Viewport-Driven Adaptive 360° Live Streaming Optimization Framework

Shuai Peng[1], Jialu Hu[1], Han Xiao[1], Shujie Yang[1], Changqiao Xu[1]
[1]State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing 100876, China

Virtual reality (VR) video streaming and 360° panoramic video have received extensive attention in recent years, which can bring users an immersive experience. However, the ultra-high bandwidth and ultra-low latency requirements of virtual reality video or 360° panoramic video also put tremendous pressure on the carrying capacity of the current network. In fact, since the user's field of view (a.k.a viewport) is limited when watching a panoramic video and users can only watch about 20%∼30% of the video content, it is not necessary to directly transmit all high-resolution content to the user. Therefore, predicting the user's future viewing viewport can be crucial for selective streaming and further bitrate decisions. Combined with the tile-based adaptive bitrate (ABR) algorithm for panoramic video, video content within the user's viewport can be transmitted at a higher resolution, while areas outside the viewport can be transmitted at a lower resolution. This paper mainly proposes a viewport-driven adaptive 360° live streaming optimization framework, which combines viewport prediction and ABR algorithm to optimize the transmission of live 360° panoramic video. However, existing viewport prediction always suffers from low prediction accuracy and does not support real-time performance. With the advantage of convolutional network (CNN) in image processing and long short-term memory (LSTM) in temporal series processing, we propose an online-updated viewport prediction model called *LiveCL* which mainly utilizes CNN to extract the spatial characteristics of video frames and LSTM to learn the temporal characteristics of the user's viewport trajectories. With the help of the viewport prediction and ABR algorithm, unnecessary bandwidth consumption can be effectively reduced. The main contributions of this work include: (1) a framework for 360° video transmission is proposed; (2) an online real-time viewport prediction model called *LiveCL* is proposed to optimize 360° video transmission combined with a novel ABR algorithm, which outperforms the existing model. Based on the public 360° video dataset, the tile accuracy, recall, precision, and frame accuracy of *LiveCL* are better than those of the latest model. Combined with related adaptive bitrate algorithms, the proposed viewport prediction model can reduce the transmission bandwidth by about 50%.

***Index Terms*—360° video streaming, video streaming framework, viewport prediction, adaptive bitrate streaming, virtual reality.**

I. Introduction and motivation

The concept of VR was born as early as the 1950s, but due to the limited resources and lack of technology at that time, the development of virtual reality (VR) was limited. According to International Data Corporation's (IDC) research results [1] on the domestic XR (Augmented Reality, Virtual Reality, Mixed Reality) [2] market, the business end industry applications represented by entertainment scenes, life scenes, and even educational scenes are more and more popular. VR headsets can enhance entertainment experience, facilitate life, and enrich teaching methods to make up for the lack of scenes. The rise of 5G communication technology has greatly promoted the development of streaming media services, especially 360° panoramic video and VR video. VR videos can capture the scene in all directions, allowing the viewer to dynamically change the viewing position during playback, and obtain an immersive watching experience. Therefore, more and more video content providers, including YouTube and Facebook, provide 360° video-on-demand content and actively develop 360° video-related technologies. However, due to the huge challenges in the collection, transmission, and playback of panoramic video, applications related to panoramic videos cannot be popularized in the entertainment life of the public, and the details are as follows.

Manuscript received December 5, 2021; revised January 18, 2022. Corresponding author: Shujie Yang, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (email: sjyang@bupt.edu.cn).

Firstly, panoramic video is different from traditional video in many aspects such as collection, projection, and compression. The collection process requires multiple cameras to capture cooperatively. In order to transmit panoramic video, it is necessary to project the 3D panoramic video onto a two-dimensional plane. Equirectangular projection (ERP) [3] is one of the most commonly used projection methods and is also computational-friendly because each rectangle in the equirectangular projection has the same solid angle. After projection, the video needs to be encoded and compressed during the transmission process. Generally, the AVC/H.264 [4] encoding method is used. Designing appropriate collection, projection, and compression methods can reduce spatial and temporal redundancy while ensuring video quality, reduce transmission bandwidth consumption, and increase system capacity.

Secondly, the bandwidth for transmitting a 4k 360° panoramic video to the client and allowing users to watch in all directions is 400Mbps, while the traditional 4K video streaming requires only 25Mbps of bandwidth [5]. Panoramic videos require a very high video resolution to provide users with a high-quality watching experience, which is normally greater than 4K. However, the current global average fixed bandwidth and mobile bandwidth download speeds are 113.25Mps and 63.15Mbps [6]. Therefore, in a dynamic time-varying network environment, transmitting 360° videos to the client is very challenging and it's more difficult to meet bandwidth requirements.

This work mainly focuses on the scene of transmitting live panoramic video stream, aiming to reduce the transmission bandwidth. It is conceivable that when users are watching 360° panoramic videos, they can obtain the viewing angle by moving their heads while wearing head-mounted devices, tilting or rotating the phone, or dragging the mouse. In this scenario, only a part of the whole video, approximately 90° to 180° in the viewport, can be enjoyed. Therefore, the video content outside the user's viewport is wasted during this period. In a VR live broadcast scenario, if the panoramic video stream is completely transmitted to the user without considering the user's viewport, the user will cache a large amount of useless information and cause waste of network resources. Motivated by this, we propose an online-updated viewport prediction method, which can predict users' region of interest (ROI). In addition, we introduce a novel ABR algorithm [7] to adjust the download bitrate according to the real-time network and client status. These two key technologies support our panoramic video streaming framework, in which we deliver the video content inside the users' ROI to the client in high quality and the remaining video content in low quality. Compared with all high-quality transmission of panoramic video, this scheme can reduce the transmission bandwidth and the load capacity of the server to a certain extent.

Our main contributions can be concluded into the three following aspects:

- A new live panorama video transmission framework is proposed, which is the whole detail process from video acquisition, video preprocessing, video transmission to video display. Based on the framework, we design a panoramic video playback platform.
- A novel algorithm is proposed, where an innovative viewport prediction model called **LiveCL** is combined with an adaptive bitrate algorithm to optimize the transmission. The live viewport prediction model outperforms the existing model because of the dynamic adjustment criterion threshold mechanism.
- Based on the public panoramic video data, an improvement is found in the algorithm. A novel ABR algorithm for 360° video is also proposed. Utilizing the viewport prediction combined with the ABR algorithm, our proposed framework can reduce the bandwidth consumption by about 50%.

The viewport prediction model in this paper is inspired by LiveDeep, which is a viewport prediction mechanism proposed by Feng *et al.* [8] for live VR streaming for the first time. However, LiveDeep predicts numerous redundant tiles that users will not watch, which increases bandwidth consumption. Compared with LiveDeep, there are three main changes. Firstly, we use the modified AlexNet combined with ResNet's idea as the CNN module in LiveCL, which avoids the loss of information in the forward propagation. Secondly, we feed users' fixation point maps into the LSTM module, while LiveDeep only considers users' viewing coordinate points. Thirdly, we introduce a dynamic adjustment criterion threshold mechanism for LiveCL, which can reduce the number of predicted redundant tiles. In addition, we use a tile-based ABR algorithm to make bitrate decisions based on viewport prediction. We will explain and elaborate the above differences in detail in section IV, V and VI.

