Optimizing HAS for 360-degree videos

Christian Moldovan, Frank Loh, Michael Seufert, Tobias Hoßfeld

Optimizing HAS for 360-Degree Videos

Christian Moldovan, Frank Loh, Michael Seufert and Tobias Hoßfeld
University of Würzburg, Chair of Communication Networks, Würzburg, Germany
{christian.moldovan, frank.loh, michael.seufert, tobias.hossfeld}@uni-wuerzburg.de

Abstract—In recent years, an increasing number of Internet-based applications have been released that use virtual reality for education, training, gaming, and various forms of entertainment. When transmitting an omnidirectional 360° video over the Internet, it consumes considerably more data compared to traditional video streaming. To overcome the high network requirements and still reach a high Quality of Experience, HTTP adaptive streaming technology is considered for 360° video streaming. In contrast to traditional streaming, the adaptation logic considers not only the current network conditions, but also the viewport of the user, i.e., which part of the 360° sphere the user is currently focusing on. However, as the viewport of the user might change anytime, the adaptation logic is required to accurately predict the viewport in the next seconds of the playback to allow for an efficient and smooth streaming with a high visual quality.

In this paper, we present novel linear programs that determine the optimal visual quality, which is reachable in a given network scenario using different approaches for viewport prediction. Our results are an important contribution for designing adaptation logics for 360° video streaming, which allow for efficient data transmission in the network while reaching a high QoE in VR applications.

I. INTRODUCTION

Video streaming is the predominant form of traffic in today's Internet [20]. In the last decade, demands, challenges, and technologies for streaming have been changing tremendously. Streaming is no longer sporadic but has reached mainstream, with streaming content having evolved from short clips to TV content or feature films. Moreover, streaming is not only consumed at home, but to also in mobile scenarios, where network conditions might heavily fluctuate. To account for such fluctuations, HTTP Adaptive Streaming (HAS) was introduced. It allows to download different segments of the video in different qualities. Thus, the video can adapt to the current network conditions to avoid stalling, i.e., playback interruptions due to buffer underrun, which is the most severe degradation of Quality of Experience (QoE) and a major threat to user satisfaction [21].

Also, in the next years, the rise of streaming will continue. One major reason for this are the upcoming use cases and applications of virtual reality (VR), as VR requires streaming of omnidirectional videos, or so called 360° video, which are typically consumed on head mounted displays (HMDs). [19] expects the compound annual growth rate of the VR market to be 58.54% from 2018 to 2023. Moreover, the number of VR users in the US already reached 22.5 million in 2017 (11.5 million users of HMDs) and is expected to grow up to 57.1 million (30.6 million users of HMDs) in 2021 [4]. From a

conceptional perspective, a 360° video can be regarded as a spatial concatenation of several individual videos, so called tiles, each showing a dedicated part of the 360° sphere. A 360° video stream is technically composed of many individual video streams of tiles, such that every tile can be downloaded and played out in an individual quality [17].

The usage of HAS technology for 360° videos is beneficial for both end users, streaming providers, and network operators. On the one hand, similar to classical HAS, the video stream can adapt to the current network conditions and avoid stalling, which is a major QoE degradation. On the other hand, streaming providers and network operators can save resources on their servers and in the network by reducing the bit rate – and consequently the visual quality – of tiles, which are not in the viewport or focus of the end user. This would allow to serve more users with the same resources. However, there is a trade-off, as streaming providers and network operators still need to deliver the best possible streaming experience to their end users. Thus, the adaptation logic, which decides on the streamed quality of each tile, has to be well designed to address this trade-off.

A straightforward approach to reduce the required resources for 360° streaming is to consider the viewport of the user, i.e., which part of the 360° sphere the user is currently focusing on. Tiles are streamed in the best possible quality only if they are in the focus of the user, and in a reduced quality if they are out of focus. If the viewport of the user would be known at any time during the streaming, an optimal adaptation strategy could then be obtained. In this paper, we define and solve this optimization problem for omnidirectional videos in a given network scenario by a linear program (LP). It can be used as a benchmark tool that is helpful to evaluate adaptation strategies for 360° videos.

In a practical use case, however, the viewport of the user is not known in advance. Thus, it is necessary to predict the viewport in the next seconds of the playback and rely on these predictions for adaptation decisions. In this paper, we modify the previously mentioned LP to consider the practical, but possibly imperfect viewport prediction using three different prediction approaches. Then, we compare the results with the optimal solution as a benchmark and with the current state of the art. Our results show that a simple statistics-based approach performs best among all other tested methods in this scenario.

