Towards Optimal Real-time Volumetric Video Streaming: A Rolling Optimization and Deep Reinforcement Learning Based Approach

Jie Li, *Member, IEEE*, Huiyu Wang, Zhi Liu, *Senior Member, IEEE*, Pengyuan Zhou, *Member, IEEE*, Xianfu Chen, *Member, IEEE*, Qiyue Li, *Senior Member, IEEE*, Richang Hong, *Member, IEEE*

***Abstract*—Volumetric video provides users with a good viewing experience of six degrees of freedom (DoF) and has wide applications in many fields such as teleconferencing and online games. However, the huge data volume and strict latency requirements of point cloud video, the most popular representative of volumetric video, pose a challenge to its transmission. Existing point cloud video transmission algorithms usually segment a long video by every one or several group of frames, predict network bandwidth and FoV information, then perform adaptive transmission by solving the quality of experience (QoE) optimization problem. However, such segmentation neglects the impact of current optimization decisions on the subsequent video streaming process, as well as the large prediction error for a long interval, severely degrading user's QoE. Moreover, the complex constrained optimization problem makes the solution time too long to meet the real-time video streaming requirements. To this end, in this paper, we propose a rolling prediction-optimization-transmission (POT) framework, which makes predictions of network bandwidth and FoV in each short rolling window to reduce prediction error. And our framework takes into account the upper bounded QoE contribution of the subsequent point cloud video to improve the system performance. In addition, we design a deep reinforcement learning based real-time solver to make decisions of the fixed structure optimization problem in each roll, allowing our system to run in real-time. We have performed simulations and experiments, and the results show that our solution outperforms existing methods.**

***Index Terms*—Point cloud video, video streaming, real-time transmission, QoE model, deep reinforcement learning, rolling optimization**

I. Introduction

Volumetric video is expected to be the next generation of video, offering an immersive viewing experience with six degrees of freedom (DoF), being a key application of 6G cellular networks with great promise in areas such as healthcare, entertainment, and education [1], [2]. Point cloud video is one of the most popular manifestations of volumetric video. It can form holographic three-dimensional video in space, and users can freely watch the video from various angles, providing 6DoF immersive viewing experience [3], [4], [5].

However, point cloud video streaming has extremely high bandwidth and low latency requirements [6], [7], [8]. For example, for a point cloud video with 30 frames per second (fps), when the number of points is about 760,000 per frame, the amount of data in one single frame without any compression is about 12 MB, which means that the bandwidth requirement for the video streaming is as high as 2.9 Gbps [9]. In addition, to avoid users' discomfort due to rapid head rotation, the motion-to-photon (MTP) delay needs to be less than 20 milliseconds [10], which poses a great challenge for real-time point cloud video streaming.

To address these challenges, most of the point cloud video streaming over HTTP methods use tiling technologies, as widely used in VR/360 video streaming [11]. These methods first cut the video content into tiles and encode them at different quality levels [12]. Then the system needs to use the future user's field of view (FoV) and network bandwidth information to select the appropriate video representation by optimizing the quality of experience (QoE) using adaptive bitrate (ABR) algorithms [13], [14].

If the future FoV and network bandwidth can be perfectly obtained, the ABR algorithm can maximize the user's QoE. An intuitive approach is to predict the information for the whole video length, but this does not take into account the impact of long-term prediction inaccuracies. The commonly used traditional ABR algorithms only predict the information of one future group of frames (GoF) when solving the optimization problem [15], [16]. However, such a system cannot respond in a timely manner to sudden network bandwidth fluctuations or changes in user's FoV for future GoFs, making it difficult for the system to cope with complex and changing environments. Some researchers extend the traditional ABR, and their schemes divide the point

Jie Li, Huiyu Wang and Richang Hong are with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China (e-mail: lijie@hfut.edu.cn; wanghuiyu@mail.hfut.edu.cn; hongrc.hfut@gmail.com).

Zhi Liu is with The University of Electro-Communications, Tokyo, Japan (e-mail: liuzhi@uec.ac.jp).

Pengyuan Zhou is with University of Science and Technology of China, Hefei, China. (e-mail: pyzhou@ustc.edu.cn).

Xianfu Chen is with VTT Technical Research Centre of Finland, Finland. (e-mail: xianfu.chen@vtt.fi).

Qiyue Li is with the school of Electrical Engineering and Automation, Hefei University of Technology, and Engineering Technology Research Center of Industrial Automation Anhui Province, Hefei, Anhui 230009, China. (e-mail: liqiyue@mail.ustc.edu.cn).

cloud video into several segmentations by every several GoFs, perform FoV and network bandwidth prediction and then try to find optimal decisions in each segmentation [17]. This method is able to cope with the impact of fluctuations in predictor variables on decision-making, however, it does not take into account the current GoF decisions' impact on the future GoFs.

Furthermore, the above-mentioned studies cannot guarantee the performance of the real-time streaming of point cloud videos. In a practical Dynamic Adaptive Streaming over HTTP (DASH) based system, the transmission delay is mainly composed of two parts: the downloading and decoding time of point cloud video contents, and the time of solving the formulated optimization problem. The former is related to the data volume of the point cloud video and the condition of network bandwidth, while the latter is influenced by the scale of the optimization problem and the complexity of the solution algorithm. Usually, the streaming optimization problem is a large-scale binary integer programming problem. Traditional optimization methods, such as the branch-bound algorithm [18], [19], are often very slow to solve such complex problems, which makes it difficult to guarantee real-time performance. Traditional heuristic algorithms such as the ant colony algorithm [20] and genetic algorithm [21], [22] have relatively poor solution quality and degrade user's QoE. Some researchers propose deep reinforcement learning (DRL)-based solutions to solve the optimization problem [23], [24], which can effectively improve the solution speed and meet the requirements of a real-time streaming system. The DRL models try to decide the quality levels of all the tiles in one-time parallel solving. However, in a point cloud video streaming system, there are very complicated constraints and a huge number of decision variables in the optimization problem, which makes it very hard for the parallel DRL models to learn the optimal policy.

To address the above issues, we propose a rolling prediction-optimization-transmission (POT) framework to maximize users' QoE, which considers the objective quality, rebuffering time and quality switches. For each decision of the kth GoF, we segment the long point cloud video into two parts, the N GoFs starting from k, and the remaining GoFs. we only perform FoV and network bandwidth prediction for the first N GoFs to avoid long prediction errors. We also fully consider the impact of decisions at kth GoF on the subsequent video. Unlike the traditional ABR methods, we transform the original problem into a rolling optimization problem, then analyze the nature of the two optimization problems and the gaps in their solutions. We prove that there is an upper bound of the QoE contribution provided by the remaining GoFs, although non-negligible. Thus we reformulate the QoE optimization problem for the whole video into a rolling N GoF lengths optimization problem. In our POT framework, by performing short-time prediction, optimization and transmission at each GoF, we can achieve optimal streaming for the whole video length.

In addition, we propose a DRL-based optimization problem solver, Serial Cyclic Dueling Deep Q-Network (SC-DDQN), to meet the real-time requirements of the streaming system. Regarding the large solution space and complex constraints of N GoF lengths optimization problem, SC-DDQN learns the solving of the optimization problem with the same structure and only several changing parameters such as FoV and network bandwidth. Besides, SC-DDQN selects the quality level of each tile serially and sequentially, instead of determining the quality levels of all tiles at once. It reduces the instability of DRL model due to the complex constraints of the optimization problem. By embedding SC-DDQN solver into our POT framework, our system can achieve optimal real-time volumetric video streaming. We conduct extensive simulations, as well as an experimental system with Microsoft HoloLens 2 to verify performance of the proposed scheme. The results have shown the advantages of the proposed scheme. The main contributions of this paper are as follows:

1) A rolling POT framework is proposed to avoid the problems caused by the video segmentation and large prediction errors of long interval. Our framework considers the impact of current optimization decisions on future GoFs, and uses rolling to prediction-optimization-transmission in each GoF to cope with the fluctuation of bandwidth and FoV.
2) SC-DDQN is designed to solve the optimization problem of in rolling POT framework in real-time. It is able to reduce the model instability caused by the complex constraints, learn the solving procedure of the fixed structure optimization problems and realize real-time solutions.
3) Extensive simulations and experiments are conducted and the results have shown the advantages of the proposed scheme. We have also made all user datasets and source programs public for future research.

The remainder of this paper is organised as follows. Section II reviews relevant researches in the area of point cloud video streaming and related optimization methods. Section III introduces our real-time point cloud streaming system and formulates the problem. In section IV, we describe the proposed rolling POT framework. The detailed design process of DRL based real-time SC-DDQN solver is illustrated in section V. In section VI, we design comparative simulations and HoloLens based experiments to verify the performance of our system. Finally, section VII concludes our paper.

II. Related work

A. Point cloud video streaming

To cope with the challenges of point cloud video streaming, there have been many studies in terms of acquisition, coding, tiling and transmission optimization.

Some commercial volumetric video capture and processing systems, e.g., Evercoast[1], Mantis Vision[2], 4Dviews[3] and Metastage[4], triangulate point clouds producing sequences of polygonal meshes, to generate 3D representations of a scene in volumetric video capture and processing systems, and can provide more accurate 3D models for advanced post-processing tasks and higher quality point cloud video service. The high-quality point cloud acquisition systems also attract a lot of attention from the academic community. For example, several studies have focused on optimizing acquisition systems for point cloud capture based on triangulation[25][26][27], to obtain high-quality raw point clouds which bring user a better viewing experience.

