Delay-Optimized Multi-User VR Streaming via End-Edge Collaborative Neural Frame Interpolation

Sushu Yang, Peng Yang, *Member, IEEE*, Jiayin Chen, *Member, IEEE*, Qiang Ye, *Senior Member, IEEE*, Ning Zhang, *Senior Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—In this paper, with the objective of significantly increasing the frame rate of virtual reality (VR) videos, we design an efficient end-edge collaborative VR streaming system which consists of three modules: frame similarity analysis, offloading decision making, and collaborative frame interpolation. In specific, frame similarity analysis tries to eliminate redundant frames based on perceived quality assessment, so that the required number of interpolated frames can be reduced without deteriorating visual quality. Then, an end-to-end (E2E) delay optimization problem is formulated to obtain the optimal offloading strategy, by balancing the transmission and computing burden of neural frame interpolation via end-edge collaboration. Furthermore, the E2E delay of the proposed system is theoretically analyzed based on queuing theory. Our analysis reveals that, the proposed collaborative distribution of interpolation tasks between edge and end devices are effective to achieve the minimal E2E delay of streaming VR videos. Extensive experimental results demonstrate that the proposed system can significantly improve the frame rate of VR videos, while maintaining timely VR content delivery in various networking conditions.**

***Index Terms*—Mobile edge computing, end-to-end delay, virtual reality, neural frame interpolation.**

I. Introduction

Virtual reality (VR) videos have attracted growing global interest in place of traditional two-dimensional (2D) videos in the past decade [1]. With the help of head-mounted devices (HMD), e.g., HTC Vive and Facebook Oculus, users can immerse themselves in a virtual environment and explore content in different directions by moving their heads around as if in the real world. However, in contrast with 2D videos played on flat screens, the spherical viewing range of VR content represented by substantial volumes of data poses technical challenges on timely content delivery, due to the dynamic and scarce network bandwidth. It is reported that the bandwidth consumption of VR delivery is around 80 times that of 2D video streaming [2]. To enable VR streaming in current networks, tile-based streaming method is proposed by partitioning VR videos into non-overlapping tiles which can be flexibly delivered and processed according to varying network conditions and user behaviors [3].

Compared with watching 2D videos [4] [5], users become more sensitive to visual quality (e.g., frame rate and resolution) when exposed to immersive VR videos. As a consequence, users with a headset may easily suffer from dizziness, eye fatigue, and nausea if video playback is not smooth [6]. Those factors in turn deteriorate the quality of experience (QoE) of watching VR videos. This symptom is called VR sickness which gets even worse with longer exposure time. Unfortunately, as reported by [7], more than 80% of users feel discomfort when experiencing current VR services.

Frame rate is a key performance indicator of VR streaming system. It is recommended that [6] increasing the frame rate of VR videos to 90 frames per second (fps) can effectively relieve sickness. Besides, when it is difficult to maintain high frame rate and high resolution at the same time due to limited network capacity, providing resources to improve frame rate is more critical. In addition, a higher frame rate indicates that more video information can be included within the same amount of time. For VR videos that contain high-motion contents, e.g., sports events and live games, the motion changes in the virtual world should be updated more frequently with a higher frame rate to decrease visual artifacts and enable smoother viewing experience. It has been verified that VR videos with a high frame rate (e.g., 90 fps) can provide higher user-perceived quality than low frame rate (e.g., 30 fps) ones through extensive subjective tests [8]. Moreover, although existing commercial VR headsets with a refresh rate ranging from 60 to 120 Hz are qualified to support high frame rate VR videos[1], the frame rate of source VR videos is merely 25 fps or 30 fps. This mismatch results in a series of critical quality issues which can be alleviated by frame rate enhancement. Motivated by the above observations, in this paper, we focus on increasing the frame rate of VR videos, in order to provide enhanced VR video experience.

In contrast to low frame rate VR videos, streaming VR videos with high frame rate over bandwidth-scarce networks is challenging due to the demand of delivering substantially high

This work is supported in part by the Natural Science Foundation of China under Grant 62001180, in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2022QNRC001, in part by the Natural Science Foundation of Hubei Province of China under Grant 2021CFB338, and in part by the Fundamental Research Funds for the Central Universities, HUST, under Grant 2021XXJS014. (*Corresponding author: Peng Yang.*)

Sushu Yang and Peng Yang are with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China (e-mail: {ssyang, yangpeng}@hust.edu.cn).

Jiayin Chen is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: jiayin.chen2018@gmail.com).

Qiang Ye is with the Department of Computer Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada (e-mail: qiangy@mun.ca).

Ning Zhang is with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON N9B 3P, Canada (e-mail: ning.zhang@uwindsor.ca).

Xuemin (Sherman) Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

[1]https://www.vive.com/

volumes of data. The data rate of an encoded 4K VR video at 60 fps can reach 100 Mbps [9] while the response to user's headset movement should be contained in a millisecond level. The corresponding rendering time (rendering a 1440p frame takes over 10 ms [10]) which is positively correlated with the number of frames also increases, making VR's strict delay satisfaction challenging. In order to alleviate the burden of bulk transmission from the cloud server, a neural video frame interpolation technique is exploited to increase the frame rate of VR videos in a cost-effective manner. Specifically, frame interpolation synthesizes intermediate frames between any two consecutive frames. Those interpolated frames are then inserted into the pristine frame sequence before playback. In this way, the overall transmission delay can be reduced. However, generating high-quality interpolated frames often relies heavily on neural networks (NNs), which is computationally intensive beyond the capacity of current VR HMDs. Remote cloud server is powerful yet delivering the interpolated video frames from the cloud is subject to unacceptable transmission latency.

Mobile edge computing (MEC) is a key enabler of NN-interpolated VR delivery [11] [12]. By deploying an edge server in the vicinity of users, the frame interpolation task can be offloaded to the edge where the computing resources are abundant compared with the HMDs. Besides, an accurate estimation of bandwidth can also be made from the edge thus facilitating adaptive offloading decisions [13]. Inspired by those observations, we design an end-edge collaborative VR delivery system, where the frame interpolation task is properly distributed between the edge and the HMDs to fully exploit the end-edge coordination gain. Nevertheless, offloading frame interpolation task to the edge for reducing workloads on the HMDs is not always beneficial to reducing latency, especially in the case of ultra-high frame rate improvement. Since the frames generated by interpolation are uncompressed with substantial data volume, the transmission of the edge-interpolated frames from the edge to the HMDs may mitigate the benefits from task offloading to the edge. In order to trade off the transmission delay with interpolation delay under different network conditions, the end-to-end (E2E) delay, which consists of transmission delay and interpolation delay, is analyzed at the tile level, taking into account the impact of available bandwidth and computing resources. Moreover, considering the heavy data traffic for a multi-user case, queuing delay of tile frames on the edge is provided based on the modeling of VR video traffic. The optimal offloading ratio of the frame interpolation task is then derived according to the E2E delay analysis to guarantee timely VR delivery.

Considering the diversity among different tiles owing to the wide viewing range of VR videos, we employ this feature to further reduce resource consumption while maintaining improved QoE. Through calculating the similarity of consecutive frames, whether to interpolate or not can be determined for every frame pair. Frame interpolation can be considered only when notable motion changes are detected, namely, adjacent frames are with low similarity. The main contributions of this paper are summarized as follows:

- We design an end-edge collaborative VR content delivery system to improve user's QoE through providing high frame rate VR videos. In order to reduce the amount of data to be transmitted, neural video frame interpolation is utilized to reconstruct videos from low frame rate to high frame rate, in joint consideration of the computing resources on the edge and HMDs to ensure timely delivery. Moreover, in order to further relieve the computation and transmission burden of neural frame interpolation tasks, frame similarity evaluation is exploited to avoid interpolating unnecessary frames within frame pairs with high similarity.
- We analyze the E2E delay of the proposed system in single-user and multi-user cases, which includes transmission delay, computing delay and edge queuing delay based on the modeling of VR video traffic. The E2E delay analysis reveals that the distribution of interpolation tasks between edge and end devices can effectively contain the streaming latency of VR videos. Therefore, an optimal offloading strategy is developed to minimize the lowest E2E delay under various network conditions.

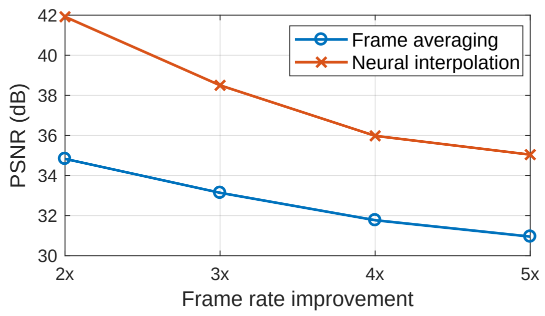


Fig. 1: Effect of different frame interpolation methods.

Fig. 2: Frame averaging (left) with PSNR = 35.88 dB and neural interpolation (right) with PSNR = 42.57 dB.

The remainder of this paper is organized as follows. First, related works are reviewed in Section II. Section III describes the system model and Section IV presents the E2E delay analysis under dynamic network conditions, which leads to the design of optimal interpolation strategy. In Section V, the proposed end-edge collaborative VR delivery system design is introduced, following the E2E delay analysis. Experimental results and performance analysis are given in Section VI. Finally, we conclude this paper and discuss our future work in Section VII.

II. Related Work

Extensive research efforts have been devoted to VR video streaming, including encoding, tiling, viewpoint prediction, and adaptive streaming [2] [14] [15]. The attempts on video encoding aim to reduce the data size of video files to lower the requirement on transmission rate. Considering that only a small portion of the panoramic scene is viewed by the user at a time, various viewpoint prediction algorithms have been proposed to predict user's field of view (FoV) at different moments. Additionally, combined with efficient tiling and adaptive streaming strategies, video tiles are delivered at different quality levels to save bandwidth, while maintaining certain level of visual quality. These works focus on the scenario of one server and one user, without considering edge nodes. However, enabling promising VR experience can be very time- and resource-consuming, further quality improvement such as video frame interpolation is even more challenging.

