VARFVV: View-Adaptive Real-Time Interactive Free-View Video Streaming with Edge Computing

Qiang Hu, Qihan He, Houqiang Zhong, Guo Lu, Xiaoyun Zhang, Guangtao Zhai, Yanfeng Wang

***Abstract*—Free-view video (FVV) allows users to explore immersive video content from multiple views. However, delivering FVV poses significant challenges due to the uncertainty in view switching, combined with the substantial bandwidth and computational resources required to transmit and decode multiple video streams, which may result in frequent playback interruptions. Existing approaches, either client-based or cloud-based, struggle to meet high Quality of Experience (QoE) requirements under limited bandwidth and computational resources. To address these issues, we propose VARFVV, a bandwidth- and computationally-efficient system that enables real-time interactive FVV streaming with high QoE and low switching delay. Specifically, VARFVV introduces a low-complexity FVV generation scheme that reassembles multiview video frames at the edge server based on user-selected view tracks, eliminating the need for transcoding and significantly reducing computational overhead. This design makes it well-suited for large-scale, mobile-based UHD FVV experiences. Furthermore, we present a popularity-adaptive bit allocation method, leveraging a graph neural network, that predicts view popularity and dynamically adjusts bit allocation to maximize QoE within bandwidth constraints. We also construct an FVV dataset comprising 330 videos from 10 scenes, including basketball, opera, etc. Extensive experiments show that VARFVV surpasses existing methods in video quality, switching latency, computational efficiency, and bandwidth usage, supporting over 500 users on a single edge server with a switching delay of 71.5ms. Our code and dataset will be publicly available.**

***Index Terms*—free-view video, view-adaptive streaming, QoE, bit allocation, computational complexity, real-time.**

I. Introduction

INTERACTIVE free-view video (FVV) is an emerging technology that allows users to freely choose their viewing angle as if they were present in the scene. FVV captures dynamic 3D scenes using multiple closely spaced, time-synchronized cameras, encoding the data into multiple video streams to provide panoramic scenes. This technology enables immersive live streaming with smooth inter-view switching and dynamic bullet-time effects, making it highly promising for large-scale broadcasts such as concerts, sports, and interactive teaching.

Delivering FVV over mobile networks faces several challenges regarding video quality, view switching delays, transmission bandwidth, and computational resources, which are often in conflict with one another. Achieving high-quality video requires increased bandwidth and processing power to manage tasks such as loading, decoding, and rendering video streams. However, the inherent limitations of mobile networks and devices restrict their capacity to provide sufficient bandwidth and processing power, particularly in bandwidth-limited or resource-constrained environments. As video quality improves (e.g., higher bitrate or resolution), the bandwidth needed for transmission increases proportionally, which can result in network congestion and playback interruptions. Additionally, frequent view switching demands real-time data loading, decoding, and rendering, imposing substantial computational strain on both servers and mobile devices and often leading to latency and degraded user experience. Efforts to optimize one aspect, such as reducing bandwidth usage or minimizing switching delays, may inadvertently compromise video quality or increase computational demands, thereby intensifying performance issues. These intricate trade-offs underscore the necessity of developing effective strategies to balance conflicting factors and to ensure seamless and efficient delivery of FVV over mobile networks.

Early methods [1], [2] for FVV systems typically involve transferring all views to the client, either encoded together or separately. Although these approaches ensure that all views are available, they are highly inefficient, leading to excessive bandwidth consumption and high decoding burdens on the client side. Such inefficiencies often degrade the Quality of Experience (QoE), particularly in bandwidth-constrained or resource-limited scenarios. In response, more recent

Qiang Hu, Houqiang Zhong, Guo Lu, Xiaoyun Zhang, Guangtao Zhai, and Yanfeng Wang are with the School of Electronics Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China.

Qihan He is with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China

Manuscript received May 15, 2024.

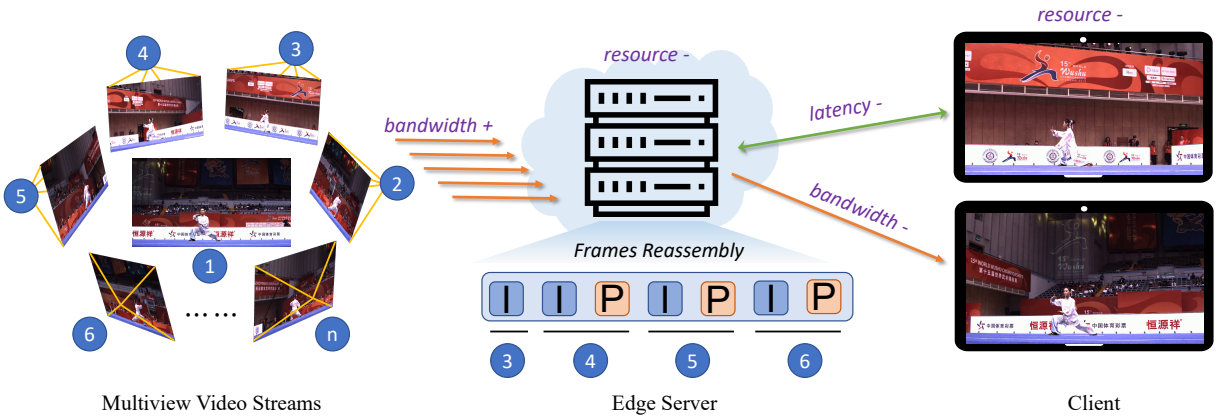


Fig. 1. Demonstration of our VARFVV. VARFVV is a bandwidth-efficient and low-complexity interactive FVV streaming system and employs an edge server to reassemble multiview frames continuously in time and inter-view based on the user's view track to obtain the FVV stream. VARFVV further delivers the generated FVV stream to the user with low latency through WebRTC. This approach significantly reduces transmission bandwidth and computational resources needed at the edge server and client.

solutions, such as HTTP Adaptive Streaming (HAS) or DASH-based techniques [3]–[6], dynamically fetch some videos only adjacent to the current view, instead of all views, to reduce bandwidth usage. The video client, however, needs to clear the current buffer and re-buffer a fixed number of new frames before playback resumes when a user continuously and rapidly switches views. The delay in resuming video playback often has a negative impact on the viewing experience.

Cloud-based delivery systems [7], [8] have been developed to optimize transmission bandwidth and reduce client-side computational complexity in FVV streaming. These systems decode multiple video streams at the edge server and employ dedicated encoders to transcode the user-selected views, effectively minimizing bandwidth consumption and ensuring smooth view switching. However, as the number of users grows, the reliance on individual encoders for each user substantially increases the computational load on the server, leading to scalability and resource challenges. For example, an NVIDIA RTX4000 GPU with a hardware-based encoder (NVENC) can encode approximately 20 videos at 1080P@25FPS or 5 videos at 4K@25FPS in real-time simultaneously[1]. Supporting 500 users at 1080P@25FPS would require approximately 25 RTX4000 GPUs. Therefore, it is essential to explore more efficient strategies for FVV live systems.

In this paper, we propose a novel view-adaptive real-time interactive FVV streaming system with edge computing or VARFVV, which achieves low switching delay and high QoE while maintaining low-cost computation and transmission, as illustrated in Fig. 1. We realize this through two main innovations. First, we design a low-complexity FVV generation scheme that reassembles multiview video frames at the edge server based on user-selected view tracks, avoiding the need for transcoding. This reassembly approach greatly accelerates the generation process compared to traditional transcoding-based cloud methods. The generated FVV stream is then delivered to users in real-time through WebRTC [9], ensuring minimal latency. By reassembling and transmitting a single video stream to the client, VARFVV significantly reduces both transmission bandwidth consumption and the computational load on the server and client, enabling the system to efficiently support a large number of users simultaneously experiencing high-quality UHD FVV live services on mobile devices.

Second, we introduce a popularity-adaptive bit allocation scheme to further enhance the overall QoE within a constrained bit budget. Our approach begins by formulating a QoE-aware optimization problem aimed at maximizing the user's QoE. To solve this problem, we introduce a graph neural network (GNN) based view popularity prediction method, where each view is modeled as a vertex within VARFVV. Transitions between vertices are used to characterize users' view trajectories, allowing us to estimate the popularity of each view. Based on this prediction, we design a novel bit allocation scheme that dynamically allocates more bits to perceptually popular views by borrowing bits from less popular ones. This adaptive approach optimizes the video quality of the most viewed perspectives, ensuring that the system maximizes the perceived visual quality for users and effectively balances bit resources to enhance the overall QoE.

In summary, our contributions are as follows:

- We propose VARFVV, a bandwidth- and computationally-efficient system that enables

[1]https://developer.nvidia.com/nvidia-video-codec-sdk

TABLE I
COMPARISON OF FVV STREAMING METHODS ACROSS KEY DIMENSIONS. OUR VARFVV ACHIEVES LOW VIEW SWITCHING LATENCY AND BANDWIDTH USAGE, HIGH VIDEO QUALITY, AND MINIMAL RESOURCE USAGE AT BOTH THE CLIENT AND EDGE SERVER.

Method	View Switching Latency↓	Video Quality↑	Bandwidth Usage↓	Client Computing Resource↓	Edge Server Computing Resource↓
HASFVV [4]	High	Low	High	High	-
Yao *et al.* [5]	High	Low	High	High	-
DASHFVV [23]	High	Low	High	High	-
EdgeEncodingFVV [7]	Low	High	Low	Low	High
HybridFVV [8]	Low	High	Low	Low	High
Hu *et al.* [31]	Low	High	Low	Low	High
VARFVV (Ours)	**Low**	**High**	**Low**	**Low**	**Low**

real-time interactive FVV streaming with high QoE and low switching delay, providing a scalable solution for delivering immersive FVV services in real-world scenarios.

