# DEEP REINFORCEMENT LEARNING-BASED RATE ADAPTATION FOR ADAPTIVE 360-DEGREE VIDEO STREAMING

*Nuowen Kan, Junni Zou, Kexin Tang, Chenglin Li, Ning Liu, Hongkai Xiong**

School of Electronic Information & Electrical Engineering, Shanghai Jiao Tong University, China

## ABSTRACT

In this paper, we propose a deep reinforcement learning (DRL)-based rate adaptation algorithm for adaptive 360-degree video streaming, which is able to maximize the quality of experience of viewers by adapting the transmitted video quality to the time-varying network conditions. Specifically, to reduce the possible switching latency of the field of view (FoV), we design a new QoE metric by introducing a penalty term for the large buffer occupancy. A scalable FoV method is further proposed to alleviate the combinatorial explosion of the action space in the DRL formulation. Then, we model the rate adaptation logic as a Markov decision process and employ the DRL-based algorithm to dynamically learn the optimal video transmission rate. Simulation results show the superior performance of the proposed algorithm compared to the existing algorithms.

***Index Terms*—** 360-degree videos, adaptive streaming, rate adaptation, deep reinforcement learning

## 1. INTRODUCTION

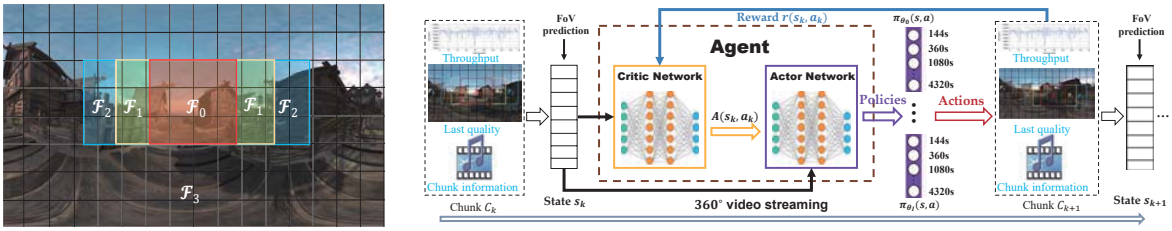
In recent years, 360-degree videos that provide immersive sensation by placing viewers at the center of a 360-degree scene have become increasingly popular, particularly in live sports and games. Through wearing a head-mounted display (HMD), the viewer can freely adjust his/her head orientation to control the field of view (FoV) during the video playback. Compared to traditional videos, 360-degree videos have much higher resolution and thus require more bandwidth to deliver the entire scene, which may result in rebuffering (i.e., playback interruption due to the empty buffer) under the bandwidth-constrained environment. Considering that a viewer's FoV only covers a partial area of the full 360-degree scene at a given time, [1, 2] propose a tile-based partial delivery approach that predicts the viewer's FoV in advance and delivers only the corresponding tiles according to the FoV prediction. This however imposes novel challenges to the adaptive streaming of 360-degree videos. For instance, for streaming traditional videos, the possible video quality reduction or stalling under poor channel conditions can be avoided by existing rate adaptation methods [3] via moderately reducing the bitrate to enlarge the buffer occupancy when the network throughput is sufficient. In adaptive 360-degree video streaming, however, an erroneous FoV prediction may result in a mismatch between the prefetched content in the buffer based on the FoV prediction and the content covered by the real FoV of viewers. Moreover, the quality in the real FoV will not be updated until the prefetched video content is played, which leads to an *FoV switching latency*. Due to the dynamics and uncertainties of the FoV and network condition, it is therefore essential to strike a trade-off between the FoV switching latency, bandwidth efficiency, video playback quality and the risk of rebuffering. In this paper, we focus on how to adaptively control the bitrates to achieve such an optimal trade-off for 360-degree video streaming.