There are some studies focusing on point cloud coding. [28] introduces a point cloud codec Draco, which can significantly reduce the computational complexity and decrease the latency caused by coding and decoding compared with other coding and decoding methods [29], [30]. However, its compression ratio is low and requires larger bandwidth. For 3D-tiling techniques, [31] introduces a system specifically for point cloud video, which chunks point cloud video into video slices and removes or reduces the quality level of certain tiles based on the user's viewpoint and the distance of the viewpoint relative to the chunks, thus improving the user's viewing experience with limited network bandwidth. [9] proposes a completely different saliency-based hybrid tiling scheme and builds a prototype system to verify its performance, improving the average QoE by about 23% compared to conventional 3D-tiling technique. L. Wang et al. [32] propose a new QoE formula based on the user's FoV and spatial location, inter-tile occlusion, and rendering device's resolution and transforms the optimization problem into a multi-choice knapsack problem to achieve a lower computational complexity. B. Han et al. [33] present a comprehensive study of point cloud video streaming over mobile and propose three visibility optimization schemes of great importance, namely viewport visibility (VV), occlusion visibility (OV) and distance visibility (DV), which can reduce data usage by an average of 40% with almost no loss of actual video viewing quality. The authors of [11] design a scheme to allocate the computational and communication resources for maximizing the QoE of the point cloud video streaming, where the server prepares compressed chunks (which occupy more computational resources) and uncompressed chunks (which occupy more communication resources) of different quality levels, and weighs the type and level of chunks to be transmitted based on network throughput and CPU processing power. There are also some researches on streaming in the area of VR video and traditional video, e.g.,[34] focuses on multi-view 2D video streaming transmission. [35] investigates the optimization transmission of 360° video, and it formulates this elaborate optimization problem by taking into account the video type, user's viewing probability, and FoV distribution, and solves it using the divide-and-conquer approach. [36] formulates the peer-to-peer (P2P) video on demand (VoD) streaming into an optimization problem that maximizes video bitrates under bandwidth constraints and solves it using a distributed greedy algorithm. However, the above studies tend to consider the prediction information of only one future GoF when calculating the QoE, which makes it difficult for the system to cope with complex network environments. Instead, our study calculates the QoE of N future GoFs and performs rolling optimization to predict the fluctuations of the predictor variables so that it can cope with different network environments.

[1]https://evercoast.com/
[2]https://mantis-vision.com/3d-studio-3iosk/
[3]https://www.4dviews.com/
[4]https://metastage.com/our-tech/

B. Optimization Methods in Video Streaming

There are some studies in the field of video streaming that apply rolling optimization methods or DRL algorithms for system optimization. For example, X. Yin et al. use the idea of rolling optimization to optimize the traditional video streaming [37], and use a table lookup-like approach to solve the optimization problem quickly in a practical scenario. The authors of [38] propose a RL algorithm called pensive, which differs from the traditional ABR algorithm in that this RL algorithm does not rely on a predefined model and various assumptions about the environment. In order to cope with the complexity and variability of video and mobile wireless channels, [39] proposes a deep Q-learning framework for dash video streaming with very good performance and fast convergence. [40] proposes the HD3 algorithm to cope with the high quality and low latency requirements of real-time video streaming, which is able to simultaneously determine the bit rate, target buffer level, and continuous latency limit values for complex decision problems in discrete-continuous hybrid space, and outperforms the traditional DRL algorithm. [41] uses a heuristic algorithm to reduce solving time, it remains challenging to meet the real-time requirements for point cloud video transmission problems with large search spaces. A recent work [42] proposes a federated reinforcement learning solution to address privacy concern for optimized point cloud video streaming experience. However, these studies do not fully consider the impact of future GoFs on the decisions of current GoFs, and these DRL algorithms tend to solve all decision variables in parallel at once, which is difficult to apply in point cloud video streaming with large problem sizes and complex constraints. Our study considers the impact of future GoFs and solves the optimization problem using a serial approach. It reduces the instability of DRL model due to the complex constraints of the optimization problem.

III. System Overview and Problem Formulation

A. System Overview

We consider a DASH based real-time point cloud video streaming system, which uses a client-server (C/S) architecture. In this system, the server groups the original point cloud video frames into K GoFs and divides the video content of each GoF into C tiles. Then each tile is encoded into up to L different quality level representations. At GoF k, $k \in K$, the server obtains downloading request of quality level l of tile c from the client, and then sends the corresponding encoded tiles to the client.

At the client side, the rolling POT framework divides the long point cloud video into two parts: the video from GoF k to GoF $k+N$ and the remaining part after GoF $k+N$. We only perform FoV and network bandwidth prediction for the first N GoFs to avoid long prediction errors, where N is much smaller comparing with K. Then the POT framework uses the prediction information to optimize the QoE within the rolling window, which is the duration of N GoFs, i.e, at each step of the rolling process, the QoE in the prediction interval N is first optimized so as to obtain the solutions of N GoFs. Then the solution of the first GoF is used as the decision variable, and the next GoF (although the decision variables for the next N GoFs are all decided) is transmitted accordingly. After the transmission of next GoF is completed, the prediction window is shifted forward by one GoF to continue this rolling process.

Unlike traditional ABR approaches, our framework fully considers the impact of the decision of the kth GoF on subsequent videos by proving that the QoE contribution provided by the remaining GoFs is upper bounded. For the optimization problem of in each roll, we design a DRL based solver to make decisions in real-time. Then the optimal representation of GoF k is sent to the server as download request. After all the tiles in GoF k are downloaded and decoded, the video content is put into the buffer and waits to be played. Fig. 1 illustrates the architecture of this real-time point cloud video streaming system.

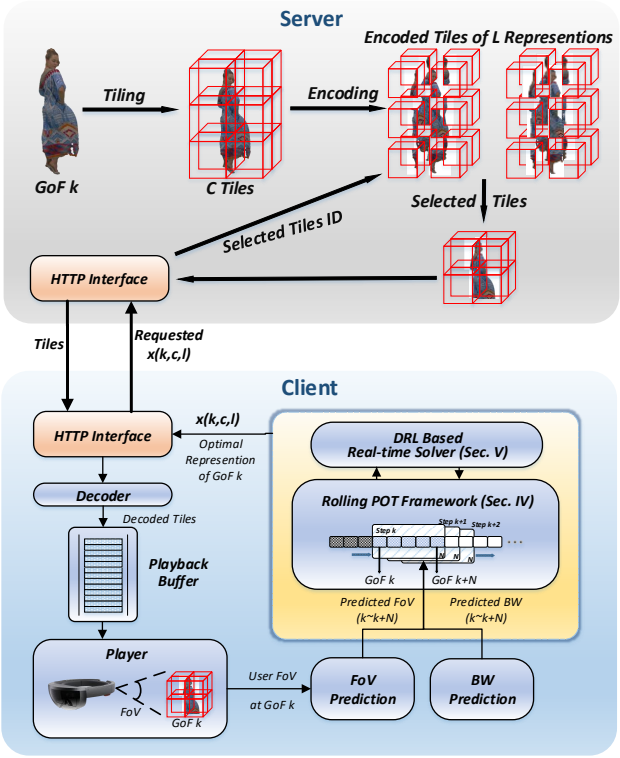


Figure 1. Architecture of the rolling POT framework based real-time point cloud video streaming system.

B. QoE model of Point cloud video

According to [11], [43], [44], the QoE model for point cloud video streaming should consider three main factors that have a large impact on user's viewing experience, which are the objective quality of tiles, impairment of rebuffer and quality switches. Thus, we denote the QoE of GoF k as QoE_k, which can be calculated as follows:

$$QoE_k = w_1 * Q_k^T - w_2 * P_k^S - w_3 * S_k^W \tag{1}$$

where Q_k^T represents the objective quality of GoF k, while P_k^S and S_k^W are the impairment of rebuffer and quality switches, respectively. w_1, w_2, and w_3 are the weights of these three components.

1) Objective quality for point cloud video: In general, the objective quality of each GoF can be measured by the peak signal-to-noise ratio (PSNR) of the tiles residing in the user's FoV. Thus, we define the objective quality Q_k^T of the GoF k, as follows:

$$Q_k^T = \sum_{c=1}^{C} \sum_{l=1}^{R} \left(q_{k,c,l} \times x_{k,c,l} \times f_{k,c}^V \right) \tag{2}$$

where c and l represent the tile number and quality level, respectively. $x_{k,c,l}$ is a binary variable, $x_{k,c,l} = 1$ means tile c with quality level l in GoF k is transmitted, otherwise $x_{k,c,l} = 0$. $f_{k,c}^V$ is also a binary variable that represents whether tile c in GoF k resides inside user's FoV. $f_{k,c}^V = 1$ means the tile c in GoF k resides inside user's FoV, and $f_{k,c}^V = 0$, otherwise. Here we assume that the FoV information and $f_{k,c}^V$ is obtained with perfect prediction. $q_{k,c,l}$ denotes the PSNR of tile c in GoF k with quality levels l, and it can be calculated at server side in advance.

2) Impairment of rebuffer: Let $s_{k,c,l}$ be the amount of data of tile c with quality level l in GoF k. Then the total transmission time of GoF k can be calculated as:

$$T_k^W = \frac{S_k}{BW_k} = \frac{\sum_{c=1}^{C} \sum_{l=1}^{R} \left(s_{k,c,l} \times x_{k,c,l} \times f_{k,c}^V \right)}{BW_k} \tag{3}$$

where BW_k is network bandwidth during the transmission of GoF k, which is considered to be obtained by

perfect prediction.

