculus Rift) on the market. On the other hand, the David_MMSys18 dataset obtains a significantly more extensive range of accumulated angular motion in every 5-second scanpath. We thus regard the David_MMSys18 dataset as the most challenging one and use it in our experiments for ablation study. Finally, all these datasets are used for comparison between our method and other state-of-the-art methods.

B. Implementation Details

Here we use Cartesian coordinates (*i.e.*, x, y, z) rather than polar coordinates (*i.e.*, θ, ϕ) to represent viewport position since the former retains continuous between ± 1 in x, y, z dimensions on the sphere, while the latter has a periodic issue which $-\pi = \pi$ in θ coordinate. We set the input historical window length $H = 1$ second and output prediction window length $F = 5$ seconds since the professional streaming systems (*e.g.*, Facebook) download video segments at least 5 seconds before the playout time [14]. Precisely, our model forecasts a user's viewport scanpath in the forthcoming 5 seconds by considering his/her previous 1-second viewport scanpath. The sample rate is 25 elements per second, implying that the model inputs a 25-element sequence and outputs a 125-element sequence.

We use David_MMSys18 to conduct an ablation study of our method. Following the settings in Romero_PAMI21 [6], we selected ten videos viewed by 28 observers for training and five videos viewed by the other 29 observers for testing. Our transformer employs $N = 1$ layer encoder with $h = 8$ multi-head attention and model length $d_{model} = 512$, where $d_k = d_v = d_{model}/h = 64$ following [9]. We used batch size 5, learning rate 2×10^{-7} with 0.99 decay rate, and Adam

TABLE III: Quantitative results of the ablation study. All the scores are shown in average great circle distance (in rad.) from the 1^{st} to the 5^{th} second, which the lower value indicates the better prediction accuracy. The best scores are shown in **bold**.

	1^{st} s	2^{nd} s	3^{rd} s	4^{th} s	5^{th} s
SPE, L_{pos}	0.269	0.668	0.952	1.117	1.191
SPE, $L_{pos+vel}$	**0.239**	**0.637**	**0.934**	**1.095**	**1.139**
LPE, L_{pos}	0.354	0.749	1.004	1.174	1.293
LPE, $L_{pos+vel}$	0.401	0.774	1.018	1.179	1.287
input window H=0.5s	**0.219**	0.638	0.959	1.176	1.329
input window H=1.0s	0.239	**0.637**	**0.934**	**1.095**	**1.139**
input window H=1.5s	0.319	0.700	0.965	1.141	1.251
input window H=2.0s	0.307	0.706	0.988	1.170	1.272
Enc 1 layer	**0.239**	**0.637**	0.934	**1.095**	**1.139**
Enc 2 layers	0.258	0.652	**0.920**	**1.095**	1.191
Enc 3 layers	0.283	0.673	0.945	1.125	1.210
Ours (SPE, $L_{pos+vel}$)	**0.239**	**0.637**	**0.934**	**1.095**	**1.139**
Ours, SM, Enc 1 layer	0.355	0.765	1.021	1.174	1.271
Ours, SM, Enc 2 layers	0.313	0.752	1.056	1.247	1.362
Ours, SM, Enc 3 layers	0.278	0.850	1.279	1.506	1.604

optimizer. We implemented the model with PyTorch on a computer embedded with an Intel Core i7-7700 CPU and an Nvidia GTX 1080 GPU. We trained the model for 300 epochs taking about 1.2hr training time on the David_MMSys18 dataset.

C. Evaluation Metrics and Baseline

Referring to exiting methods [1]–[6], we selected three widely-used metrics: average grate circle distance, the average ratio of overlapping tiles, and mean overlap for evaluation. Average great-circle distance computes the distance between the predicted point $\hat{P}_t = (\hat{\theta}_t, \hat{\phi}_t)$ and groudtruth point $P_t = (\theta_t, \phi_t)$ on a sphere. Mean overlap computes the average overlap ratio of intersection over the union between predicted and ground truth viewport area in a given prediction window. The FoV size is set to 100°. The average ratio of overlapping tiles measures accuracy in terms of the percentage of overlapping tiles between predicted and ground truth viewports. We follow Nguyen_MM18 [4] to set (9, 16) tiles in a frame. The lower great circle distance, the higher mean overlap score, and the higher ratio of overlapping tiles indicate a more accurate prediction.