- We introduce a low-complexity FVV generation scheme that reassembles multiview video frames instead of transcoding, significantly reducing the computational load on the edge server. Experimental results show that a single edge server with an AMD Ryzen 7 3700 CPU@3.6 GHz can support over 500 users simultaneously experiencing FVV.
- We develop a spatial-temporal graph network to predict view popularity and implement a popularity-based bit allocation algorithm, maximizing overall QoE by prioritizing frequently viewed perspectives.

II. RELATED WORK

A. Interactive Free-View Video Streaming

Early methods [1], [2] achieve free-view video by transferring all video streams to a client. Even though the multiview videos can be compressed by exploiting the dependencies [10], [11], transmitting all videos to a client is not optimal because the client only uses a small portion of them. Receiving all video streams also results in high bandwidth and consumption for the client. Some methods [12], [13] employ redundant coding structures, where each original frame is encoded into multiple versions, appropriately trading off expected transmission rate with storage, to facilitate view switching. However, in scenarios where clients frequently switch views, both methods require re-buffering a fixed number of new frames to resume playback.

To enhance bandwidth efficiency, adaptive streaming techniques [3]–[6], [14]–[23] have been proposed for multiview and free-view video systems. These techniques enable the transmission of videos with varying numbers and bitrates. For instance, Zhang *et al.* [4] address the limitation of equal bitrate by jointly optimizing reference views and bitrates to select the optimal subset of different views and bitrates. A DASH-based FVV streaming client is proposed to reduce view switching latency by prefetching a set of adjacent images [23]. However, even with the simultaneous use of 9 decoders, the view switching time remains at least 800 ms. With the advancement of deep learning technologies, several methods [24]–[30] have leveraged deep learning-guided resource multiplexing and view semantic information to further enhance bandwidth efficiency. Although the adaptive streaming approaches partially reduce the bandwidth and client-side computation, they still needs to receive and decode redundant views.

Cloud-based delivery systems [7], [8], [31] offer a promising solution by transferring bandwidth and computational pressure to the server. In these systems, the edge server uses dedicated encoders to transcode the user-selected view into an FVV stream, transmitting only the requested target stream to the client. For example, Dong *et al.* [7] propose an elastic system architecture with low-latency features to support generic interactive video applications at near-user edges by extending the directed acyclic graph model with switch vertices. However, when the number of users is large, the edge server may face significant computing resource demands for transcoding. In contrast, our method focuses on reassembling video frames instead of transcoding them at the edge, thus maintaining the original video quality without imposing heavy computational demands. Tab. I presents a comparative analysis of FVV streaming methods across key performance metrics: view switching latency, video quality, bandwidth usage, client computing resources, and edge server computing resources. The proposed VARFVV achieves low view switching latency and bandwidth usage while maintaining high video quality with minimal resource requirements for both the client and the edge server.

B. Bit Allocation

Optimal bit allocation schemes for video coding with a given visual attention have been investigated in [32]–

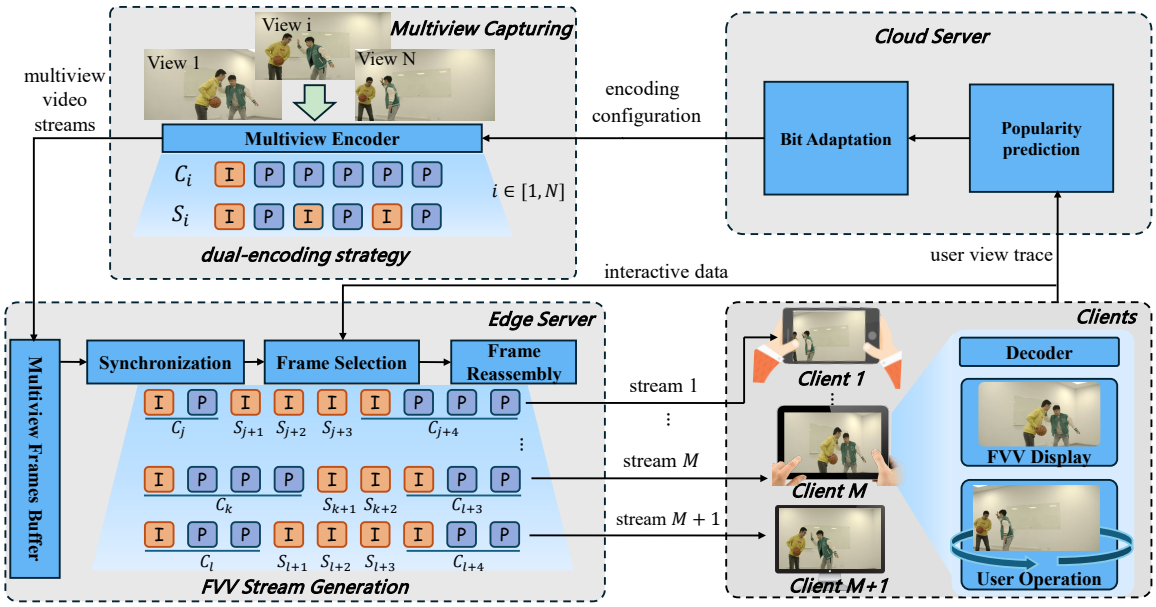


Fig. 2. Overview of the VARFVV System Architecture. Our VARFVV integrates multiview encoding, popularity-adaptive bit allocation, edge-side FVV stream generation, and client-side decoding to deliver high-quality, real-time FVV services with low computational and transmission costs.

[41]. By allocating more bits to attended or popular regions, it is possible to design video coding algorithms that can improve visual quality while meeting bandwidth constraints. Gitman *et al.* [33] develop a gaze-based video coding framework to achieve high-quality salient regions. For 360° video delivery, some work [42]–[47] proposes viewport-adaptive 360° video streaming systems to reduce the bandwidth waste. Chen *et al.* [45] introduce a viewport-aware upstream and downstream bit allocation algorithm to maximize users' QoE.

Accurately predicting view popularity is important for determining the optimal bit allocation strategy. Chakareski *et al.* [48] present a view popularity-aware bit allocation algorithm to maximize the expected video quality, but their popularity prediction method only relies on statistical data and ignores temporal characteristics. To address this, some studies use the Long Short-Term Memory (LSTM) network to predict future video popularity based on historical data [49]–[53]. For example, Maniotis *et al.* [51] employ an LSTM network to forecast the popularity evolution of the content requests and prefetch content for caching. However, LSTM-based methods focus primarily on temporal patterns, often neglecting spatial relationships among tiles that are critical for accurate predictions in 360° video.

Recent advancements in graph-based modeling for 360° video have demonstrated superior performance in predicting user viewport changes by integrating spatial and temporal dependencies. Hu *et al.* [54] model changes in user viewport trajectories as a directed weighted graph, leveraging the tile transition probability matrix to predict tile view popularity. Similarly, Xu *et al.* [55] model user viewport transitions using a graph structure and further incorporates GNNs to fuse multiple features for predicting tile popularity. These graph-based approaches capture spatial and temporal dependencies more effectively, significantly improving the accuracy of popularity predictions and enabling efficient resource allocation in 360° video streaming systems. In this paper, we extend the concept of graph-structured modeling from 360° video to FVV by representing view transitions as a weighted directed graph. Leveraging a GNN-based popularity prediction algorithm and a popularity-adaptive bit allocation strategy, our approach integrates spatial and temporal dynamics to maximize QoE under bandwidth constraints.

III. System Design and Problem Formulation

A. System Overview

The VARFVV system is designed to deliver high-quality, real-time FVV services in a cost-effective manner by integrating multiview encoding, popularity-adaptive bit allocation, edge-side FVV stream generation, and client-side decoding within a unified architecture, as shown in Fig. 2. The system captures dynamic 3D scenes from multiple views using a synchronized array of ZCAM[2] cameras, and the captured views are

[2]http://www.z-cam.com/e2/

then encoded via the "**Multiview Encoder**" using the H.264 codec through the FFmpeg[3] library. Each view (v_i) is encoded into two formats: *view-switching representation* (S_i) for rapid view changes and *view-constant representation* (C_i) for stable perspectives. This dual-encoding approach forms the foundation for continuous and seamless view switching.

To optimize the QoE for users, the system employs a popularity-adaptive bit allocation algorithm on the cloud server. The "**Popularity Prediction**" module analyzes limited historical user interaction data to forecast the future popularity of various views. According to the predicted popularity of each view, the "**Bit Adaptation**" component periodically updates the bit allocation of the multiview encoder. By borrowing bits more aggressively from perceptually less important views and allocating them to perceptually popular views, we improve the overall quality of FVV transmitted to users while meeting bandwidth constraints.

The edge-side FVV generation scheme is crucial for minimizing view switching delays and reducing transmission and computational costs. The "**Synchronization**" module aligns video frames from multiple views using their Presentation Time Stamps (PTS) to ensure accurate synchronization. When users interact with the system, such as swiping or switching views, the "**Frame Selection**" and "**Frame Reassembly**" components dynamically select and reassemble the appropriate frames into an FVV stream based on the user's chosen trajectory. Since our system only reassembles the video frames rather than performing transcoding, it maintains low computational overhead.

The generated FVV stream is transmitted to the client using WebRTC technology, which guarantees low-latency delivery and immediate response to user interactions. The client-side decoding module processes the incoming stream locally, sending interaction signals to the server as required. With the VARFVV design, our system efficiently enables users to enjoy high-quality UHD FVV live services on mobile devices over mobile networks. Detailed implementation specifics are available in Section V. Tab.II summarizes the main notations used in the following sections.

B. FVV Stream Generation

In traditional cloud-based delivery systems [7], [8], [31], each user request triggers re-encoding of the video stream on the server, leading to significant computational overhead. To alleviate this, we propose a low-complexity FVV stream generation scheme that reassembles the I-frames and P-frames from the multiview video stream at the edge server based on user-selected view paths. In video coding, I-frames are independently decodable, while P-frames rely on previous I/P-frames for decoding [56], [57]. Consequently, view switching requires waiting for the next view's I-frame, with longer I-frame intervals causing delays that degrade the user experience. To enable random access for low-latency view switching, multiview videos can be encoded with high-density I-frames at the acquisition stage. However, as I-frames require more bits than P-frames to achieve the same quality, transmitting high-density I-frames results in wasted bandwidth when the user's view is constant.