In a typical adaptive 360-degree video streaming system, an original video is temporally divided by the server into multiple chunks each of which contains a constant number of video frames, while each frame is spatially divided into multiple tiles that are encoded in different bitrates and resolutions. On the viewer's side, before downloading the chunks from the server, the bitrate of every tile is to be determined by a rate adaptation strategy. To maximize the quality of experience (QoE) under a bandwidth-constrained condition, initial steps [4, 5, 6, 7] have been made with respect to the study of adaptive 360-degree video streaming. But crucially, all these works neglect the FoV switching latency and attempt to seek the trade-off by using heuristic methods and with assumptions on the prior knowledge of the network condition.

On the other hand, with the prosperity of artificial intelligence, [8, 9, 10, 11] attempt to tackle challenges in streaming of the traditional videos by utilizing the reinforcement learning (RL). The works in [8, 9] show the ability of RL-based methods in solving the rate adaptation optimization problems for traditional video streaming. And recent works in [10, 11] outperform the traditional rate adaptation algorithms through using the deep reinforcement learning (DRL) technique. Somewhat counter-intuitively, considerable challenges might occur when directly applying the above rate adaptation algorithms to 360-degree adaptive video streaming, in which not only the bitrates of temporally divided chunks require to be selected, but the bitrate of every spatially divided tile within a frame should be simultaneously determined based on the dynamics of the viewer's FoV. This will cause a problem of *combinatorial explosion*.

To reduce the FoV switching latency and relieve the

*The work has been partially supported by the NSFC grants No. 61622112 and No. 61871267.




(a) (b)

**Fig. 1**. (a) A scalable FoV example, where $\mathcal{F}_0$ has the maximum probability to be seen by viewer. (b) Main framework of our proposed DRL-based rate adaptation algorithm for adaptive 360-degree video streaming.

combinatorial explosion problem for adaptive 360-degree video streaming, in this paper, we propose a *deep reinforcement learning-based rate adaptation algorithm* based on the asynchronous advantage actor-critic (A3C) algorithm [12] of DRL. Specifically, we define a QoE metric particular for the adaptive 360-degree video streaming, by introducing a penalty term that discourages the large buffer occupancy to reduce the possible FoV switching latency. To relieve the combinatorial explosion, we propose a *scalable FoV* method that divides the 360-degree scene into multiple scalable FoVs according to the viewing probability of tiles [5] as illustrated in Fig. 1(a), where tiles in the same FoV are assigned with the same bitrate. By doing so, an independent and low-dimensional bitrate set is achieved in each FoV. We then formulate the rate adaptation logic as a Markov decision process (MDP), which is then leveraged to adaptively select the bitrate (viewed as the *action* in RL) for every tile in the future chunks according to a variety of system dynamics (viewed as the *state*). Finally, we implement our proposed algorithm and the simulation results show that our algorithm outperforms existing algorithms in terms of the overall QoE while keeping the buffer occupancy to a properly low level.

The rest of paper is organized as follows. Section 2 describes the system models and formulates the optimization problem. The DRL-based rate adaptation algorithm design and its performance evaluation are presented in section 3 and section 4, respectively. The conclusion of this paper is given in section 5.

## 2. PROBLEM FORMULATION

We consider a typical adaptive 360-degree video streaming system. At the server, an original 360-degree video is temporally divided into multiple chunks that contains a constant number of frames (i.e., corresponding to a fixed time duration of $L$), while each frame is spatially divided into $Y \times Z$ tiles that are encoded in $M$ different bitrates. Upon receiving the downloading requests of video chunks from the clients, the server first responds to the clients with a manifest including the set of available bitrates for each tile. A rate adaptation algorithm is then designed to select the optimal bitrate for each tile according to the observed state (e.g., network throughput, buffer occupancy, etc.). After that, video chunks composed of these selected tiles with proper bitrates will then be delivered to the clients.