With the emergence of mobile edge computing, the capabilities of edge are envisioned as key enablers for delivering VR videos [16] [17]. Mehrabi *et al.* exploit edge servers to select video quality by considering both users' QoE and the fairness of bit rate allocation [18]. The recent proposal [13] develops a software framework on the edge server to run a deep reinforcement learning model for adaptive video transmission. However, these works do not exploit the unique features of VR videos thus their performances are limited. Sun *et al.* [19] design a MEC-based VR delivery model by assigning video rendering task to the edge server and pre-caching FoVs on the edge to save latency. Similarly, Dang *et al.* [20] consider joint caching and computing in fog radio access networks to maximize the mean tolerate delay. In [21], an analytical model is proposed to study the delay performance of MEC-based video streaming. However, the delay performance is not clear as there are actual queues of video tiles on the edge [22]. Note that all the works in [19]–[21] utilize edge capacity to accelerate VR rendering which mainly focuses on decoding and stitching without consideration of further quality improvement.

In this paper, different from existing approaches on enhancing the QoE of VR streaming, we investigate improving frame rate of VR videos. High frame rate can be achieved through video frame interpolation [23]–[25], which generates an intermediate frame between two reference frames. Early frame interpolation methods, such as frame averaging that based on the weighted combination of reference images, have rather low computation demands. However, the neglect of complex motion changes can lead to noticeable blurry artifacts, deteriorating user's experience. Currently, with deep learning techniques gaining growing popularity, neural network-based video frame interpolation methods become dominant. Fig. 1 gives a comparison between two interpolation methods. The performance of neural interpolation always surpasses frame averaging owing to its powerful nonlinear fitting ability, which is able to model various motion modes of complicated scenes. As shown in Fig. 2, blurry artifacts of frame averaging are easily perceptible, resulting in much lower Peak Signal to Noise Ratio (PSNR) compared with neural interpolation.

To optimize video streaming, Usman *et al.* propose to perform frame interpolation on mobile devices to reconstruct lost frames [26] [28]. Nevertheless, implementing neural interpolation [23]–[25] for better visual quality relies heavily on powerful hardware, i.e., GPU, while current VR headsets

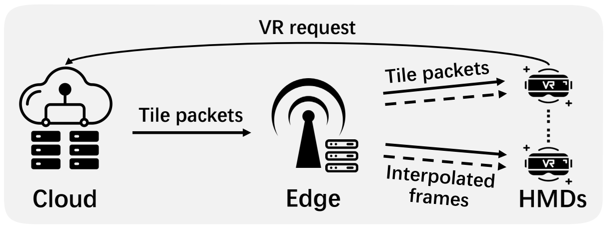


Fig. 3: An overview of the end-edge collaborative VR streaming system.

are not computationally capable of neural interpolation under VR's strict latency requirement. Witnessing the potential of MEC, an edge-assisted frame interpolation and super-resolution framework is proposed to reconstruct downsampled videos [27]. But it aims for low frame rate VR video delivery with a target frame rate at merely 30 fps, and ignores the capability of user devices.

To fully unleash available resources on both user devices and edge nodes, we exploit neural edge intelligence to significantly increase VR frame rate in this paper. Through offloading part of heavy neural frame interpolation task to the edge, the computing burden on the HMDs can be largely alleviated. Furthermore, to guarantee timely VR delivery in a cost-effective manner, the optimal offloading decision is derived based on our E2E delay analysis under varying bandwidth and computing resources. The comparison of our work with related works is listed in Table I.

III. System Model

As depicted in Fig. 3, we consider an end-edge collaborative VR delivery system with an edge server deployed in the vicinity of a set $\mathcal{M} \triangleq \{1, ...m, ..., |\mathcal{M}|\}$ of users (i.e., VR HMDs), where $|\cdot|$ denotes the set cardinality and m represents user index. Edge and HMDs are equipped with the amounts of computing resources, denoted by W' and W, respectively. Source VR videos, with an initial frame rate, f, are pre-cached and pre-coded on the cloud after being temporally segmented into short chunks with the same duration, τ, and then spatially cropped into equal-size tiles. Denote by k tile index. Upon receiving user demands, the cloud pushes corresponding VR content to the edge and the HMDs to jointly perform the neural frame interpolation tasks to achieve a required high frame rate, f^*, before playback. Let T_0 be the transmission delay enroute from the cloud to the edge and $R(t)$ be the data transmission rate from the edge to the HMDs at time t. In order to guarantee timely VR delivery, the cloud should determine the offloading ratio, according to the available bandwidth and computing resources, while achieving the lowest E2E delay. Denote by $\alpha \in [0, 1]$ the offloading ratio, namely, the percentage of frames interpolated on the edge server.

A. Communication Model

Figure 3 also illustrates the transmission process of VR tile packets. Tiles requiring frame interpolation are sent to the edge for frame rate improvement. When the edge receives bin files (video data after compression) of a designated tile, a copy

TABLE I: Related Works

Scheme	Contribution	Limitation
End + Edge [19]	Precache video tiles on the edge and assign computational tasks either to the edge server or mobile devices to save latency	The offloading strategy is at coarse granularity, leading to suboptimal solutions
End [23]	Design a real-time neural frame interpolation technique to improve video quality	Do not reveal edge intelligence to optimize video delivery
End [26]	Propose to perform frame interpolation on mobile devices to reconstruct lost frames	Do not reveal edge intelligence to optimize video delivery
Edge [27]	Propose an edge-assisted frame interpolation and super-resolution framework to reconstruct downsampled videos with low frame rate and resolution	Aim for low frame rate VR video delivery, and ignore the computing capability of user devices

of this file is delivered to the HMDs immediately. Since tiles requiring interpolation are sent to the edge, the transmission delay from the cloud to the edge, T_0, is equivalent for all tiles, regardless of whether they are interpolated on the edge or on the HMD. Denote by D_k the original data size of the raw bin file of tile k. The transmission delay of delivering a tile from the cloud to the HMDs is $T_0+\frac{D_k}{R_m(t)}$, where $R_m(t)$ is the bandwidth allocated to user m at time t. Considering the huge data volume of uncompressed tile frames, edge-interpolated frames are compressed with ratio, β (< 1), before being transmitted to the HMDs. Therefore, the delay of transmitting one edge-interpolated tile frame with data size d is $\frac{\beta d}{R_m(t)}$. For notational brevity, $R(t)$ is abbreviated as R in the following.

B. Computing Model

In our system, the computation tasks of VR videos mainly comprise video frame interpolation and video decoding. To guarantee the visual quality of interpolated frames, neural video frame interpolation is adopted in our system. Note that, for a certain neural interpolation approach, the amount of required floating operations is fixed. Hence, given the computing capacity, the runtime of the interpolation NN is constant determined by the frame resolution, which is also evaluated in our experiments. Let $\check{\tau}_m$ and $\check{\tau}'_m$ be the time required to interpolate one tile frame by an NN on the HMD and the edge given certain computing resources, respectively. They are calculated as $\frac{\phi c}{W_m}$ and $\frac{\phi' c}{W'_m}$ [29], where ϕ and ϕ' (cycles/bit) denote the computing intensity of the computing task and c (bits) is the amount of data to be processed per frame. W_m and W'_m (cycles/sec) are the m_{th} HMD computing capacity and edge computing resources allocated to user m, respectively. According to our experiments on an RTX 3080 GPU, the time of interpolating one 640×480 frame with RIFE [23] is around 10 ms. Considering that the computing resources on the edge server are much higher than that on the HMDs, the corresponding edge interpolation time is much shorter under the same level of computing intensity. The interpolation time of a tile is proportional to the required number of interpolated frames and the offloading ratio. The required amount of interpolated frames of tile k, $\check{I}_k$, is represented as

$$\check{I}_k = (f^* - f)\tau. \tag{1}$$

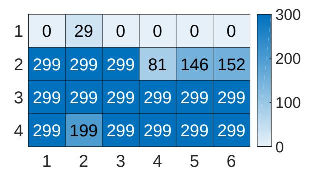
Consequently, the time of interpolating a tile on the HMDs and the edge is $(1-\alpha_{m,k})\check{\tau}_m\check{I}_k$ and $\alpha_{m,k}\check{\tau}'_m\check{I}_k$, respectively.

Since there is usually high redundancy between consecutive frames, for frame pairs with high similarity, the latter frame can be approximated by the former one. Therefore, neural frame interpolation is not performed for highly similar frame pairs. By analyzing the similarity of different frames, frame pairs with high similarity are ignored and the required amount of interpolated is thus reduced, which further alleviates the computing burden on the edge server and HMDs. Considering the high consistency with the perception of human eyes, the widely adopted metric, Structural Similarity (SSIM), is employed to calculate frame similarity [30] [31]. Extensive subjective experiments have been carried out to validate the effectiveness of SSIM by inviting various viewers to rate videos of different genres. SSIM is defined as

$$SSIM\,(x,y) = \frac{(2\mu_x\mu_y + C_1)\,(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)}, \tag{2}$$

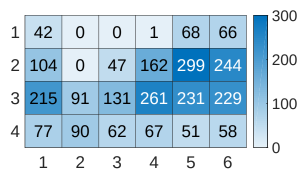
where x and y represent two images, μ and σ are mean and standard deviation of pixel intensity, respectively. C_1 and C_2 are constants. According to [32], if the SSIM between two images is lower than a threshold (e.g., 0.9), it indicates that noticeable motion changes take place, calling for supplementary frames to enhance user's QoE. However, a high value of SSIM means that the content is regarded as static. According to the subjective tests in [8], the perceptual quality of interpolated videos is worse or just slightly better than that of the source videos when the source videos contain very small amount of motion. Therefore, interpolating more frames cannot lead to enhanced experience in this condition. Consequently, no interpolation will be performed for those frame pairs. By calculating SSIM between frames, $\check{I}_k$ can be reduced for every tile while maintaining enhanced quality.