The reminder of this work is structured as follows. Section II discusses background and related work. In Section III, the LP for the optimal adaptation logic is presented. This benchmark LP, which relies on perfect knowledge about the viewport, it

is extended to compute optimal adaptation strategies for three practical, but possibly imperfect viewport prediction approaches. In Section IV, we compare the results of the LPs. Finally, Section V concludes the paper.

II. RELATED WORK ON 360° VIDEO STREAMING

A. Adaptive Video Streaming

Video streaming is based on a download of video data to a video buffer and concurrent playback from that buffer. However, if the throughput decreases below the video bit rate, the buffer might deplete, which results in a playback interruption, also called stalling or rebuffering. The introduction of HAS made it possible to adjust the video bit rate to the currently available downlink bandwidth in order to avoid stalling events. Therefore, each video is available in different video bit rates, reached by different encoding settings. Although stalling events are the most severe degradation of the perceived user's Quality of Experience (QoE), also the initial delay before the playback start and the visual quality of the video [21], [22] have to be considered.

In addition to this temporal adaptation, the current video encoding H265, which was introduced in 2013 [11], allows to also spatially split a video into tiles and encode each tile with a different bit rate [14]. Using this principle, regions of the video that are of low interest can be downloaded in a lower quality. This leads to an additional reduction of consumed data.

B. Linear Optimization of HAS

The first quadratic program that solved adaptive streaming was presented in [13]. In a first step the optimal quality is determined that is reachable without stalling. In a second step, the number of quality switches is minimized while playing at the optimal quality and while avoiding stalling. This program was extended in [10] to include multiple viewers who watch the same video at the same time. In [15], the program was extended to optimize the QoE fairness for multiple viewers who watch different videos asynchronously. In this paper, we focus on the first step, that optimizes the quality, and extend it for 360° videos.

C. 360° Video Streaming

Current video platforms always stream the whole video in the same visual quality, although in the case of 360° videos only a small share of the total video, the so-called viewport, is watched at any point in time by the user. To reduce the data consumption of 360° video streaming, approaches have been developed to predict which viewport the user will watch. If the viewport can be predicted accurately, it is sufficient to stream it in high visual quality while the remaining tiles can be transmitted in a lower quality, as they are less likely to be watched. However, an accurate viewport prediction is only possible if the video consists of short video segments [9], which leads to high segment overhead. A segment length of 2 seconds is suggested in [18] as a good trade-off between segment overhead and head movement prediction

accuracy. A description of the basic principles of adaptive tile-based streaming of omnidirectional video services over HTTP, available encoding options, and evaluations with respect to bit rate overhead, bandwidth requirements, and quality aspects can be found in [7]. To predict the performance of 360° videos in terms of application layer QoS and QoE the authors of [3] propose PERCEIVE, a method which uses machine learning techniques.

D. Viewport Prediction

Several works relied on viewport prediction in the context of 360° video streaming. The authors of [17] developed a framework for 360° videos that uses 35% less data while providing similar quality in slow viewport movement scenarios. First viewport-adaptive streaming algorithms were presented in [29] and [27]. The authors of [16] used a trajectory-based approach which groups past users that have a similar viewing trajectory and create a model of the viewport evolution over time for the identified groups. This model is used at prediction time for new users. The authors of [12] proposed a heatmap-based model and a trajectory-based model for viewport prediction. The authors showed that using other user's information increases the performance over only using the current user's information. A similar trajectory-based prediction scheme and an adaptation algorithm for VR videos were presented in [25]. In [2], it was suggested to investigate head movement prediction extracted from the feedback of previous users. A 360° head movement data set from 59 users was provided in [1]. The authors of [26] used a probabilistic model of viewport prediction to reduce side effects caused by wrong head movement prediction. Their approach significantly reduced stalling with small buffers. In addition, they provided an optimization problem that minimizes quality distortions and spatial quality variance. A first approach to using neural networks for viewport prediction was presented in [5]. They leveraged content- and sensor-related features for the prediction. Their approach, that was tested in a user study, consumes less bandwidth and has a shorter initial delay than other approaches. In [28], a viewport prediction model was presented that is based on a convolutional neural network (CNN). In addition, they presented a trajectory prediction model which is based on an RNN. The RNN was combined with a correlation filter-based viewport tracker that adds content awareness to increase the performance of the prediction. An approach that relies on learning contextual bandits to predict the viewport for 360° videos is presented in [8]. While other approaches follow the behavior of the current user, the authors of [6] believe that user behavior is hardly predictable and is mainly correlated to moving objects in the video. With this idea in mind, they develop a motion tracking algorithm that identifies representative moving objects. The authors of [23] separated video segments into two tiers: the basic whole video tier and the viewport tier. The whole video tier can be downloaded early and can be stored in the buffer while the viewport tier is downloaded on demand to enhance the current viewport.
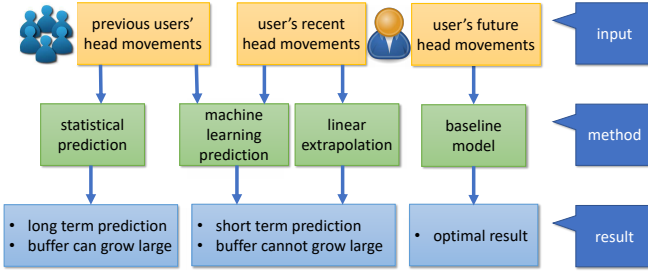
Figure 1: Overview of the four approaches with input and result. The baseline model only serves for comparison and is not applicable for practical purposes since it requires exact future knowledge.