In this paper, we assume that the viewers' viewport is predicted in advance, by using the FoV prediction method in [5, 13]. Based on the prediction, we further denote a scalable FoV set, $\{\mathcal{F}_0, \mathcal{F}_1, \cdots, \mathcal{F}_I\}$, which is sorted in a decreasing order of the probability that each FoV is watched by the viewer. An example of the scalable FoV set is illustrated in Fig. 1(a), where four different levels of FoVs are shown in the panoramic frame. Tiles in $\mathcal{F}_0$ own the sovereign priority in streaming, while tiles in $\mathcal{F}_3$ can be ignored when the transmission resource is limited.

We denote by $R_{y,z,k} \in \mathcal{R}$ the bitrate for the $y$-th row and $z$-th column tile $T_{y,z}$ in the $k$-th chunk $C_k$, where $\mathcal{R}$ is the set of available bitrates. The quality of the tile $T_{y,z}$ in chunk $C_k$ is denoted as $q_{y,z,k} = g(R_{y,z,k})$, where the function $g(x)$ maps a bitrate to the video quality. Hence, the quality of chunk $C_k$ can be calculated as:

$$Q_k = \sum_{i=0}^{I} \mathcal{P}_i \sum_{T_{y,z} \in \mathcal{F}_i} q_{y,z,k}, \tag{1}$$

where $\mathcal{P}_i$ is derived from the viewing probability model in [5] and represents the average viewing probability of tiles in $\mathcal{F}_i$.

Moreover, let $h_k$ denote the average network throughput while downloading the chunk $C_k$, and $u_k^i$ be the size of $\mathcal{F}_i$ (i.e., the sum of the tile size in each FoV). The downloading time of $C_k$ can be calculated as $d_k = \sum_{i=0}^{I} u_k^i / h_k$. Next, we denote the buffer occupancy when the client starts to download $C_k$ as $B_k \in [0, B_{\max}]$, where $B_{\max}$ is the maximum buffer size. Hence, the dynamic buffer occupancy can be formulated as:

$$B_{k+1} = \left(B_k - d_k\right)_+ + L, \tag{2}$$

where the notation $(x)_+ = \max\{x, 0\}$ ensures that the term is nonnegative. Note that if the term $B_k - d_k$ is negative, the client has no video remaining in the buffer and the chunk $C_k$ has not been downloaded completely yet, which is called a rebuffering event. Inversely, if the term $B_k - d_k$ is positive, there are some future chunks prefetched in the buffer, which will cause a possible FoV switching latency if an erroneous

viewport prediction exists and the value of the term represents the duration of prefetched chunks.

To jointly take the video playback quality, FoV switching latency, rebuffering time and quality variation into consideration, we define the QoE metric of a chunk $C_k$ as follows:

$$\text{QoE}_k = Q_k - \beta(B_k - d_k)_+ - \lambda\big(d_k - B_k\big)_+ - \mu\sum_{i=0}^{I}\Delta_i, \quad (3)$$

where $\beta$ controls the strength of limiting the client to prefetch chunks, $\lambda$ and $\mu$ are the penalty weights of rebuffering time and quality variation, respectively, $\Delta_i = \sum_{T_{y,z}\in\mathcal{F}_i} q_{y,z,k} - q_{y,z,k-1}$represents the quality variation which can result in an annoying effect on watching experience.

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

$$\begin{aligned}\max_{R_{y,z,k}\in\mathcal{R}} \quad & \sum_{k=\tau}^{\infty}\gamma^{k-\tau}\text{QoE}_k \\ \text{s.t} \quad & R_{y,z,k} = R_{y',z',k}, \forall\, T_{y,z}, T_{y',z'} \in \mathcal{F}_i,\end{aligned} \quad (4)$$

where the objective function represents the discounted accumulative QoE starting from the chunk $C_\tau$, the constraint specifies that the tiles in the same FoV are with the same bitrate and $\gamma \in [0, 1)$ is an exponential discount factor.

## 3. DEEP REINFORCEMENT LEARNING BASED RATE ADAPTATION

To solve the optimization problem, we propose a DRL-based rate adaptation algorithm for adaptive 360-degree video streaming. The framework of our proposed algorithm is shown in Fig. 1(b). Specifically, an agent, consisting of an actor network and a critic network, interacts with the environment which can be described as a MDP, namely a tuple ⟨state, action, state transition probability, reward⟩. At each step, the agent first observes the state from the environment. For each FoV, the actor network generates a policy, then an action is chosen to apply in the environment, which results in a reward feeding back to the agent and the environment's switching to a new state.