A comparative experiment between high- and low-motion VR videos has been conducted to see tile diversity with the threshold set to 0.9. A set of VR videos are classified using conventional optical flow method [33], videos with high average field strength are considered to contain high-motion contents, and vice versa. Each of those videos lasts for 10 seconds, and is spatially cropped into 4×6 tiles, at a frame rate of 30 fps. In total, each video contains 300 frames. By calculating SSIM of the tiles of each video, we can obtain the required amount of frames to be interpolated for smooth VR playback experience. As shown in Fig. 4, the inter-frame difference of videos with high motion is so large that the required amount of interpolated frames can be as high as 299 if doubling frame rate is expected, which means that one additional frame should be interpolated into any two consecutive frames. In contrast, contents change mildly in



	1	2	3	4	5	6
1	0	29	0	0	0	0
2	299	299	299	81	146	152
3	299	299	299	299	299	299
4	299	199	299	299	299	299

(a) High motion



	1	2	3	4	5	6
1	42	0	0	1	68	66
2	104	0	47	162	299	244
3	215	91	131	261	231	229
4	77	90	62	67	51	58

(b) Low motion

Fig. 4: The number of frame pairs requiring interpolation.

slow-motion videos, pixels in previous frames can be reused to approximate subsequent frames while hardly impairing the user-perceived visual quality. As for low-motion VR videos, since the background content takes up a large portion of the entire scene, the corresponding tiles almost remain still through the whole video. Hence, the frame rate of those tiles can be suppressed to save bandwidth. Note that, the decision of interpolation of each frame can be capsulated in the video description file, and streamed along with the video files. As the data size of the interpolation information is insignificant, the corresponding transmission delay can be ignored.

Since interpolated frames are generated with reference to uncompressed frames, video tiles should be decoded before frame interpolation. Denote by τ_d and τ_d' the time required to decode one tile frame on the HMDs and the edge, respectively. They can be recorded through tests on real-world VR videos. Similarly, the time of decoding a tile is $\tau_d \tau f$ on the HMDs and $\tau_d' \tau f$ on the edge.

C. Problem Formulation

Thanks to the computing capacity of edge servers, computation-intensive neural interpolation tasks can be partially offloaded before playback on the HMDs. However, it also incurs additional processing and transmission delay, especially considering the computational complexity of the NNs used for interpolation, as well as the inflated data volume after increasing frame rate. In order to guarantee timely VR video streaming, the E2E delay of delivering VR videos should be carefully analyzed. Denote by $T_{m,k}$ and $T'_{m,k}$ the E2E delay for delivering tile k requested by user m with HMD-only and edge interpolation, respectively, the final E2E delay of delivering tile k is $\max\{T_{m,k}, T'_{m,k}\}$. Consequently, the optimal offloading ratio can be derived from the following min-max optimization problem, given by

$$\min_{\alpha} \ \max\{T_{m,k}, T'_{m,k}\}, \tag{3}$$

where $T_{m,k}$ and $T'_{m,k}$ are comprised of transmission delay, decoding delay, and interpolation delay. Next, we provide an E2E delay modeling of the proposed frame interpolation system in single-user and multi-user scenarios, accounting for the bandwidth dynamics, and the constrained computing resources on both edge server and the HMDs.

IV. E2E Delay Modeling

A. Single-User Scenario

In a single-user scenario, the edge serves for a single user, thus the E2E delay of delivering one tile with HMD-only interpolation, T, is given by

$$T_k = T_0 + \frac{D_k}{R} + \tau_d \tau f + (1-\alpha_k)\check{\tau}\check{I}_k. \tag{4}$$

where user index is omitted for brevity. The first two terms indicate the transmission delay from the cloud to the edge and that from the edge to the HMD. The last two terms represent the decoding delay and interpolation delay, respectively. Similarly, the E2E delay of delivering the frames generated via end-edge collaborative interpolation, T', is given by

$$T'_k = T_0 + \lceil \alpha_k \rceil \tau_d' \tau f + \alpha_k \check{\tau}' \check{I}_k + \frac{\alpha_k \beta d \check{I}_k}{R}, \tag{5}$$

where $\alpha_k \beta d \check{I}_k$ is the data size of the total frames generated by edge interpolation. The last term of Eq. (5) corresponds to the transmission delay of delivering all edge-interpolated frames of a tile to the HMDs. Note that if the interpolation ratio α_k of a certain tile equals 0, the corresponding bin file is not decoded on the edge server. In case of $\alpha_k > 0$, the objective of accelerating end-edge collaborative frame interpolation boils down to the min-max optimization problem (3). In order to solve problem (3), the deciding factors on the gap between T_k and T'_k should be analyzed, which helps to characterize the impact of bandwidth and computing resources on the E2E delay. Define $\Delta T_k \triangleq T_k - T'_k$, we have

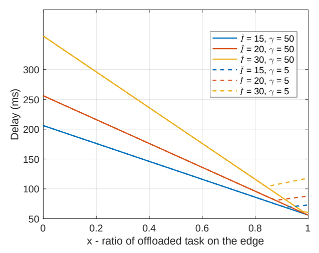
$$\Delta T_k = \frac{D_k - \alpha_k \beta d \check{I}_k}{R} + (\tau_d - \lceil \alpha_k \rceil \tau_d')\tau f + \check{\tau}\check{I}_k - \alpha_k(\check{\tau} + \check{\tau}')\check{I}_k. \tag{6}$$

According to our experimental results, which have also been shown in [10], the HMD decoding time, τ_d, can be as small as 0.7 ms, which is even smaller when frames are decoded on the edge. The difference, $\tau_d - \tau_d'$, is thus of smaller order than that of transmission and interpolation delay, and can be neglected. Therefore, $\Delta T_k \approx \frac{D_k - \alpha_k \beta d \check{I}_k}{R} + \check{\tau}\check{I}_k - \alpha_k(\check{\tau} + \check{\tau}')\check{I}_k$. From this, the solution to (3) can be obtained in a divided-and-conquer manner. Since the lowest E2E delay depends on the larger one of T_k and T'_k, it is achieved when the two delay values are equal, namely, when their difference equals 0. Hence, by letting $\Delta T_k = 0$, we have

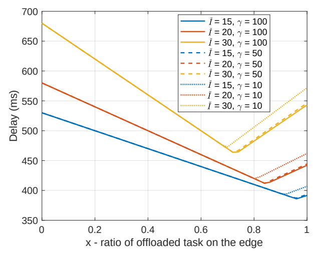
$$\alpha_k = \frac{(D_k/R) + \check{\tau}\check{I}_k}{(\beta d \check{I}_k/R) + (\check{\tau} + \check{\tau}')\check{I}_k}. \tag{7}$$

In case $\Delta T_k \geq 0$, the E2E delay of HMD-only frame interpolation dominates. From (4), the ratio, α_k, should be made close to (7) from below. In contrast, when $\Delta T_k < 0$, the E2E delay of edge frame interpolation dominates, and the optimal value of α approaches (7) from above. In what follows, two typical cases are analyzed to demonstrate the impact of bandwidth and computing resources.

Case 1: Bandwidth-sufficient and computing-constrained. In this case, the data rate, R, is sufficiently large. The two terms of transmission delay, D_k/R and $\beta d \check{I}_k/R$, in (7) are thus negligible. The value of ratio α_k is determined by the interpolation speed on both the HMDs and the edge, i.e., $\alpha_k \to \frac{\check{\tau}}{\check{\tau}+\check{\tau}'}$. That is, when the edge is equipped with higher computing power facilitating rapid neural interpolation, i.e., a smaller value of $\check{\tau}'$, the value of α_k increases and more interpolated frames should be generated from the edge. This



(a) Bandwidth = 50 Mbps



(b) Bandwidth = 5 Mbps

Fig. 5: Delay under different network capacities.

is validated in Fig. 5(a), where γ denotes the ratio of $\check{\tau}$ to $\check{\tau}'$. Note that the optimal offloading ratio decreases with the increase of the number of interpolated frames.

Case 2: Bandwidth-constrained and computing-sufficient. In this case, the transmission delay is significant, while the delay of frame interpolation on both the HMDs and the edge, $\check{\tau}\check{I}_k$ and $\check{\tau}'\check{I}_k$, in (7) are negligible. The value of ratio α_k is determined by the data size of edge-interpolated frames, i.e., $\alpha_k \rightarrow \frac{D_k}{\beta d\check{I}_k}$. That is, when the transmission link from the edge to the HMD is experiencing poor channel conditions, the ratio α_k should decrease with the total amount of interpolated frames, as shown in 5(b). It is also observed from $\alpha_k \rightarrow \frac{D_k}{\beta d\check{I}_k}$ that efficient video compression with lower value of β also contributes to more frames generated from the edge.

B. Multi-User Scenario

In practice, the cloud provides VR streaming services to multiple users simultaneously. Multiple tiles for different HMDs are continuously streamed via the edge server, leading to packet queuing on the edge [22] [34]. Once a tile arrives at the edge, it joins the processing queue before decoding and frame interpolation. In order to characterize packet arrival and queue accumulation, experiments have been done to figure out the tile frame size distribution. The frame size distribution of a VR video lasting for 30 s is shown in Fig. 6. According to [35], video traffic of H.264 codecs can be modeled with Gamma distribution and this is consistent with our tests on H.265 video stream. Owing to the additive property of Gamma distribution, the inter-arrival time of tile frames, follows Gamma distribution as well. Hence, the tile flow of user $m \in \mathcal{M}$ can be characterized as a GI/G/1 queuing process [36], upon which the average queuing delay can be approximated to shed light on the overall E2E delay[2]. For tile flow of the user m, suppose that tile frames, denoted by l_1, ..., l_n (n is frame index), ..., arrive at the processing queue on the edge server where they are processed in the order of their arrival. Denote by a_n the inter-arrival time of l_n and l_{n+1} and s_n the processing time of l_n (i.e., the time for decoding and interpolation). Let q_n be the waiting time of l_n in the processing queue and u_n denote

[2]Queuing theory focuses on the long-term performance of a system by studying network dynamics. Therefore, it is beneficial to understand the operating characteristics and performance bottlenecks of the system, and to develop targeted improvement plans accordingly. Although the solution derived from the optimization problem based on queuing theory may not be accurate when evaluating the short-term performance of the system, the average system performance can be guaranteed.