III. Optimal Adaptation for 360° Videos

In this section, we first define a baseline linear program for 360° video streaming, i.e., tile-based video streaming. This linear program computes the optimal adaptation strategy in a given network scenario if the viewport of the user would be known. As in practice the viewport cannot be known in advance, we extend the baseline program to take into account three approaches for viewport prediction. An overview of the three approaches is given in Figure 1.

A. Baseline Linear Program for Optimal 360° Adaptation

In this problem, we download a set of n video segments in order. Each segment consists of m spatial tiles. To have a complete video, each tile of every segment must be downloaded exactly once, compare Eq. (2). Each tile can be downloaded in one of r resolutions or quality layers. The size in bytes of tile s of segment i in resolution j is defined as S_{ijs}. The quality of a segment is defined by its value function w_{ijs}.

For each segment i there exists a deadline D_i^2 until which it must be completely downloaded to allow seamless video streaming without stalling. The first segment must be completely downloaded within the allowed initial delay $D_1^2 = T_0$. The deadline of each consecutive segment i is defined as $D_i^2 = T_0 + (i-1) \cdot \tau, \quad \forall i > 1$, where τ is the duration of a segment.

The data volume that can be downloaded between time D_i^1 and D_i^2 is defined as $V(D_i^1, D_i^2)$. The sum of the volume of any consecutively downloaded tiles must not be greater than the available data volume until the last segments deadline, compare Eq. (4). In omnidirectional videos, the user only sees a subset of all tiles. Thus, we define the viewport Y_i of a segment i which includes all tiles of which at least one pixel is viewed during the segment. For the objective function only the viewed tiles are relevant. For the sake of simplification, we consider each tile equally important, even if it is viewed briefly or partially. The goal of the objective function (1) is to maximize the sum of the quality of all viewed tiles of all video segments while avoiding stalling. We formulate the corresponding binary linear program as follows:

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{s \in Y_i} w_{ijs} x_{ijs} \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{r} \sum_{s \in Y_i} x_{ijs} = 1 \tag{2}$$
$$\forall i \in \{1, \ldots, n\}$$
$$x_{ijs} \in \{0, 1\} \tag{3}$$
$$\forall i \in \{1 \ldots n\}, j \in \{1 \ldots r\}, s \in \{1 \ldots m\}$$
$$\sum_{i=1}^{k} \sum_{j=1}^{r} \sum_{s \in Y_i} S_{ijs} x_{ijs} \leq V(0, D_i^2) \tag{4}$$
$$\forall k \in \{1, \ldots, n\}$$

B. Optimal 360° Adaptation under Viewport Prediction

In a practical use case, the viewport of the user is not known in advance. Thus, it is necessary to predict the viewport in the next seconds of the playback and rely on these predictions for adaptation decisions. Thus, we extend the baseline program to take into account three approaches for viewport prediction, which are also presented in Figure 1:

- a statistics-based approach that uses other users' viewports to determine the viewport probability for each tile during each video segment,
- a linear extrapolation of the head movement trajectory of the current user, and
- a deep neural network that trains with previous users' head movement sequences during the same video before being applied to the current user's trajectory.

