SR360: Boosting 360-Degree Video Streaming with Super-Resolution

Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang and Di Wu
School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
Corresponding author: Miao Hu (humiao5@mail.sysu.edu.cn)

ABSTRACT

360-degree videos have gained increasing popularity due to its capability to provide users with immersive viewing experience. Given the limited network bandwidth, it is a common approach to only stream video tiles in the user's Field-of-View (FoV) with high quality. However, it is difficult to perform accurate FoV prediction due to diverse user behaviors and time-varying network conditions. In this paper, we re-design the 360-degree video streaming systems by leveraging the technique of super-resolution (SR). The basic idea of our proposed *SR360* framework is to utilize abundant computation resources on the user devices to trade off a reduction of network bandwidth. In the SR360 framework, a video tile with low resolution can be boosted to a video tile with high resolution using SR techniques at the client side. We adopt the theory of deep reinforcement learning (DRL) to make a set of decisions jointly, including user FoV prediction, bitrate allocation and SR enhancement. By conducting extensive trace-driven evaluations, we compare the performance of our proposed SR360 with other state-of-the-art methods and the results show that SR360 significantly outperforms other methods by at least 30% on average under different QoE metrics.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → Virtual reality.

KEYWORDS

360-degree video streaming, deep reinforcement learning, quality of experience, super-resolution

ACM Reference Format:
Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang and Di Wu. 2020. SR360: Boosting 360-Degree Video Streaming with Super-Resolution. In *30th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'20), June 10–11, 2020, Istanbul, Turkey*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3386290.3396929

1 INTRODUCTION

The fast upgrade of Virtual Reality (VR) devices and the increasing number of VR gamers have become the major driving forces to propel the growth of 360-degree videos. In the 360-degree video applications, users can explore omni-directional video contents and have immersive viewing experience, by turning their head-mounted displays (HMDs). Besides, 360-degree videos provide a higher resolution of video frames than that of conventional videos. According to the report by Cisco [3], it is predicted that the traffic of VR will increase at a 60% compound annual growth rate thru to 2021, and 43 million VR headsets will be shipped annually by 2022. A number of world-leading video service providers (e.g., YouTube, Facebook) already started to support 360-degree videos.

To achieve a fairly immersive experience, the video resolution of 360-degree video needs to be at least 8K, which corresponds to a network throughput of 400 Mbps and above. However, as only a small portion of each video frame will be displayed on the screen at a specific moment, it is a kind of wastage of bandwidth to stream the entire frame with high quality. A number of methods [7, 11, 17] adopted tile-based schemes, which split the frame into tiles and only stream video tiles in the user's *Field-of-View (FoV)*. To support viewport adaptive streaming, different tiles are allocated with different bitrates. Spatial representation description (SRD) [15] was designed as an enhancement to the MPEG-DASH standard. Some work [16, 23] proposed to allocate tiles in the FoV with a higher bitrate and other tiles with a lower bitrate.

Despite the aforementioned work, there are still quite a few challenges: (1) *Insufficient bandwidth.* The quality of 360-degree video is highly dependent on the network bandwidth between the server and the client. In case that the bandwidth resource is insufficient and highly dynamic, user QoE will degrade significantly [8]. (2) *Erroneous FoV prediction.* Due to diverse user behaviors and the impacts of video genres, it is difficult to make accurate FoV prediction. The prediction error will directly affect the user QoE. (3) *Coupling influential factors.* There are many factors that can influence tile bitrate selection, such as time-variant network conditions, head movement orientations, and so on. It is challenging to take all these dynamic factors into account. (4) *Multi-dimensional QoE optimization.* To provide a good enough QoE, we need to jointly optimize a set of multi-dimensional QoE metrics, including high video quality, low rebuffering time, smooth video playback, etc.