Let $p_{k,c,l}$ be the number of points of tile c with quality level l in GoF k. According to our previous work [15], the decoding time of tiles is positively correlated with the point number of tiles. Thus, we define the decoding time of the GoF k as:

$$T_k^C = F\left(\sum_{c=1}^{C}\sum_{l=1}^{R}\left(p_{k,c,l} \times x_{k,c,l} \times f_{k,c}^V\right)\right) \tag{4}$$

where $F()$ is the mapping function of the number of points to the decoding time. Let $[x]_+ := max(x, 0)$, then the playback buffer dynamic can be expressed as:

$$bf_{k+1} = T^p + [bf_k - T_k^W - T_k^C]_+ \tag{5}$$

where $T^p = f/fps$ represents the playback time of each GoF, which is a constant. f denotes the number of frames in a GoF, and fps is the number of frames that are drawn per second by the player. $bf_k \in [0, bf^M]$ means that the playback buffer should be less than the upper limit bf^M. Then the rebuffering time generated during the transmission of GoF k can be calculated as the time duration of playback buffer is drained.

$$P_k^S = [T_k^W + T_k^C - bf_k]_+ \tag{6}$$

3) Impairment of quality switches: The content of adjacent GoFs sometimes fluctuates in objective quality, which has a negative viewing experience for the users. And the greater the difference in quality level of the tiles, the closer the tiles to the viewer, the greater the impact on the viewer. Therefore, we define the quality switches as follows.

$$\begin{aligned} S_k^W = &\textstyle\sum_{c=1}^{C}(\left|\sum_{l=1}^{R} l \times x_{k,c,l} - \sum_{l=1}^{R} l \times x_{k-1,c,l}\right| \\ &\times D_{k,c} \times \min(f_{k,c}^V, f_{k-1,c}^V)) \end{aligned} \tag{7}$$

where $D_{k,c}$ is the coefficient weighting the distance, defined as:

$$D_{k,c} = \frac{B_{k,c}^S}{d_{k,c}} \tag{8}$$

Here $d_{k,c}$ is the distance from the tile c in the GoF k to the viewer and $B_{k,c}^S$ denotes the diagonal length of the bounding box of tile c in GoF k. Obviously, the further the tile is from the viewpoint, or the smaller the tile quality level fluctuates, the smaller the effect of quality switches.

C. Problem Formulation

For a point cloud video with K GoFs (where K is a usually very large number), we can construct the optimization problem $P1$ by optimizing the user's average quality of experience.

$$\mathbb{P}1: \max_{x_{k,c,l}} QoE = \max_{x_{k,c,l}}\left(\frac{1}{K}\sum_{k=1}^{K} QoE_k\right) \tag{9}$$

$$\text{s.t. } x_{k,c,l} \in [0,1] \tag{9a}$$

$$\sum_{l=1}^{L} x_{k,c,l} = 1, \forall k, \forall c \tag{9b}$$

$$bf_k \in [0, bf^M], \forall k \tag{9c}$$

$$Eq.2 - Eq.8 \tag{9d}$$

where constraint Eq. 9a indicates that the decision variable $x_{k,c,l}$ is a binary variable. Eq. 9b indicates that only one quality level can be selected for each tile.

To solve the problem $P1$, we need to predict FoV and bandwidth information for K GoFs in advance. However, the existing algorithms can only guarantee the prediction accuracy in a short time range, and the quality of prediction results decreases as the prediction interval increases [45], which will severely degrade user's QoE, especially for the usually long volumetric video with a large K.

In addition, the presence of a large number of constraints and complex search space of decision variables make problem $P1$ to be a binary integer programming problem of high complexity. This poses a great challenge for real-time solving. When the point cloud video is long, the prediction interval is further stretched and the problem size is further increased, solving problem $P1$ in one go is not a good choice, which leads to longer solution time and poorer solution quality.

Traditional naive ABR algorithms [46] predict FoV and bandwidth, and then solve the optimization problem for only one GoF. It avoids the impact of inaccurate long interval predictions, but is unable to make decisions about changes in FoV and bandwidth in advance. Another kind of algorithms [17] extend one step ABR by predicting and solving the optimization problem for every N GoFs. This algorithm is able to predict future FoV and bandwidth changes, but does not consider the impact of current GoF decisions on future GoFs. To address the above issues, we present a rolling POT framework in the next section.

IV. Rolling POT framework for point cloud video streaming

In this section, we describe the rolling POT framework in detail. The general flow of the whole framework is shown in Fig.2. Our framework makes a decision before the client initiates a download request for each GoF, and first needs to predict the bandwidth and user's FoV for N GoFs interval, and then uses the optimization algorithm to find the optimal decision variables of next N GoFs. Then the next GoF is transmitted based on this decision. After the transmission completes, the prediction window is shifted forward by one GoF and the above process continues to be repeated.

Figure 2. Prediction-optimization-transmission framework.

Note that our rolling POT framework differs from traditional ABR algorithms in that our framework considers the impact of the current step's decisions on video playback beyond the truncation step, and we will show that there is an upper bound on the impact of the current decisions on the truncation step.

For the state of buffer,

$$bf(k+1) = f(bf(k), u(k)), \quad bf(1) = bf_1 \tag{10}$$

where $bf(k) \in \mathbb{R}$ and $u(k) \in \mathbb{R}^2$ denote the buffer capacity and the tile selection variable of the system at step k, respectively. $u(k)$ is the matrix of $x(k-1)$ and $x(k)$ vertically spliced, i.e., $u(k) = \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix}$. For each fixed k, $u(k)$ represents a two-dimensional matrix, and $u(k,c,l)$ denotes the elements of the tile c and quality level l of the matrix $u(k)$ at step k. $f()$ represents the form on the right side of the equals sign of Eq. 5. The tile selection variables and buffer capacity constraints in Eq. 10 are as follows:

$$bf(k) \in [0, bf^M], k \geq 1 \tag{11}$$

$$\sum_{l=1}^{R} u(k,c,l) = 1, c \in [1,C], k \geq 1 \tag{12}$$

For the Eq.9, we have,

$$QoE = \frac{1}{K}\sum_{k=1}^{K} QoE_k = \frac{1}{N}\sum_{k=1}^{N} QoE_k + \frac{1}{K}\sum_{k=1}^{K} QoE_k - \frac{1}{N}\sum_{k=1}^{N} QoE_k \tag{13}$$

We define $E(bf(k)) = (C * L * w_1 * q_{max} * bf(k))/bf^M$. Obviously, $E(bf(k))$ is a positive definite function satisfying $E(0) = 0$ and $E(bf(k)) > 0$.

Then we define the set $\Omega := \{bf(k) \in R \mid E(bf(k)) \leq C * L * w_1 * q_{max}\}$, and we have Theorem 1.

Theorem 1. *$\forall k, bf(k) \in \Omega$, i.e. the playback buffer is stable.*

Proof. Obviously, $\Omega \in [0, B]$. We have:

$$E(bf(s)) - E(bf(j)) = \frac{C \times L \times w_1 \times q_{\max}}{bf^M} \times (bf(s) - bf(j))$$
$$= C \times L \times w_1 \times q_{\max} \times \frac{1}{bf^M} \times (bf(s) - bf(j))$$

Considering:

$$\begin{aligned} bf(j) - bf(s) &= \left[bf(j-1) - T^W_{j-1} - T^C_{j-1}\right]_+ \\ &- \left[bf(s-1) - T^W_{s-1} - T^C_{s-1}\right]_+ \\ &\geq 0 - \left(bf(s-1) - T^{\pi}_{s-1} - T^C_{s-1}\right) \\ &\geq T^W_{s-1} + T^C_{s-1} - bf^M \geq -bf^M \\ &\Rightarrow bf^M \geq bf(s) - bf(j) \end{aligned}$$

So we can get:

$$\begin{aligned} &E(bf(s)) - E(bf(j)) = \\ &- C \times L \times w_1 \times q_{\max} \times \frac{1}{bf^M} \times (bf(j) - bf(s)) \\ &\leq -C \times L \times w_1 \times q_{\max} \times \frac{1}{bf^M} \times bf^M \\ &= -C \times L \times w_1 \times q_{\max} \end{aligned}$$

And because:

$$C \times L \times w_1 \times q_{\max} \geq \frac{1}{s-j}\sum_{k=j}^{s} QoE_k$$

Therefore:

$$E(bf(s)) - E(bf(j)) \leq -\frac{1}{s-j}\sum_{k=j}^{s} QoE_k, \quad N+1 \geq s \geq j \geq 1 \tag{14}$$

So, the positive definite function $E(bf(k))$ satisfies the Hamilton-Jacobi-Bellman (HJB) inequality 14.