The rest of the paper is organized as follows. Section II discusses related work. Section III presents our proposed live VR video streaming framework. Section IV designs the viewport prediction model and section V proposes a tile-based ABR algorithm. Section VI provides our simulation results, such as prediction accuracy metric, bandwidth savings metric, and so on. Section VII summarizes the significance of our proposed framework and explains the unresolved issues and possible future improvements.

II. Related Work

Compared with traditional video, panoramic video transmission needs to consume additional bandwidth resources. The existing strategies for reducing transmission bandwidth can be divided into two categories: reduction before video transmission and reduction during video transmission. Bandwidth reduction before video transmission can be achieved through multicast technology [9], which can multicast the same video content to the corresponding edge server based on the consistency of certain user viewing behaviors. This technology can enhance the collaboration between the edge server and the central server and is beneficial for improving the system capacity. Since multicast technology is not the main focus of this work, the details will not be explained extensively. Another way to reduce the transmission bandwidth before transmission is viewport prediction. As mentioned before, the video segment inside the ROI is transmitted in a higher resolution and the remaining areas are in a lower resolution after predicting the users' ROI. In the process of video transmission, a special adaptive bitrate algorithm for panoramic video is generally used to select the download bitrate suitable for the network environment at that time. It is conceivable that when the network conditions are constantly changing, the network throughput also changes from time to time. This phenomenon is exacerbated in cellular networks in different regions. For this reason, video providers generally encode panoramic video streams into different bitrate levels in advance and divide them into different video segments. The following in section II is a brief overview of related works on panoramic video transmission, viewport prediction, and ABR algorithm.

Panoramic Video Streaming Framework. Similar to the transmission of traditional real-time 2D video, a server-client architecture is normally used. If more clients are involved, an edge server may be needed to reduce the delay for the client to obtain the video stream. Federico Chiariotti [10] presents a Dynamic Adaptive Streaming over HTTP (DASH, a.k.a MPEG-DASH) standard based 360° video streaming transmission architecture and uses the Omnidirectional Media Format (OMAF) standard to extend DASH and the streaming system by specifying the spatial nature of video segments, which supports common projection methods and tile-based streaming. But Federico Chiariotti did not point out the specific technical details in the paper. Yaqoob *et al.* [11] and Shafi *et al.* [12]

propose a DASH-based panoramic video streaming process, including panoramic video collection and stitching, panoramic video projection, panoramic video encoding, panoramic video packaging and transmission, panoramic video decoding and rendering, and so on. They also mention several adaptive streaming strategies, such as buffer-based adaptive streaming, throughput-based adaptive streaming, viewport-based streaming, and tile-based streaming. However, they did not explain the specific implementation details and the dependencies of each technical detail, which cannot verify the performance in the actual environment. He *et al.* [13] demonstrate whether the current network environment supports panoramic video streaming. Zink *et al.* [5] describe the current challenges of panoramic video transmission and propose a brief delivery framework for offline panoramic video streaming. Fan *et al.* [14] propose a more comprehensive offline panoramic video delivery framework. However, most of these existing delivery frameworks are all for offline panoramic videos and do not verify the actual technical details.

Viewport Prediction. Many efforts have been made in the field of viewport prediction, which can be roughly divided into two categories: based on user trajectory and based on video content. Petrangeli *et al.* [15] use a clustering algorithm to classify user trajectories into different clusters, where the users' vision trajectories in each cluster are similar for a certain period of time, and then perform regression fitting on the viewport data for each cluster. However, since the temporal characteristics of the user trajectory are not considered, their model can't learn the time change trend of the user trajectory. Bao *et al.* [16] proposed a linear regression model combined with a 3-layer perceptron model to predict the user's future viewport center, but the length of the prediction window is only 100$\sim$500ms. Xu *et al.* [17] use a long short-term memory (LSTM) [18] to encode the user's historical viewing trajectory, and combine the visual features to predict the user's viewport in the next 1s. Feng *et al.* [19] realize motion detection and user viewing feature selection and tracking in panoramic video based on the relationship between the user's viewport of interest and the moving objects in the video, and then update the prediction model based on user feedback error recovery. However, due to the limitations of the motion detection algorithm, the model can only achieve good results in the scene with easily distinguished foreground and background. If the scene captured by the motion camera in real time or under a dynamic background, the accuracy of the prediction will drop sharply and even the bandwidth consumption will be huge. The models mentioned above are based on the assumption that a large amount of users' historical viewing trajectory data are available in an offline environment. However, when a user is watching live VR video streaming, only a small amount of user viewing data can be obtained, which greatly limits the training and prediction accuracy of viewport prediction models. Feng *et al.* [20] also used AlexNet [21] to extract video tiles that users are more interested in from different video segments, and modify the traditional convolutional neural network (CNN) architecture to achieve the model's real-time updating and the user's viewport prediction for each video. CNN cannot capture the temporal relationship between the video frame and the user's trajectory, so the model cannot accurately predict the user's viewport when the user switches the viewport quickly. Based on [20], Feng *et al.* [8] again add long short-term memory (LSTM) to extract the temporal features of the user's viewport trajectory and propose the ***LiveDeep*** model. This model directly takes the union of the prediction results of CNN and the prediction results of LSTM as the final predicted user's viewport. This model improves the prediction performance compared to the previous model [20] using CNN alone, but it predicts numerous redundant tiles that users will not watch and the bandwidth reduction is not obvious. In addition, the model does not consider the salient feature of video frames and it is difficult to accurately predict the user's future viewport area in the dynamic scene where the user frequently switches the viewport.

ABR Algorithm. Since network conditions will change over time, especially in cellular mobile networks, the throughput in the network will also change accordingly. Therefore, a video is usually divided into video segments, each video segment contains a few seconds of video content, and each