In this paper, we propose to re-design the 360-degree video streaming systems by leveraging the technique of super-resolution (SR). The basic idea is to trade the client side's computation resources for a reduction of bandwidth consumption. Today's GPU components on mobile devices (or desktop PCs) are powerful enough to speedup advanced neural network operations [22]. By performing SR operations, we can boost the quality of low-resolution video tiles to high-resolution video tiles at the client side. To reduce the SR processing time, we apply the super-resolution operations at the tile

NOSSDAV'20, June 10–11, 2020, Istanbul, Turkey

ACM ISBN 978-1-4503-7945-8/20/06…$15.00
https://doi.org/10.1145/3386290.3396929

level. We also propose a deep reinforcement learning (DRL)-based model to predict a user's FoV, allocate bitrates for each tile, and make decision on whether to enhance a video by SR or not. Overall, our contributions in this paper can be summarized as follows:

- We design a novel framework for 360-degree video streaming applications called *SR360*, in which super-resolution is exploited to boost 360-degree video streaming at the client side and thus significantly reduce the consumption of network bandwidth.
- We propose to use deep reinforcement learning to jointly optimize multiple tasks, including FoV prediction, bitrate allocation for each transmitted video tile, SR enhancement decision per tile, and so on.
- We conduct extensive real trace-driven evaluations on the proposed algorithm, and our results show that SR360 can achieve significant improvement compared with other state-of-the-art methods. Especially, the QoE measure can be improved by over 30%.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the system overview, and the details of algorithm design. Section 4 illustrates the evaluation results. Finally, Section 5 concludes the paper and outlooks future research directions.

2 RELATED WORK

The research on 360-degree video is pretty active in recent years. The previous work related to our paper can be categorized into the following three aspects:

FoV prediction. Previous approaches commonly used a FoV predictor to predict a user's viewport in the next video chunk, and then allocate proper bitrate for each video tile according to the prediction results. DRL360 [23] compared LSTM model for viewport prediction with LR algorithm, CNN model and KNN algorithm. The comparison result showed that LSTM model performs the best. Nguyen *et al.* [14] proposed to combine panoramic saliency maps with user head orientation history for head movement prediction. CLS [21] integrated the past fixations with cross-user region-of-interest (ROI) derived from their historical fixations. However, due to diverse user behaviors, the prediction error cannot be completely avoided.

Adaptive bitrate algorithm. When the availability of network bandwidth is highly dynamic, it is effective to use adaptive streaming and allocate different bitrates for each tile. Petrangeli *et al.* [16] adopted a heuristic approach, which aimed at maximizing bandwidth utilization and allocating tiles in the viewport with the highest quality. Rubiks [6] and Pano [5] used an MPC-based optimization framework to optimize user QoE in the chunk level. A few studies [9, 23] directly optimized user QoE based on deep reinforcement learning. The DRL-based algorithms have been shown to achieve better performance in 360-degree video streaming.

Super-resolution. The purpose of super-resolution is to recover a high-resolution image from a single or multiple lower resolution image(s). In recent studies [10, 12], deep neural networks (DNNs) have been adopted to learn the mapping from low-resolution images to high-resolution images. NAS [22] was the first video-on-demand system to apply super-resolution DNNs to improve the performance of conventional video streaming. In spite that the technique of SR has been invented for many years, there is very limited work to apply SR to 360-degree video streaming due to its high-bandwidth and low-latency requirements.

Our paper attempts to show the feasibility of boosting the performance of 360-degree video streaming with SR technique.

3 FRAMEWORK OF SR360 VIDEO SYSTEMS

In this section, we will introduce the framework of our proposed *SR360* 360-degree video systems, and provide the details of our algorithm design.

3.1 System Overview

In the SR360 framework, we conduct super-resolution operations at the client side to enhance the quality of low-resolution frames. Thus, the server can reduce the bandwidth consumption. The framework can be illustrated by Fig. 1. In the following, we briefly introduce all the components in the framework.

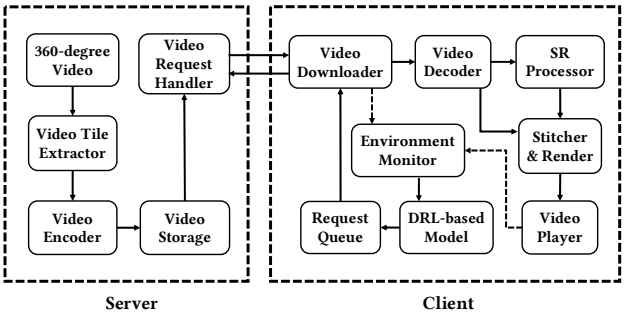


Figure 1: The *SR360* framework.