Therefore, for $\forall k, bf(k) \in \Omega$. And according to [47], the playback buffer is stable. □

Then we can obtain:

$$\frac{1}{K}\sum_{k=1}^{K} QoE_k - \frac{1}{N}\sum_{k=1}^{N} QoE_k \leq E(bf(N+1)) \tag{15}$$

Substituting Eq. 15 into Eq. 13 :

$$\frac{1}{K}\sum_{k=1}^{K} QoE_k \leq \frac{1}{N}\sum_{k=1}^{N} QoE_k + E(bf(N+1)) \tag{16}$$

Substituting it into Eq. 13

$$QoE \leq \frac{1}{N}\sum_{k=1}^{N} QoE_k + E(bf(N+1)) \tag{17}$$

Therefore, we define the objective function as

$$QoE_k^N := \frac{1}{N}\sum_{k=1}^{N} QoE_k + E(bf(N+1)) \tag{18}$$

Obviously, this is an upper bound of problem P1, and it can be transformed into problem $P2$ as follows:

$$P2 : \max_{\bar{u}(\cdot)} QoE_k^N \tag{19}$$

$$QoE_k^N = \frac{1}{N}\sum_{i=1}^{N} QoE_{k+i|k} + E(\overline{bf}(k+N+1 \mid k) \tag{20}$$

s.t.

$$\overline{bf}(k+i+1|k) = f(\overline{bf}(k+i|k), \overline{u}(k+i|k)) \tag{21a}$$

$$\overline{u}(k+i|k)) \in \{0,1\}, i \in [0,N] \tag{21b}$$

$$\overline{bf}(k+i|k)) \in [0,B], i \in [0,N] \tag{21c}$$

$$\overline{bf}(k+N|k)) \in \Omega \tag{21d}$$

where N is the prediction step length of the optimization problem from GoF k, $\overline{bf}(k+i|k)$ is the prediction dynamic state where the buffer starts from $bf(k)$ under the action of the prediction solver variable $\overline{u}(\cdot)$. Assume that the optimization problem P2 has a solution at GoF k, noted as:

$$U^*(k) = \begin{bmatrix} \bar{u}^*(k \mid k) \\ \bar{u}^*(k \mid k+1)^N \\ \vdots \\ \bar{u}^*(k+N-1 \mid k) \end{bmatrix}^N \tag{22}$$

Taking the first element of $U^*(k)$ as an optimal solution of the original problem P1 at step k.

$$u^*(k) := \bar{u}^*(k \mid k) \tag{23}$$

Dynamics of buffer is

$$bf(k+1) = f\left(bf(k), u^*(k)\right) \tag{24}$$

Accordingly, the specific flow of our rolling POT framework is shown in Algo. 1.

Algorithm 1: Rolling POT Framework

Input: $bf(1)$: initial buffer status
Output: $u(k)$: the optimized solution that meets the binary constraint
Initialize: $u(1), f^V(1), bf(1), N$
Denote the initial capacity of $bf(1)$ as buffer and use it as the input to the system.
for $k = 1$ *to* K **do**
  Predict $BW(k) \sim BW(k+N), f^V(k) \sim f^V(k+N)$
  Solve the solution of the N-step optimization problem $\bar{u}^*(k \mid k) \sim \bar{u}^*(k+N \mid k)$
  Take the solution of the first step as the output, i.e. $u(k) = \bar{u}^*(k \mid k))$
  Substitute $u(k)$ into Eq. 10 to obtain the depth of the buffer $bf(k+1)$ at the next step, i.e., $bf(k+1) = f(bf(k), u(k))$;
  $bf(k) = bf(k+1)$
end

V. SC-DDQN: DRL based real-time solver

In the rolling POT framework, to perform real-time streaming, we have to solve the QoE optimization problem with a range of N GoFs in one GoF time. The dimension of the solution is $N*C*L$, which makes the solution space of the problem remain large. Traditional optimization algorithms such as Lagrange multiplier method and dynamic programming [48] are able to find the optimal solution, but the computation time cannot meet the requirements of real-time point cloud video systems. Note that although the problem $P2$ is relatively large in search space of decision variables, the problem structure is fixed in each solution, while only the variable parameters changing, such as the prediction results of FoV, BW and the point cloud tiles related variables $q_{k,c,l}, p_{k,c,l}$. Thus we design a DRL network, SC-DDQN, to explore the optimal solutions under fixed problem structure and different combinations of variable parameters. After the training completes, the algorithm can give the solution under the variable parameters of the input problem in real-time.

A. Reconstruction of Problem P2

First, from problem $P2$ of Eq. 19, we have:

$$\max_{x_{k,c,l}} \frac{1}{N}\sum_{i=1}^{N} QoE_{k+i|k} \tag{25}$$

In solving the optimization problem 25, we need to determine the quality levels of $N*C$ tiles at GoF k. SC-DDQN uses a serial structure to determine the quality level of one tile at a time, and obtains the complete solution of the optimization problem for GoF k after cycling $N*C$ times. The details about $state, action, reward$ are as follows.

State: Denote $\vec{x}$ to be the one-dimensional expansion of the multidimensional vector x. Then we define that at each cycling step m, $m \in [1, N*C]$, *state* S_m is composed in the following way.

$$S_m = [Y, \vec{x_m}] \tag{26}$$

where $Y = [\vec{q}, \vec{f^V}, \vec{s}, \vec{f^{V'}}, \vec{p}, \vec{x'}, \vec{bf}, \vec{D}, \vec{BW}]$, $\vec{f^{V'}}, \vec{x'}$ represent the FoV information and decision result of the previous GoF. x_m denotes the decision variables of the tiles (containing the determined tiles and the undetermined tiles) when the serial solution process proceeds to step m.

Action: First, we define the set of quality levels of all tiles as $\mathbb{T} := [t_1 \cdots t_m \cdots t_{N*C}]$, where t_m denotes the

quality level of the tile m. Then we define *Action* $a_m := t_m$. For constraint 9b, we set the constraints on the *action* a_m at step m:

$$a_m \subseteq \mathbb{T} \backslash [t_1 \cdots t_{m-1}] \tag{27}$$

The above equation indicates that the quality level cannot be selected from the selected tiles.

State transition: After executing *action* under the previous *state* S_m, new decision variables $x_{m+1}^{\rightarrow}$ is obtained according to the determined tiles, thus forming the new state $S_{m+1} = [Y, x_{m+1}^{\rightarrow}]$.

Reward: We define *reward* using the increment of QoE after executing *action*:

$$reward_m = (95 - (QoE(S_m^{\rightarrow}) - QoE(S_{m-1}^{\rightarrow})))^2 \tag{28}$$

where $QoE(S_{m-1}^{\rightarrow}), QoE(S_m^{\rightarrow})$ are the QoE values calculated by the decision variables $x_{m-1}^{\rightarrow}$ and $x_m^{\rightarrow}$ before and after the execution of the action, respectively. In addition, if the action a_m does not satisfy the constraint 27, let $reward_m = -10000$.

B. Network Structure

We design the network structure of SC-DDQN as shown in Fig.3. FC stands for fully connected network and all FC layers have 761 neurons. The input and output layers, and the hidden layer all use linear rectification units. The state transmission process of this solution is also illustrated. The action a_1 is obtained at step 1, and then the next state is computed, using the new state for step 2, and so on until the whole decision variable $x_{k,c,l}$ is obtained after $m = N \times C$ steps. We use the randomly generated 100,000 sets of random parameters Y to form the training set. In training, the network reads the training set and then starts training for up to 200,000 episodes, and the training set is trained in 200,000 episodes in a loop. The capacity of experience pool e is 200,000, batch size = 256, learning rate = 1.5×10^{-3} , discount factor $\gamma = 1$. The specific training process is indicated by Algo. 2, where Le represents the number of episodes trained, $Ls = N \times C$ represents the number of steps per action, and Lt represents the number of training sets. The output process of SC-DDQN is also shown in Algo. 3. We also make the experimental code available at GitHub[5].

VI. Simulations and Experiments

To verify the performance of the rolling POT framework and SC-DDQN, we perform extensive simulations, and conduct real experiments using Microsoft HoloLens 2.

Experimental video: In our simulations and experiments, we choose Basketball (denoted as Video1) as the high-resolution test video and Soldier (denoted as Video2) as the low-resolution test video [49], as shown in Fig. 4. Each video contains 300 frames, and 10 frames

[5]https://github.com/FreezingWHY/SC-DDQN/tree/master

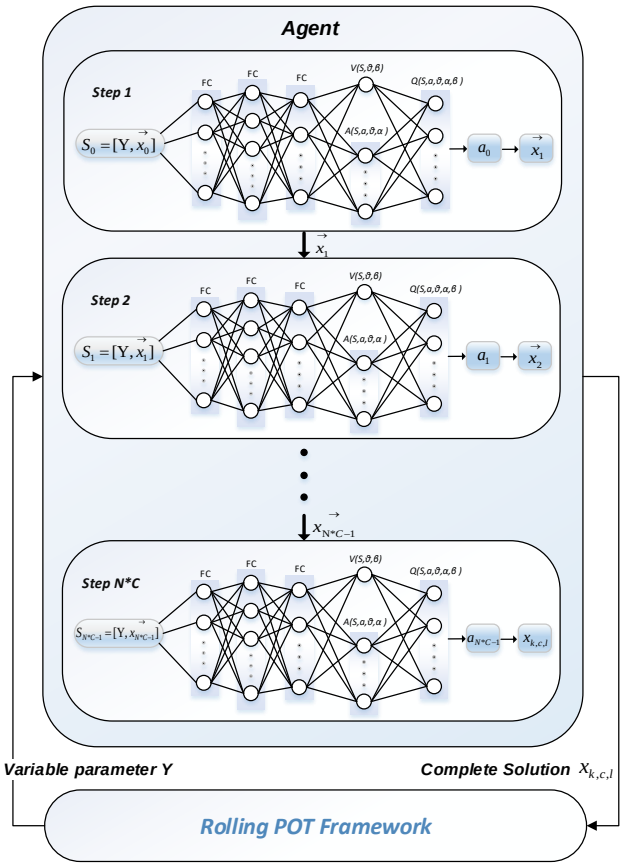


Figure 3. DRL workflow and SC-DDQN structure.

make up of one GoF. Each GoF is divided into C = 2×2×2 tiles, and encoded into L = 4 different quality representations using the Video Point Cloud Coding (V-PCC) software [29]. During the encoding, the parameters of geometry (geometryQP) and texture (textureQP) of these 4 representations are set as: l_1 : (32, 42), l_2 : (28, 37), l_3 : (24, 32), and l_4 : (20, 27), respectively.