### 3.1. MDP in adaptive 360-degree video streaming

We characterize and define the environment of adaptive 360-degree video streaming system as the state $s_k = (\vec{h}, \vec{d}, b_k, e_k, \vec{R_k}, \vec{\tau_k}, \vec{\nu_k})$. Specifically, $\vec{h}$ is the average network throughput measurements for the past $p$ chunks; $\vec{d}$ is the downloading time of the past $p$ chunks; $b_k$ is the buffer occupancy when the client starts to download $C_k$; $e_k$ is a flag bit representing the end of chunk sequence; $\vec{R_k}$ is a vector consisting of the bitrates at which the tiles were downloaded in the last chunk; $\vec{\tau_k}$ and $\vec{\nu_k}$ represent the position of scalable FoV and the available bitrates of each FoV.

The agent takes the state as inputs of the actor and critic neural network and chooses an action $a_k \in \mathcal{R}$ based on a policy $\pi : \pi(a_k|s_k) \to [0, 1)$ which is the probability distribution of taking an action $a_k$ at the state $s_k$. However, if we follow the common setting that uses a policy to select an action at each step, the action space with respect to adaptive 360-degree video streaming increases to $M^{Y\times Z}$, which results in a combinatorial explosion. To tackle this problem, we define an action $a_k^i$ which has $M$ different choices of bitrates for $\mathcal{F}_i$ independently. Therefore, there are $I$ actions chosen simultaneously at every step, corresponding to the $I$ FoVs. Let $a_k = \{a_k^0, a_k^1, \cdots, a_k^I\}$ denote the action for chunk $C_k$, accordingly, let $\pi^i(a_k^i|s_k)$ denote the policy of choosing actions for $\mathcal{F}_i$. In summary, we define a policy and an action for each FoV, and the policies satisfy:

$$\pi(a_k|s_k) = \prod_{i=0}^{I}\pi^i(a_k^i|s_k). \quad (5)$$

Upon applying actions to the environment, an immediate reward $r(s_k, a_k)$ is fed back to the agent. In our algorithm, we define the reward as $r(s_k, a_k) = \text{QoE}_k$.

### 3.2. Training with policy gradient method

In an A3C algorithm setting, the actor neural network approximates the policy $\pi(a|s)$ using $\pi_\theta(a|s)$ with respect to parameter $\theta$. On the other hand, with parameters $\theta_v$, the critic neural network estimates the state value function $V^{\pi_\theta}(s;\theta_v)$, i.e., the expected total reward starting from state $s$ when the agent selects actions following the policy $\pi_\theta(a|s)$.

In order to derive the optimal policies, the parameters of the actor and critic neural network should be trained with the policy gradient method [14], which aims to maximize the objective function in Equation (4) by continuously adjusting the parameters of the neural network. The gradient of the objective function of actor network with respect to parameter $\theta$ can be formulated as [12]:

$$\nabla_\theta\mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta\log\big(\pi_\theta(a|s)\big)A^{\pi_\theta}(s,a)], \quad (6)$$

where $A^{\pi_\theta}(s,a) = r(s,a) + \gamma V^{\pi_\theta}(s+1;\theta_v) - V^{\pi_\theta}(s;\theta_v)$ is the advantage function representing how better the action $a$ is than other actions drawn from policy at the state $s$.