The server mainly handles video encoding and delivers data according to client requests. The input video is stored with the standard equirectangular projection. *Video Tile Extractor* divides the video frames into tiles and *Video Encoder* encodes the tiles with all candidate rates. The tiles are stored in the *Video Storage*. When the *Video Request Handler* receives video requests, it streams requested video contents to the client.

The client is responsible for video decoding, video enhancement, FoV prediction, bitrate selection, and so on. *Video Decoder* decodes tiles after the tiles being downloaded by the *Video Downloader*. Then, the *SR Processor* enhances the lower-resolution tiles inside the user's viewport. If all the downloaded tiles have the highest resolution or the client has no available GPU resource, the tiles will be directly stitched and rendered by the *Stitcher* and *Renderer*. Next, the *Environment Monitor* records the environment states of *Video Downloader* and *Video Player*. *DRL-based Model* takes the states as the input and performs actions, including FoV prediction, tile bitrate selection and decision on whether to enhance video quality by SR or not. The *Request Queue* makes a request according to actions of the *DRL-based Model* and *Video Downloader* sends requests for video chunks to the server.

3.2 Content-Aware Super-Resolution

The content-aware super-resolution can be seen as a type of video compression over the entire video. It can learn the features that map the low-resolution video to the high-resolution video. However, it is impractical to build a universal super-resolution model for 360-degree videos given the diversity among different videos.

To address the above challenge, we consider a content-aware SR model in which an independent SR model is built for each video. In this case, we can achieve near-zero training error due to the inherent overfitting property of DNN. We extend the super-resolution neural network in [22], which supports multi-scale super-resolution (x1, x2, x3, x4) and uses separate network for each resolution. Instead, we use a full DNN model rather than a scalable DNN, because the size of a full DNN is much smaller than the size of a 360-degree video chunk and the full DNN model can be pre-fetched by the client before video playback starts. Furthermore, in each residual block of SR model, a convolutional layer is followed by a ReLU activation function and we use Batch normalization to speed up the learning.

However, when the DNN model is extended to all 360-degree video frames, the processing time cannot satisfy the real-time requirement. Additionally, due to the limited FoV, it is a waste of client-side computation resources processing the whole 360-degree video frame. Therefore, we choose to apply the DNN model to each tile and compare the *non-tiling* case with the *tiling* case. For the whole 360-degree video frame, i.e., the *non-tiling* case, the training data contains pairs of a low resolution (e.g., 360p, 480p, 720p, 1080p) and the original highest resolution (e.g., 4K) image. For each tile, i.e., the *tiling* case, low resolutions include 130p, 180p, 270p, 480p and the highest resolution is 540p (the tiling setting is 6x4). The DNN models are offline trained on desktop PC with an Nvidia GTX 1080 GPU. We update the DNN parameters to minimize the difference between the DNN's output and the target high resolution image.

We obtain the processing time of the above two SR schemes and the results are shown in Table 1. The processing time for each 1-second chunk (or tile) includes three parts: *decoding*, *super-resolution* and *re-encoding*. As shown in the table, the processing time for a chunk of the whole frame is much higher than that for a single tile. Given the results, we adopt the *tiling* scheme for super-resolution. Furthermore, owing to the limited client-side computation resources, the tiles inside the predicted viewport have the highest priority to be enhanced by SR model.

We consider a typical adaptive 360-degree video streaming system. At the server side, a 360-degree video is temporally segmented into multiple chunks. Each chunk, with a time duration T, contains a series of frames. Each frame is spatially divided into N equal-sized tiles where the tile sequence number $i \in \{1, 2, \cdots, N\}$ is numbered from the top left to the bottom right. In this paper, our objective is to maximize user QoE by determining the bitrate for each tile and enhancing the video quality by SR model.