(a) Basketball (b) Soldier

Figure 4. Illustration of experimental video.

Bandwidth trace: To verify the performance of our framework in real network scenarios, we choose four 5G network traces with different fluctuations, both of which are collected from the major mobile operators in

Algorithm 2: Training Process of SC-DDQN

Initialize: Total number of episodes of training *Le*. Number of steps needed to solve each episode *Ls*. Total number of training sets *Lt*. Network weight θ^{pred}, θ^{target}. Probability ϵ. Target net update interval *o*.
for *k = 1 to Le* **do**
  load the first (*Le* % *Lt*) set of training sets as parameters *C* for this episode;
  if *k*%*o* == 0 **then**
    Let $\theta^{target} \leftarrow \theta^{pred}$;
  end
  Update exploration probability ϵ;
  for *m = 1 to Ls* **do**
    Select *action* a_m: Select a random action $a_m \subseteq \mathcal{A} \backslash \mathcal{A}'$ with probability ϵ;
    Select the action a_m with probability $1-\epsilon$ using $a_m = \max_a Q(S_m, a; \theta^{pred})$;
    Execute action a_m, observe $reward_m$, new state S_{m+1};
    Deposit the quaternion $(S_m, a_m, reward_m, S_{m+1})$ to the experience pool *e*;
    Randomly draw mini batch $(S_m^{'}, a_m^{'}, reward_m^{'}, S_{m+1}^{'})$ from experience pool *e* ;
    if $S_{m+1}^{'}$ *is not the end state* **then**
      $y = r_m^{'} + \gamma \max_{a_{m+1}} Q(S_{m+1}^{'}, a_{m+1}^{'}; \theta_{target})$;
    end
    else
      $y = r_m^{'}$;
    end
    Update the network parameters θ^{pred} using gradient descent: $\nabla_\theta L(\theta) = \left(y - Q\left(S, a; \theta^{\text{pred}}\right)\right) \nabla_\theta Q\left(S, a; \theta^{\text{pred}}\right)$,where $L(\theta) = \left(y - Q\left(S, a; \theta^{pred}\right)\right)$;
  end
end

Ireland [50]. The range of the four bandwidth traces are as follows: BW1: 150 Mbps to 300 Mbps, BW2: 0 Mbps to 600 Mbps, BW3: 450 Mbps to 600 Mbps, and BW4: 0 Mbps to 1200 Mbps. These values are presented in the Table I along with their corresponding mean and standard deviation.

FoV trajectory: We build a FoV dataset that records the position of the user and the viewing direction for each frame. The FoV dataset contains the perspective information of 40 users (20 males and 20 females) watching Video1 and Video2. Each video contains 150 frames and is played in a loop in Unity 2020.3.1. FoV data is collected in real-time and consists of six sets of parameters, including the user's current spatial position (x, y, z) and head pitch Euler angles (pitch, yaw, roll).

Algorithm 3: Output Process of SC-DDQN

Input: The variable parameter *Y* of problem *P*2;
Output: Decision Variables $x_{k,c,l}$;
Initialize: The initial one-dimensional expansion of decision variable $\vec{x_0}$ is an all-0 vector;
Forming $\vec{x_0}$ and *Y* into an initial state S using Eq. 26; **for** *m = 0 to* $N \times C$ **do**
  The state *S* is sent to the trained neural network as input;
  Observe the network output and set the 0-1 variable of the corresponding position in $\vec{x_m}$ from 0 to 1,and get $\vec{x}_{m+1}$ and *Y*;
  Build new state $S^{'}$ by $\vec{x}_{m+1}$ and *C*, and Let $S = S^{'}$;
end
Transform $\vec{x_{N \times C}}$ into a three-dimensional matrix $x_{k,c,l}$, and output the result;

Table I
MEAN AND STANDARD DEVIATION OF DIFFERENT NETWORK BANDWIDTHS

	Mean	Std
BW1(Mbps)	199.5589	9.4171
BW2(Mbps)	213.5832	69.1551
BW3(Mbps)	523.393	50.8957
BW4(Mbps)	655.9592	285.9844

We also make the FoV data available on GitHub[6].

Bandwidth and FoV prediction: We use two different approaches to generate predictions of bandwidth and FoV with different accuracies, as follows.

1) **Different prediction error level**: Our rolling POT framework requires prediction of bandwidth and FoV for *N* GoF before each rolling optimization. Short-term prediction is fairly simple with high accuracy. However, the accuracy of long-term prediction is relatively poor, and the presence of cumulative prediction errors leads to greater uncertainty, which will severely degrade QoE [51]. Based on [52], we build a similar long short-term memory (LSTM) network for the prediction. To verify the impact of different accuracies caused by predictors, we modify the number of neurons in each layer in the LSTM network to achieve five bandwidth and FoV prediction error levels. Table II and Table III show the root mean square error (RMSE) of the bandwidth and FoV prediction results, respectively.

Table II
RMSE OF BANDWIDTH PREDICTION ERROR UNDER DIFFERENT ACCURACY LEVEL (MBPS)

	Level 1	Level 2	Level 3	Level 4	Level 5
BW	10.22	15.81	23.65	31.22	39.08

2) **Different prediction window**: Once the prediction algorithm is selected, the prediction errors are re-

[6]https://github.com/Yong-Chen94/6DoF_Video_FoV_Dataset

Table III
RMSE of FoV Prediction Error under different accuracy Level ((m) for X, Y, Z / (degree) for Pitch, Yaw, Roll)

	Level 1	Level 2	Level 3	Level 4	Level 5
X	*0.15*	*0.17*	*0.22*	*0.24*	*0.28*
Y	*0.15*	*0.18*	*0.21*	*0.24*	*0.27*
Z	*0.15*	*0.17*	*0.19*	*0.22*	*0.25*
Pitch	*6.45*	*9.53*	*11.84*	*14.88*	*17.77*
Yaw	*7.39*	*9.76*	*12.88*	*17.20*	*18.32*
Roll	*8.50*	*9.85*	*14.05*	*16.92*	*18.62*

lated to the prediction window N. With the increasing of N, the prediction error of LSTM network will become larger. Thus we select the LSTM network with the highest accuracy, and test prediction error under different prediction window. The RMSE performances with different windows are shown in Table IV and Table V.

Table IV
RMSE of Bandwidth Prediction Error with different prediction window (Mbps)

N	**4**	**5**	**6**	**7**	**8**	**9**	**10**
BW	6.80	10.22	12.69	20.94	26.98	37.59	47.35

Table V
RMSE of FoV Prediction Error with different prediction window ((m)for X, Y, Z / (degree) for Pitch, Yaw, Roll)

N	**4**	**5**	**6**	**7**	**8**	**9**	**10**
X	0.12	0.15	0.16	0.20	0.24	0.27	0.30
Y	0.11	0.15	0.15	0.20	0.25	0.26	0.29
Z	0.12	0.15	0.15	0.19	0.23	0.27	0.31
Pitch	5.97	6.45	7.02	10.79	14.56	16.11	17.65
Yaw	6.25	7.39	8.15	11.76	15.36	17.03	18.69
Roll	6.98	8.51	7.86	12.38	16.89	17.42	17.95

Comparison Methods: To verify the performance of the proposed streaming system, denoted as **POT+DRL**, we select four methods as our competitors.

1) **ABR+OPT**: The traditional single-step ABR method used in [15], in which the branch-bound (BB) algorithm [53], which can yield optimal solution for combinatorial optimization problems, is used for solving the binary programming problem for each GoF.
2) **nABR+OPT**: nABR method is an extension of the traditional single-step ABR algorithm, where the streaming process is segmented by every N GoFs for optimization and transmission [17]. The BB algorithm is used as optimization solver for each segmentation.
3) **POT+OPT**: The rolling POT framework is used, and the BB algorithm is used to solve the optimization problem in each roll.
4) **nABR+DRL**: The difference from nABR+OPT method is that each in each segmentation, SC-DDQN is used to solve the optimization problem [54].

Similar to [55], in all our subsequent comparisons, we set the weights in the QoE model to $w_1 = 1$, $w_2 = 1000$ and $w_3 = 1$. For each set of experiments in the simulation, we run up to 1,000 tests to eliminate the influence of random factors.

We also set up several experiments to compare the performance of our SC-DDQN solver with the Natural DQN [56], particle swarm optimization (PSO) algorithm [57], and BB algorithm.

A. Simulation Performance

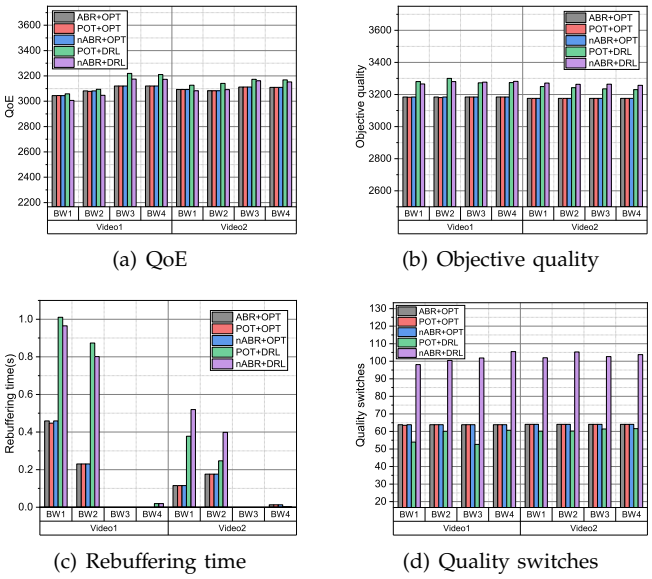
a. Performance evaluation of the rolling POT framework

QoE related metrics: We show the performance of these five methods in terms of the QoE related metrics (QoE, objective quality, rebuffering time and quality switch) in Fig. 5.