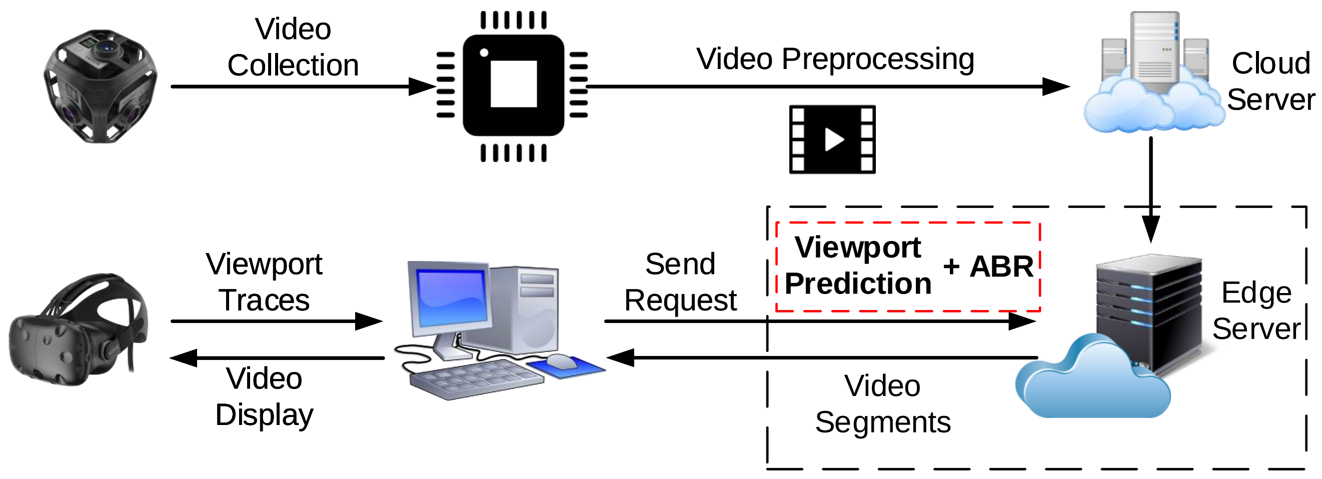


Fig. 1: Live 360° video streaming framework

video segment will be encoded into different bitrate levels. In this way, clients can switch the video bitrate to an appropriate bitrate according to their own network conditions to avoid stalling events and other situations when the network status changes. The adaptive bitrate algorithm can help clients automatically choose the appropriate video quality. The traditional adaptive bitrate algorithm has been relatively mature, but it is mainly aimed at traditional 2D videos. There are currently four types of adaptive bitrate algorithms: rate-based, buffer-based, hybrid model, and learn-based algorithms. Rate-based algorithms mainly predict the available bandwidth based on historical throughput. For example, in RB [7], the segment fetch time in the past is used as the basis for bitrate selection. Buffer-based algorithms make decisions based on the client's buffer occupancy. For example, in BOLA [22], bitrate adaptation is formulated as a utility maximization problem, and Lyapunov optimization is used to minimize stall events and maximize video quality. The algorithm of hybrid models mainly combines rate-based and buffer-based algorithms. Common algorithms are MPC [23], DYNAMIC [24], etc. Learning-based algorithms mainly use reinforcement learning or machine learning to make bitrate decisions, such as Pensieve [25], Fugu [26], etc. For panoramic video, the adaptive bitrate algorithm mainly makes decisions for the tiles in a video. Since the adaptive bitrate algorithm is not the main focus of this work, a modified version of a classical ABR algorithm is used in our work.

III. Live VR Video Streaming Framework

The live panoramic video streaming mainly involves video acquisition, video preprocessing, video transmission, and video display. Fig. 1 shows the overall transmission framework.

Video Acquisition. Generally, a professional panoramic camera is used to capture video content in all directions. Different from the traditional 2D video capture method, panoramic video needs to be captured with a professional camera which generally has multiple cameras and a wide viewing angle. After shooting, the pictures in various directions are stitched to form a panoramic picture. The different panoramas are composed into video segments, after which video segments are sent to the preprocessing stage. Here we use the professional panoramic camera Insta360 Pro2[1] to capture the video. It has 6 independent lenses and can record 8K panoramic video.

Video Preprocessing. After acquiring the stitched panoramic video, we need to project the video before transmitting it. Converting 2D videos to 3D, the projection methods [27] are involved. Common projection methods include equirectangular projection (ERP) [3], cube projection [28] (CMP, often used for 360° videos in Youtube), barrel projection [29] (BRL, proposed by Facebook), even hybrid equi-angular cubemap projection [30], etc. Fig. 2 shows the image of a same panoramic video frame of $skiing$ video introduced in section VI after ERP and CMP projection. With the help of Insta360 Pro2, the projection stage can be doned automatically and the default projection method is ERP. The projected video can be encoded with the traditional video coding standard, such as AVC/H.264 [4]. In order to adapt to the dynamic time-varying network conditions, the complete 360° video is often cut into different video segments, and then each video segment is encoded with a different bitrate, which is often used in traditional video transmission. In our work, we innovatively proposed the tile-based coding of video segments to reduce the bandwidth of 360° video transmission, that is, the video frames in each video segment are divided into tiles, and then each tile is encoded with a different bit rate. In the actual viewing process, the video tiles that users will watch are predicted based on the viewport prediction model we proposed, and then the bit rate of tiles is dynamically adjusted with the help of the tile-based ABR algorithm.

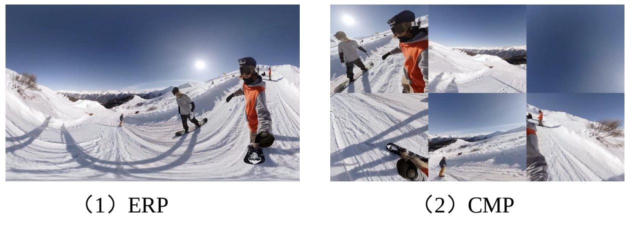


Fig. 2: The same video frame projected by ERP and CMP

Video Transmission. The transmission of panoramic video is similar to that of traditional 2D video. Generally, it is transmitted to the cloud server via Ethernet, WIFI, or cellular network. The cloud server transmits the video stream to the edge server, and then the client pulls the 360° video stream after preprocessing from the nearest edge server. In order to realize our adaptive transmission system, we use the currently popular Dash.js[2] player as the fundamental frame of our system. Dash.js player can allow us to embed our own ABR algorithm, and play 360° video in combination with A-Frame[3] framework. In the actual transmission process, the Dash.js player generally downloads the Media Presentation Description (MPD) file of the panoramic video stream from the server, and then the adaptive algorithm makes bitrate decision according to the predicted user's viewport, MPD file, and network status. The client can obtain the video segment in the server according to the selected rate most suitable for the network status. Then videos needs to be displayed through a device that supports panoramic video playback.

Video Display. In order to give users an immersive experience, 360° videos generally need to be rendered with professional headsets. At present, professional headsets on the market include HTC VIVE [31], Sony PlayStation VR [32], Oculus Rift [33], Google Cardboard, [34] and so on. In addition to professional head-mounted devices (HMDs), traditional PCs and mobile devices can also support the playback of panoramic video, but the user experience of these

[1]https://www.insta360.com/product/insta360-pro2/

[2]https://github.com/Dash-Industry-Forum/dash.js

[3]https://aframe.io/

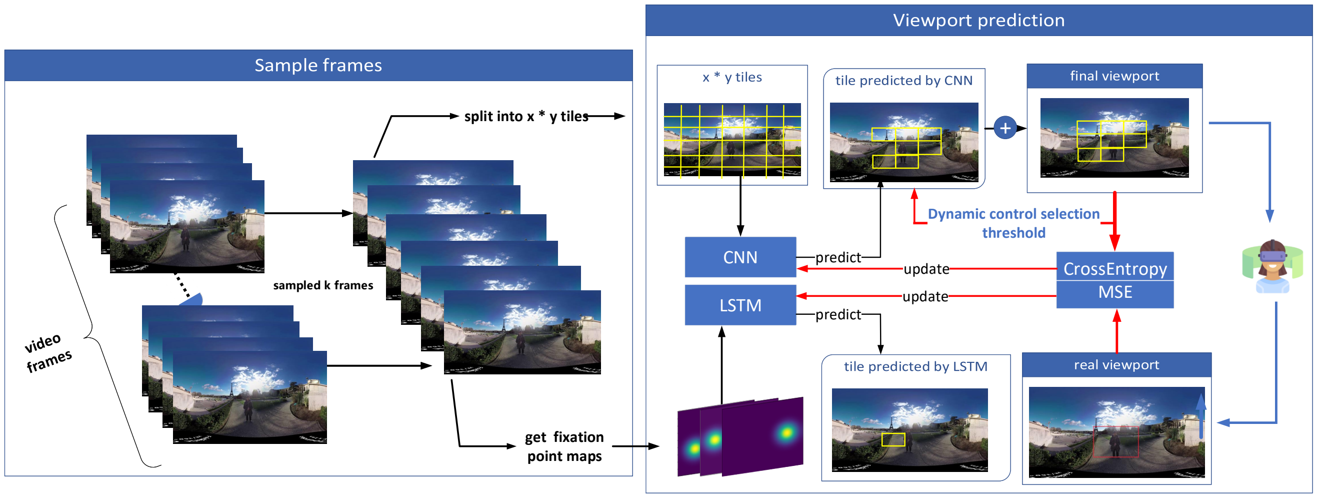


Fig. 3: **LiveCL** model architecture

is not as good as head-mounted devices. In our system, we use a professional HTC VIVE Pro Eye[4] device to watch 360° videos. The device can connect to the Dash.js player to render VR video in real-time through WebXR API[5] embedded in A-Frame framework. In addition, the device has the function of eye-tracking. We can use the device to capture the users' eye position data and then feed it back to the server, and the server updates the viewport prediction model in real-time.