3.3 Problem Formulation

In this section, we will first define the QoE metrics to evaluate the performance of 360-degree video streaming and then formulate the optimization problem. Let us define $QoE_{1,k}$ as the average viewport quality, $QoE_{2,k}$ as the rebuffering time and $QoE_{3,k}$ as the average viewport quality change. The QoE metric of a chunk k can be given as follows:

$$\mathrm{QoE}_k = \alpha QoE_{1,k} - \beta QoE_{2,k} - \omega QoE_{3,k}, \tag{1}$$

where α is the weight of viewport quality, β and ω are the penalty weights of rebuffering time and the average viewport quality change. Each QoE metric is described as below.

3.3.1 Average Viewport Quality. As only the content inside the viewport can be viewed, the average viewport quality is the average bitrate of all tiles in the user's real viewport. Let $v_i^k \in \{0, 1\}$ denote an indicator to indicate whether the i-th tile of the k-th chunk is in the viewport. Let r_i^k be the bitrate for the i-th tile of the k-th chunk. To reflect the SR-based quality enhancement, the perceived bitrate can be calculated based on [22]:

$$\tilde{r}_i^k = SSIM^{-1}(SSIM(DNN(r_i^k))), \tag{2}$$

where $DNN(r_i^k)$ represents the quality enhancement by SR model. SSIM is the average structural similarity [20] to measure the tile quality in the chunk level, and its reverse $SSIM^{-1}$ maps an SSIM value back to the tile bitrate. To create the mapping, we measure the SSIM of original tiles at each bitrate (or resolution) and use piece-wise linear interpolation (e.g., $(r_0, SSIM_0)$, $(r_1, SSIM_1)$, ..., $(r_i, SSIM_i)$), where $r_i \in \mathcal{R}$ and $\mathcal{R}$ denotes the available choices of bitrates. If the tile is not enhanced by SR model, $\tilde{r}_i^k = r_i^k$. We have the average viewport quality of the k-th chunk as:

$$QoE_{1,k} = \frac{\sum_{i=1}^{N} v_i^k \tilde{r}_i^k}{\sum_{i=1}^{N} v_i^k}, \tag{3}$$

where N is the number of tiles in each chunk.

3.3.2 Rebuffering Time. The rebuffering time can be obtained by:

$$QoE_{2,k} = (D_k + P_k - B_k)_+, \tag{4}$$

where $(x)_+ = max\{x, 0\}$ means that the term is non-negative. Note that D_k is the downloading time of the k-th chunk, P_k denotes the processing time taken to enhance the tiles by SR model, B_k denotes the buffer occupancy when the client starts to download the k-th chunk. If the next chunk has not been completely downloaded or the SR model is still processing the tiles when the buffer occupancy is empty, it would cause rebuffering. Here D_k can be calculated by:

$$D_k = \frac{\sum_{i=1}^{N} s_i^k}{N_k}, \tag{5}$$

where s_i^k is the file size for the i-th tile of the k-th chunk and N_k denotes the average bandwidth for the k-th chunk. Next, we provide the formulation of $B_k \in [0, B_{max}]$, where B_{max} is the maximum buffer size. The dynamic buffer occupancy can be formulated as:

$$B_{k+1} = (B_k - D_k)_+ + T, \tag{6}$$

where T is the duration of a downloaded video chunk.

3.3.3 Average Viewport Quality Change. This metric represents the average bitrate difference between adjacent chunks, which can be measured by:

$$QoE_{3,k} = \left| \frac{\sum_{i=1}^{N} v_i^k \tilde{r}_i^k}{\sum_{i=1}^{N} v_i^k} - \frac{\sum_{i=1}^{N} v_i^{k-1} \tilde{r}_i^{k-1}}{\sum_{i=1}^{N} v_i^{k-1}} \right| \tag{7}$$

Table 1: The SR processing time for 1-sec content of two schemes.