We can observe that the QoE related metrics of Video2 is higher than Video1, this is because Video2 is with less points and thus has a smaller size. Under the same network conditions, the streaming of Video2 is easier.

As illustrated in Fig. 5(b) and Fig. 5(c), POT+DRL and nABR+DRL tend to choose higher quality tiles than the other three methods, while also may generate long rebuffering time. However, the proposed method, POT+DRL, has a higher overall QoE value than nABR+DRL, due to the fact that our method has fewer video quality switches. This demonstrates that our framework is more able to take into account future bandwidth and FoV forecast impacts in every GoF decision, and can effectively reduce quality switches. Furthermore, when playing Video2, the difference in QoE metrics between our method POT+DRL and the other methods is higher than Video1, due to the smaller amount of data in Video2, which brings less rebuffer penalty, and the fact that POT+DRL has the least number of video quality switches. Combining the three factors of QoE, our POT+DRL method achieves the best performance.

Impact of prediction error level: We validate the impact of prediction error level caused by different prediction algorithms, and the results are shown in Fig. 6. It can be seen that when the prediction is relatively accurate, the performance of ABR+OPT, POT+OPT, nABR+OPT is better than that of POT+DRL and nABR+DRL. This is because when the prediction results are accurate, the first three methods can always obtain optimal decision results, and thus achieve better QoE. With the heuristic algorithm, the global solution cannot be guaranteed, especially when search space is huge. Thus a slightly lower QoE is obtained. However, when the prediction error is relatively large, the solution quality of the optimal solution algorithm is greatly affected, but our SC-DDQN solver performs more consistently and is more resistant to large prediction errors. This occurs due to the SC-DDQN's incorporation of imperfect predictions of network bandwidth and FoV in the training dataset. The performance of POT+DRL and nABR+DRL fluctuates only in a small range with increasing prediction error levels, and the overall performance remains unchanged.

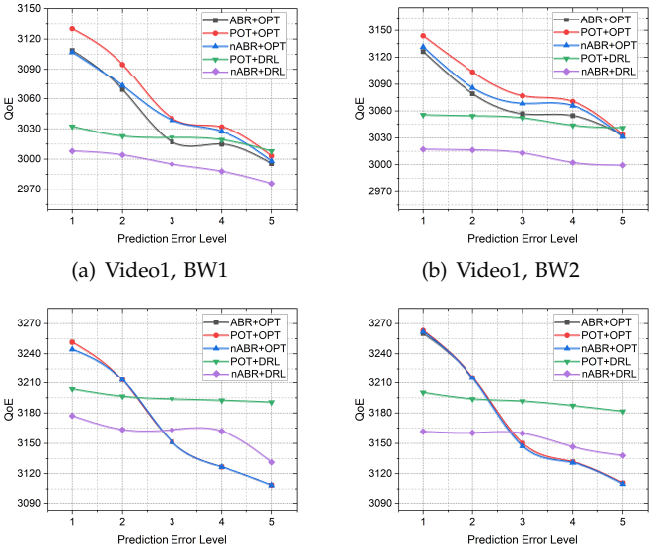


(a) QoE (b) Objective quality

(c) Rebuffering time (d) Quality switches

Figure 5. QoE performance comparison of different streaming methods.

In addition, it can be seen from the Fig. 6 that with rolling POT framework. POT+OPT works better than ABR and nABR, indicating that our rolling framework considers the impact of current GoF decisions on future GoFs and obtains better performance. In fact in the following section we will show that the solving time of optimal algorithm is unacceptable, and our SC-DDQN solver is more applicable in real-time streaming scenarios.

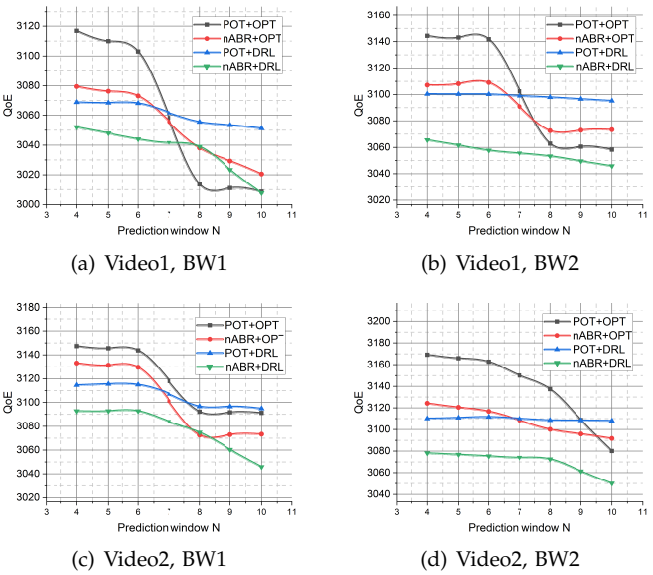


(a) Video1, BW1 (b) Video1, BW2

(c) Video2, BW1 (d) Video2, BW2

Figure 6. Performance comparison under different prediction error levels.

Impact of prediction window: With the prediction error in Table IV and V, we can evaluate the performance impact of different prediction window. Since traditional ABR algorithm does not take into account the prediction window, we compare the performance of the four algorithms, POT+OPT, nABR++OPT, POT+DRL, and nABR+DRL, and the results are shown in Fig. 7.



(a) Video1, BW1 (b) Video1, BW2

(c) Video2, BW1 (d) Video2, BW2

Figure 7. Performance comparison under different predict window.

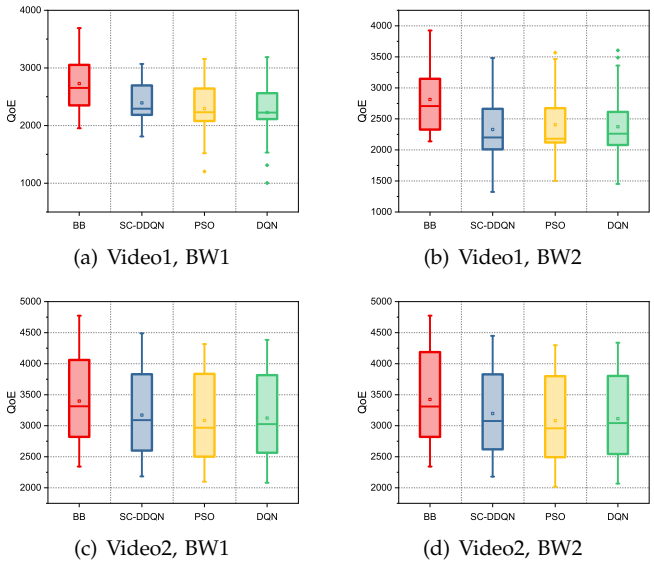
It can be seen that smaller prediction window produces more accurate bandwidth and FoV prediction results, with the POT+OPT algorithm exhibiting the best performance, closely followed by the nABR+OPT, both outperforming POT+DRL and nABR+DRL. As prediction window N increases, the prediction accuracy tends to decrease. Consequently, the optimal algorithm cannot generate high quality solution based on inaccurate prediction data, leading to gradual declines in performance of the POT+OPT and nABR+OPT algorithms. Although the performance of the POT+DRL shows a slight decrease, it still outperforms nABR+DRL in terms of overall performance.

b. Performance comparison of different optimization problem solvers

We randomly select 50 GoFs and analyze the performance of our SC-DDQN solver from two perspectives, the QoE value calculated in one GoF rolling step and the solution time of each rolling optimization. And the results are shown in Fig. 8 and Fig. 9.

Fig. 8 shows the solution quality of four algorithms. The optimal branch-bound algorithm performs better than all other algorithms. It can be seen that our SC-DDQN solver outperforms the nature DQN algorithm in all cases. The mean solution quality of SC-DDQN is better than that of PSO algorithm for Video2, and is almost equal for Video1. This is due to the fact that our SC-DDQN prefers to select high quality tiles to obtain better QoE.

Fig. 9 shows the solution time of the four algorithms. Although the optimal branch-bound achieves best QoE, its solution time is much higher than the others, and it



(a) Video1, BW1 (b) Video1, BW2

(c) Video2, BW1 (d) Video2, BW2

Figure 8. Optimization quality comparison with different solvers.

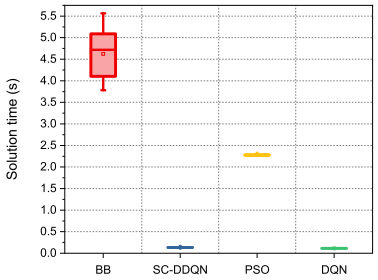


Figure 9. Solution time comparison with different solvers.

fluctuates too much to be applied in a real-time streaming system. The other three algorithms have almost no fluctuation in time. The SC-DDQN takes almost the same time as the DQN algorithm, while achieving higher QoE value. It means our POT+DRL method can guarantee a good solution quality when consuming a stable and much smaller solution time, and can be applied in a practical point cloud video streaming system.