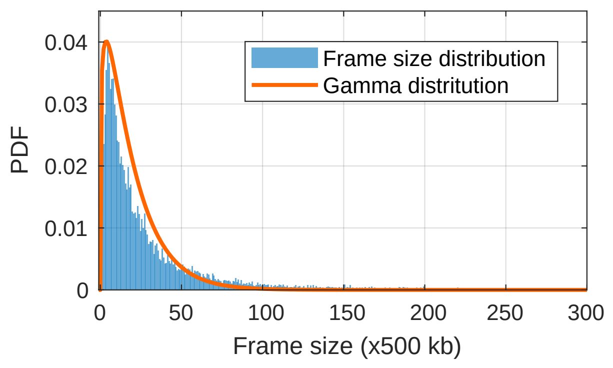


Fig. 6: Distribution of tile frame size.

the time difference between processing time and inter-arrival time, which equals $s_n - a_n$. It is clear that

$$q_{n+1} = (q_n + u_n)^+, \tag{8}$$

where $x^+ = max\{x, 0\}$ holds for any real number x. Eq. (8) indicates that the waiting time of frame l_{n+1} is positive if the waiting time and processing time of l_n is larger than the inter-arrival time between l_n and l_{n+1}. Otherwise, q_{n+1} equals zero, indicative of no queuing of l_{n+1}.

Denote by λ_m the average arrival rate, the average inter-arrival time is $1/\lambda_m$ which is related to the mean value of the Gamma distribution. Note that different video content corresponds to different Gamma distributions. The average processing time can also be defined as $1/\mu_m$. μ_m is the average processing rate of the edge server and is mainly determined by the edge interpolation time, $\check{\tau}'_m$, which is determined according to the allocated computing resources in proportion to the amount of frame interpolation tasks of user m. If the average processing time of tile frames is less than the average inter-arrival time, q_n converges to a random variable q in distribution. Therefore, the mean waiting time (namely, queuing delay) of the processing queue on the edge is $E(q)$.

Let $z = q + u$ be the difference between the time a tile frame leaves the queue and the time the next frame arrives at the queue, where u and u_n have the same distribution and are independent of q. Thus z^+ represents the situation where a tile frame arrives at the queue before its last frame leaves so that it has to wait for processing. Let $z^- = (-z)^+$, which denotes the situation where the last frame leaves the queue before the next one arrives. Clearly, we have $z = z^+ - z^-$, leading to the following equality

$$E(z) = E(z^+) - E(z^-) = E(q) + 1/\mu_m - 1/\lambda_m. \tag{9}$$

Due to (8), $z^+ = (q + u)^+$ has the same distribution as q, indicating $E(z^+) = E(q)$. Hence, we obtain

$$E(z^-) = 1/\lambda_m - 1/\mu_m. \tag{10}$$

It can be observed that if the arrival rate of tile frames is smaller than the processing rate of the edge server, $E(z^-) > 0$ meaning no queuing occurs statistically.

According to [36], the relationship of the variance of z, z^+ and z^- is as follows

$$\begin{aligned} &var(z^+) + var(z^-) \\ &\quad = var(q+u) - 2E(z^+)E(z^-) \\ &\quad = var(q) + var(u) - 2E(q)[1/\lambda_m - 1/\mu_m], \end{aligned} \tag{11}$$

where $var(\cdot)$ represents the operation of variance. Note that $var(z^+) = var(q)$ because of the same distribution. Therefore, the average queuing delay can then be derived as

$$E(q) = \frac{var(u) - var(z^-)}{2[1/\lambda_m - 1/\mu_m]}, \tag{12}$$

which yields the upper bound of $E(q)$,

$$E(q) \le \frac{[(\sigma_m^{(a)})^2 + (\sigma_m^{(s)})^2]}{2[(1/\lambda_m) - (1/\mu_m)]}, \tag{13}$$

where $(\sigma_m^{(a)})^2$ and $(\sigma_m^{(s)})^2$ denote the variance of inter-arrival time (i.e., Gamma distribution) and that of processing time.

Lemma 1: With a normalizing fraction [36], the upper bound can be scaled down to an approximation of q since the derivatives of the upper bound and the Pollaczek-Khintchine formula are equal, given by

$$\begin{aligned} q &\approx \frac{1 + [\sigma_m^{(s)}\mu_m]^2}{\mu_m^2/\lambda_m^2 + [(\sigma_m^{(s)})\mu_m]^2} \cdot \frac{[(\sigma_m^{(s)})^2 + (\sigma_m^{(a)})^2]}{2[(1/\lambda_m) - (1/\mu_m)]} \\ &= \frac{\rho_m^2[1 + (\sigma_m^{(s)})^2\mu_m^2][(\sigma_m^{(a)})^2\lambda_m^2 + \rho_m^2(\sigma_m^{(s)})^2\mu_m^2]}{2\lambda_m(1-\rho_m)[1 + \rho_m^2(\sigma_m^{(s)})^2\mu_m^2]}, \end{aligned} \tag{14}$$

where ρ_m equals $\frac{\lambda_m}{\mu_m}$, representing the nonempty probability of the processing queue.

The proof of Lemma 1 is given in Appendix A.

Generally, the speed of NN interpolation is mainly determined by the input size, namely, the resolution of frames. Without loss of generality, we assume the resolution of interpolated frames remains constant. The variance, $(\sigma_m^{(s)})^2$, thus tends to zero. Consequently, the average queuing delay of one frame is

$$q = \frac{\rho_m^2(\sigma_m^{(a)})^2\lambda_m}{2(1-\rho_m)}. \tag{15}$$

Since the decoding time ($<$ 1 ms) per frame is much shorter than the interpolation time (tens of ms), the queuing delay of a tile on the edge server mainly depends on the required number of interpolated frames and can be given by

$$Q_{m,k} = \alpha_{m,k}\check{I}_k q, \tag{16}$$

where $\alpha_{m,k}\check{I}_k$ is the amount of frames interpolated on the edge. As a result, the problem of accelerating end-edge collaborative frame interpolation becomes

$$\min_{\alpha_{m,k}} \ \max\{T_{m,k}, T'_{m,k} + Q_{m,k}\}. \tag{17}$$

In practice, the arrival rate of tile packets, λ_m, at the edge can be approximated by the data rate, R_{ce}, from the cloud to the edge. Following the derivation of Eq. (7), similar analysis on the difference, $T_{m,k} - (T'_{m,k} + Q_{m,k})$, can be performed in order to obtain the optimal ratio of edge-interpolated frames, which is given by

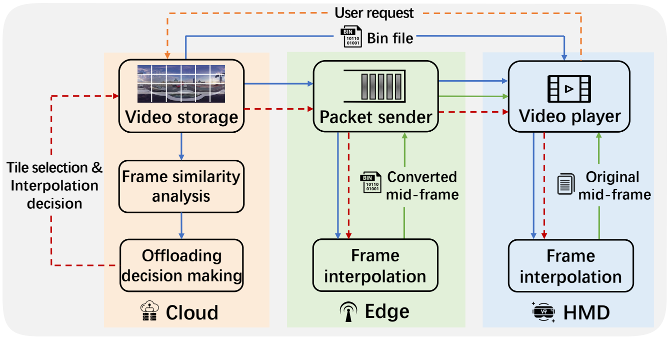


Fig. 7: System architecture.

$$\alpha_{m,k} = \frac{(D_k/R_m) + \check{\tau}_m\check{I}_k}{(\beta d\check{I}_k/R_m) + (\check{\tau}_m + \check{\tau}'_m + q)\check{I}_k}. \tag{18}$$

When bandwidth is sufficient and computing resources are the bottleneck, the offloading ratio $\alpha_{m,k} \to \frac{\check{\tau}_m}{(\check{\tau}_m + \check{\tau}'_m + q)}$. On the contrary, queuing delay can be neglected thus $\alpha_{m,k} \to \frac{D_k}{\beta d\check{I}_k}$.

V. System Design

In this section, the architecture of the proposed end-edge collaborative VR delivery system is presented based on the E2E delay analysis. The details of the functionality of the cloud, edge and HMDs are shown in Fig. 7, including source VR video preparation, packet transmission, etc. The proposed system consists of three core building blocks. (1) *Frame similarity analysis.* As temporally adjacent frames may be quite similar to each other, there is no need to interpolate for every frame pair to enhance video playback smoothness. Through similarity analysis, the least number of frame pairs that require frame interpolation can be calculated while achieving the same quality of high frame rate VR videos. (2) *Offloading decision making.* To trade off the corresponding transmission delay and interpolation delay resulting from frame interpolation, an offloading decision making module is deployed on the cloud to decide the ratio of frame interpolation task offloaded to the edge under bandwidth and computing resources constraints. Also, this module predicts the set of tiles that are likely to be in user's FoV after receiving user request. (3) *Collaborative frame interpolation.* A neural interpolation model is employed to enhance VR frame rate according to the interpolation decision obtained from the offloading decision making module. Considering the substantial data volume to be transmitted after interpolation, interpolated frames (i.e., mid-frames) are encoded via fast intra-frame compression. When all the required mid-frames are ready, the video player inserts them into the original video sequence displaying for the user.