Scheme	Non-tiling				Tiling			
Input Resolution	360p	480p	720p	1080p	130p	180p	270p	480p
Processing Time (sec)	2.64	2.72	2.74	2.73	0.10	0.10	0.11	0.11

In order to maximize the long-term QoE, the optimization problem for adaptive 360-degree video streaming can be formulated as follows:

$$\max_{\{r_1^k, r_2^k, \cdots, r_N^k\}} \quad \sum_{k=1}^{M} \mathrm{QoE}_k \qquad \text{s.t.} \quad (2)(5)(6). \tag{8}$$

where M is the total number of video chunks.

3.4 DRL-driven Algorithm Design

In the above problem, we need to optimize multiple objectives and make a set of decisions (including FoV prediction, bitrate selection, etc). As a subfield of machine learning, reinforcement learning (RL) can automate decision-making and optimize the long-term reward. Based on the observed states of a given system, RL model guides how the agent ought to take actions in an environment in order to maximize the cumulative reward. Deep learning further enables RL to handle the scenarios with high-dimensional state space and action space [2].

In this section, we propose a DRL-based rate adaptation algorithm to solve the optimization problem. Different from the previous DRL-based algorithms for 360-degree video streaming, our algorithm doesn't contain any additional FoV estimator. At each step, the agent not only makes the bitrate decision but also predicts the viewport of the user for the next chunk. Moreover, the agent can decide whether to process the tiles inside the viewport by SR model or not. Note that the bitrate decision determines the tiles inside the predicted viewport and the lowest quality level will be chosen automatically for tiles outside the predicted viewport.

States: After the download of each chunk k, the state s_k can be represented by:

$$s_k = (R_k, B_k, \vec{x_k}, \vec{d_k}, \vec{\mathrm{fov}_k}, \vec{vp}_k, c_k). \tag{9}$$

The state s_k is the input of the DRL neural networks. Specifically, R_k is the average bitrate of the tiles inside the real viewport in the last downloaded video chunk; B_k is the current buffer occupancy; $\vec{x_k}$ is the average network throughput for the past p chunks; $\vec{d_k}$ is the download time of the past p chunks; $\vec{\mathrm{fov}_k}$ is the vectors of m versions of the available file sizes for viewport areas; $\vec{vp}_k$ represents the tile sequence numbers of the past p viewpoint; c_k is the number of chunks left.

Actions: The DRL-based model takes the action for the next chunk. Let $a_k = \{a_k^1, a_k^2, a_k^3\}$ denote the action for the k-th chunk. $a_k^1 \in \{1, 2, \cdots, N\}$ denotes the tile sequence number of the viewpoint for the k-th chunk, where N is the number of tiles in each chunk. Once a_k^1 is determined, the user's viewport can be mapped. $a_k^2 \in \mathcal{R}$ denotes the available choices of bitrates for the tiles inside viewport. $a_k^3 \in \{0, 1\}$ determines whether tiles inside the viewport of the k-th chunk to be enhanced by SR model or not. The DRL-based model seeks to predict the tile sequence number of the viewpoint, determine the rate for the tiles inside the viewport and choose whether to enhance the tiles or not. So there are actual three actions chosen simultaneously at each step.

Rewards: The reward of the k-th chunk is defined as follows:

$$\tau_k = \alpha QoE_{1,k} - \beta QoE_{2,k} - \omega QoE_{3,k} - \mu A_k. \tag{10}$$

$$A_k = (\frac{1}{acc_k + \epsilon} - 1)_+, \tag{11}$$

where μ is the penalty weight about the viewport prediction error. A_k is the penalty item of the viewport prediction error for the k-th chunk, which means that the lower the accuracy is, the higher the punishment is. acc_k is the viewport prediction accuracy for the k-th chunk. ϵ is a small number to deal with the situation when acc_k goes to zero. We add this item to the reward for the purpose of better making the viewport prediction.