B. *Real Experiments*

We also build a point cloud video streaming system to verify the feasibility of the proposed scheme, as shown in Fig. 10. We use a laptop as the DASH server and deploy a Nginx based HTTP web server, where the encoded point cloud videos are stored. A PC is used as the client with a Microsoft HoloLens 2 as the playback device. The connection between client PC and HoloLens 2 is made by way of holographic projection technology in Unity engine version 2020.3.17. We use the Mixed Reality Toolkit 2 (MTRK2) provided by Microsoft. The transmission between server and client are performed via HTTP protocol and a Xiaomi Network Router 4A WiFi AP, which supports up to 1200Mbps bandwidth.

VPCC-TMC2-v7 is used for the encoding and decoding. Python3.6 and TensorFlow-GPU2.1 are used to implement our rolling optimization framework, SC-DDQN real-time solver and LSTM based bandwidth and FoV prediction. In addition, we implement Natural DQN, particle swarm and optimal branch-bound algorithm as our competitors.

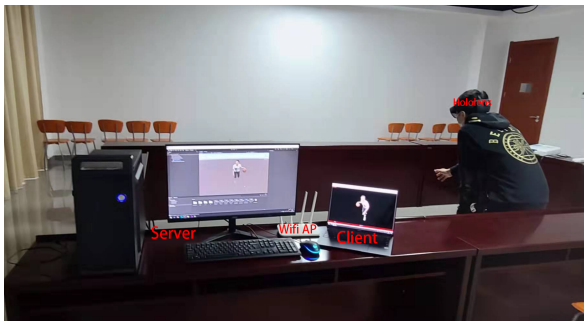
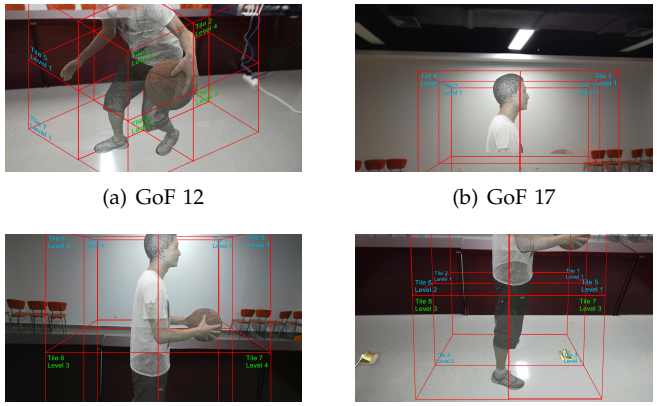


Figure 10. Experimental environment.



(a) GoF 12 (b) GoF 17

(c) GoF 18 (d) GoF 19

Figure 11. Point cloud video frames played in HoloLens 2. The tiling information and quality level of each tile are also marked (The larger the number of quality level, the better the video quality, indicating smaller geometryQP and textureQP setup in encoding and decoding software).

To show the effectiveness of our POT+DRL method in the real streaming system, we randomly select several frames of a single GoF and three consecutive GoFs played in HoloLens 2 for a visual check, as illustrated in Fig. 11. As can be seen from Fig. 11(a), the qualities of tiles 1, 2, 3 and 4, which are closer to the viewer's point of view, are significantly higher than that of tiles 5, 6, 7 and 8. This is mainly because in our framework the distance from a tile to the point of view will have impact to quality switches and QoE, as shown in Eq. 8. Fig. 11(b), Fig. 11(c) and Fig. 11(d) show three consecutive frames in one GoF. In Fig. 11(b), our framework chooses to transmit tiles 1, 2, 5 and 6 of the upper body, which is only partially visible within the FoV of HoloLens 2, and limited by network situation, only low quality level can be transmitted. In Fig. 11(d), while the user's

point of view moves to the lower body, our framework considers that frequent quality switches will have a negative impact on QoE, and choose not to select the same quality level of tile 1, 2, 5 and 6 while requesting better quality of tile 7 and 8, which achieves a better overall QoE.

During the experiments, we measure the mean values of BW1 and BW2 throughout the entire video playback, which are 231.8Mbps and 337.2Mbps, respectively. Moreover, we also measure the average solving, downloading, and decoding time of the applicable algorithms in a real environment, which are listed in Table VI,VII and VIII.

Table VI
AVERAGE SOLVING TIME FOR DIFFERENT ALGORITHMS

Average solving time (s)					
	ABR+BB	**POT+BB**	**nABR+BB**	**POT+DRL**	**nABR+DRL**
V1B1	0.8161	4.4731	0.9050	0.1322	0.1300
V1B2	0.7968	4.4588	0.9120	0.1334	0.1337
V2B1	0.7944	4.4592	0.9099	0.1340	0.1347
V2B2	0.7983	4.4442	0.9127	0.1346	0.1333

Table VII
AVERAGE DOWNLOADING TIME FOR DIFFERENT ALGORITHMS

Average downloading time (s)					
	ABR+BB	**POT+BB**	**nABR+BB**	**POT+DRL**	**nABR+DRL**
V1B1	0.2183	0.2180	0.2210	0.2340	0.2343
V1B2	0.2129	0.2073	0.2073	0.2319	0.2285
V2B1	0.2068	0.2068	0.2068	0.2169	0.2212
V2B2	0.2037	0.2027	0.2020	0.2156	0.2194

Table VIII
AVERAGE DECODING TIME FOR DIFFERENT ALGORITHMS

Average decoding time (s)					
	ABR+BB	**POT+BB**	**nABR+BB**	**POT+DRL**	**nABR+DRL**
V1B1	0.1164	0.1163	0.1177	0.1239	0.1241
V1B2	0.1172	0.1147	0.1145	0.1256	0.1237
V2B1	0.1112	0.1112	0.1112	0.1159	0.1180
V2B2	0.1112	0.1108	0.1106	0.1161	0.1176

V1B1 denotes the experimental results for Video1 under BW1, while V1B2, V2B1, and V2B2 represent the same conditions. Regarding the solving time, both the DRL model using in the POT framework and the nABR algorithm have relatively short solving time. However, the average solving time of POT+BB method is the longest among the methods using conventional mathematical optimization algorithms. Consequently, considering solving time, the second term of the QoE metric will be significantly penalized. Traditional ABR algorithms have smaller optimization problem sizes for each GoF, and the nABR algorithm only solves an optimization problem for each th GoFs, leading to its average solving time shorter than POT+BB but still significantly longer than DRL algorithms. Traditional mathematical algorithms generate optimal solutions, which result in shorter average download and decoding times for their decisions.

We also calculate the QoE of different methods, taking the actual video rendering and the optimization solving time into account, and the results are shown in Fig. 12. The negative value of QoE is caused by the frequent rebuffering interruptions after considering the optimization solving time. Our POT+DRL method has a great advantage in solving time, and the QoE value in the real system is higher than that of other methods.

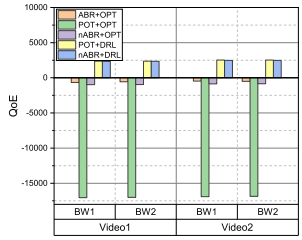


Figure 12. QoE performance comparison of different methods in the experimental system.

VII. CONCLUSION

In this paper, we propose a POT framework for real-time point cloud video streaming, which is able to maintain stable performance under complex and variable network conditions. We propose a DRL based real-time solver SC-DDQN, which can solve the optimization problem of our framework in a real-time manner. Extensive simulations are conducted and the results demonstrate that the performance of our framework is better than those of conventional algorithms. In addition, we have built a experimental system using HoloLens2 and the experimental results confirm the effectiveness of the proposed framework.

ACKNOWLEDGMENT

This work is supported in part by grants from the National Natural Science Foundation of China (52077049), the Anhui Provincial Natural Science Foundation (2008085UD04).

REFERENCES

[1] J. van der Hooft, M. T. Vega, T. Wauters, C. Timmerer, A. C. Begen, F. De Turck, and R. Schatz, "From capturing to rendering: Volumetric media delivery with six degrees of freedom," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 49–55, 2020.

[2] S. Dijkstra-Soudarissanane, K. E. Assal, S. Gunkel, F. t. Haar, R. Hindriks, J. W. Kleinrouweler, and O. Niamut, "Multi-sensor capture and network processing for virtual reality conferencing," in *Proceedings of the 10th ACM Multimedia Systems Conference*, 2019, pp. 316–319.

[3] X. Xu and E. Steinbach, "Towards real-time modeling and haptic rendering of deformable objects for point cloud-based model-mediated teleoperation," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014, pp. 1–6.

[4] S. Subramanyam, J. Li, I. Viola, and P. Cesar, "Comparing the quality of highly realistic digital humans in 3dof and 6dof: A volumetric video case study," in *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2020, pp. 127–136.

[5] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2017.

[6] Z. Liu, Q. Li, X. Chen, C. Wu, S. Ishihara, J. Li, and Y. Ji, "Point cloud video streaming: Challenges and solutions," *IEEE Network*, vol. 35, no. 5, pp. 202–209, 2021.

[7] F. P. Brooks, "What's real about virtual reality?" *IEEE Computer graphics and applications*, vol. 19, no. 6, pp. 16–27, 1999.

[8] Y. Xu, W. Hu, S. Wang, X. Zhang, S. Wang, S. Ma, Z. Guo, and W. Gao, "Predictive generalized graph fourier transform for attribute compression of dynamic point clouds," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 5, pp. 1968–1982, 2021.

[9] J. Li, C. Zhang, Z. Liu, R. Hong, and H. Hu, "Optimal volumetric video streaming with hybrid saliency based tiling," *IEEE Transactions on Multimedia*, pp. 1–1, 2022.