To address this issue, we introduce a dual-encoding strategy at the acquisition stage, where each view v_i ($1 \leq i \leq N$) is encoded into two representations: a *view-switching representation* labeled as S_i and a *view-constant representation* labeled as C_i, as shown in Fig. 3. The former uses a static group of pictures (GoP) size 2 ($\hat{g}_{i,j} = 2$) and is used for view-switching scenarios, while the latter uses a static GoP size 25 ($g_{i,j} = 25$) and is used for view-constant scenarios. The encoded multiview streams are transmitted to the edge server, where they are dynamically reassembled based on the user's view paths. When a user's current

[3]https://ffmpeg.org/

TABLE II
NOTATION

Symbol	Physical Meaning
v_i	The i-th view
S_i	The *view-switching representation* of v_i
C_i	The *view-constant representation* of v_i
$S_{i,j}$	The j-th video chunk in S_i
$C_{i,j}$	The j-th video chunk in C_i
$x_{i,j}$	Actual popularity of $C_{i,j}$
$\hat{x}_{i,j}$	Actual popularity of $S_{i,j}$
$p_{i,j}$	Predicted popularity of $C_{i,j}$
$\hat{p}_{i,j}$	Predicted popularity of $S_{i,j}$
$R_{i,j}$	Allocated bits for $C_{i,j}$
$\hat{R}_{i,j}$	Allocated bits for $S_{i,j}$

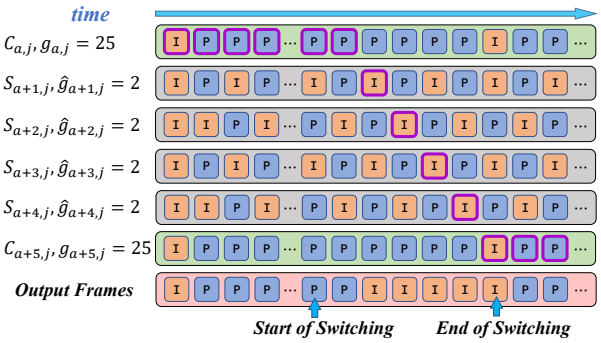


Fig. 3. GoP structure design for each stream and view switching strategy.

view is fixed at v_i for the j-th video chunk, the edge server extracts the compressed video frames directly from $C_{i,j}$ as the output frames and transmits them to the user. If the user switches the view from v_i to v_m, the edge server immediately selects a time-continuous I-frame from $S_{i+1,j}$ to $S_{m,j}$ and reassembles these frames consecutively in time to generate the FVV stream transmitted to the user. After view switching is completed, the edge server continues to multiplex the video frames in $S_{m,j}$ as the output frame until it encounters a time-synchronized I-frame in $C_{m,j}$, at which point it starts to reuse the frames in $C_{m,j}$. This process enables continuous, seamless, and fast switching of user views with an almost negligible increase in computational complexity. It is worth noting that when i is an even number, we insert an I-frame at the starting position of $S_{i,j}$ so that the I-frames between two adjacent *view-switching representations* are staggered, which effectively reduces the switching delay.

C. QoE Metrics

We aim to maximize the QoE with limited bandwidth resources by determining the bit allocation for each view at a specific video chunk j. $R_{i,j}$ and $\hat{R}_{i,j}$ denote the allocated bits for the j-th video chunk in C_i and S_i, respectively, where $i \in \{1, 2, \cdots, N\}$ is the view number. We model the total QoE of FVV for a video chunk j as follows:

$$\begin{aligned}&\mathbf{QoE_j}(R_{1,j}, \ldots, R_{N,j}, \hat{R}_{1,j}, \ldots, \hat{R}_{N,j}) = \\ &\mathbf{QoE_{1,j}} - \mu_1 \mathbf{QoE_{2,j}} - \mu_2 \mathbf{QoE_{3,j}}\end{aligned} \tag{1}$$

where $\mathbf{QoE_{1,j}}$ denotes the QoE on video quality, $\mathbf{QoE_{2,j}}$ and $\mathbf{QoE_{3,j}}$ represent inter-view and temporal quality switching, respectively. These three QoE features are essential to characterizing the QoE of FVV. μ_1 and μ_2 are weighted factors.

Video Quality. Following the previous work [54], [58], we adopt a logarithmic model to depict the relationship between QoE and allocated bits.

$$\mathbf{QoE_{1,i,j}} = log(1 + R_{i,j}/\eta) \tag{2}$$

where η is a model parameter. Considering all *view-switching representations* and *view-constant representations*, we obtain the QoE on video quality for all views as follows:

$$\mathbf{QoE_{1,j}} = \sum_{i=1}^{N} x_{i,j} log(1+R_{i,j}/\eta) + \sum_{i=1}^{N} \hat{x}_{i,j} log(1+\hat{R}_{i,j}/\hat{\eta}) \tag{3}$$

where $x_{i,j}$ and $\hat{x}_{i,j}$ are the view popularity of C_i and S_i in the j-th video chunk, respectively. $\hat{\eta}$ is a model parameter. As $x_{i,j}$ and $\hat{x}_{i,j}$ are unknown a prior, we propose a novel approach in Section IV-A that leverages historical view requests to obtain the predited popularity $p_{i,j}$ and $\hat{p}_{i,j}$. Then Eq. 3 can be written as:

$$\mathbf{QoE_{1,j}} = \sum_{i=1}^{N} p_{i,j} log(1+R_{i,j}/\eta) + \sum_{i=1}^{N} \hat{p}_{i,j} log(1+\hat{R}_{i,j}/\hat{\eta}). \tag{4}$$

Inter-view Quality Switching. When users switch views in an FVV system, they encounter variations in video quality due to the transitions between different view representations, each potentially encoded with varying bitrates. These transitions can occur between two consecutive *view-switching representations* or between a *view-constant representation* and a *view-switching representation*. Excessive bitrate changes between two consecutive representations can potentially lead to degraded visual quality when switching views. To measure and control these variations, we define the inter-view quality switching metric as follows:

$$\begin{aligned}\mathbf{QoE_{2,j}} = \mu_3 &\sum_{i=2}^{N} \hat{p}_{i,j} (\hat{R}_{i,j} - \hat{R}_{i-1,j})^2 + \\ &\sum_{i=2}^{N} \hat{p}_{i,j} (\hat{R}_{i,j}/\hat{\eta} - R_{i-1,j}/\eta)^2\end{aligned} \tag{5}$$

where μ_3 is a weighted factor. The first term in the equation quantifies the quality variation between consecutive *view-switching representations*, calculating the squared difference in bitrate between two successive view-switching events, weighted by the predicted popularity $\hat{p}_{i,j}$. The second term evaluates the transition between a *view-constant representation* and a *view-switching representation*. The metric $\mathbf{QoE_{2,j}}$ is designed to capture the magnitude of bitrate changes between consecutive representations, enabling the system to monitor and regulate these fluctuations, thus ensuring visual consistency and improving the overall QoE for users.

Temporal Quality Switching. Temporal quality switching refers to the variation in video quality between consecutive video chunks when the user's viewing angle remains fixed, as represented in the *view-constant representation* C_i. Significant changes in the bitrate between these consecutive chunks can cause discomfort or adverse physiological effects, such as dizziness or motion sickness, particularly in immersive environments like FVV. To accurately quantify these variations, we define the temporal quality switching metric as follows:

$$\mathbf{QoE_{3,j}} = \sum_{i=1}^{N} p_{i,j} (R_{i,j} - R_{i,j-1})^2. \tag{6}$$

where $R_{i,j}$ and $R_{i,j-1}$ denote the bitrate of the current and previous video chunks, respectively. The squared difference between the bitrates of consecutive video

chunks reflects the magnitude of the variation, with larger fluctuations penalized more heavily. This design prioritizes video quality stability over time, reducing the risk of visual inconsistency.

D. Optimization Problem Formulation

We can now formulate a QoE-aware optimization problem for FVV adaptive bit allocation.

$$\begin{aligned} \textbf{arg} \quad & \textbf{max} \quad \mathbf{QoE_j}(R_{1,j}, \ldots, R_{N,j}, \hat{R}_{1,j}, \ldots, \hat{R}_{N,j}) \\ \textbf{s.t.} \quad & \sum_{i=1}^{N} R_{i,j} + \hat{R}_{i,j} \leq R_j \\ & R_{\mathbf{min}} \leq R_{i,j} \leq R_{\mathbf{max}}, \hat{R}_{\mathbf{min}} \leq \hat{R}_{i,j} \leq \hat{R}_{\mathbf{max}} \end{aligned} \tag{7}$$

The objective of our adaption logic is to optimize the bit allocation $R_{i,j}$ and $\hat{R}_{i,j}$ ($i \in \{1, 2, \cdots, N\}$) for a video chunk j, so that the total QoE measured on the j-th video chunk is maximized. The constraint restricts an upper bound on the sum of allocated bits R_j, and requires that the allocated bits $R_{i,j}$ and $\hat{R}_{i,j}$ should not fall below $R_{\mathbf{min}}$ and $\hat{R}_{\mathbf{min}}$, respectively, which correspond to the lowest available video quality. To prevent excessive use of bits, $R_{i,j}$ and $\hat{R}_{i,j}$ should also not exceed $R_{\mathbf{max}}$ and $\hat{R}_{\mathbf{max}}$. The optimal bit allocation problem in Eq. 7 can be broken down into two parts. The first is to predict the view popularity $p_{i,j}$ and $\hat{p}_{i,j}$, while the second is to assign bit based on the predicted view popularity. In Section IV, we propose a tractable approach to solve Eq. 7.