Policy and training algorithm: The agent makes its action decision based on the policy $\pi : (s_k, a_k) \rightarrow [0, 1]$, which is the probability distribution of taking an action a_k at the state s_k. We use a neural network to obtain the policy $\pi_\theta : (s_k, a_k)$, where θ are the parameters of the neural network. We use the state-of-the-art actor-critic method A3C [13] as our training algorithm. Given the policy network parameters θ, the gradient can be computed by:

$$\nabla_\theta \mathbb{E}_{\pi_\theta} \left[\sum_{k=0}^{n} \gamma^k \tau_k \right] = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) A^{\pi_\theta}(s, a) \right], \tag{12}$$

where $A^{\pi_\theta}(s, a)$ is the advantage function indicating how better the action a is than other actions drawn from the policy at the state s.

In practice, the agent samples a trajectory of the tile bitrate selection and empirically estimates an unbiased $A^{\pi_\theta}(s_k, a_k)$ as $A(s_k, a_k)$. The actor network parameters can be updated with the learning rate ρ as below:

$$\theta = \theta + \rho \sum_k \nabla_\theta \log \pi_\theta (s_k, a_k) A(s_k, a_k) \tag{13}$$

The critic network can be updated as follows:

$$\theta_v = \theta_v - \rho_v \sum_k \left(\tau_k + \gamma V^{\pi_\theta}(s_{k+1}; \theta_v) - V^{\pi_\theta}(s_k; \theta_v) \right)^2, \tag{14}$$

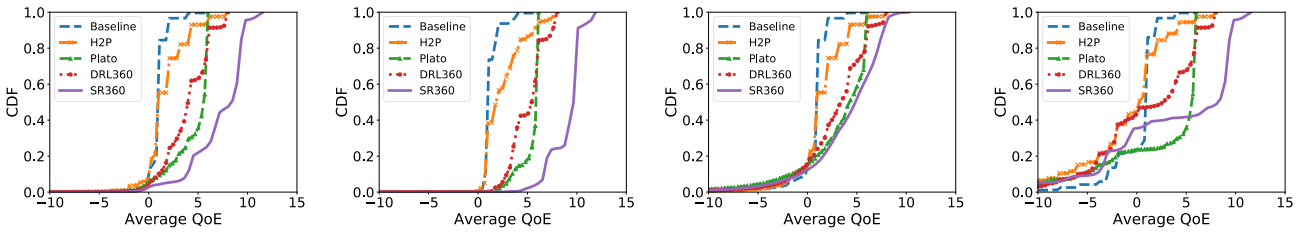
where $V^{\pi_\theta}(.; \theta_v)$ is the output of the critic network, ρ_v is the learning rate, and τ_k is the reward at time step k.

4 EVALUATION

In this section, we evaluate the performance of our model by using a 360-degree video player simulator with real traces.

4.1 Experiment Settings

DRL model settings: For the actor network, SR360 passes $p = 8$ past bandwidth measurements, chunk download time and tile sequence numbers of viewpoint to three identical 1D convolution



(a) (1, 1, 1) (b) (1, 0.25, 0.25) (c) (1, 4, 1) (d) (1, 1, 4)

Figure 2: The CDFs of the average QoE value under four QoE objectives.

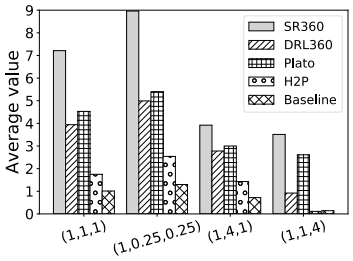


Figure 3: Comparison of different algorithms on various QoE objectives.

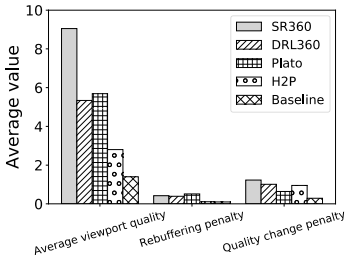


Figure 4: Comparison of different algorithms on the average values of the individual terms in the QoE metric with weights (1, 1, 1).