[10] T. Thanh Le, J.-B. Jeong, S. Lee, J. Kim, and E.-S. Ryu, "An efficient viewport-dependent 360 vr system based on adaptive tiled streaming," 2021.

[11] J. Li, C. Zhang, Z. Liu, W. Sun, and Q. Li, "Joint communication and computational resource allocation for qoe-driven point cloud video streaming," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[12] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[13] A. Elgabli and V. Aggarwal, "Fastscan: Robust low-complexity rate adaptation algorithm for video streaming over http," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 2240–2249, 2020.

[14] 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, 2021.

[15] J. Li, C. Zhang, Z. Liu, R. Hong, and H. Hu, "Optimal volumetric video streaming with hybrid saliency based tiling," *IEEE Transactions on Multimedia*, 2022.

[16] A. E. Essaili, D. Schroeder, E. Steinbach, D. Staehle, and M. Shehada, "Qoe-based traffic and resource management for adaptive http video delivery in lte," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 988–1001, 2015.

[17] L. Yu, T. Tillo, and J. Xiao, "Qoe-driven dynamic adaptive video streaming strategy with future information," *IEEE Transactions on Broadcasting*, vol. 63, no. 3, pp. 523–534, 2017.

[18] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 335–346, 2019.

[19] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University*, pp. 2006–07, 2007.

[20] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*. Springer, 2019, pp. 43–55.

[21] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[22] M. Dorigo and K. Socha, "An introduction to ant colony optimization," in *Handbook of Approximation Algorithms and Metaheuristics, Second Edition*. Chapman and Hall/CRC, 2018, pp. 395–408.

[23] G. Xiao, M. Wu, Q. Shi, Z. Zhou, and X. Chen, "Deepvr: Deep reinforcement learning for predictive panoramic video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1167–1177, 2019.

[24] S. Li, C. She, Y. Li, and B. Vucetic, "Constrained deep reinforcement learning for low-latency wireless vr video streaming," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 01–06.

[25] O. Schreer, I. Feldmann, S. Renault, M. Zepp, M. Worchel, P. Eisert, and P. Kauff, "Capture and 3d video processing of volumetric video," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 4310–4314.

[26] V. Leroy, J.-S. Franco, and E. Boyer, "Multi-view dynamic shape refinement using local temporal integration," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3113–3122.

[27] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Transactions on Graphics (ToG)*, vol. 34, no. 4, pp. 1–13, 2015.

[28] "Draco: 3d data compression," Google, 2018. [Online]. Available: http://github.com/google/draco

[29] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[30] T. M. Eugene d'Eon, Bob Harrison and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset,," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006,*, January 2017.

[31] J. Park, P. A. Chou, and J. N. Hwang, "Volumetric media streaming for augmented reality," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2019.

[32] L. Wang, C. Li, W. Dai, S. Li, J. Zou, and H. Xiong, "Qoe-driven adaptive streaming for point clouds," *IEEE Transactions on Multimedia*, pp. 1–1, 2022.

[33] B. Han, Y. Liu, and F. Qian, "Vivo: visibility-aware mobile volumetric video streaming," in *MobiCom '20: The 26th Annual International Conference on Mobile Computing and Networking*, 2020.

[34] J. Lim and M. Kim, "An optimal adaptation framework for streaming multiple video objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 5, pp. 699–703, 2008.

[35] C.-L. Fan, S.-C. Yen, C.-Y. Huang, and C.-H. Hsu, "On the optimal encoding ladder of tiled 360° videos for head-mounted virtual reality," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1632–1647, 2021.

[36] Y. He and L. Guan, "Solving streaming capacity problems in p2p vod systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1638–1642, 2010.

[37] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, p. 325–338, aug 2015.

[38] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 197–210.

[39] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[40] X. Jiang and Y. Ji, "Hd3: Distributed dueling dqn with discrete-continuous hybrid action spaces for live video streaming," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2632–2636.

[41] N. Kan, J. Zou, C. Li, W. Dai, and H. Xiong, "Rapt360: Reinforcement learning-based rate adaptation for 360-degree video streaming with adaptive prediction and tiling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1607–1623, 2022.

[42] Y. Gao, P. Zhou, Z. Liu, B. Han, and P. Hui, "Fras: Federated reinforcement learning empowered adaptive point cloud video streaming," *arXiv preprint arXiv:2207.07394*, 2022.

[43] W. Huang, Y. Zhou, X. Xie, D. Wu, M. Chen, and E. Ngai, "Buffer state is enough: Simplifying the design of qoe-aware http adaptive video streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 590–601, 2018.

[44] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 187–198.

[45] M. J. Khan, A. Bentaleb, and S. Harous, "Can accurate future bandwidth prediction improve volumetric video streaming experience?" in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 2021, pp. 1041–1047.

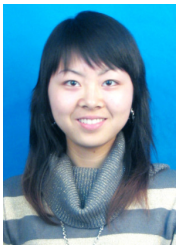
[46] L. Wang, C. Li, W. Dai, J. Zou, and H. Xiong, "Qoe-driven and tile-based adaptive streaming for point clouds," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 1930–1934.

[47] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[48] M. Michalos, S. Kessanidis, and S. Nalmpantis, "Dynamic adaptive streaming over http." *Journal of Engineering Science & Technology Review*, vol. 5, no. 2, 2012.

[49] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies-a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, vol. 7, no. 8, p. 11, 2017.

[50] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5g dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, p. 303–308.
[51] A. Sorjamaa and A. Lendasse, "Time series prediction using dirrec strategy." in *Esann*, vol. 6. Citeseer, 2006, pp. 143–148.
[52] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.
[53] V. I. Norkin, G. C. Pflug, and A. Ruszczyński, "A branch and bound method for stochastic global optimization," *Mathematical programming*, vol. 83, no. 1, pp. 425–450, 1998.
[54] X. Wei, M. Zhou, S. Kwong, H. Yuan, S. Wang, G. Zhu, and J. Cao, "Reinforcement learning-based qoe-oriented dynamic adaptive streaming framework," *Information Sciences*, vol. 569, pp. 786–803, 2021.
[55] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.
[56] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.
[57] F. Van Den Bergh *et al.*, "An analysis of particle swarm optimizers," Ph.D. dissertation, University of Pretoria, 2007.



Jie Li received the B.E., in electronic engineering in 2006 and Ph.D. degree in 2011 from University of Science and Technology, China. She is currently an Associate Professor at Hefei University of Technology, Anhui, China. From March 2012 to Sep. 2012, she was a research assistant in National Institute of Informatics, Japan. Her research interest includes cognitive radio network, wireless video streaming.

Huiyu Wang received the B.E., in electronic engineering from the Hefei University of Technologyin 2020. He is currently a postgraduate at Hefei University of Technology, Anhui, China since July 2020. His research interest includes reinforcement learning and volumetric video streaming.

Zhi Liu (S'11-M'14-SM'19) received his Ph.D. degree in informatics in National Institute of Informatics. He is currently an Associate Professor at The University of Electro-Communications. His research interest includes video network transmission and mobile edge computing. He is now an editorial board member of Springer wireless networks and IEEE Open Journal of the Computer Society. He is a senior member of IEEE.

Pengyuan Zhou received his PhD from the University of Helsinki. He was a Europe Union Marie-Curie ITN Early Stage Researcher from 2015 to 2018. He is currently a research associate professor at the School of Cyber Science and Technology, University of Science and Technology of China (USTC).

Xianfu Chen received his Ph.D. degree with honours from Zhejiang University, China, in March 2012. Since April 2012, he has been with the VTT Technical Research Centre of Finland, Oulu, Finland, where he is currently a Senior Scientist. His research interests cover various aspects of wireless communications and networking, with emphasis on human-level and artificial intelligence for resource awareness in next-generation communication networks. Dr. Chen was the recipient of the 2021 IEEE Communications Society Outstanding Paper Award and the 2021 IEEE Internet of Things Journal Best Paper Award. Dr. Chen serves as an Editor for IEEE Transactions on Cognitive Communications and Networking as well as Microwave and Wireless Communications, an Academic Editor for Wireless Communications and Mobile Computing, an Associate Editor for China Communications, and served as a Member of the First Editorial Board of Journal of Communications and Information Networks. Dr. Chen has served as the guest editor for several international journals, including IEEE Wireless Communications Magazine. Dr. Chen served as a TPC Chair/Track Co-Chair/TPC member for a number of IEEE ComSoc flagship conferences.

Qiyue Li (M'11-SM'21) received his Ph.D. degree in communication and information system from the University of Science and Technology of China in 2008. He joined the School of Electrical Engineering and Automation, Hefei University of Technology in 2011, where he is now a professor. His current research interests are mobile edge computing, video streaming, smart grid communication, cyber-physical power system. He is a member of the Standing Committee of IEEE PES Power System Communication and Network Security Technical Committee.

Richang Hong (Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2008. He is currently a Professor with the Hefei University of Technology. His research interests include multimedia content analysis and social media, in which he has coauthored more than 100 publications. He is a member of the ACM and an Executive Committee Member of the ACM SIGMMChina Chapter. He was the recipient of the Best Paper Award at the ACM Multimedia 2010, the Best Paper Award at the ACM ICMR 2015, and the Honorable Mention of the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award 2015. He was an Associate Editor for the IEEE TRANSACTIONS ON BIG DATA, ACM Transactions on Multimedia Computing, Communication and Applications and the Steering Committee Member of the Multimedia Modeling Conference (MMM).