1) Statistical Long-Term Prediction: The first method assumes that most users share a common interest in a subset of all tiles during a segment. The idea is to download tiles in a higher resolution if they are watched more frequently. Since this method is independent of the current viewport, we can download tiles early and fill the buffer. For this, we require the frequency P_{isu} that a tile s of segment i is viewed by a viewer u. We then determine the probability $P_{is} = E[P_{isu}]$ that a tile is viewed over all previous users. For any new user, we use this probability P as a weight for the objective function. Consequently, in the linear program, the input of the algorithm changes as P and m have to be additionally considered, and the objective function (1) must be replaced with (5):

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{r} \sum_{s=1}^{m} P_{is} j x_{ijs}. \tag{5}$$

2) Linear Extrapolation: For this method, we assume that the head movement of viewers follows a linear trajectory over a short period of time. We extrapolate the head rotations of the three head rotation angles θ, ψ, ϕ, i.e., yaw, pitch and roll. A prediction horizon of $p \in \{2, 3 \ldots 10\}$ seconds is used to estimate the viewport during the next p seconds. While a short prediction window leads to more accurate results, a prediction window smaller than 2 s is not recommended since the video must be split into shorter segments which leads to
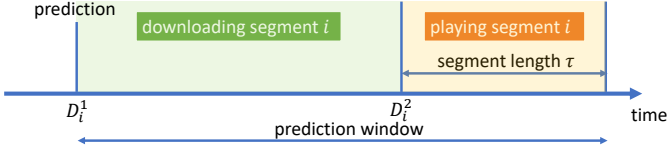
Figure 2: Sketch of the deadlines D_i^1, D_i^2 and prediction window in linear extrapolation and machine learning prediction.

worse encoding efficiency according to [9]. Therefore, we only use $p \geq 2$ for comparison with other methods.

However, if we base the download purely on live predictions, tiles must be downloaded very late, and the buffer cannot be filled for more than the prediction window p. This is expressed through the point in time D_i^1 at which the prediction is done. In Figure 2, we see that for any prediction window p and segment length τ it is $D_i^1 = D_i^2 + \tau - p$. We therefore have to replace the viewport Y with the predicted viewport Y^* and we add the point in time D_i^1 where the prediction occurs as the earliest possible download time. Constraints (4) must be replaced with the following constraint (6).

$$\sum_{i=l}^{k} \sum_{j=1}^{r} \sum_{s \in Y_i^*} S_{ijs} x_{ijs} \leq V(D_i^1, D_k^2) \qquad (6)$$
$$\forall l \in \{1, \dots, n\}, \forall k \in \{l, \dots, n\}.$$

3) Machine Learning Prediction: In the following, we assume that the head movement patterns of users are similar for the same video and not necessarily linear. Nevertheless, we can still rely on the same linear program that was used for linear extrapolation described in Section III-B2.

To obtain a non-linear viewport prediction, we will rely on machine learning to learn head movement patterns from previous viewers and to predict them for the next viewer. For this, we use a neural network that consists of a 4×1000 input layer, three hidden RNN layers with 256 neurons each and a 4×1000 output layer. The RNN uses a softsign activation function. We experimented with many different configurations for the RNN and used these since they returned the best results. The input consists of the four parts q_1, q_2, q_3, q_4 that define the quarternion that describes the head rotation as described in [1]. To speed up the RNN, we quantize the head movement traces of the users to the 1000 nearest values each.

During training, in each epoch we randomize the order in which the training users are selected. Training is conducted for 2000 epochs for each combination of parameters. We tested a prediction horizon of $p \in \{2, 3, \dots 10\}$ seconds to estimate the viewport during the next p seconds. Due to the same encoding efficiency problem described in Section III-B2, we use $p \geq 2$ for comparison with other methods.

The RNN returns a quarternion for each time step. From it we determine the viewport Y_i that the user is expected to view during segment i. We also experimented with a CNN with an LRU, but it performed worse than the RNN. Thus, we do not investigate this approach further in this paper.

IV. PERFORMANCE EVALUATION OF VIEWPORT PREDICTION

A. Performance Evaluation Methodology

For the evaluation, 51 head movement traces from [2] are used for the viewport prediction. The head movement traces start 40 s after the start of the video and are 70 s long. For the evaluation we only consider these parts of the video. To ensure that training and testing data is not correlated in the neural network, we use the head movement traces from 40 users exclusively for training and eleven users for testing. For the sake of comparability, we only used these eleven head movement traces for all evaluations.