IV. Viewport Prediction

To consider the load pressure on the cloud server during the actual transmission, we can deploy the viewport prediction module at the edge server. As mentioned in the *video transmission* subsection, videos will be transmitted to the cloud server after preprocessing, and then the cloud server will transmit videos to each edge server. Users will pull requested video segments from the nearest edge server where viewport prediction module can predict the video tiles that users will watch, and this part of the video content can be transmitted in high definition. In the following, we will mainly introduce our viewport prediction module in detail.

A. Motivation

In the VR live broadcast scenario, the video stream data is captured in real time. It is difficult to train and update a perfect model because of the lack of a large amount of user history viewing data. On the other hand, the live broadcast scene has strict real-time characteristics, and the time overhead of common methods is relatively large. Therefore, the offline viewport prediction model is difficult to apply to live streaming. Most of the existing methods for predicting the user's viewport are aimed at Video-on-demand (VOD). In the VOD scenario, a large number of users have watched the panoramic video in the past, and the video streaming and a large amount of user trajectory data can be obtained in advance. So the existing model can be trained offline. When new users watch panoramic video, a well-trained model can predict the future viewport of users, which can solve the problem of viewport prediction in most VOD scenarios. However, it can not be realized in the scene of live 360° video streaming. First of all, the video streams in VR live broadcast scenarios are captured in real time, and there is also a lack of a large amount of user viewing data, so there is no way to train the model in advance. For this reason, we need an online training model that can be updated in real time to solve the above problems. In addition, this model should meet the basic conditions of the live prediction model: online training, long-term iterative update, and real-time prediction.

Although Feng *et al.* [8] proposed the online training model LiveDeep, the user's viewport predicted by the model is too large, and many tiles that users do not watch are predicted. Therefore, the model does not perform well in bandwidth saving. The model trains different tiles of VR video frames in real time by modifying the traditional CNN architecture, and takes the changes in the viewport of user into account by using LSTM [18]. The author selects the union set of CNN and LSTM prediction results as the final prediction result. In addition, the model has a cold start problem. For the first video segment, the prediction effect will be poor because the model is not trained and uses randomly initialized parameters to predict the user's viewport. The model simply trains small tiles by dividing the sampled video frame into $5 * 5$ tiles, which considers the spatial features of video frame separately. Moreover, LSTM only processes the user's viewport coordinates in a past certain time period, which considers the temporal characteristics alone. LiveDeep model does not consider the spatio-temporal characteristics between the user's viewport trajectory and the video frame as a whole. When there are multiple salient objects in the video, it is difficult for the model to predict the user's viewport in the next time period. Since CNN is good at extracting the spatial features of video frames and LSTM is good at learning the temporal features of users' trajectory data, we propose a ***LiveCL*** model that mainly includes CNN module and LSTM module. In addition, ***LiveCL*** jointly considers the temporal and spatial characteristics between the user's viewport trajectory and the video frame.

[4]https://www.vive.com/us/product/vive-pro-eye/overview/
[5]https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API

B. *Model architecture*

Our model aims to reduce the bandwidth of 360° video streaming by predicting the user's viewport. The realization process pays attention to the basic requirements of the real-time scenes mentioned above such as the relatively short training period, online training, and real-time updating. The overall architecture of our model can refer to Fig. 3. It is mainly divided into two parts: video frame sampling and model training. The **LiveCL** model is based on the CNN module and the LSTM module is supplemented to realize the prediction of the user's viewport. Therefore, it is very important to choose a suitable CNN module. We implement the prediction function of our CNN module by modifying the traditional AlexNet[21] architecture. The main difference between this model and the LiveDeep [8] is that we process fixation maps of user's gaze point from the last video segment through the LSTM module, which considers the spatio-temporal characteristics of the user's viewport trajectory. We have also noticed that the LiveDeep model predicted numerous redundant tiles. So we introduce a mechanism for dynamically adjusting the threshold, which is to dynamically adjust our threshold according to the number of predicted tiles.

In order to meet the real-time requirements of VR live video streaming, we uniformly sample k frames for the video segment. Because a video segment contains a large number of similar video frames, training all video frames will consume a lot of time. We divide the sampled video frame into $x * y$ tiles spatially, so as to implement the tile-based ABR algorithm later. Each training of the **LiveCL** model feeds $k * x * y$ tiles to the CNN module to fully extract the spatial features of the video frame. In order to consider the spatio-temporal characteristics between user's viewport trajectory and the video frame, we feed the k fixation maps of user's gaze point in the user's last video segment to the LSTM module of the **LiveCL** model, which jointly considers the spatio-temporal characteristics between user's viewport trajectory and the video frame.

C. *Training process*

1) *CNN module*

The main function of the CNN module is to determine whether each tile is of interest to the user. Fig. 4 shows the overall architecture of the CNN module. Here we use a network similar to Alexnet, and then reduce the number of parameters in traditional AlexNet by modifying the number of input and output channels in the convolution layer. Using the idea of ResNet [35], we feed the result of convolution layer L5 and convolution layer L6 into the pooling layer, which can avoid the loss of information in the forward propagation to a certain extent.

Since the user viewing history data of the first video segment can not be obtained while viewing the live VR video, the user viewport of the first video segment needs to be predicted using randomly initialized parameters. After the user views the first video segment, the user views the viewport trajectory data back to the server, which acts as a training label based on the tiles in the user's viewing area. The specific

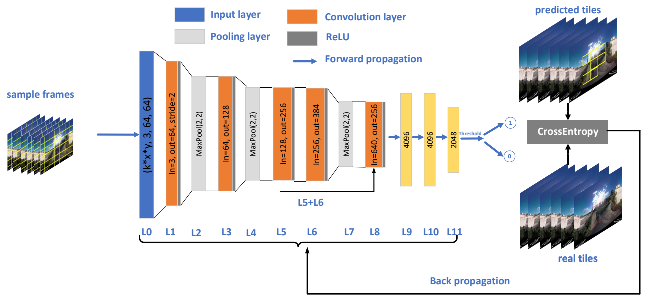


Fig. 4: CNN module

process is as follows: k frames are sampled from the current video segment, and each frame is divided into $x * y$ tiles. In order to reduce the training time, we resize each tile into (64, 64), and then feed all the tiles in sampled frames into the CNN module. We use the cross entropy function as the loss function to calculate the error between the predicted tile and the real tile, and use the Adaptive Moment Estimation (Adam) optimizer to update all the parameters of the CNN module. For the next video segment, the updated model can be used to predict the tile of interest to the user. Here we need to strictly control the criteria for determining the block of interest. When the output value of a certain tile is greater than a certain threshold, we determine it as a tile of interest to the user. The threshold is initially 0.5. In order to meet the real-time requirements, we set the learning rate of the model to 0.0001 and the epoch to 15 during the initial training. If the loss is less than 0.15 during the training process, we will terminate the training early, so that the training time can be reduced as much as possible while meeting a relatively high training accuracy.