Unlike [54], which focuses on tile-based bitrate allocation for 360° video streaming, our approach is tailored for bitrate allocation across multiple camera views in FVV streaming. This shift from tile-level optimization to inter-view transitions requires additional constraints and metrics specifically designed for dynamic view switching.

IV. Optimal Bit Allocation

A. Popularity Prediction

Here, we introduce a novel method based on historical data to predict view popularity $p_{i,j}$ and $\hat{p}_{i,j}$. For *view-constant representations*, where users typically maintain a fixed viewing angle after selecting a preferred view, a baseline method leverages the ground truth popularity of each view ($x_{i,j-1}$) obtained from the previous video chunk as the predicted popularity of the next video chunk ($p_{i,j}$). We refer to this method as the previous popularity carryover (PPC) model.

However, for *view-switching representations*, the popularity of views can change rapidly as users switch their focus between different views, resulting in highly nonlinear and complex patterns. Traditional methods like LSTM networks [49], [50] capture temporal dynamics from historical data, but they fail to account for the spatial relationships between views, which are crucial for accurate view popularity prediction in FVV. Inspired by recent works [54], [55] that model user viewport changes as directed weighted graphs in 360° video tiles, we extend Guo *et al.*'s attention-based spatial-temporal GNN approach [59], originally designed for traffic intersection prediction, to FVV, enabling the capture of both temporal and spatial dependencies.

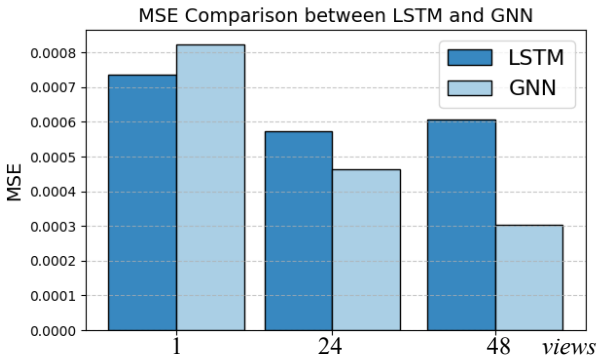


Fig. 4. Comparison of prediction accuracy between LSTM and GNN methods for view popularity prediction across different numbers of views.

We preliminarily compare the performance of LSTM and GNN methods for predicting view popularity across varying numbers of views (single view, 24 views, and 48 views) in Fig. 4. The results demonstrate that LSTM's accuracy does not improve with an increasing number of views, due to its limited ability to capture spatial relationships between views. In contrast, GNN's graph-based modeling effectively addresses this issue, leading to more accurate predictions. Our adapted GNN effectively captures both spatial correlations between views and temporal dynamics of user preferences, enabling accurate prediction of $\hat{p}_{i,j}$ for dynamic view-switching scenarios.

Our attention-based spatial-temporal GNN models each view as a vertex in the graph G, with the popularity of each view in a video chunk as an attribute associated with the vertex, as shown in Fig. 5. An undirected edge is established between vertices representing views that are adjacent based on camera connectivity. The adjacency matrix $\mathbf{A}$ is constructed to reflect these camera connections, capturing the spatial relationships between views in the graph structure. This scheme can be expressed as:

$$\begin{aligned} G &= (V, E, \mathbf{A}) \\ |V| &= N, \mathbf{A} \in \mathbb{R}^{N \times N} \end{aligned} \tag{8}$$

where N is the total number of views, V represents the

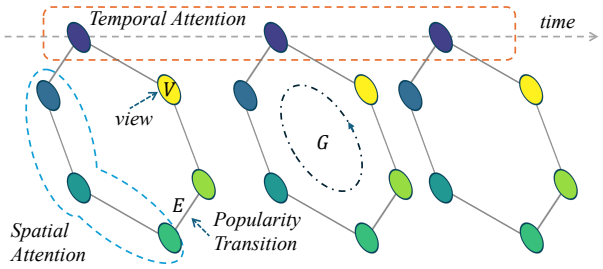


Fig. 5. Graph-based model for view popularity prediction with spatial and temporal attention.

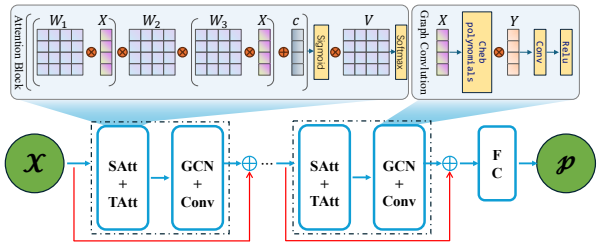


Fig. 6. Popularity prediction network structure. $\mathcal{X}$ and $\mathcal{P}$ denote the historical and future popularity of various views, respectively, with red arrows indicating residual connections.

set of vertices, E denotes the edges based on the logical view switching between cameras. For each frame, spatial attention is applied to aggregate information from neighboring nodes, while temporal attention aggregates data from the same node across different time steps. The attention mechanism enables the model to prioritize relevant spatial and temporal features, improving its ability to capture spatial correlations (e.g., adjacent views) and temporal dynamics (e.g., changes in user preferences). This design ensures that the model remains robust and effective, even in scenarios with sudden changes in user behavior, as verified in the experimental section.

We represent the popularity of S_i in the j-th chunk as $\hat{x}_{i,j} \in \mathcal{R}, i \in N$. $\mathbf{X}_j = (\hat{x}_{1,j}, \hat{x}_{2,j}, \cdots, \hat{x}_{N,j})$ denotes the popularity of all nodes for the j-th video chunk. $\mathcal{X}_\tau = (\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_\tau)$ represents the historical popularity of all nodes over τ chunks. As shown in Eq. 9, graph neural network F maps the historical popularity $\mathcal{X}_\tau$ to future view popularity sequences $\mathcal{P}^d$ over the next d chunks. $\mathcal{P}^d = (\mathbf{P}_{\tau+1}, \mathbf{P}_{\tau+2}, \cdots, \mathbf{P}_{\tau+d})$, where $\mathbf{P}_{\tau+1} = (\hat{p}_{1,\tau+1}, \hat{p}_{2,\tau+1}, \cdots, \hat{p}_{N,\tau+1})$.

$$\mathcal{P}^d = F(\mathcal{X}_\tau) \tag{9}$$

Fig. 6 shows the architecture of the popularity prediction module, which comprises two main components: spatial-temporal attention and graph convolution. Specifically, the attention part can be exemplified by the spatial attention, which includes a set of learnable parameters denoted as $\mathbf{W}_y$, $\mathbf{c}_y$, and $\mathbf{V}_y$ in the network layer.

$$\mathbf{Y} = \sigma((\mathcal{X}_\tau \mathbf{W}_{y,1})\mathbf{W}_{y,2}(\mathbf{W}_{y,3}\mathcal{X}_\tau)^T + \mathbf{c}_y)\mathbf{V}_y \tag{10}$$

$$e^{\mathbf{Y}}_{i,k} = exp(\mathbf{Y}_{i,k}) \tag{11}$$

$$\mathbf{Y}'_{i,k} = \frac{e^{\mathbf{Y}}_{i,k}}{\sum_{k=1}^{N} e^{\mathbf{Y}}_{i,k}} \tag{12}$$

where $\mathbf{Y}' \in \mathcal{R}^{N \times N}$ is the spatial attention matrix of graph nodes. It is obtained by applying the attention mechanism to the graph's adjacency matrix $\mathbf{A}$ and can dynamically adjust the weights between vertices during graph convolutions. Temporal attention is similar to spatial attention.

$$\mathbf{Z} = \sigma((\mathcal{X}_\tau \mathbf{W}_{z,1})\mathbf{W}_{z,2}(\mathbf{W}_{z,3}\mathcal{X}_\tau)^T + \mathbf{c}_z)\mathbf{V}_z \tag{13}$$

$$\mathbf{Z}'_{i,k} = \frac{e^{\mathbf{Z}}_{i,k}}{\sum_{k=1}^{N} e^{\mathbf{Z}}_{i,k}} \tag{14}$$

where $\mathbf{W}_z, \mathbf{c}_z, \mathbf{V}_z$ are learnable parameters. $\mathbf{Z}'$ is the temporal attention matrix in a time slice.

Similar to [59], we define a simplified spatial graph convolution as:

$$g_s * G(\mathcal{X}_\tau) = \sum_{m=1}^{M} s_m(T_m(\tilde{L}) \odot \mathbf{Y}')\mathcal{X}_\tau \tag{15}$$

where $*G$ represents a graph convolution operation(GCN). M denotes the number of neighbours used in the convolution, s is the convolution kernel parameter, and $T_m(\tilde{L})$ is the Chebyshev polynomial term defined on $\mathcal{X}$. To incorporate temporal information, we apply the temporal attention matrix $\mathbf{Z}'$ to $g_s * G(\mathcal{X}_\tau)$, and perform a standard convolution operation in a time chunk using $g_t * G$. Multiple spatial-temporal blocks are stacked to extract a larger range of dynamic spatial-temporal correlations. Finally, a fully connected layer with a ReLU activation function is appended to ensure that the output of each component has the same dimension and shape as the forecasting target.

$$\mathcal{P}^d = \mathbf{W} \odot \text{Relu}(\text{FC}(g_t * G(\mathbf{Z}' g_s * G(\mathcal{X}_\tau)))) \tag{16}$$

where $\mathbf{W}$ is a learnable parameter. We exploit a mean absolute error (MAE) loss function to learn the parameters of GNN F.

$$\mathcal{L}(j) = \frac{1}{N}\sum_{i=1}^{N} |\hat{p}_{i,j} - \hat{x}_{i,j}| \tag{17}$$