To assess the effectiveness of each method, we use a simple baseline (no-prediction method) which uses the repetition of the last element in the input scanpath as an output scanpath.

D. Ablation Study

Table III reports the quantitative results in average great circle distance for prediction accuracy in each second. This ablation study is organized into four parts. The first row in the table investigates the impact of sinusoidal/learnable positional embedding (denoted SPE/LPE) and combination loss function (denoted L_{pos} and $L_{pos+vel}$). From the results, we can see that sinusoidal outperforms learnable positional embedding in all 5-second sequences. Comparing the combination loss function, we can see that adding velocity MSE to position MSE improves the prediction in sinusoidal positional embedding. The second row in the table illustrates the results of different input historical window lengths. It shows that the shorter window length leads to better short-term prediction but worse long-term prediction. We set the window length to 1 second as it achieves the best results most of the time. The third row in the table examines the effects of different transformer encoder (denoted Enc) layers. In vanilla Transformer [9], which uses six layers encoder to extract higher-order features between elements, we discover that, in our work, only using one-layer encoder performs satisfactorily. More layers of encoder do not improve the results.

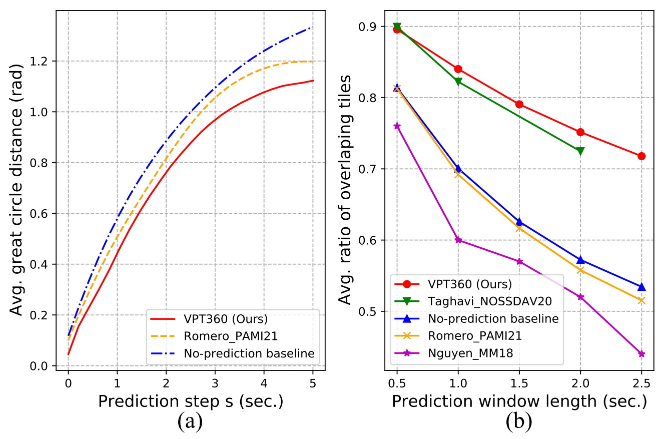

Fig. 4: Comparison results on (a) David_MMSys18 and (b) Wu_MMSys17 dataset, respectively.

From the results above, we can conclude that, given 1-second past scanpath, using sinusoidal position embedding, combination loss function, and one layer transformer encoder brings about the highest accuracy in 5-second scanpath prediction. Referring to other existing methods using saliency maps to improve the prediction, we integrate ground truth saliency maps of future frames to see if it contributes to better prediction. We use the method proposed in Romero_PAMI21 [6] to flatten the saliency maps of the next frame into one dimension and concatenate it with position embedded input sequence. We then use the transformer encoder to encode the concatenation of position embedded sequence and flattened saliency map.

The fourth row of Table III demonstrates the results of integrating ground truth saliency maps (denoted SM) with encoders in a different number of layers. We can see that the performance is not improved by simply combining the saliency maps. A better integration method is required.

E. Comparison with the State of the Arts

Fig. 4a, Fig. 4b, and Table IV present the performance of our VPT360 compared with state-of-the-art methods and no-prediction baseline on three datasets, respectively. All the training and test sets follow the same settings in all compared methods for a fair comparison.

Fig. 4a compares our method with Romero_PAMI21 [6] on the David_MMSys18 dataset. It shows that our method achieves the best result in the entire 5-second scanpath prediction. Fig. 4b compares our method with a cluster-

TABLE IV: Comparison with Xu_PAMI18: Mean Overlap scores of FoV prediction, prediction window length F ≈ 30ms (1 frame). The best score is shown in **bold** and the second-best score is shown in underline.