A. System Workflow

The workflow of the proposed system is shown in Algorithm 1. Specifically, source VR videos are stored on the cloud after being segmented and compressed at tile level. Through frame similarity analysis on each frame pairs, those with

Algorithm 1 Workflow of the Proposed System

```
Input: User request
Output: Tile set $S$ to be transmitted, required frame rate $f^*$
    and offloading ratio $\alpha$
1: for Each chunk of the requested VR video do
2:     Given user request and historical user information
3:     Predict the set of tiles $S$ to be transmitted
4:     for Each tile in $S$ do
5:         Determine $f^*$ by obtaining SSIM between frames
           and derive $\alpha$ based on Eq. (3) or (17)
6:         Offload frame interpolation task to the edge and the
           HMDs according to $\alpha$
7:         Deliver edge-interpolated frames to the HMDs
8:     end for
9: end for
```

relatively low similarity are thought to contain high motion. Therefore, supplementary mid-frames are needed to improve video playback smoothness, calling for extra frame interpolation tasks. When receiving user's demand for a specific VR video at a target frame rate, the cloud server predicts which set of tiles are to be delivered to the user, and determines relevant task offloading ratio of each tile based on current network conditions and the result obtained from the frame similarity analysis module. Then the predicted set of tiles along with the offloading decisions are sent to the edge server and HMDs together. Based on the offloading decisions, the frame interpolation module deployed on the edge and HMDs performs the corresponding interpolation tasks. Edge-interpolated mid-frames are later transmitted to the HMDs and inserted into the original video sequence to provide improved VR video experience for the user.

B. Frame Similarity Analysis

Recall that SSIM is utilized to quantify similarity of different frames due to its consistency with the perception of human eyes. It is noteworthy that other similarity metrics can also be applied in this module. Every two consecutive tile frames are carefully examined and compared to get their similarity. Based on their similarity, we determine into which frame pairs interpolated mid-frames will be later inserted to improve playback smoothness. If SSIM is lower than a threshold (set to 0.98 according to the experiments in Section VI), extra interpolated frames are required to improve video quality. On the contrary, if neighboring frames are with a larger SSIM value, the inter-frame difference can hardly be distinguished thus the benefit of interpolation is marginal. In this case, frame averaging will be employed in real time to keep a constant frame rate across all of the tiles. Note that videos with scene switches can cause low SSIM between adjacent frames as well, but this can be solved by dedicated scene change detection technologies [37]. For simplicity, we consider VR videos with no scene switches in this paper.

C. Offloading Decision Making

After obtaining the similarity comparison results of the video frames, an offloading decision making module is activated. The functionality of this module is two-fold. On the one hand, based on historical traces of user requests, this module predicts user's FoV, i.e., the subset of tiles that are most likely to be requested in the next. On the other hand, this module determines the percentage of the frame interpolation task to be completed on the edge and the HMDs. To strike a balance between the transmission delay and interpolation delay, the frame interpolation task should be properly split between the edge and the HMDs in accordance with the previous E2E delay analysis. This split ratio is calculated by the offloading decision making module based on the computing resources on the edge and the HMDs, as well as the available bandwidth. Information about network conditions and HMD capacities as well as user request can be collected via uplink transmission with negligible delay because of the small data volume [21].

The final interpolation decision consists of both the number of interpolated mid-frames and the insert positions. This decision is sent to the edge server along with the requested tiles. In certain conditions, not all of the tiles in the predicted user FoV require frame interpolation. As shown in Fig. 7, tiles with relatively static content are directly sent to the HMDs rather than waiting in the edge processing queue.

D. Collaborative Frame Interpolation

Upon the receiving of all of the data packets of a certain tile, the edge or the HMDs decode the frames and input them into the frame interpolation module. To cater for the stringent delay requirement of VR video streaming, the state-of-art real-time video frame interpolation neural network, RIFE [23], is adopted in our system[3]. For the edge server, the data to be transmitted after interpolation consists of the pristine video tile and raw tile frames, both of which have much larger data size compared with the compressed ones. For example, the data size of an 8-bit 640 $\times$ 480 tile frame in original YUV420 format is 3.6 Mb while that of a bin file of an entire tile containing 30 frames can be as low as 200 Kb. Since RIFE encodes the output image into PNG (lossless encoding with comparatively low compression ratio) format to maintain fidelity, transmitting those PNGs with limited network bandwidth is still a burden.

In order to further reduce the transmission latency, the edge server transfers the copies of the tile bin files to the HMDs immediately after receiving them so that only the interpolated tile frames are transmitted after decoding and processing on the edge. Subsequently, the frames generated via edge interpolation are encoded before delivering to the HMDs. For instance, intra-frame compression of the H.26x coding standards [38] can help to effectively reduce the data size with slightly loss of details and negligible encoding time. We have tested the encoding speed of intra-frame compression using FFmpeg[4] under different quantization parameters (QP). Results in Table II show that the average time of encoding an 640$\times$480 tile frame is less than 2 ms. Note that, the decoding

[3]All the experiments are conducted using RIFE as the interpolation network, due to its relatively low complexity and acceptable frame quality. In case interpolated frames with higher quality are expected, alternative interpolation NNs [24] [25] can also be considered with higher time consumption.

[4]http://www.ffmpeg.org/.

TABLE II: Intra-frame encoding time

QP	15	22	27	32
Time (ms)	1.62	1.24	1.05	1.01

time is even less than that of encoding, thus can be ignored as well [10]. Another benefit of applying intra-frame coding lies in the encoding independence of each frame, which allows more flexible decoding before video playback.

VI. Performance Evaluation

We now evaluate the E2E delay performance of the proposed system and the effects of different system parameters are also studied under various experimental settings.

A. Experimental Setup

As listed in Table III, we select a set of 4K VR videos from the dataset in [14] and YouTube[5] (Dogs without viewpoint traces) covering different genres, and crop them into 6×4 equal-size video tiles as recommended in [39] with a duration of 1 s. All video tiles are encoded at a QP value of 22[6], which is the best quality level according to some recent studies [2] [15]. The average decoding time of a video tile on the HMD using FFmpeg is around 20 ms after testing on a personal computer with an Intel i9-10900K CPU. The interpolation time of RIFE for one tile frame is around 10 ms using an RTX 3080 GPU. These settings are used for the HMD. Since the computing capacity of edge server is tens of times that of the HMD [21], the time consumption of decoding and interpolation on the edge is set to one-tenth of that on the HMD, in case of no parallel accelerators are used. The compression ratio of edge-interpolated frames is set to 1/2, and the original data size of edge-interpolated frame is set to 1000 kb. Real-world network traces, as shown in Fig. 8, in static (60 Mbps on average) and mobile (33 Mbps on average) cases from the dataset in [40] are selected to emulate practical bandwidth variation between the edge and the HMD. Besides, one square wave-like trace is synthesized with mean throughput at 50 Mbps.

Baselines. (1) *HMD* [23]: This baseline directly transmits all low frame rate video chunks to the HMDs, and executes all the frame interpolation tasks on a single HMD for each user, without edge intelligence. (2) *Edge* [27]: This algorithm deploys a neural interpolation model on an edge server to implement frame interpolation. Then it delivers all the edge interpolated frames to the HMDs, without consideration of task offloading between the edge and HMDs. (3) *HMD-Edge (T)* [19]: The computing capacity of the edge and HMD are both taken into account. The interpolation task is offloaded completely either to the edge or to the HMD, namely, the offloading ratio is $\in \{0, 1\}$. For fair comparison, frame similarity analysis is applied for these baselines and ours. Our algorithm is named *HMD-Edge* in the following.

[5]https://www.youtube.com/watch?v=D01wilu5j_M.

[6]The E2E delay analysis in Section IV holds under different encoding configurations since the source video tiles are completely decoded before interpolation. The amount of I, P and B frames only affect the value of decoding delay and video data size.

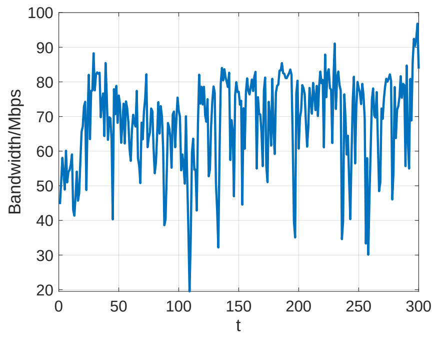
TABLE III: A Summary of Video Sequences

Video	**Type**	**Resolution**	**Length**	**frame rate**
Dive	scenery	3840x1920	29 s	30 fps
Grey	scenery	3840x1920	29 s	30 fps
Entrance	sports	3840x1920	30 s	25 fps
Sea	sports	3840x1920	30 s	30 fps
Plane	navigation	3840x1920	30 s	30 fps
Dogs	animal	3840x1920	30 s	60 fps

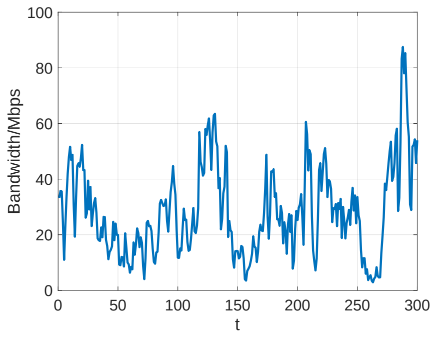
B. Performance Comparison

In order to evaluate the effectiveness of the proposed system, we compare the performance of the baselines and ours. Firstly, in order to verify the impact of different SSIM values as the interpolation threshold, we downsample video “Dogs" into 30 fps and compare the reconstruction quality and E2E delay of different schemes. Fig. 9(a) shows the quality of RIFE and frame averaging. Due to the strong fitting capability of neural networks, RIFE always outperforms frame averaging, especially when interpolated frames are added to every pair of consecutive frames (i.e., SSIM = 1). However, the time consumption increases dramatically with higher demand of video quality as shown in Fig. 9(b). Under such circumstances, partially offloading heavy interpolation task to the edge can significantly reduce the E2E delay. In comparison with the HMD-only and Edge-only schemes, the HMD-Edge scheme achieves a delay reduction by up to 51% and 14%, respectively, on account of the joint utilization of the computing resources on the edge and HMD. Since the task offloading ratio is derived at finer granularity in our work, i.e., mid-frames of a tile are generated by the edge and HMD jointly, our algorithm performs better than HMD-Edge (T). Based on the above results, we choose 0.98 as the similarity threshold to decide whether to interpolate or not in the following experiments. Note that other thresholds can also be employed, depending on the delay requirement of applications, network conditions and the computation capability of the HMD.