B. Popularity-adaptive Bit Allocation

With the predicted popularity of video chunks, we propose a bit allocation scheme for each representation of VARFVV. We construct the Lagrange function with

the Karush-Kuhn-Tucker (KKT) conditions to solve Eq. 7, as:

$$
\begin{aligned}
&L(\lambda, R_{1,j}, \ldots, R_{N,j}, \hat{R}_{1,j}, \ldots, \hat{R}_{N,j}) \\
&= \sum_{i=1}^{N} p_{i,j} log(1 + R_{i,j}/\eta) + \sum_{i=1}^{N} \hat{p}_{i,j} log(1 + \hat{R}_{i,j}/\hat{\eta}) \\
&- \mu_1 \cdot \mu_3 \sum_{i=2}^{N} \hat{p}_{i,j} (\hat{R}_{i,j} - \hat{R}_{i-1,j})^2 \\
&- \mu_1 \cdot \sum_{i=2}^{N} \hat{p}_{i,j} (\hat{R}_{i,j}/\hat{\eta} - R_{i-1,j}/\eta)^2 \\
&- \mu_2 \sum_{i=1}^{N} p_{i,j} (R_{i,j} - R_{i,j-1})^2 \\
&+ \lambda \left(R_j - \sum_{i=1}^{N} (\hat{R}_{i,j} + R_{i,j}) \right)
\end{aligned} \tag{18}
$$

where λ is the Lagrange multiplier, and R_j is the target number of bits for j-th time slot.

$$
\begin{aligned}
R_j &= \frac{R_{avg} \times (N_{coded} + SW) - R_{coded}}{SW} \\
R_{avg} &= R_{tar} \times T_d
\end{aligned} \tag{19}
$$

where R_{tar} is the target bitrate for all representations of VARFVV, and T_d is the time duration of a video chunk. N_{coded} and R_{coded} denote the number of the coded time slots and bit cost for all the coded representations, respectively. SW is the size of sliding window, which aims to make bitrate adjustment smoothly. Then we let $\frac{\partial L}{\partial R_{i,j}} = 0$, $\frac{\partial L}{\partial \hat{R}_{i,j}} = 0$.

$$
\begin{aligned}
\frac{\partial L}{\partial R_{i,j}} =& \frac{p_{i,j}/\eta}{1 + R_{i,j}/\eta} - 2\mu_2 p_{i,j} (R_{i,j} - R_{i,j-1}) \\
&- \lambda = 0
\end{aligned} \tag{20}
$$

$$
\begin{aligned}
&\frac{\partial L}{\partial \hat{R}_{i,j}} = \frac{\hat{p}_{i,j}/\hat{\eta}}{1 + \hat{R}_{i,j}/\hat{\eta}} - 2\mu_1 \mu_3 \hat{p}_{i,j} (\hat{R}_{i,j} - \hat{R}_{i-1,j}) \\
&- 2\mu_1 \hat{p}_{i,j} (\hat{R}_{i,j}/\hat{\eta} - R_{i-1,j}/\eta)/\hat{\eta} - \lambda = 0
\end{aligned} \tag{21}
$$

We use the dichotomy methodology to obtain the accurate value of λ, $R_{i,j}$ and $\hat{R}_{i,j}$, as shown in Algorithm 1. Our bit allocation strategy dynamically allocates bits to each video chunk based on its popularity. The perceptually less popular views are compromised to save more bits for the views with more perceptual relevance. We can thus achieve a higher visual quality of FVV under the same target bits condition.

Algorithm 1 Popularity-adaptive Bit Allocation

Input: λ_{min}, λ_{max}, R_j, η, $\hat{\eta}$, ϵ
Output: λ, $R_{i,j}$, $\hat{R}_{i,j}$

while *Ture* **do**
  $\lambda = (\lambda_{max} + \lambda_{min})/2$
  for i = *1* : N **do**
    Derive the view popularity $p_{i,j}$, $\hat{p}_{i,j}$
  end for
  Calculate R_j by Eq. 19
  for i = *1* : N **do**
    Calculate $R_{i,j}$, $\hat{R}_{i,j}$ by solving Eq. 20, 21
  end for
  if $\sum_{i=1}^{N} (\hat{R}_{i,j} + R_{i,j}) > R_j$ **then**
    $\lambda_{min} = \lambda$
  else
    $\lambda_{max} = \lambda$
  end if
  Iteration until
    $-\epsilon R_j < R_j - \sum_{i=1}^{N} (\hat{R}_{i,j} + R_{i,j}) < \epsilon R_j$
    or achieve max iteration number
end while

V. System Implementation

A. Capturing and Adaptive Encoding

As shown in Fig.7, we adopt a large array of time-synchronized and closely spaced ZCAM cameras to capture the same dynamic 3D scene from N different views. Each camera is equipped with a hardware synchronization unit that can trigger all cameras to shoot simultaneously. After the cameras are installed, they are calibrated to determine the exact camera parameters, including internal and external parameters. To provide a seamless visual experience for clients switching between views during the FVV live-streaming service, we utilize CUDA through a C++ interface to calibrate the images at the outset.

Fig. 7. A set of multiview capturing system.

We implement "**Multiview Encoder**" using the FFmpeg library, specifically utilizing the H.264 encoder. The corrected pictures for each view are encoded into *view-switching representation* and *view-constant representation*. Each representation of the stream records a PTS for each frame, which is calculated based on the acquisition time of that frame. The choice of the H.264 encoder balances compression efficiency and computational complexity, ensuring the overall real-

time performance and stability of the FVV system.

Meanwhile, we collect users' historical trajectories. We record the number of times each view is viewed by users at each video chunk. The popularity of each view is calculated according to the number of times each view is viewed by users in each video chunk. The cloud server predicts view popularity on view-constant and view-switch representations. According to the optimal solution, VARFVV optimizes the bit allocation of each view and then updates the coding configurations. These multiview video streams are encoded and transmitted to the edge server in real-time. At this time, the bits contained in video chunks from different views are different due to the different popularity of users' viewing, and more bits are allocated to video chunks viewed by more users.

B. Synchronization and Frame Reassembly

The transmission speed of multiview video streams in the network is fluctuating, which results in variability in the time that each video stream arrives at the edge server. To avoid inconsistencies between views when switching views, we implement a real-time synchronization method that achieves high tolerance for time differences between different video streams.

We use the FFmpeg library to demultiplex the received live streams and obtain the PTS of each frame. We then synchronize frames in multiview video streams according to PTS. The specific process is as follows: (i) Create sync buffers for each live stream to receive video frames. (ii) Start a thread to synchronize the frames in the buffers based on the PTS. The frames in the sync buffers are shared by all users. We select the desired frames from the sync buffers based on each user's view trajectory and current PTS, and reassemble them into a live stream. For example, when a user's view remains unchanged, the edge selects time-consecutive frames from the *view-constant representation* of that view and reassembles them into a live stream for the user. When a user switches views, the edge selects frames from the *view-switching representation* corresponding to the user's viewing track and reassembles them into the live stream. As we only demultiplex and reassemble video streams instead of transcoding, the increase in computational complexity is almost negligible.

C. Interaction and Distribution

We implement a signaling server to enable users to establish a WebRTC connection between edge and client. When a client sends a request to the signaling server to experience the FVV service, the signaling server assigns an edge server with service resources to the client. Then a connection is established through a discovery and negotiation process. The client can request video data from the edge server through interactive operations such as play and swipe. For example, when the client swipes to the left, it sends an interaction signal to the corresponding edge server, which generates an FVV stream consistent with the user's selected trajectory in both time and view. The generated FVV stream is delivered to the client via WebRTC. Our approach extracts and transmits the desired frames based on user-selected trajectories, which effectively reduces the transmission bandwidth.

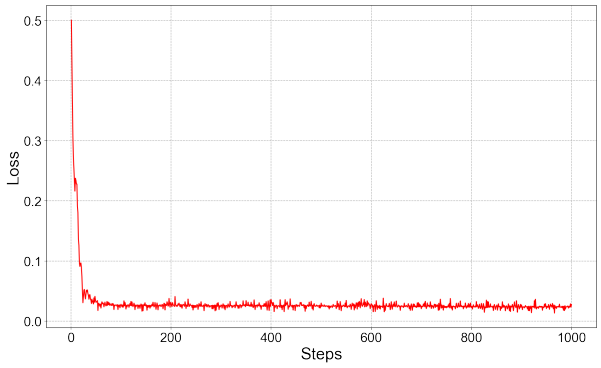


Fig. 8. Loss curve for training process.

VI. Experimental Results

A. Training

To evaluate the performance of VARFVV, we construct an FVV dataset comprising 330 videos from 10 scenes, including basketball, opera, martial arts, etc. We recruit 82 participants to use our VARFVV to freely view these videos and record their view traces. A cloud server with an AMD Ryzen 7 3700X CPU@3.60 GHz and an NVIDIA 2080ti GPU is used for popularity training and prediction. Our view popularity prediction method combines PPC and GNN, where PPC uses $x_{i,j-1}$ as $p_{i,j}$, while GNN is trained online with historical data samples of $\hat{x}_{i,j}$ during the initial 10 seconds. We configure the GNN training with $M = 2$, utilizing an MAE loss function alongside a learning rate of 0.005 and a batch size of 32. GNN undergoes 50 epochs during the initial training to achieve optimal results.

After the initial training, the obtained weights are used to predict the popularity distribution of users. During the online learning phase, each new data frame is immediately added to the training set, and the GNN model parameters are updated in real-time, ensuring continuous adaptation to the latest data. The analysis of the training process, as illustrated in Fig.8, demonstrates a consistent reduction in the MAE loss of the GNN as the number of steps increases. Notably, the loss function converges and approaches zero after only a few steps.

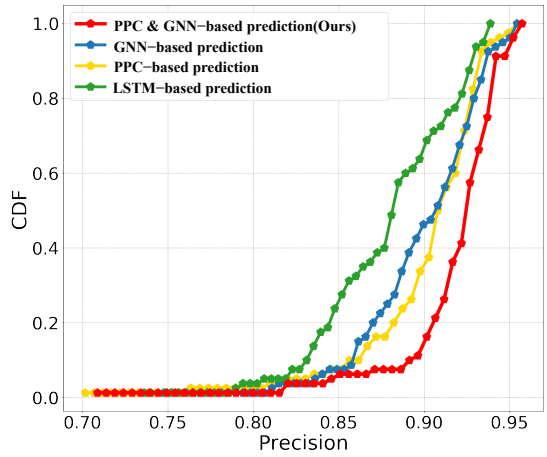


Fig. 9. The CDF of popularity prediction precision on the dataset where the length of a video chunk is one second.