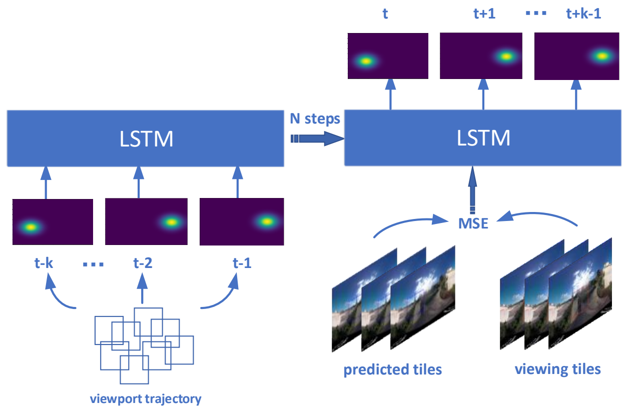


Fig. 5: LSTM module

2) *LSTM module*

The main function of the LSTM module is to assist the prediction of the CNN module, referring to the LSTM section of the **LiveCL** model architecture in Fig. 3. Fig. 5 shows the structure of our LSTM module. Specifically, the fixation maps in the previous video segment viewed by users are input into the LSTM module in chronological order, so that LSTM [18] can learn the temporal characteristics of the user's viewport trajectory. During training, after the user watches the video segment, the viewport trajectory data is fed back to the server.

The server generates fixation maps based on the viewport trajectory data and feeds them to the LSTM module. The number of hidden layers in the LSTM module is 6 and the total number of layers is 2. The initial learning rate is set to 0.001, and the training epoch is set to 50. We use the mean square error (MSE) as the loss function to calculate the error between the predicted fixation map and the real fixation map, and use the Adaptive Moment Estimation (Adam) optimizer to update all the parameters of the LSTM module. We dynamically adjust the criterion threshold of the CNN module according to the prediction result of the LSTM module, which solves the problem of the excessive number of tiles predicted by LiveDeep to a certain extent. We combine the user's viewport area predicted by the CNN module and the LSTM module and locate the index of the tiles in it. We adjust the threshold according to the following rules: we assume that the total number of tiles in a frame is m ($m = x * y$) and the number of predicted tiles is n. We can adjust the threshold according to the size of the ratio $p = \frac{n}{m}$. The larger the p, the larger the criterion threshold. For specific rules, please refer to Table I.

TABLE I: Threshold adjustment rules

ratio p	threshold
$0.5 \leqslant p \leqslant 1$	0.8
$0.3 \leqslant p < 0.5$	0.7
$0.24 \leqslant p < 0.3$	0.6
$0 \leqslant p < 0.24$	0.5

In order to take the spatial characteristics of users' viewport trajectory changes into account, we convert fixation point coordinates into a two-dimensional matrix called fixation point maps. We use the user viewport trajectory dataset published by Wu *et al.* [36], which contains 18 videos for 48 different levels of users to watch, and records the timestamp of 48 users watching different videos and its corresponding head orientation and head position. This dataset logs quaternions (qx, qy, qz, qw) to represent the direction of movement of the user's viewport. Specifically, we use Formula 1 to convert the quaternion into the direction of the user's viewport, where (x, y, z) is the direction vector of viewport after quaternion conversion.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 * qx * qz + 2 * qy * qw \\ 2 * qy * qz + 2 * qx * qw \\ 1 + 2 * qx^2 - 2 * qy^2 \end{bmatrix} \tag{1}$$

In order to obtain the projection point coordinates (xp, yp) in projected frame after equiangular rectangular projection (ERP), it is necessary to calculate the vertical angle ϕ and horizontal angle θ of the direction vector. We can get the projection point coordinates according to Formula 2:

$$\begin{bmatrix} xp \\ yp \end{bmatrix} = \begin{bmatrix} \frac{\varphi}{360} * W \\ \frac{(1-\sin\theta)}{2} * H \end{bmatrix} \tag{2}$$

Finally, we can use the Gaussian Blur function $GaussianBlur()$ in OpenCV[6] to convert the two-dimensional coordinates after ERP projection into a fixation point map. For the process mentioned above, please refer to Fig. 6.

[6]https://docs.opencv.org/4.5.0/d4/d86/group__imgproc__filter.html

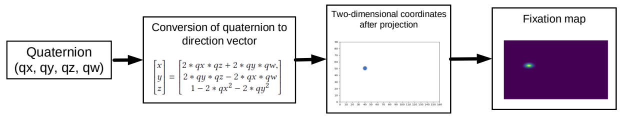

Quaternion (qx, qy, qz, qw)
Conversion of quaternion to direction vector
$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 * qx * qz + 2 * qy * qw, \\ 2 * qy * qz - 2 * qx * qw \\ 1 - 2 * qx^2 - 2 * qy^2 \end{bmatrix}$
Two-dimensional coordinates after projection
Fixation map

Fig. 6: The workflow of obtaining the fixation point map

V. Proposed ABR algorithm

Traditional adaptive bitrate algorithms mainly focus on 2D videos, which makes a bitrate decision for the entire video before downloading every segment. Directly using traditional ABR algorithms may be a naive solution for 360° videos and every tile will have the same quality. As for panorama videos, videos are normally cut into segments and each segment is separated into tiles before transmission. Taking into consideration of the different priorities of tiles, tiles can be chosen with different bitrates. This motivates the design of a novel adaptive bitrate algorithm for 360° videos.

The current study of traditional adaptive bitrate algorithms mainly falls into different groups, as explained earlier. Different types of algorithms can have individual characteristics. Rate-based algorithms can directly match the network throughput, while buffer-based algorithms can significantly reduce the potential stalling events. Hybrid algorithms can take advantage of both rate-based and buffer-based algorithms. Dynamic[24] is one of the hybrid algorithms, which is also one of the adaptive bitrate algorithms in Dash.js.

The theory of Dynamic is the combination of RB [7] and BOLA [22]. Before downloading every segment, the algorithm will first check the buffer occupancy. If the buffer level is above the threshold, BOLA will be used to make bitrate decisions. In this circumstance, the buffer is relatively stable, and BOLA can make a higher bitrate choice. If the buffer level is below the threshold, throughput (RB) will be used. This often occurs when user starts to watch a video or the network condition drop quickly. With RB, the bitrate decisions are made based on the past network throughput, and can directly match the network. BOLA in this situation may choose the lowest bitrate for a long time.