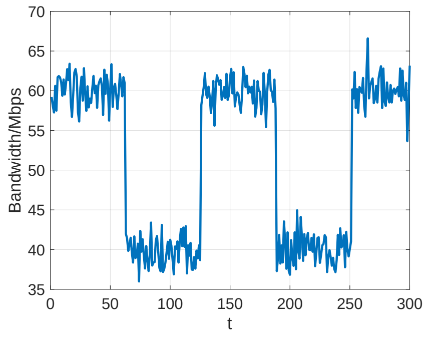
Figure 10(a)-10(c) illustrate the delay performance under different network traces. Higher bandwidth results in lower E2E delay by reducing the transmission delay of delivering mid-frames, especially for tiles with heavy interpolation tasks. It can be clearly seen that the long tail (corresponding to tiles with a large amount of mid-frames to be interpolated) in Fig. 10(b) decreases in Fig 10(a) and 10(c). However, if the interpolation task is excessively heavy, the delivery of interpolated frames may mitigate the benefits of offloading task to the edge. For example, as shown in Fig. 10(a), the HMD-only scheme sometimes outperforms the Edge-only scheme. Therefore, it is important to derive a rational offloading strategy to balance the computing cost and transmission cost as analyzed in Section IV. Since VR contents near the equator of the spherical viewing range usually contain a lot of motion, while those in the polar regions are with relatively low motion, the required amount of interpolated frames of different tiles are of high diversity. As a result, the consequent E2E delay of equatorial tiles and polar tiles can be quite different, leading to unsmoothness of the CDF plots. Moreover, the adopted network traces exhibit significant fluctuations, which also has



(a) Static trace (60 Mbps)

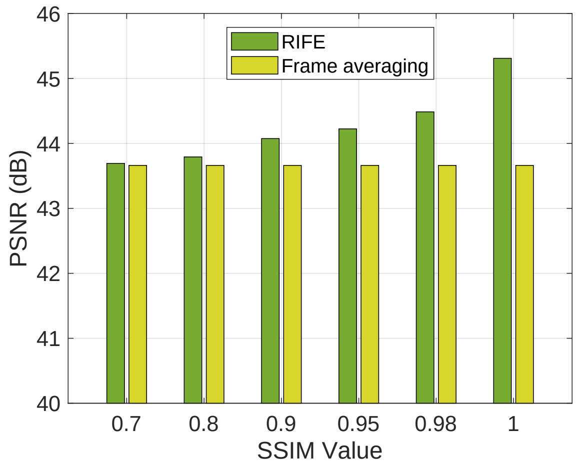


(b) Mobile trace (33 Mbps)

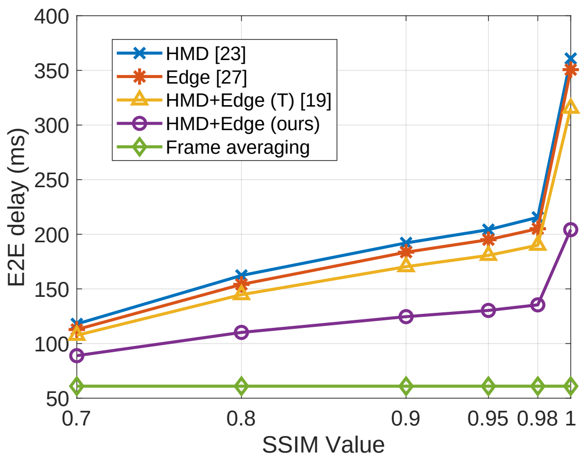


(c) Synthesized trace (50 Mbps)

Fig. 8: Snapshot of network traces.



(a) Quality of different interpolation methods.



(b) E2E delay of different interpolation methods.

Fig. 9: Impact of different SSIM values.

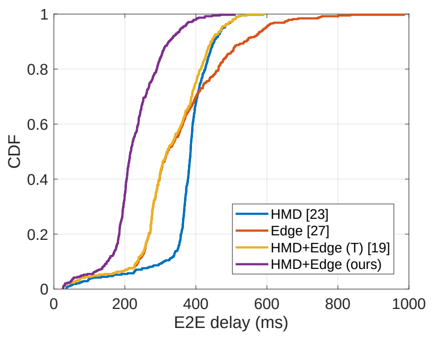
an impact on the delay performance.

Meanwhile, the capabilities of different schemes to interpolate tile frames are evaluated. Considering that the prefetching time (i.e., prediction time window of viewpoint algorithms) of video tiles should be carefully determined to ensure high viewpoint prediction accuracy while preventing stalls, we evaluate under different prefetching time settings as in [3] [41]. The average number of interpolated mid-frames for a chunk of different VR videos is presented in Fig. 11(a) and 11(b). Under different time settings, the HMD-Edge scheme always achieves the best performance and the performance gain grows with the increase of prefetching time. With the prefetching time setting to 500 ms, the transmission delay of tile bin files accounts for a large portion thus there is little remaining time to perform frame interpolation. Consequently, the edge server with far more computing resources is more likely to deal with part of the task while it is tough for the HMD to undertake within a short time. Since the amount of interpolated frames is relatively small within 500 ms, it is more beneficial to offload interpolation tasks to the Edge. Hence, the performance of Edge and HMD-Edge (T) are almost equivalent. With longer prefetching time as illustrated in Fig. 11(b), the gaps among different schemes are further enlarged thus it is advantageous to realize the computational potential of employing user HMD. However, prefetching tiles too early may cause decrease in viewpoint prediction accuracy, which means more missed tiles should be retransmitted leading to larger transmission delay of delivering bin files. This in turn deteriorates the interpolation performance of all schemes.

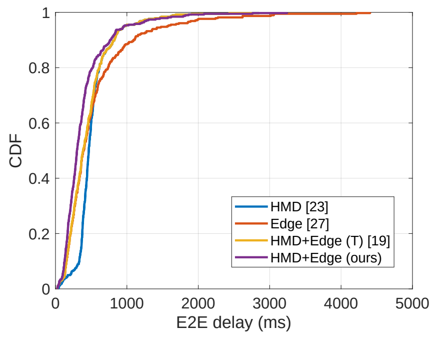
C. *Impact of System Parameters*

The delay analysis in Section IV implies that, the E2E delay is affected by various factors. Hence, we carry out experiments under different parameter settings to figure out their influences on the system performance in the case of enhancing frame rate of video “Plane" from 30 fps to 90 fps. Since we mainly study the potential of end-edge collaboration, the following experiments focus on the comparisons between the Edge-only scheme and HMD-Edge scheme and the conclusions will be similar when involving other schemes.

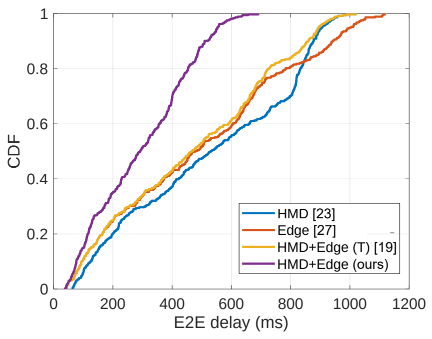
Impact of video data size: The selected VR video is encoded into different quality levels with different data size by adjusting the QP from 22 to 32. The CDF of the E2E delay with different video data size is shown in Fig. 12(a). It can be seen that the HMD-Edge scheme undoubtedly outperforms the Edge-only scheme in all conditions since it not only exploits the strong computing resources on the edge, but also releases the pressure of transmitting the mid-frames from the edge to the user by partially offloading the frame interpolation task to



(a) Static trace (60 Mbps)

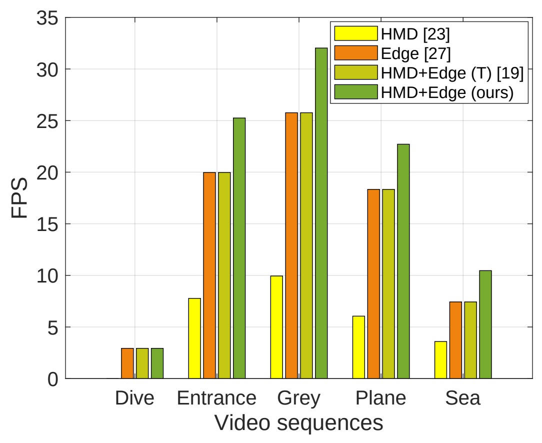


(b) Mobile trace (33 Mbps)

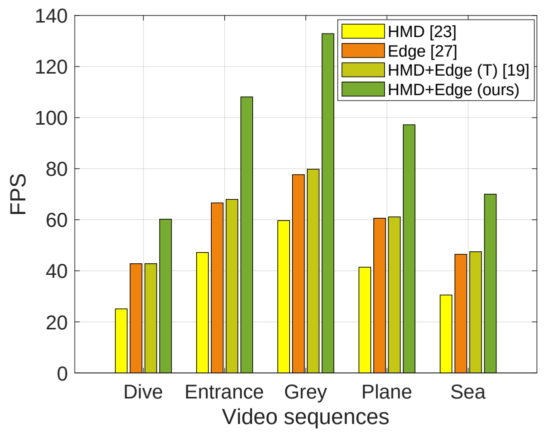


(c) Synthesized trace (50 Mbps)

Fig. 10: Delay performance under different network traces.



(a) Prefetching time = 500 ms.



(b) Prefetching time = 1000 ms.

Fig. 11: Interpolation capability under different prefetching time.

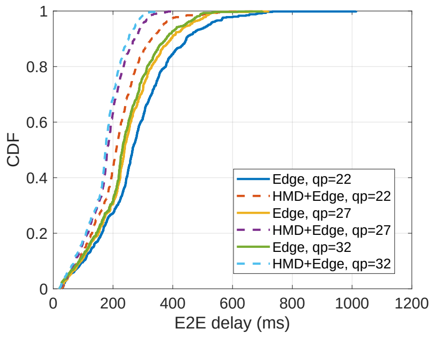
both sides. For each scheme, the E2E delay decreases with the growth of QP as a larger QP leads to smaller video data size at the cost of deteriorating video quality. However, the delay reduction resulted from reducing video data size gradually becomes marginal, since the E2E delay is dominated by the delivery of edge-interpolated frames and interpolation delay.