In practice, we apply the following update rule to train the parameters $\theta_i$ of each policy $\pi_{\theta_i}(a_k^i|s_k)$:

$$\begin{aligned}\theta_i \leftarrow \theta_i + \rho\sum_k \nabla_{\theta_i}\log\big(\prod_{i=0}^{I}\pi_{\theta_i}(a_k^i|s_k)\big)A^{\pi_\theta}(s_k,a_k) \\ +\beta_H\nabla_{\theta_i}H(\pi_{\theta_i}(\cdot|s_k)),\end{aligned} \quad (7)$$

where $\rho$ is the learning rate of the actor neural network, and $H(\cdot)$ is the entropy of the policy at each time step for ensuring the exploration, accordingly the hyperparameter $\beta_H$ controls the strength of exploration.

The update rule of parameter $\theta_v$ of the critic network follows the standard temporal difference method [15],

$$\theta_v \leftarrow \theta_v - \rho'\sum_k \nabla_{\theta_v}[A^{\pi_\theta}(s_k,a_k)]^2, \quad (8)$$

where $\rho'$ is the learning rate of the critic neural network.

## 4. PERFORMANCE EVALUATION

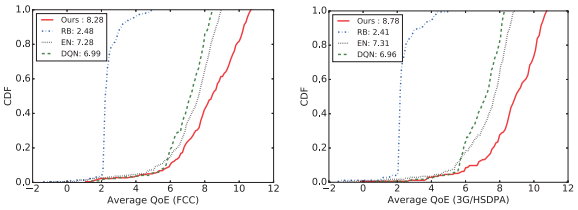
In our proposed algorithm, the actor and critic neural network have the same architecture except for the output layer. At the input layer, convolution layers are placed to process the state, including the last $p = 8$ measurements of throughput, the last $p = 8$ downloading time, the $M = 6$ available bitrates of each FoV and other inputs. Then the results of the input layer are merged to three full connection layers. The output layer in the actor neural network generates the policies with the Softmax activation function, while the output layer of the critic neural network generates the state value with a linear activation function. During the training, the discount factor is set as $\gamma = 0.99$, which implies that current actions will be influenced by 100 future steps. The entropy factor $\beta_H$ is controlled to decay $0.99$ times per $10^3$ iterations. And the learning rate of the actor and critic neural network is set as $10^{-4}$ and $10^{-3}$, respectively.

We generate the adaptive 360-degree video streaming with available bitrate set for each tile as $\{300, 700, 1600, 3700, 8600, 20000\}$ kbps. We divide the video into $80$ chunks with a duration of $1s$ and $16 \times 8$ tiles with the same size. In addition, the trace of viewport follows [13] and we transform it to a scalable FoV with four levels. For the QoE metric, we set $q_{y,z,k} = \log(R_{y,z,k})$ since the marginal improvement of quality decreases when the bitrates increase. We set the penalty weight defined in Eq (3) as $\mu = 0.1$, $\lambda = 8.0$, and $\beta = 2$, which means a rebuffering time of $1s$ or a prefetched video of $4s$ in the buffer receives the same penalty as a bitrate reduction of 3000 kbps.

To demonstrate the performance, we compare the proposed algorithm with the following algorithms: 1) rate-based algorithm (RB), which predicts the throughput by historical statistics and then selects the highest available bitrate under the predicted throughput; 2) enumerated algorithm (EN), which employs buffer occupancy observation and throughput prediction to select the bitrate that maximizes the given QoE metric over the next chunk using an enumerated method; 3) DQN, which is similar to our algorithm except that it uses the deep Q-learning [12] to learn the policies instead of A3C. The datasets of the network throughput traces used in this paper include a broadband dataset (FCC) and a 3G/HSDPA mobile dataset, which have been used in [3, 11]. The results of experiments are shown below, noting that the DQN and our algorithm have the same iterations in training.