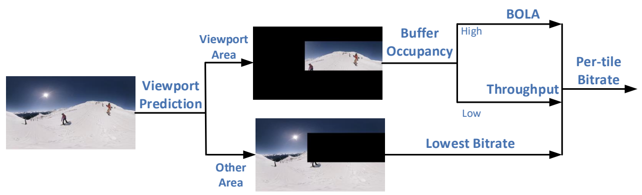


Fig. 7: Viewport-driven ABR algorithm

Moving from 2D to live panorama videos, certain modifications are made based on the original version of Dynamic. Combining the results from the **LiveCL** model, the overall process of the proposed viewport-driven adaptive bitrate algorithm is shown in Fig. 7. After the viewport prediction, the tiles can be split into two parts, tiles cover the predicted viewport area and other tiles. For tiles that cover the viewport area, those tiles have relatively higher priority and should be transmitted in higher quality. To make the proper bitrate decisions, the algorithm will first check the buffer occupancy. Similar to Dynamic, it chooses BOLA when buffer is stable while utilizing RB when buffer is about to deplete. However, there are mainly two modifications to suit panorama video in live broadcast scenarios. First, the bitrate adaptations are more fine-grained. Taking into consideration of different statuses of tiles, bitrates of tiles within a frame can be different. As tiles are transmitted parallel through HTTP, the bitrate adaptations are made for every tile. Second, the threshold of buffer occupancy is decreased. Compared to video on demand, video players in live streaming have smaller buffer and the video segments are also shortened. With shorter segments, the delay can be reduced. Also, users can enjoy higher quality earlier when the network condition becomes better. As for tiles in other areas, the tiles still need to be transmitted, in case of some sudden unpredicted head movements. The chosen bitrate of those tiles can be encoded with the lowest quality. The proposed ABR algorithm is shown in Algorithm 1.

Algorithm 1 Bitrate adaptation for 360 video

```
repeat
  before downloading a new segment
  for tiles within a segment do
    if tile is in the predicted viewport then
      check buffer occupancy $b$
      if $b \geq B_{threshold}$ then
        choose bitrate based on BOLA
      else
        choose bitrate based on throughput
      end if
    else
      choose the lowest bitrate
    end if
  end for
until end of video
```

VI. Results and Evaluation

All the data of the simulation experiment are based on the 360° video dataset contributed by Wu *et al.* [36]. Table II shows the information about the 360° video in our experiment. We implement *LiveCL* model with PyTorch 1.8 in Python 3.7 of windows 10 and use CUDA to accelerate the training process.

TABLE II: Video information

Video name	Content	Category	Length
Conan1	Connan360°-Sandwidth	Performance	$2'44''$
Skiing	Freestyle Skiing	Sport	$3'20''$
Alien	Google Spotlight-Help	Film	$4'53''$
Conan2	Conan360°-Weird AI	Performance	$2'52''$
Surfing	GoPro VR-Tahiti Surf	Sport	$3'25''$
War	The fight for Falluja	Documentary	$10'55''$
Cooking	360° Cooking Battle	Performance	$7'31''$
Rhinos	The Last of the Rhinos	Documentary	$4'52''$

We mainly compare the proposed *LiveCL* with *LiveDeep*, so the number of tiles divided in the experiment is $5 * 5$ mentioned in [8]. Fig. 8 shows the result of some frames predicted by *LiveCL* and *LiveDeep*. (a)∼(c) are the prediction results of *LiveCL*, where the blue box is the predicted tile, and the red box is the tile actually viewed by the user; (d)∼(f) are the prediction results of *LiveDeep*, where the cyan box is the predicted tile, and the red box is the tile actually viewed by the user. As you can see from the Fig. 8, the number of tiles predicted by *LiveCL* is much less than that predicted by *LiveDeep*. In order to verify the effectiveness of our proposed model, we choose tile, frame, time, and bandwidth metrics as our evaluation metrics, which will be described in detail below.

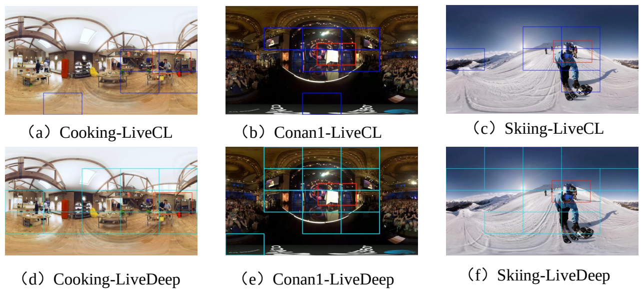


Fig. 8: The result of viewport predicted by *LiveCL* and *LiveDeep*: (a)∼(c) by *LiveCL* and (d)∼(f) by *LiveDeep*

A. Tile metric

For the convenience of the following description, we assume that the total number of tiles divided by a frame is m ($m = 5 * 5 = 25$), the number of predicted tiles is n, the number of tiles actually viewed by the user is t, the number of predicted tiles which are viewed by the user is a, and the number of predicted tiles which are not viewed by the user is b ($n = a + b$). To this end, we calculate the following metrics:

- ***TileAccurancy*** indicates the proportion of the number of tiles correctly predicted by the user. Refer to Formula 3.

$$TileAccurancy = \frac{a + (m - t - b)}{m} \tag{3}$$

- ***TileRecall*** represents the user's experience. The higher it is, the better the user's viewing experience is. Refer to Formula 4.

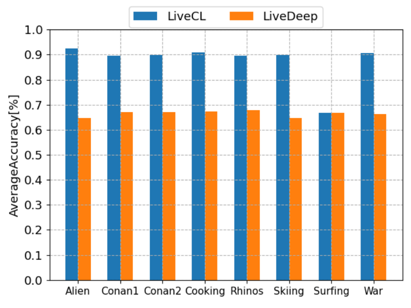
$$TileRecall = \frac{a}{t} \tag{4}$$

- ***TilePrecision*** indicates the level of bandwidth. The higher the index, the more obvious the bandwidth reduction. Refer to Formula 5.

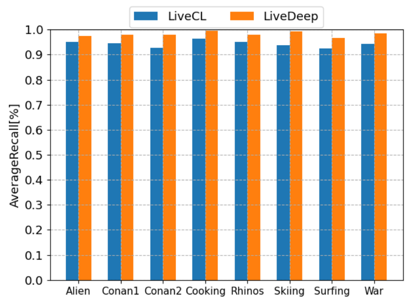
$$TilePrecision = \frac{a}{n} \tag{5}$$

In Fig. 8(c), these parameters are m=25, n=6, t=4, a=4 and b=2. The metrics are TileAccurancy=92%, TileRecall=100% and TilePrecision=66.67%.

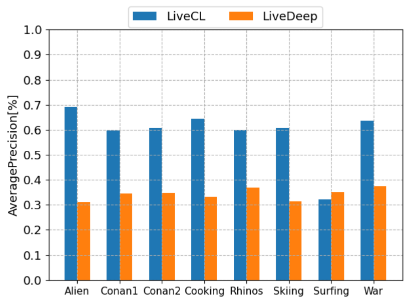
Something needs to be noted that three metrics of tile can be calculated for each frame. For a video, we calculate all these metrics of all sampled frames and then take the average value as the average metrics of the video. Since the dataset [36] provides viewport data for 48 users, we end up averaging these metrics across all users for a single video.