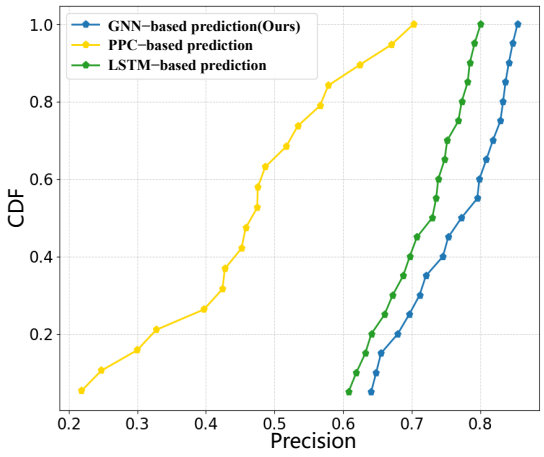


Fig. 10. The CDF of popularity prediction accuracy under sudden changes in user behavior during view switching.

TABLE III
TRAINING TIME AND PREDICTION TIME

Scene	View	Training Time	Prediction Time
Basketball	23	1.49s	0.15ms
Opera	48	1.56s	0.19ms
Martial Arts	48	1.87s	0.15ms
Teaching	13	1.52s	0.13ms
Average	-	1.61s	0.16ms

Tab. III provides insights into the training and prediction times. The training time reflects the duration to train 50 epochs on 10 seconds of data, averaging 1.61s. This indicates a swift training and update process. The average prediction time is 0.16ms, showcasing the model's real-time capability in predicting popularity distribution results.

B. Evaluation

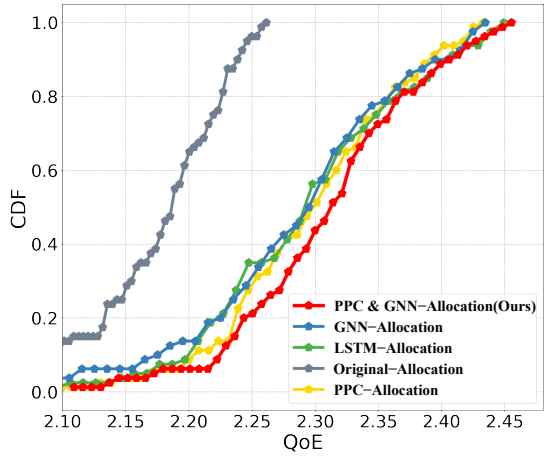
Precision of View Popularity Prediction. We evaluate our view popularity prediction method in comparison to an LSTM-based prediction method [51]. We also conduct an ablation study to individually evaluate the performance of the proposed PPC and GNN-based view popularity predictions. Inspired by [60], we use the precision of popularity prediction to investigate the performance.

$$Precision(j) = 1 - \sqrt{\frac{\sum_{i=1}^{N}(p_{i,j} - x_{i,j})^2 + \sum_{i=1}^{N}(\hat{p}_{i,j} - \hat{x}_{i,j})^2}{2N}}. \quad (22)$$

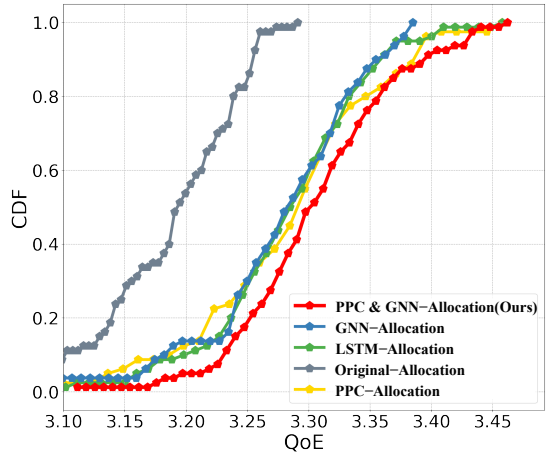
Fig. 9 demonstrates the cumulative distribution function (CDF) of precision in predicting popularity. Our popularity prediction scheme outperforms the other three schemes, primarily due to its ability to model both view-constant and view-switching scenarios. Specifically, PCC is effective for view-constant scenarios by leveraging the popularity of previous chunks, but struggles with view-switching due to its inability to capture dynamic user behavior changes. LSTM captures temporal dynamics in view-switching scenarios but fails to account for spatial relationships between views, limiting its performance. In contrast, GNN captures both spatial correlations and temporal dynamics through a graph-based representation, enabling it to better adapt to user focus shifts during view switching and achieve higher accuracy. This also aligns with the results in Fig. 4. However, GNN lacks explicit optimization for view-constant scenarios. Our method combines PCC for constant views and GNN for dynamic views, ensuring superior prediction accuracy across both contexts.

We also evaluate the performance of view popularity prediction under continuous and sudden changes in user behavior during view switching. For this experiment, we use a 20-second sequence from the opera scenario, where users continuously switch views, simulating dynamic changes in behavior. As shown in Fig. 10, our GNN-based prediction maintains competitive accuracy even under these challenging conditions, outperforming both PPC and LSTM. This is due to its ability to model spatial correlations between views and capture the dynamic changes in user preferences during view switching. However, as expected, the prediction accuracy for stable interactions (shown in Fig. 9) is higher than that for scenarios with sudden changes, due to the increased variability introduced by rapid shifts in user behavior.

User QoE. We evaluate the proposed bit allocation strategy against four alternatives: original-allocation, PPC-allocation, LSTM-allocation, and GNN-allocation.



(a) The total number of bits per second is $10 \cdot N$ Mbps ($N = 23$).



(b) The total number of bits per second is $20 \cdot N$ Mbps ($N = 23$).

Fig. 11. The CDF of user QoE on the dataset where the length of a video chunk is one second.

The original-allocation strategy distributes equal bits for all the views, while the other methods implement the popularity-adaptive bit allocation using different popularity prediction algorithms. For our method, we specify the tuning parameters for user QoE: $\eta = 1, \hat{\eta} = 4, \mu_1 = 1, \mu_3 = 1, \mu_2 = 1/16, \epsilon = 0.005, \lambda_{min} = 0, \lambda_{max} = 100$. By applying these parameters as described in Algorithm 1, we solve for the optimal $R_{i,j}$ and $\hat{R}_{i,j}$.

Fig. 11 presents the measured user QoE for each scheme, where each scheme utilizes the same total number of bits across all representations. We observe that our proposed bit allocation scheme achieves the best QoE compared to the others, both in high and low bit scenarios, validating the effectiveness of the proposed approach. The original-allocation method consistently performs worse than the others because the popularity-adaptive bit allocation allows us to borrow bits more aggressively from less popular views and allocate them to more popular views. Additionally, the proposed bit allocation scheme guarantees a higher minimum QoE than the other schemes, indicating its ability to maintain a minimum level of user satisfaction when watching free-view videos. These results further demonstrate the efficacy of our proposed popularity prediction model, which is identical to the results presented in Fig. 9.

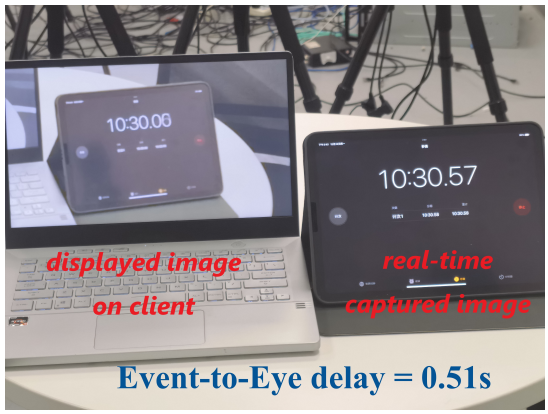


Fig. 12. Event-to-eye delay visualization in VARFVV. This figure illustrates the delay comparison between the VARFVV client device and the capture side, highlighting the timestamps used to calculate the event-to-eye delay.

Delay. We assess three critical delay metrics that impact the FVV viewing experience: start-up delay, view-switching delay, and event-to-eye delay. The start-up delay is the time taken from sending a streaming request to the appearance of the first video frame on the screen [61]. The view-switching delay refers to the duration between requesting a view switch and the display of the desired view [4]. Event-to-eye delay measures the time for an event, captured by the camera, to be displayed on the client's screen [61].