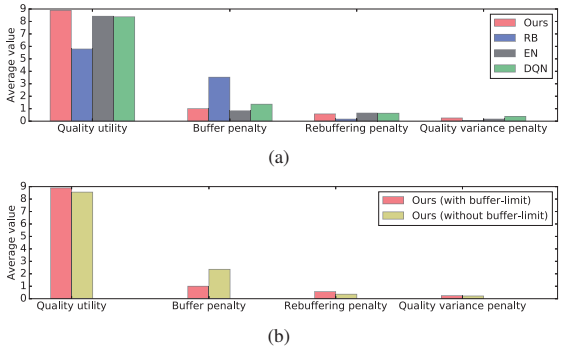
Fig. 2 shows the cumulative distribution function (CDF) over the average QoE (i.e., the mean of $\text{QoE}_k$ over a throughput trace) tested in the two datasets. It can be seen that our algorithm can always achieve an optimal trade-off when selecting the bitrates and a higher average QoE than others. In comparison, the average QoE achieved by our algorithm is at least $13.7\%$ higher than the others in the FCC dataset and $20.1\%$ higher than the others in the 3GP/HSDPA dataset.

Fig. 3(a) shows the average value of all the individual terms in the QoE metric for the FCC datasets. As shown, our algorithm presents a good capability of balancing the video rebuffering events and the FoV switching latency. Also, our algorithm has a higher quality utility than the other algorithms while keeping the rebuffering penalty and the buffer penalty at a low level. Fig. 3(b) shows that the penalty term of buffer occupancy in the QoE metric effectively limits the playback buffer (i.e, reduces the possible FoV switching latency) while only a slight impact on the rebuffering penalty is incurred.




(a) (b)

**Fig. 2**. Comparison on the CDF vs. average QoE, for (a) the FCC and for (b) the 3G/HSDPA dataset.




(a)

(b)

**Fig. 3**. (a) Comparison of different algorithms on the average values of the individual reward terms in the QoE metric, and (b) comparison on the average values of the individual reward terms in the QoE metric with and without the penalty term of buffer occupancy $\beta(B_k - d_k)_+$.

## 5. CONCLUSION

In this paper, we have presented a DRL-based rate adaptation approach for adaptive 360-degree video streaming. In particular, our algorithm could achieve an optimal trade-off between the FoV switching latency, bandwidth efficiency, video playback quality and the risk of video stall, by the optimal selection of bitrate for each tile before the client's downloading of the video chunks. The simulation results have shown that our algorithm outperformed the comparison algorithms in terms of the overall QoE and the FoV switching latency over different network throughput datasets.

## 6. REFERENCES


[1] M. Hossenini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," *CoRR*, vol. abs/1609.08729, 2016.

[2] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Lim, "MPEG DASH SRD: Spatial relationship description," in *Proceedings of the 7th International Conference on Multimedia Systems*, New York, NY, USA, 2016, MMSys '16, pp. 5:1–5:8, ACM.

[3] 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, pp. 325–338, Aug. 2015.

[4] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proceedings of the 2017 ACM on Multimedia Conference*, New York, NY, USA, 2017, MM '17, pp. 315–323, ACM.

[5] C. Liu, N. Kan, J. Zou, Q. Yang, and H. Xiong, "Server-side rate adaptation for multi-user 360-degree video streaming," in *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, pp. 3264–3268.

[6] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," in *Proceedings of the 2017 ACM on Multimedia Conference*, New York, NY, USA, 2017, MM '17, pp. 943–951, ACM.

[7] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proceedings of the 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct 2017, pp. 1–6.

[8] M. Claeys, S. Latré, J. Famaey, T. Wu, W. V. Leekwijck, and F. D. Turck, "Design and optimisation of a (FA) Q-learning-based HTTP adaptive streaming client," *Connection Science*, vol. 26, no. 1, pp. 25–43, 2014.

[9] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for DASH clients," in *Proceedings of the 7th International Conference on Multimedia Systems*, New York, NY, USA, 2016, MMSys '16, pp. 8:1–8:12, ACM.

[10] 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, Dec 2017.

[11] 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*, New York, NY, USA, 2017, SIGCOMM '17, pp. 197–210, ACM.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, New York, USA, Jun 2016, vol. 48, pp. 1928–1937, PMLR.

[13] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *Proceedings of the 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2017, pp. 1–9.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, 1999, NIPS'99, pp. 1057–1063, MIT Press.

[15] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA, USA, 1st edition, 1998.