(a) TileAccuracy of different videos

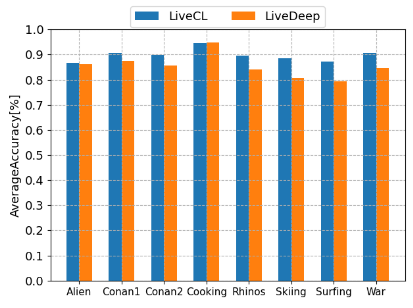


(b) TileRecall of diffent videos

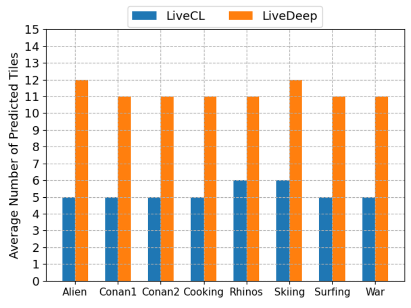


(c) TilePrecision of different videos

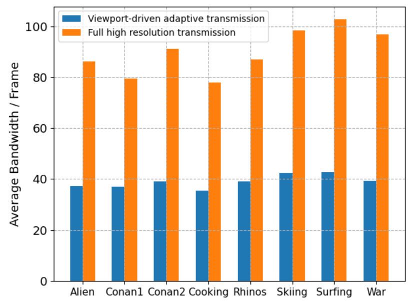
Fig. 9: Tile metrics comparison between *LiveCL* and *LiveDeep*



(a) FrameAccuracy of different videos



(b) Number of predicted tiles in different videos



(c) Bandwidth of each frame in different videos

Fig. 10: (a) Average *FrameAccurancy* comparison; (b) Average number of predicted tiles comparison; (c) Average bandwidth of each frame between *LiveCL*-driven adaptive transmission and full high-resolution transmission

Fig. 9 shows the metrics comparison between *LiveCL* and *LiveDeep*: the graph on the left shows the ***TileAccurancy*** per video, the graph in the middle shows the ***TileRecall*** per video, and the graph on the right shows the ***TilePrecision*** per video. From the Fig. 9, we can see that the *TilePrecision* metric of *Surfing* video is relatively low, because the video scene is mainly a dynamic scene of surfing, and the user's viewport switches more frequently when watching this video. It is difficult for *LiveCL* to maintain a high *TilePrecision* while maintaining a relatively high *TileRecall*. Here **LiveCL** chose a higher *TileRecall* to provide users with a perfect viewing experience as much as possible, but the corresponding bandwidth consumption will also increase a lot because many redundant tiles that users will not watch are predicted.

As mentioned above, we introduce a mechanism of dynamically adjusting the criterion threshold in *LiveCL* to solve the problem of numerous redundant tiles which are predicted by *LiveDeep*. Redundant tiles are not viewed by users. This is also reflected in the *TileAccurancy* and *TilePrecision* metrics in Fig. 9, where the *LiveCL* model predicts tiles with greater *TileAccurancy* and *TilePrecision* than the *LiveDeep* model. In addition, the *TileRecall* metric in Fig. 9 reflects that the *LiveCL* model is slightly lower than the *LiveDeep* model. This is also caused by the excessive number of tiles predicted by the *LiveDeep* model, that is, the tiles predicted by the *LiveDeep* model basically completely cover the user's viewport, but there are also a lot of redundant tiles at the same time which will not be viewed by the user.

B. Frame metric

Similar to the tile metrics, the frame metric reflects the accuracy of the model prediction in a certain video. We stipulate that for a certain sampled frame, only when the predicted viewport area completely covers the actual viewport area viewed by the user, this frame can be counted as an accurate prediction frame, which is also mentioned in [8]. We count the accurately predicted frames among all sampled frames for a certain video, and calculate ***FrameAccurancy*** according to Formula 6, where *correctFrame* is the number of all accurately predicted frames and *sampleFrame* is the number of all sampled frames.

$$FrameAccurancy = \frac{correctFrame}{sampleFrame} \tag{6}$$

The higher the *FrameAccuracy*, the higher the accuracy of the model predicting the viewport area of the user watching a certain video. Something should also be noted that for a certain video here, we count the *FrameAccurancy* metric of 48 users, and then take the average value as the average *FrameAccurancy* of the video. Fig. 10 (a) shows the *FrameAccuracy* metric for different videos. The *FrameAccuracy* metrics of the *LiveCL* model and the *LiveDeep* model are basically close, which means the two models can predict the future viewport of users in most cases.

C. Bandwidth metric

In order to actually compare the difference in the number of predicted tiles that users are interested in between the two models, we count the changes in the number of predicted tiles in all sampled frames. As before, it is also the average of all users for a certain video. Fig. 10 (b) shows the comparison of the number of tiles predicted by the two models. We can see that **LiveCL** predicts that the average number of tiles that users are interested in is 6, which is also the effect of the dynamic adjustment criterion threshold mechanism mentioned above. Correspondingly, **LiveDeep** predicts that the average number of tiles that users are interested in is 11. Compared to *LiveDeep*, *LiveCL* transmits fewer high-resolution tiles and reduces transmission bandwidth more significantly. In order to simulate the transmission bandwidth in a real scenario, we use *ffmpeg*[7] to encode all the frames that are divided into tiles into four bit rate levels for 8 videos in Table II. Table III shows the average bandwidth in *bps* units consumed by transmitting each tile with different bitrate levels for different videos.

TABLE III: Tile bitrates [bps] at different levels

Video name	Level1	Level2	Level3	Level4
Conan1	18286.0	58193.0	135869.0	2466442.0
Skiing	38100.0	149997.0	389970.0	7781989.0
Alien	20417.0	71849.0	188486.0	7781989.0
Conan2	24370.03	88758.32	238364.4	4275851.87
Surfing	50935.38	209369.63	517066.71	10611551.4
War	33806.0	131147.0	338441.0	5844169.0
Cooking	25740.0	80397.0	197771.0	3189401.0
Rhinos	16597.0	57806.0	142524.0	3620377.0

The ABR algorithm makes a decision on a certain level as the bitrate of downloading a certain tile according to the predicted viewport area and the state of the network. For example, a higher bitrate is selected for tiles in the user's viewport area, and the bitrate of tiles farther from the field of view is sequentially reduced. We transmit the tiles in the predicted viewport area at the *Level2* bitrate, and the rest of the tiles are transmitted at the *Level1* bitrate. Fig. 10 (c) shows bandwidth consumption of transmitting a frame between viewport-driven adaptive transmission and full high-resolution transmission. (All tiles are transmitted at the *Level2* bitrate in full high-resolution transmission.) Note that the bandwidth value in Fig. 10 (c) is given in scale, so it is a bit different from the values in Table III.

D. Time consumption

The live broadcast scene is very demanding on time. As mentioned above, panoramic video is generally transmitted in video segments, so the model training and prediction time should not exceed the video segment time. The timeline of the video segment in our experiment is 2s. The specific hardware configuration in our experiment is an Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz CPU, 32.0 GB RAM, and NVIDIA GeForce RTX 3060 GPU. Fig. 11 shows the time of *LiveCL* processing each video segment for different videos, which proves that the processing time of each video segment is less than 2s. So *LiveCL* meets the basic requirements of the real-time live broadcast environment. The *Conan2* video's average processing time of each video segment in Fig. 11 is the longest,

[7]https://www.ffmpeg.org/ffmpeg-all.html

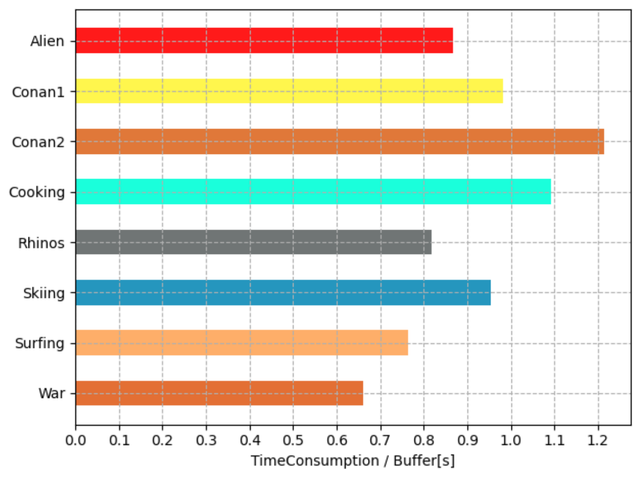


Fig. 11: Average time consumption in *LiveCL*

which is 1.11s. It may be because the video is a shot of an action science fiction movie scene, and there are more objects involved in sample frames. Therefore, the CNN and LSTM modules are trained in all epochs to minimize the loss between the predicted viewport area and the user's actual viewport area.