Method	KingKong	SpaceWar2	StarryPolar	Dancing	Guitar	BTSRun	InsideCar	RioOlympics	SpaceWar	CMLauncher2	Waterfall	Sunset	BlueWorld	Symphony	WaitingForLove	Average
Xu_PAMI18 [3]	0.809	0.763	0.549	0.859	0.785	0.878	0.847	0.820	0.626	0.763	0.667	0.659	0.693	0.747	0.863	0.753
No-prediction baseline	0.974	0.963	0.906	0.979	0.970	0.983	0.976	0.966	0.965	0.981	0.973	0.964	0.970	0.968	0.978	0.968
Romero_PAMI21 [6]	0.974	0.964	0.912	0.978	0.968	0.982	0.974	0.965	0.965	0.981	0.972	0.964	0.970	0.969	0.977	0.968
VPT360 (Ours)	**0.981**	**0.978**	**0.975**	**0.986**	**0.983**	**0.988**	**0.983**	**0.983**	**0.980**	**0.983**	**0.979**	**0.979**	**0.980**	**0.981**	**0.984**	**0.982**

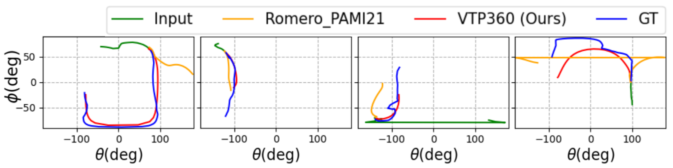

Fig. 5: Four examples of viewport scanpath predicted by our VPT360 and Romero_PAMI21 on the David_MMSys18 dataset.

based method Taghavi_NOSSDAV20 [2], two deep-learning-based methods Nguyen_MM18 [4], and Romero_PAMI21 on Wu_MMSys17 dataset. The results show the prediction accuracy in terms of average ratio of overlapping tiles in various prediction window lengths. Our method obtains a close result as Taghavi_NOSSDAV20 in a 0.5-second prediction window while outperforms Taghavi_NOSSDAV20 and the other methods in the prediction window longer than 0.5 seconds. It is noted that the result of Taghavi_NOSSDAV20 is in the prediction window from 0.5 to 2 seconds as reported in the paper [2]. In Table IV, to compare with Xu_PAMI18 [3], which predicts head movement in the next frame, we set the prediction window into one frame (approximately 30ms) and use mean overlap as an accurate measurement. We can see that our method significantly outperforms Xu_PAMI18 in all 15 test videos. Moreover, the scores of Xu_PAMI18 are lower than that of the no-prediction baseline in all 15 test videos, which implies that its complex architecture does not contribute to remarkable prediction ability.

We can conclude that our VPT360 achieves superior prediction accuracy to state-of-the-art methods in both short-term and long-term prediction from the comparison results. Fig. 5 visualizes four examples of viewport scanpath predicted by our VPT360 and Romero_PAMI21 on the David_MMSys18 dataset.

V. Conclusion

This paper introduced a novel transformer-based long-term viewport prediction method for 360° video, namely VPT360. We process the user's viewport scanpath as a time-dependent sequence and model the time dependencies to predict future viewport scanpath. By exploiting the self-attention mechanism in the transformer to compute the impact between every two elements in a sequence, we efficiently model long-term time dependencies in the viewport scanpath without any other video content information. Our ablation study validated the usage of sinusoidal position embedding, combination loss function, and 1-layer transformer encoder processing 1-second viewport scanpath contributes to the highest accuracy in 5-second prediction. Our VPT360 requires the least learnable parameters and achieves the highest accuracy on short-term and long-term prediction over three widely-used datasets compared with other state-of-the-art methods. In future work, we intend to develop an effective yet simple method to integrate the saliency maps of 360° video to increase the prediction accuracy and benefit the streaming system.

References

[1] 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)*, 2018, pp. 157–160.

[2] 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.

[3] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting head movement in panoramic video: A deep reinforcement learning approach," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[4] A. Nguyen, Z. Yan, and K. Nahrstedt, "Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction," in *ACM Multimedia Conference for 2018 (ACMMM2018)*, 2018.

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

[6] M. F. Romero Rondon, L. Sassatelli, R. Aparicio-Pardo, and F. Precioso, "Track: A new method from a re-examination of deep architectures for head motion prediction in 360-degree videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[7] S. Afzal, J. Chen, and K. K. Ramakrishnan, "Characterization of 360-degree videos," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, ser. VR/AR Network '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–6.

[8] C. Ozcinar, J. Cabrera, and A. Smolic, "Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 217–230, 2019.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[11] 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*, ser. MMSys'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 193–198.

[12] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, and P. L. Callet, "A dataset of head and eye movements for 360° videos," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 432–437.

[13] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 647–657, 2010.

[14] C. Zhou, Z. Li, and Y. Liu, "A measurement study of oculus 360 degree video streaming," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, ser. MMSys'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 27–37.