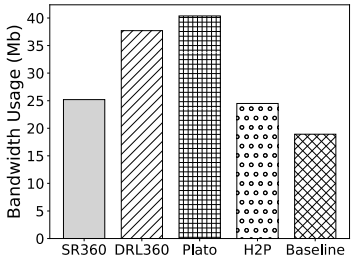


Figure 5: Comparison of different algorithms on the average bandwidth usage per chunk on the QoE metric with weights (1, 1, 1).

layers (CNN) with 128 filters, each of size 4 with stride 1. The file size of the next viewport is passed to another 1D-CNN with the same shape. The last average bitrate, the current buffer size and the number of chunks left are passed to three linear layers, each of which has 128 neurons. The outputs from these layers are then passed to a linear layer whose neurons are the same as the size of action space. The critic network uses the same NN architecture, but its final output is only one neuron. We set the discount factor $\gamma = 0.99$. The learning rates for the actor and the critic are both set to 0.0001.

Videos and head movement traces: We collect seven videos from the dataset [4], which includes fast-paced and slow-paced videos. The dataset includes head movement traces collected from 59 users watching 70-sec 360-degree videos on the Razer OSVR HDK2 HMD. We randomly select two videos as the testing set, and the rest of the videos are taken as the training set. The original 3840 × 2160 video is re-encoded using the kvazaar encoder [1], which is an open-source software for H.265. For each 1-sec chunk, we divide the video into 6 × 4 tiles (i.e., $N = 24$) and the resolution of each original tile is 540p (640 × 540). The videos have five candidate rates, i.e., $|\mathcal{R}| = 5$. Each tile is assigned with 1Mbps (130p), 2Mbps (180p), 4Mbps (270p), 6Mbps (480p) and 8Mbps (540p).

Bandwidth traces: The 4G/LTE measurements dataset [19] was selected to simulate network throughput. The throughput in the dataset ranges from 0 Kb/s to 95 Mb/s. The duration ranges from 166 sec to 758 sec. A total of 40 logs were collected, covering 5 hours of active monitoring.

We compare the performance of the proposed methods with four state-of-the-art 360-degree video streaming algorithms.

- **Baseline** [18], which allocates rates for the entire video, without the tile-based allocation. It uses linear regression (LR) to predict the bandwidth and selects the maximal rate within the bandwidth estimation.
- **H2P** [16], which allocates the rates for the tiles with a heuristic algorithm.
- **Plato** [9], which uses the LSTM based neural network model to predict the viewport in the future. It makes bitrate decisions for both viewport and non-viewport areas by the actor-critic algorithm.
- **DRL360** [23], which leverages the LSTM model to predict the bandwidth values and the viewport in the future, and allocates the rates for tiles inside the viewport by the actor-critic algorithm.

QoE objectives. According to [23], we use four sets of parameters for (α, β, ω) to indicate four QoE objectives with various preferences, that is (1, 1, 1), (1, 0.25, 0.25), (1, 4, 1), (1, 1, 4), indicating comprehensive consideration, maximizing the quality of viewports, minimizing the rebuffer time and reducing the viewport quality change respectively.

4.2 Experiment Results

We compare the proposed SR360 approach with other algorithms on each QoE objective mentioned in the experiment settings. Fig. 2 shows the cumulative distribution function (CDF) of the average

QoE for each algorithm under all four QoE objectives. Fig. 3 shows the average QoE achieved by each algorithm on the entire testing data. The results demonstrate that SR360 outperforms the best of existing algorithms. In comparison, the average QoE achieved by SR360 is at least 30% higher than that of other methods. In particular, SR360 achieves the highest average QoE when trying to maximize the quality in the viewport in terms of QoE objective, i.e., (1, 0.25, 0.25). If paying more attention to rebuffering time in the QoE objective, i.e., (1, 4, 1) or focusing more on the viewport quality change in the QoE objective, i.e., (1, 1, 4), the average QoE achieved by each algorithm degrade correspondingly but SR360 still performs better than other algorithm.