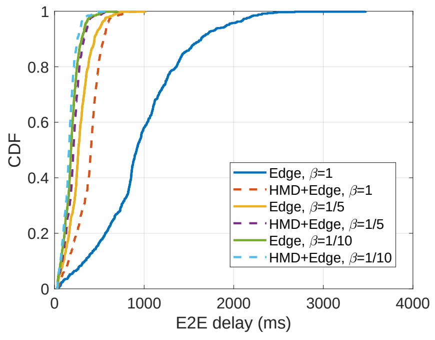
Impact of compression ratio: Fig. 12(b) compares the E2E delay of the Edge-only scheme with that of the HMD-Edge scheme under different settings of compression ratio of the interpolated mid-frames. If those mid-frames are delivered without further compression, the corresponding E2E delay is largely affected by the transmission delay, especially for the Edge-only scheme which needs to transmit all the mid-frames to the user according to the interpolation decision. However, slight compression of the pristine interpolated mid-frames can lead to a huge delay reduction with almost negligible encoding and decoding time cost and loss of quality. With a higher compression ratio, the data to be transmitted after completing the frame interpolation task on the edge becomes less thus lowering the requirement of bandwidth and further reducing the E2E delay. But later, the delay reduction resulting from the growth of compression ratio rarely increases because the time consumption of the frame interpolation task is gradually dominant over the transmission delay.

Impact of edge computing capacity: As previously mentioned, γ denotes the ratio of the processing rate of the edge to that of the HMD. The results of evaluating under different settings of edge computing capacity are exhibited in Fig. 12(c). It can be distinctly observed that both schemes benefit from the growth of computing resources on the edge server. Same as increasing the compression ratio, when the required number of the interpolated mid-frames is small or the data size of the tile bin files is not large, more computing resources can eliminate the computational deficiency of the HMD. For example, the green line with $\gamma = 20$ is above the red line with $\gamma = 4$ at the beginning of the CDF curves, pointing out the necessity to assign more frame interpolation task to the edge. Besides, the advantage of appending more computing resources is in decline since the transmission of the mid-frames gradually becomes the bottleneck of VR streaming.

D. Multi-User Scenario

To verify the performance in the multi-user case, we conduct experiments under the impact of the number of users and different parameter settings. Fig. 13(a) illustrates the average E2E delay when users watch the same video, “Dive" or “Plane” (different videos correspond to different Gamma distributions). The E2E delay increases with the number of users since bandwidth and computing resources are equally allocated to multiple users, leading to an increase of transmission delay


(a) Impact of video data size.


(b) Impact of compression ratio.

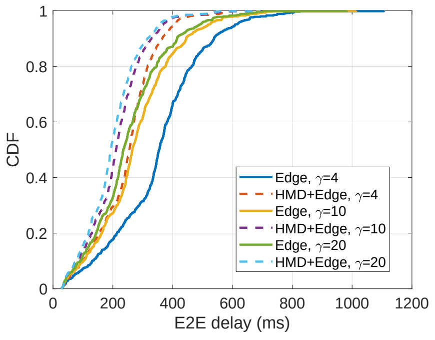
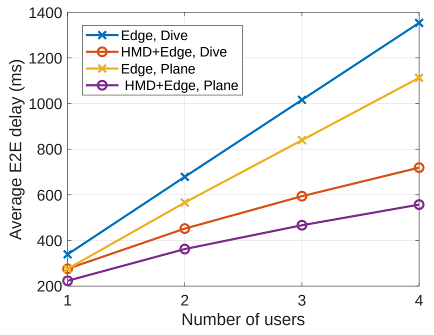
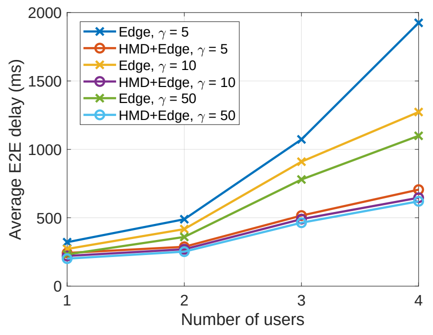

(c) Impact of edge computing capacity.

Fig. 12: Impact of system parameters.


(a) Average E2E delay under different videos.


(b) Impact of edge computing capacity.

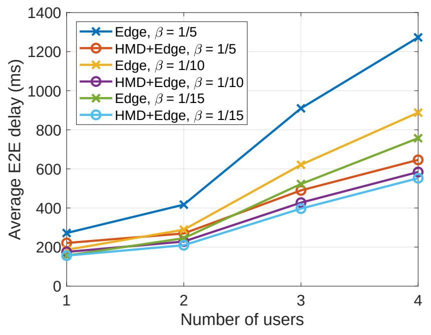

(c) Impact of compression ratio.

Fig. 13: Average E2E delay under different number of users watching different videos.

and interpolation delay for a single user. Besides, edge queuing delay of video packets grows due to concurrent video traffic. As shown in Fig. 13(a), when multiple users concurrently request VR content from the same source, the potential of utilizing one edge server to share the frame interpolation task decreases. Therefore, in this context, more edge servers should be deployed in the vicinity of users to jointly increase the frame rate of VR videos or deploy more computing resources on a single edge server.

The E2E delay performance when users watch different VR videos are illustrated in Fig. 13(b) and 13(c), where computing and bandwidth resources are allocated in proportion to the amount of required interpolated frames. Deploying more computing resources on the edge server undoubtedly reduces the E2E delay by alleviating the edge interpolation and queuing burden. However, the gain from increasing computing resources becomes marginal, since the transmission of video tiles or interpolated mid-frames is the bottleneck in this condition. Therefore, to further reduce the E2E delay, the impact of compression ratio is also shown in Fig. 13(c). Similar to adjusting compression ratio in the single-user case, further compression of edge interpolated frames significantly reduces the delay of delivering interpolated frames. In some cases, the Edge-only scheme with smaller compression ratio even achieves lower delay compared with the HMD-Edge scheme with higher compression ratio. For instance, the green curve (β = 1/15) is below the red curve (β = 1/5) when the number of users is small.

E. Discussion

Based on the above experiments, the superiority of the proposed end-edge collaboration scheme can be clearly noted under different network conditions and system settings. By jointly exploiting the computing resources on the edge and HMDs, the HMD-Edge scheme always achieves the lowest E2E delay when the computation-intensive interpolation task is the bottleneck of VR video streaming. However, if the bandwidth is not sufficient to guarantee timely delivery of edge-interpolated frames, it is more beneficial to offload more tasks to the HMDs, suggested by the experimental results. This conclusion also holds in the multi-user scenario. Besides, since multiple users compete for bandwidth and computing resources, more edge nodes should be involved to further provide improved services under this circumstance.

For the future work, we will investigate the application of the proposed system with the assistance of multiple edge servers. The cost of deploying edge nodes should be taken into account to yield more practical solutions. Besides, energy consumption of edge servers and mobile devices cannot be ignored, since the computing resources are limited and the demand of green and sustainable development is gaining growing popularity.

VII. Conclusion

In this paper, we have proposed an efficient end-edge collaborative VR video streaming system through neural frame interpolation to significantly increase VR frame rate for improving visual quality. To cope with the great demand for computing resources, an edge server and several VR HMDs have been jointly applied to implement the computation-intensive neural interpolation tasks, guaranteeing timely VR content delivery. Besides, we have modeled the queuing of video frames using Gamma distribution to characterize heavy video traffic in multi-user scenario. Based on the E2E delay analysis of the proposed system, an optimal task offloading strategy is developed to minimize the E2E delay via balancing computation and transmission burden on the edge and HMDs. Finally, real-world video sequences have been used to evaluate the system performance and the results demonstrate the effectiveness under various network conditions.

APPENDIX A
PROOF OF LEMMA 1

In Section IV-B, the upper bound of average queuing delay is given in Eq. (13), where VR video tile flow is characterized as a GI/G/1 queuing process. In the special M/G/1 case (with Poisson arrivals, i.e., $\sigma_m^{(\mathrm{a})} = 1/\lambda_m$), the mean queue length proves to be

$$L = \frac{\rho_m^2 + [\lambda_m \sigma_m^{(\mathrm{s})}]^2}{2(1-\rho_m)}, \tag{19}$$

which is called the Pollazcek-Khintichine formula [42]. According to Little's rule, the mean waiting time in the M/G/1 queue is thus

$$q' = L/\lambda_m = \frac{\rho_m^2 + [\lambda_m \sigma_m^{(\mathrm{s})}]^2}{2\lambda_m(1-\rho_m)}. \tag{20}$$

It can be noted that the derivative of Eq. (20) with respect to $\sigma_m^{(\mathrm{s})}$ is equivalent to that of the upper bound given by Eq. (13). Therefore, the actual queuing delay in the GI/G/1 queue can be approximated by its boundary with a normalization factor [36], which can be further derived based on the special M/G/1 case. By substituting $\sigma_m^{(\mathrm{a})} = 1/\lambda_m$ into Eq. (13), the right term scaled down by a normalization factor, should be equal to Eq. (20). The normalization factor can thus be derived as $\frac{1+[\sigma_m^{(\mathrm{s})}\mu_m]^2}{1/\rho_m^2+[(\sigma_m^{(\mathrm{s})})\mu_m]^2}$. Consequently, the average queuing delay of the GI/G/1 queuing process is

$$q \approx \frac{1+[\sigma_m^{(\mathrm{s})}\mu_m]^2}{1/\rho_m^2+[(\sigma_m^{(\mathrm{s})})\mu_m]^2} \cdot \frac{[(\sigma_m^{(\mathrm{s})})^2 + (\sigma_m^{(\mathrm{a})})^2]}{2[(1/\lambda_m)-(1/\mu_m)]}, \tag{21}$$

which concludes the proof.

REFERENCES

[1] C. Liu, K. Wang, H. Zhang, X. Li, and H. Ji, "Rendered tile reuse scheme based on FoV prediction for MEC-assisted wireless VR service," *IEEE Trans. Netw. Sci. Eng.*, 2023.