To accurately measure these delays, we record the intervals between the requests and significant changes in the pixel values of the frames. Our analysis revealed that the average start-up delay in the VARFVV is approximately 195.1 ms, while the view-switching delay is notably shorter at about 71.5 ms. Notably, this view-switching delay is typically too brief to be noticed by the human eye. Fig. 12 illustrates the event-to-eye delay, which is approximately 0.51s. This figure also provides a comparative analysis of the timestamps on the VARFVV client side and the capture side. For instance, the timestamp on the VARFVV client reads 10:30.06, whereas the capture side shows 10:30.57, confirming an event-to-eye delay of around 0.51s. The low latency feature of VARFVV, as evidenced by these delay measurements, is a key factor in ensuring an

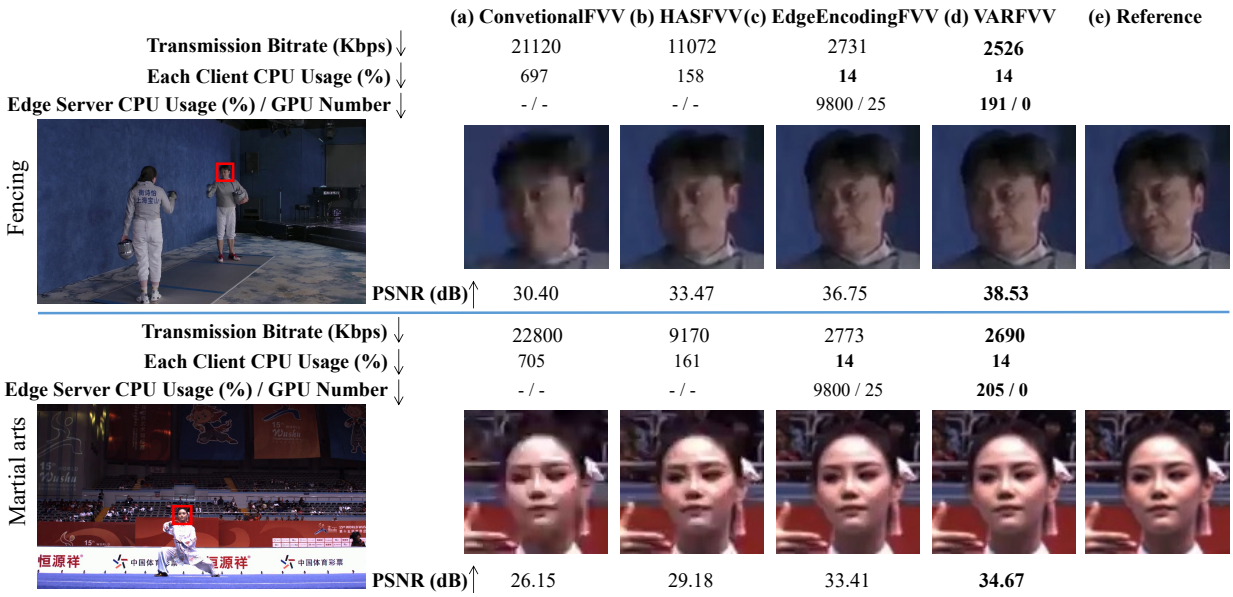


Fig. 13. Comparison of our VARFVV with ConventionalFVV [1], [2], HASFVV [4], and EdgeEncodingFVV [7] in the "Low-interactivity Scenario". The number of views of the "Fencing" scene and the "Martial Arts" scene is 48. (Simulate 500 users.)

immersive experience for users.

C. Comparison

We compare the performance of VARFVV with ConventionalFVV [1], [2] (receiving all views at the client), HASFVV [4] (receiving HTTP adaptive streaming with 10 views at the client), and EdgeEncodingFVV [7] (transcoding frames at the edge server). The performance is evaluated in terms of PSNR, transmission bitrate, and computational resources including client CPU, edge server CPU, and GPU usage. The GPU usage refers to the GPU number required. We simulate 500 users experiencing FVV concurrently to assess edge server computational resources. For a fair comparison, we maintain identical transmission bitrate for VARFVV and EdgeEncodingFVV. However, as ConventionalFVV and HASFVV require clients to receive a larger number of video streams, the bitrate per stream is reduced accordingly. In addition, EdgeEncodingFVV requires frame transcoding and therefore we configure each edge server with an AMD Ryzen 7 3700X CPU@3.60 GHz and two RTX4000 GPUs. All methods are evaluated using the same hardware and software configurations for edge servers and clients. We categorize scenarios based on user interactivity levels: (i) "Low-interactivity Scenario": users' navigation windows are almost constant. (ii) "High-interactivity Scenario": users keep switching views [4].

Fig. 13 presents the results of our VARFVV compared to other methods in the "Low Interactive Scenario" at 1080p@25FPS. Unlike ConventionalFVV and HASFVV, which require high bandwidth and substantial CPU usage to receive and decode multiple streams, our VARFVV achieves superior visual quality with lower transmission bitrate and reduced client CPU usage. Furthermore, compared to EdgeEncodingFVV, our approach demonstrates higher visual quality. Since we reassemble video frames instead of transcoding them at the edge, maintaining the original video quality without imposing heavy computational demands.

Fig. 14 illustrates the results of our VARFVV compared to other methods in the "High Interactive Scenario" at 1080p@25FPS. Our VARFVV achieves much better visual results than ConventionalFVV and HASFVV with lower transmission bitrate and lower client CPU usage. Despite exhibiting a lower PSNR than EdgeEncodingFVV, VARFVV maintains comparable subjective quality, especially during rapid view switching, where human perception is less sensitive to distortions. However, EdgeEncodingFVV demands substantially higher computational resources than VARFVV to achieve similar visual outcomes. Specifically, our single edge server does not require GPUs, whereas EdgeEncodingFVV's edge servers need 25 RTX4000 GPU cards, each capable of encoding up to 20 videos at 1080p@25fps, to serve 500 users. Furthermore, VARFVV proves more CPU-efficient, with only 153% edge server CPU usage compared to EdgeEncodingFVV's 4700%, primarily utilized for decoding tasks. Leveraging the continuous reassembly of multiview video frames without edge-side decoding and encoding enables each server to efficiently serve up to 500 users si-

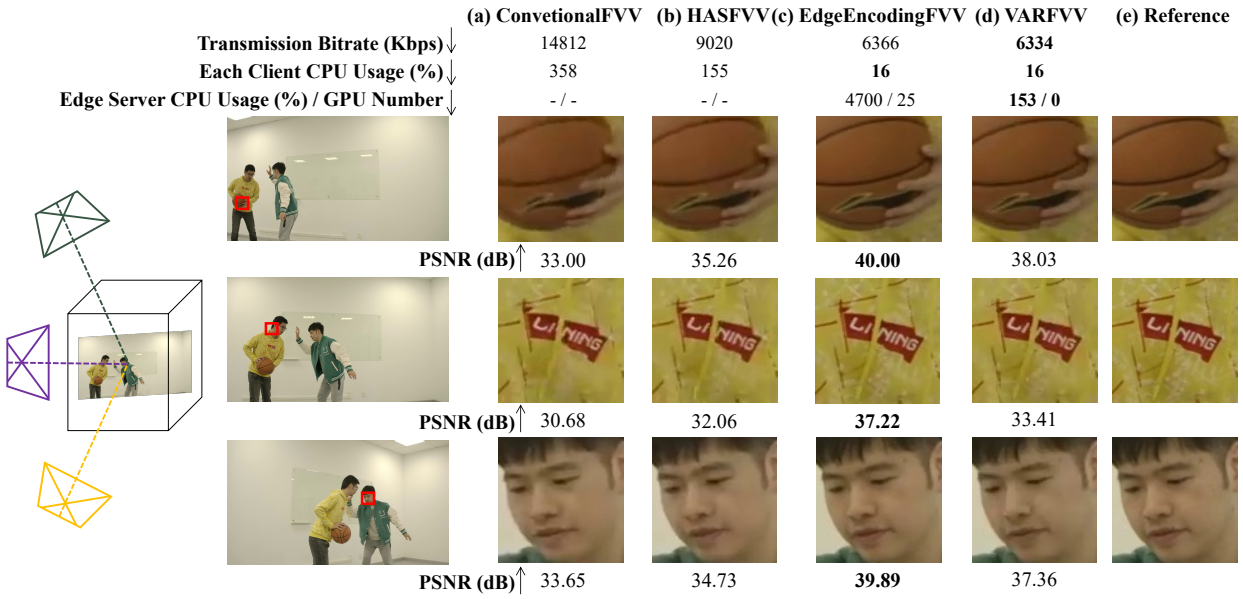


Fig. 14. Comparison of our VARFVV with ConventionalFVV [1], [2], HASFVV [4], and EdgeEncodingFVV [7] in the "High-interactivity Scenario". The number of views of "Basketball" scene is 23. (Simulate 500 users.)

TABLE IV
QUANTITATIVE COMPARISON WHEN 500 USERS USE AN FVV SERVICE AT 4K RESOLUTION WITH 23 WIEWS.

Method	Each Client CPU Usage (%) ↓	Edge Server CPU Usage (%) / GPU Number ↓
HASFVV	494	- / -
EdgeEncodingFVV	45	52560 / 100
VARFVV	**45**	**232 / 0**

multaneously, significantly reducing computational demands.

Furthermore, we conduct experiments to evaluate computational costs on videos at 4K@25FPS when 500 users use our VARFVV, HASFVV, and EdgeEncodingFVV. Tab. IV shows the results. On a client CPU, our VARFVV, as well as EdgeEncodingFVV, consumes 45% while HASFVV uses 494% because HASFVV does not exploit an edge server. On edge server, our technique occupies 232% without GPU cards whereas EdgeEncodingFVV uses 52560% with 100 GPU cards. These results show that our VARFVV is lightweight and computation-efficient on both clients and edge servers.

VII. CONCLUSION

We propose and implement VARFVV, a novel FVV streaming system that achieves low switching delay and high QoE while maintaining low-cost computation and transmission. Our computationally efficient FVV stream generation approach significantly reduces the computational load at the edge by demultiplexing and reassembling video frames, making it ideal for large-scale, mobile-based UHD FVV experiences. Furthermore, we also design a PPC and GNN-based popularity prediction algorithm and a popularity-adaptive bit allocation algorithm for multiview coding to maximize the total QoE with a limited number of bits. Extensive experimental results indicate the effectiveness of our approach for FVV quality, computational resource, and transmission bandwidth, which compares favorably to the state-of-the-art.

REFERENCES

[1] M. Tanimoto, "Ftv (free-viewpoint tv)," in *2010 IEEE International Conference on Image Processing*, 2010, pp. 2393–2396.
[2] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint tv," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 67–76, 2011.
[3] E. Ekmekcioglu, C. G. Gurler, A. Kondoz, and A. M. Tekalp, "Adaptive multiview video delivery using hybrid networking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 6, pp. 1313–1325, 2017.
[4] X. Zhang, L. Toni, P. Frossard, Y. Zhao, and C. Lin, "Adaptive streaming in interactive multiview video systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 1130–1144, 2019.
[5] C. Yao, J. Xiao, Y. Zhao, and A. Ming, "Video streaming adaptation strategy for multiview navigation over dash," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 521–533, 2019.
[6] W. Xu, Y. Cui, and Z. Liu, "Optimal multi-view video transmission in multiuser wireless networks by exploiting natural and view synthesis-enabled multicast opportunities," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1494–1507, 2020.
[7] Y. Dong, L. Song, R. Xie, and W. Zhang, "An elastic system architecture for edge based low latency interactive video applications," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 824–836, 2021.