To better understand QoE gains obtained by SR360, we analyze its performance on each individual term for the QoE objective with weights (1, 1, 1). Fig. 4 compares SR360 with other algorithms in terms of the average viewport quality, the penalty of rebuffering and the penalty of average viewport quality change. As shown, SR360 achieves the highest viewport quality, which is the benefit obtained by our bitrate adaptation and super-resolution. Moreover, SR360 achieves similar rebuferring time with other algorithms. However, SR360 suffers a little more penalty of quality change than other schemes, it is because the decision on SR enhancement may cause video quality change between two adjacent video chunks. Overall, SR360 can jointly optimize the different terms of QoE.

Fig. 5 shows the average bandwidth usage per chunk for the quality metric with parameters (1, 1, 1). The Baseline scheme uses the least bandwidth as it allocates bitrate for the entire 360-degree video frame as regular video and the video is often allocated with a low bitrate. As shown, the bandwidth usage of SR360 is lower than that of the other two DRL-based algorithm, i.e., DRL360 and Plato. The result shows that SR360 can achieve high QoE with moderate bandwidth consumption.

5 CONCLUSION

In this paper, we leveraged super-resolution to reduce bandwidth consumption of 360-degree video streaming and proposed a deep reinforcement learning (DRL) based model to predict a user's FoV, allocate bitrates for each tile and make decision on whether to enhance a video tile by super-solution model jointly. Using real-world traces of VR head movement and network bandwidth, we evaluated the effectiveness of our proposed SR360 framework. The results show that SR360 achieves significant improvement in terms of user QoE. In the next step, we plan to investigate how to perform SR in finer granularity and further improve user QoE.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China under Grants U1911201 and 61802452, Guangdong Special Support Program under Grant 2017TX04X148, the Natural Science Foundation of Guangdong under Grant 2018A030310079, and the Fundamental Research Funds for the Central Universities under Grant 19LGZD37.

REFERENCES

[1] 2018. Kvazaar. https://github.com/ultravideo/kvazaar.
[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.
[3] Cisco. 2017. Cisco Visual Networking Index Report. (2017). http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.
[4] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 2017. 360-Degree Video Head Movement Dataset. In *Proc. of the 8th ACM on Multimedia Systems Conference (MMSys)* (Taipei, Taiwan). 199–204.
[5] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360-degree Video Streaming with a Better Understanding of Quality Perception. In *Proc. of the ACM Special Interest Group on Data Communication (SIGCOMM)* (Beijing, China). 394–407.
[6] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Rubiks: Practical 360-degree streaming for smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 482–494.
[7] Mohammad Hosseini. 2017. View-aware tile-based adaptations in 360 virtual reality video streaming. In *2017 IEEE Virtual Reality (VR)*. IEEE, 423–424.
[8] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 97–108.
[9] X. Jiang, Y. Chiang, Y. Zhao, and Y. Ji. 2018. Plato: Learning-based Adaptive Streaming of 360-Degree Videos. In *Proc. of IEEE 43rd Conference on Local Computer Networks (LCN)*. 393–400.
[10] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1646–1654.
[11] Jean Le Feuvre and Cyril Concolato. 2016. Tiled-based adaptive streaming using MPEG-DASH. In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 41.
[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 136–144.
[13] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. 1928–1937.
[14] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction. In *Proc. of the 26th ACM International Conference on Multimedia (MM)* (Seoul, Republic of Korea). 1190–1198.
[15] Omar A Niamut, Emmanuel Thomas, Lucia D'Acunto, Cyril Concolato, Franck Denoual, and Seong Yong Lim. 2016. MPEG DASH SRD: spatial relationship description. In *Proceedings of the 7th International Conference on Multimedia Systems*. 5.
[16] Stefano Petrangeli, Viswanathan Swaminathan, Mohammad Hosseini, and Filip De Turck. 2017. An http/2-based adaptive streaming framework for 360 virtual reality videos. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 306–314.
[17] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM, 1–6.
[18] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. 133–144.
[19] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.
[20] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
[21] Lan Xie, Xinggong Zhang, and Zongming Guo. 2018. CLS: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 564–572.
[22] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 645–661.
[23] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li. 2019. DRL360: 360-degree Video Streaming with Deep Reinforcement Learning. In *Proc. of IEEE Conference on Computer Communications (INFOCOM)*. 1252–1260.