VII. Conclusion and Future work

In this paper, a viewport-driven adaptive transmission framework for live 360° video is proposed. By analyzing that the biggest problem of 360° video transmission is the huge bandwidth consumption, we propose a viewport prediction model called ***LiveCL*** to predict the user's viewport area, which solves the problem that *LiveDeep* predicts many redundant tiles that users will not see. The *LiveCL* model includes CNN module and LSTM module. Combined with the prediction results of LSTM, a mechanism for dynamically adjusting the criterion threshold is introduced in the CNN module. Through a series of experiments compared with *LiveDeep*, *LiveCL* can reduce the prediction of redundant tiles as much as possible while maintaining high prediction accuracy. Combined with the ABR algorithm, our solution reduces the bandwidth of 360° video transmission by 50%. It also meets the need of live video streaming scene in the model training and prediction time. The 360° live streaming optimization framework we propose may greatly reduce the transmission bandwidth, which may bring new impetus to the popularization of VR videos.

There are still unresolved issues in our work. In subsection VI-A, the accuracy rate of *LiveCL* model prediction is still relatively low in scenarios where users frequently switch their viewport. In the future, the salient features of video frames may be considered to improve the prediction accuracy of viewport prediction module for dynamic scenes. For the situation where the model prediction is wrong, the paper does not give an analysis and solution. If the predicted viewport area in the live broadcast scene does not include the user's actual viewport area, the user's viewing experience will be very poor, and the user will even get dizzy. In addition, we need to consider how to optimize viewport prediction of the first video segment where prediction module uses random parameters. For example, transfer learning [37] can help an untrained model initial parameters.

Acknowledgment

This work is supported by the National Key R&D Program of China (2018YFE0205502) and by the National Natural Science Foundation of China (NSFC) under grant No. 62001057, 61871048 and 61872253.

References

[1] IDC Corporate USA. Idc releases 10 predictions for the ar/vr market in 2021. https://www.idc.com/getdoc.jsp?containerId=prCHC47313321, 2021.

[2] Unity Technologies. What is ar, vr, mr, xr, 360°. https://unity.com/how-to/what-is-xr-glossary, 2021.

[3] Wikipedia. Equirectangular projection. https://en.wikipedia.org/wiki/Equirectangular_projection, 2021.

[4] Ingo Bauermann, Matthias Mielke, and Eckehard Steinbach. H. 264 based coding of omnidirectional video. *Computer Vision and Graphics: International Conference*, pages 209–215, 2006.

[5] Michael Zink, Ramesh Sitaraman, and Klara Nahrstedt. Scalable 360° video stream delivery: Challenges, solutions, and opportunities. *Proceedings of the IEEE*, 107(4):639–650, 2019.

[6] Internet. Internet speeds by country 2021. https://worldpopulationreview.com/country-rankings/internet-speeds-by-country, 2021.

[7] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. Rate adaptation for adaptive http streaming. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, page 169–174, 2011.

[8] Xianglong Feng, Yao Liu, and Sheng Wei. Livedeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 800–808, 2020.

[9] Cristina Perfecto, Mohammed S. Elbamby, Javier Del Ser, and Mehdi Bennis. Taming the latency in multi-user vr 360°: A qoe-aware deep learning-aided multicast framework. *IEEE Transactions on Communications*, 68(4):2491–2508, 2020.

[10] Federico Chiariotti. A survey on 360-degree video: Coding, quality of experience and streaming. *Computer Communications*, 177:133–155, 2021.

[11] Abid Yaqoob, Ting Bi, and Gabriel-Miro Muntean. A survey on adaptive 360° video streaming: Solutions, challenges and opportunities. *IEEE Communications Surveys Tutorials*, 22(4):2801–2838, 2020.

[12] Rabia Shafi, Wan Shuai, and Muhammad Usman Younus. 360-degree video streaming: A survey of the state of the art. *Symmetry*, 12(9), 2020.

[13] Dongbiao He., Cedric Westphal., and J. J. Garcia-Luna-Aceves. Network support for ar/vr and immersive video application: A survey. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications - ICETE,*, pages 359–369. INSTICC, SciTePress, 2018.

[14] Ching-Ling Fan, Wen-Chih Lo, Yu-Tung Pai, and Cheng-Hsin Hsu. A survey on 360° video streaming: Acquisition, transmission, and display. *ACM Comput. Surv.*, 52(4), 2019.

[15] Stefano Petrangeli, Gwendal Simon, and Viswanathan Swaminathan. Trajectory-based viewport prediction for 360-degree virtual reality videos. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 157–160, 2018.

[16] 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 *2016 IEEE International Conference on Big Data (Big Data)*, pages 1161–1170, 2016.

[17] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. Gaze prediction in dynamic 360° immersive videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5342, 2018.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Xianglong Feng, Viswanathan Swaminathan, and Sheng Wei. Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–22, 2019.

[20] Xianglong Feng, Zeyang Bao, and Sheng Wei. Exploring cnn-based viewport prediction for live virtual reality streaming. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 183–1833, 2019.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[22] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, 2020.

[23] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4):325–338, 2015.

[24] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. From theory to practice: Improving bitrate adaptation in the dash reference player. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(2s):1–29, 2019.

[25] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210, 2017.

[26] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. Learning in situ: a randomized experiment in video streaming. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 495–511, 2020.

[27] Yan Ye, Elena Alshina, and Jill Boyce. *JVET-E1003: Algorithm descriptions of projection format conversion and video quality metrics in 360Lib*, 07 2018.

[28] PanoTools wikipedia. Cubic projection. https://wiki.panotools.org/Cubic_Projection, 2019.

[29] Facebook Technologies. Enhancing high-resolution 360 streaming with view prediction. https://engineering.fb.com/2017/04/19/virtual-reality/enhancing-high-resolution-360-streaming-with-view-prediction/, 2017.

[30] Jian-Liang Lin, Ya-Hsuan Lee, Cheng-Hsuan Shih, Sheng-Yen Lin, Hung-Chih Lin, Shen-Kai Chang, Peng Wang, Lin Liu, and Chi-Cheng Ju. Efficient projection and coding tools for 360° video. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):84–97, 2019.

[31] HTC Corporation. Find the right immersive experience for you. https://www.vive.com/us/product/, 2021.

[32] Sony Interactive Entertainment LLC. Immerse yourself in incredible virtual reality games and experiences. https://www.vive.com/us/, 2021.

[33] Facebook Technologies. An all-in-one headset filled with hundreds of unique experiences. https://www.oculus.com/quest-2/, 2021.

[34] Google VR. Experience virtual reality in a simple, fun, and affordable way. https://arvr.google.com/intl/en_us/cardboard/, 2021.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[36] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. A dataset for exploring user behaviors in vr spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 193–198, 2017.

[37] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.