[2] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *Proc. ACM SIGCOMM*, 2019, pp. 394–407.

[3] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive adaptive streaming to enable mobile 360-degree and VR experiences," *IEEE Trans. Multimedia*, vol. 23, pp. 716–731, 2020.

[4] A. Xiao, X. Huang, S. Wu, H. Chen, and L. Ma, "Traffic-aware rate adaptation for improving time-varying QoE factors in mobile video streaming," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2392–2405, 2020.

[5] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, 2020.

[6] "IEEE standard for head-mounted display (HMD)-based Virtual Reality (VR) sickness reduction technology," *IEEE Std 3079-2020*, pp. 1–74, 2021.

[7] S. Lee, S. Kim, H. G. Kim, and Y. M. Ro, "Assessing individual VR sickness through deep feature fusion of VR video and physiological response," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 5, pp. 2895–2907, 2022.

[8] S. Fremerey, F. Hofmeyer, S. Göring, D. Keller, and A. Raake, "Between the frames-evaluation of various motion interpolation algorithms to improve 360° video quality," in *Proc. IEEE ISM*, 2020, pp. 65–73.

[9] Y. Omori, T. Onishi, H. Iwasaki, and A. Shimizu, "A 120 fps high frame rate real-time HEVC video encoder with parallel configuration scalable to 4K," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 491–499, 2018.

[10] 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," in *Proc. ACM Multimedia*, 2020, pp. 982–990.

[11] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, 2019.

[12] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5520–5530, 2019.

[13] P. K. Mu, J. Zheng, T. H. Luan, L. Zhu, M. Dong, and Z. Su, "AMIS: Edge computing based adaptive mobile video streaming," in *Proc. IEEE INFOCOM*, 2021, pp. 1–10.

[14] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze prediction in dynamic 360 immersive videos," in *Proc. IEEE CVPR*, 2018, pp. 5333–5342.

[15] S. Yang, P. Yang, H. Wang, N. Zhang, and L. Yu, "Just-noticeable-difference based coding and rate control of mobile 360° video streaming," in *Proc. IEEE GLOBECOM*, 2021, pp. 01–06.

[16] L. Ren, Y. Laili, X. Li, and X. Wang, "Coding-based large-scale task assignment for industrial edge intelligence," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2286–2297, 2019.

[17] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, 2021.

[18] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing adaptive mobile video streaming," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 787–800, 2018.

[19] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7573–7586, 2019.

[20] T. Dang and M. Peng, "Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1594–1607, 2019.

[21] Q. Cheng, H. Shan, W. Zhuang, L. Yu, Z. Zhang, and T. Q. Quek, "Design and analysis of MEC- and proactive caching-based 360 mobile VR video streaming," *IEEE Trans. Multimedia*, vol. 24, pp. 1529–1544, 2022.

[22] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692–704, 2018.

[23] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou, "Real-time intermediate flow estimation for video frame interpolation," in *Proc. ECCV*, 2022.

[24] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proc. IEEE CVPR*, 2020, pp. 5437–5446.

[25] L. Lu, R. Wu, H. Lin, J. Lu, and J. Jia, "Video frame interpolation with transformer," in *Proc. IEEE CVPR*, June 2022, pp. 3532–3542.

[26] M. Usman, X. He, K.-M. Lam, M. Xu, S. M. M. Bokhari, and J. Chen, "Frame interpolation for cloud-based mobile video streaming," *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 831–839, 2016.

[27] C. Madarasingha and K. Thilakarathna, "Edge assisted frame interpolation and super resolution for efficient 360-degree video delivery," in *Proc. ACM Mobicom*, 2022, pp. 856–858.

[28] J. Kim, Y.-G. Kim, H. Song, T.-Y. Kuo, Y. J. Chung, and C.-C. Kuo, "TCP-friendly internet video streaming employing variable frame-rate encoding and interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1164–1177, 2000.

[29] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing

environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, 2020.

[30] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[31] Q. Feng, P. Yang, F. Lyu, and L. Yu, "Perceptual quality aware adaptive 360-degree video streaming with deep reinforcement learning," in *Proc. IEEE ICC*, 2022, pp. 1190–1195.

[32] J. Meng, S. Paul, and Y. C. Hu, "Coterie: Exploiting frame similarity to enable high-quality multiplayer VR on commodity mobile devices," in *Proc. ACM ASPLOS*, 2020, pp. 923–937.

[33] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[34] J. Chen, P. Yang, Q. Ye, W. Zhuang, X. Shen, and X. Li, "Learning-based proactive resource allocation for delay-sensitive packet transmission," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 2, pp. 675–688, 2020.

[35] S. Tanwir and H. Perros, "A survey of VBR video traffic models," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1778–1802, 2013.

[36] W. G. Marchal, "An approximate formula for waiting time in single server queues," *AIIE Trans.*, vol. 8, no. 4, pp. 473–474, 1976.

[37] S.-W. Lee, Y.-M. Kim, and S. W. Choi, "Fast scene change detection using direct feature extraction from mpeg compressed videos," *IEEE Trans. Multimedia*, vol. 2, no. 4, pp. 240–254, 2000.

[38] M. Kazemi, M. Ghanbari, and S. Shirmohammadi, "Intra coding strategy for video error resiliency: Behavioral analysis," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2193–2206, 2020.

[39] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. ACM MobiCom*, 2018, pp. 99–114.

[40] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4G LTE dataset with channel and context metrics," in *Proc. 9th ACM Multimedia Syst. Conf.*, 2018, pp. 460–465.

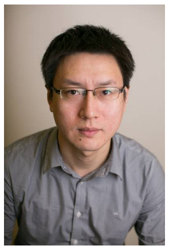
[41] Z. Huang, P. Yang, N. Zhang, F. Lyu, Q. Li, W. Wu, and X. Shen, "QoE-driven mobile 360 video streaming: Predictive view generation and dynamic tile selection," in *Proc. IEEE/CIC ICCC*, 2021, pp. 1113–1118.

[42] S. I. Gass and C. M. Harris, *Pollaczek-Khintchine formula*. New York, NY: Springer US, 2001, pp. 619–620.

Sushu Yang received her B.E. degree in Electronic Information Engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2021. She is currently working toward her M.S. degree in Information and Communication Engineering at the School of Electronic Information and Communications, HUST. Her current research interests are VR video streaming and mobile edge computing.

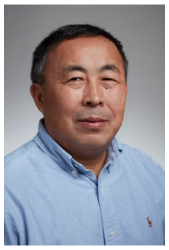
Peng Yang (Member, IEEE) received his B.E. degree in Communication Engineering and Ph.D. degree in Information and Communication Engineering from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2013 and 2018, respectively. He was with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, as a Visiting Ph.D. Student from 2015 to 2017, and a Postdoctoral Fellow from 2018 to 2019. Since 2020, he has been an Associate Professor with the School of Electronic Information and Communications, HUST. His current research focuses on mobile edge computing, video analytics, and virtual reality.

Jiayin Chen (Member, IEEE) received the B.E. degree and the M.S. degree in the School of Electronics and Information Engineering from Harbin Institute of Technology, Harbin, China, in 2014 and 2016, respectively, and the Ph.D. degree in the Department of Electrical and Computer Engineering from University of Waterloo, Waterloo, ON, Canada, in 2021. Since 2021, she has been a postdoctoral fellow with the Department of Electrical and Computer Engineering from The University of British Columbia, Vancouver, BC, Canada. Her research interests are in the area of vehicular networks, Internet of Things, and self-organized network, with current focus on sidelink communication technology based self-organized network.



Qiang Ye (Senior Member, IEEE) received the PhD degree in Electrical and Computer Engineering from the University of Waterloo, ON, Canada, in 2016. Since Sept. 2021, he has been an Assistant Professor with the Department of Computer Science, Memorial University of Newfoundland, NL, Canada. Before joining Memorial, he had been with the Department of Electrical and Computer Engineering and Technology, Minnesota State University, USA, as an Assistant Professor from Sept. 2019 to Aug. 2021 and with the Department of Electrical and Computer Engineering, University of Waterloo as a Postdoctoral Fellow and then a Research Associate from Dec. 2016 to Sept. 2019. He has published over 55 research articles on top-ranked IEEE Journals and Conference Proceedings. He is/was General and TPC co-chairs for different international conferences and workshops, e.g., IEEE VTC'22, IEEE INFOCOM'22, and IEEE IPCCC'21. He serves/served as Associate Editors of IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, Peer-to-Peer Networking and Applications, ACM/Wireless Networks, and International Journal of Distributed Sensor Networks. He also serves as the IEEE Vehicular Technology Society (VTS) Regions 1-7 Chapters Coordinator (2022-2023). He is a Senior Member of IEEE.

Ning Zhang (Senior Member, IEEE) is an Associate Professor in the Department of Electrical and Computer Engineering at University of Windsor, Canada. He received the Ph.D degree in Electrical and Computer Engineering from University of Waterloo, Canada, in 2015. After that, he was a postdoc research fellow at University of Waterloo and University of Toronto, respectively. His research interests include connected vehicles, mobile edge computing, wireless networking, and machine learning. He is a Highly Cited Researcher. He serves as an Associate Editor of IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, and IEEE SYSTEMS JOURNAL; and a Guest Editor of several international journals, such as IEEE WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. He also serves/served as a TPC chair for IEEE VTC 2021 and IEEE SAGC 2020, a general chair for IEEE SAGC 2021, a track chair for several international conferences and workshops. He received 8 Best Paper Awards from conferences and journals, such as IEEE Globecom and IEEE ICC.



Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks.

Dr. Shen received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premiers Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE INFOCOM'14, IEEE VTC'10 Fall, IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. Dr. Shen is the President of the IEEE Communications Society. He was the Vice President for Technical & Educational Activities, Vice President for Publications, Member-at-Large on the Board of Governors, Chair of the Distinguished Lecturer Selection Committee, Member of IEEE Fellow Selection Committee of the ComSoc. Dr. Shen served as the Editor-in-Chief of the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and IET COMMUNICATIONS. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.