[8] Y. Dong, L. Song, and R. Xie, "Ultra-low latency, stable, and scalable video transmission for free-viewpoint video services," *IEEE Transactions on Broadcasting*, vol. 68, no. 3, pp. 636–650, 2022.

[9] "Webrtc," 2022, https://webrtc.org/.

[10] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1461–1473, 2007.

[11] S. Shimizu, M. Kitahara, H. Kimata, K. Kamikura, and Y. Yashima, "View scalable multiview video coding using 3-d warping with depth map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1485–1495, 2007.

[12] G. Cheung, A. Ortega, and N.-M. Cheung, "Interactive streaming of stored multiview video using redundant frame structures," *IEEE Transactions on Image Processing*, vol. 20, no. 3, pp. 744–761, 2011.

[13] G. Cheung, A. Ortega, and T. Sakamoto, "Coding structure optimization for interactive multiview streaming in virtual world observation," in *2008 IEEE 10th Workshop on Multimedia Signal Processing*, 2008, pp. 450–455.

[14] L. Toni and P. Frossard, "Optimal representations for adaptive streaming in interactive multiview video systems," *IEEE Transactions on Multimedia*, vol. 19, no. 12, pp. 2775–2787, 2017.

[15] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Delay-power-rate-distortion optimization of video representations for dynamic adaptive streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 7, pp. 1648–1664, 2018.

[16] W. Zhang, S. Ye, B. Li, H. Zhao, and Q. Zheng, "A priority-based adaptive scheme for multi-view live streaming over http," *Comput. Commun.*, vol. 85, no. C, p. 89–97, jul 2016.

[17] N. Carlsson, D. Eager, V. Krishnamoorthi, and T. Polishchuk, "Optimized adaptive streaming of multi-video stream bundles," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1637–1653, 2017.

[18] Z. Gao, S. Chen, and K. Nahrstedt, "Omniviewer: Enabling multi-modal 3d dash," in *Proceedings of the 23rd ACM International Conference on Multimedia*, ser. MM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 801–802.

[19] A. Hamza and M. Hefeeda, "Adaptive streaming of interactive free viewpoint videos to heterogeneous clients," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: Association for Computing Machinery, 2016.

[20] A. Yaqoob, T. Bi, and G.-M. Muntean, "A priority-aware dash-based multi-view video streaming scheme over multiple channels," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 297–303.

[21] X. Corbillon, F. De Simone, G. Simon, and P. Frossard, "Dynamic adaptive streaming for multi-viewpoint omnidirectional videos," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 237–249.

[22] T. Su, A. Sobhani, A. Yassine, S. Shirmohammadi, and A. Javadtalab, "A dash-based hevc multi-view video streaming system," *J. Real-Time Image Process.*, vol. 12, no. 2, p. 329–342, aug 2016.

[23] S. Song, Y.-S. Park, and J. Wee, "Dash-based streaming client for view switching in free-viewpoint video systems," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, 2019, pp. 312–314.

[24] S. Wang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning with communication transformer for adaptive live streaming in wireless edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 308–322, 2022.

[25] B. Chai, J. Chen, Z. Luo, Z. Wang, M. Hu, Y. Zhou, and D. Wu, "Sdsr: Optimizing metaverse video streaming via saliency-driven dynamic super-resolution," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 4, pp. 978–989, 2024.

[26] X. Liu, M. Derakhshani, L. Mihaylova, and S. Lambotharan, "Risk-aware contextual learning for edge-assisted crowdsourced live streaming," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 740–754, 2023.

[27] J. Dai, S. Wang, K. Yang, K. Tan, X. Qin, Z. Si, K. Niu, and P. Zhang, "Toward adaptive semantic communications: Efficient data transmission via online learned nonlinear transform source-channel coding," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 8, pp. 2609–2627, 2023.

[28] N. Q. Hieu, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "When virtual reality meets rate splitting multiple access: A joint communication and computation approach," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 5, pp. 1536–1548, 2023.

[29] H. Xiao, C. Xu, Z. Feng, R. Ding, S. Yang, L. Zhong, J. Liang, and G.-M. Muntean, "A transcoding-enabled 360° vr video caching and delivery framework for edge-enhanced next-generation wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1615–1631, 2022.

[30] S. Wang, J. Dai, Z. Liang, K. Niu, Z. Si, C. Dong, X. Qin, and P. Zhang, "Wireless deep video semantic transmission," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 214–229, 2023.

[31] J. Hu, S. Guo, Y. Dong, K. Zhou, J. Xu, and L. Song, "A multi-user oriented live free-viewpoint video streaming system based on view interpolation," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*, 2022, pp. 1–6.

[32] H. Hadizadeh and I. V. Bajić, "Saliency-aware video compression," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 19–33, Jan. 2014.

[33] Y. Gitman, M. Erofeev, D. Vatolin, B. Andrey, and F. Alexey, "Semiautomatic visual-attention modeling and its application to video compression," in *Proc. IEEE ICIP*, Oct. 2014, pp. 1105–1109.

[34] C.-W. Tang, C.-H. Chen, Y.-H. Yu, and C.-J. Tsai, "Visual sensitivity guided bit allocation for video coding," *IEEE Trans. Multimedia*, vol. 8, no. 1, pp. 11–18, Feb. 2006.

[35] C. W. Tang, "Spatiotemporal visual considerations for video coding," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 231–238, Feb. 2007.

[36] Z. Wang, L. Lu, and A. C. Bovik, "Foveation scalable video coding with automatic fixation selection," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 243–254, Feb. 2003.

[37] H. Wei, X. Zhou, W. Zhou, C. Yan, Z. Duan, and N. Shan, "Visual saliency based perceptual video coding in HEVC," in *Proc. Int. Symp. Circuits Syst.*, May. 2016, pp. 2547–2550.

[38] F. Zhang and D. R. Bull, "HEVC enhancement using content-based local QP selection," in *Proc. IEEE ICIP*, Sep. 2016, pp. 4215–4219.

[39] C. Guo and L. Zhang, "A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression," *IEEE Trans. Image Process.*, vol. 19, no. 1, pp. 185–198, Jan. 2010.

[40] Q. Hu, J. Zhou, X. Zhang, Z. Gao, and M.-T. Sun, "In-loop perceptual model-based rate-distortion optimization for hevc real-time encoder," *Journal of Real-Time Image Processing*, vol. 17, no. 2, pp. 293–311, 2020.

[41] J.-B. Jeong, S. Lee, and E.-S. Ryu, "Delta qp allocation for mpeg immersive video," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 568–573.

[42] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. IEEE Int. Conf. Communications*, May 2017, pp. 1–7.

[43] K. K. Sreedhar, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Viewport-adaptive encoding and streaming of 360-degree video for virtual reality applications," in *IEEE International Symposium on Multimedia (ISM)*, Dec. 2016, pp. 583–586.

[44] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "Hevc-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proceedings of the 24th ACM*

International Conference on Multimedia, ser. MM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 601–605.

[45] J. Chen, Z. Luo, Z. Wang, M. Hu, and D. Wu, "Live360: Viewport-aware transmission optimization in live 360-degree video streaming," *IEEE Transactions on Broadcasting*, vol. 69, no. 1, pp. 85–96, 2023.

[46] Y. Huo and H. Kuang, "Ts360: A two-stage deep reinforcement learning system for 360-degree video streaming," *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2022.

[47] X. Chen, T. Tan, and G. Cao, "Macrotile: Toward qoe-aware and energy-efficient 360-degree video streaming," *IEEE Transactions on Mobile Computing*, vol. 23, pp. 1112–1126, 2024.

[48] J. Chakareski, V. Velisavljevic, and V. Stankovic, "View-popularity-driven joint source and channel coding of view and rate scalable multi-view video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 474–486, 2015.

[49] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Making content caching policies 'smart' using the deepcache framework," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 5, p. 64–69, jan 2019.

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

[51] P. Maniotis and N. Thomos, "Tile-based edge caching for 360° live video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4938–4950, 2021.

[52] Q. Cheng, H. Shan, W. Zhuang, L. Yu, Z. Zhang, and T. Q. S. Quek, "Design and analysis of mec- and proactive caching-based 360° mobile vr video streaming," *IEEE Transactions on Multimedia*, vol. 24, pp. 1529–1544, 2022.

[53] R. Zhang, J. Liu, F. Liu, T. Huang, Q. Tang, S. Wang, and F. R. Yu, "Buffer-aware virtual reality video streaming with personalized and private viewport prediction," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 694–709, 2022.

[54] M. Hu, J. Chen, D. Wu, Y. Zhou, Y. Wang, and H.-N. Dai, "Tvg-streaming: Learning user behaviors for qoe-optimized 360-degree video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 10, pp. 4107–4120, 2020.

[55] X. Xu, X. Tan, S. Wang, Z. Liu, and Q. Zheng, "Multi-features fusion based viewport prediction with gnn for 360-degree video streaming," in *2023 IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom)*, 2023, pp. 57–64.

[56] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[57] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[58] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang, "A quality-of-experience index for streaming video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 154–166, 2017.

[59] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[60] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1252–1260.

[61] J. Yi, M. R. Islam, S. Aggarwal, D. Koutsonikolas, Y. C. Hu, and Z. Yan, *An Analysis of Delay in Live 360° Video Streaming Systems*. New York, NY, USA: Association for Computing Machinery, 2020, p. 982–990.