We implemented the optimization program in Gurobi in Matlab. For the value function w_{ijs}, we chose the resolution j. This means, we focus on maximizing the resolution. To create a realistic streaming experience, we used the eight throughput traces that were recorded according to [24], concatenated them and used consecutive intervals of 100 seconds. This resulted in eleven samples, one sample for each user of the testing group, that is evaluated. We scaled down the throughput with a factor of 0.05 so that a realistic adaptation scenario is obtained.

For our experiments, we downloaded the YouTube video *2OzlksZBTiA* in eight resolutions: 144p, 240p, 360p, 480p, 720p, 1080p, 1440p, and 2160p. Using ffmpeg, we determined the segment size in Bytes and the segment duration of 5.33 s for each segment in each quality. We divided each segment into three subsegments with a length of 1.78 s. This allows for shorter deadlines to download the segments, which is necessary for a short prediction horizon in the RNN and the extrapolation. Furthermore, we partition each segment into 64 tiles (8x8) with equal bit rate to allow for spatial adaptation.

In the following, performance evaluation results are given considering this video, the head movement traces, and the bandwidth traces as presented above. A one-to-one mapping was used for the traces, i.e., the same bandwidth trace was always used with the same head movement trace.

B. Comparison of Viewport Prediction Approaches

Figure 3 gives an overview of how the five methods perform, that we compare in this study. For each user, the mean resolution of the tiles in the viewport is calculated for each approach. At the y-axis, the mean for all users is presented while we use 95% confidence intervals. The same confidence intervals are used all other figures. On the x-axis you can see the prediction window. The highest reachable resolution lies between 1080p and 1440p on average, shown by the blue line as baseline. The current YouTube approach (green) of naively downloading the whole video in the same quality can only reach an average resolution of slightly above 480p on average. If we determine a probabilistic viewport using the viewing behavior of other users (red), we can reach a resolution between 720p and 1080p on average, which is in the middle between the optimal and the current approach. The independence of the prediction window is a large advantage of this approach since it allows a large buffer. This helps to avoid stalling and keep a steady resolution if the network is unstable.
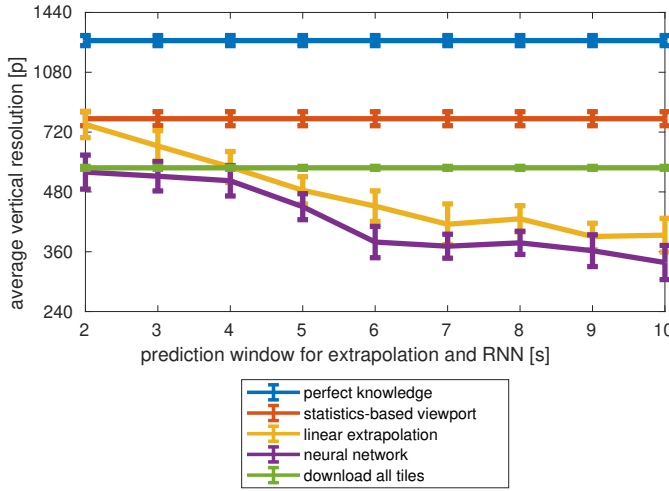
Figure 3: Impact of the prediction window size on mean quality over all users. Blue, red and green curves do not depend on prediction and are only given for comparison.

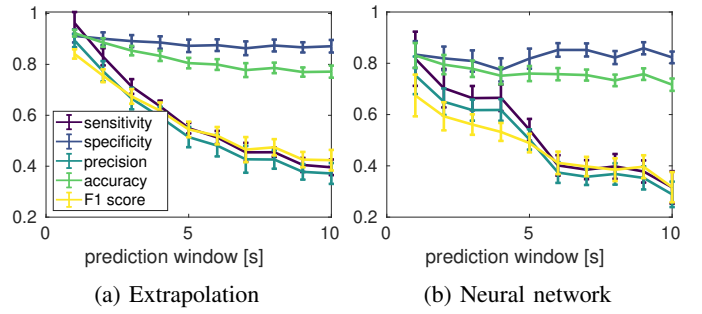
(a) Extrapolation (b) Neural network

Figure 4: Mean value of the classification functions for the neural network and the extrapolation. The value of the y-axis is described in the legend.

The neural network approach performs worst in this case. For a prediction window up to 4 s, the confidence intervals overlap with the current YouTube performance (green). Thus, no statistically significant difference can be detected. For larger prediction windows the performance becomes even worse. Comparing the linear extrapolation approach with the neural network approach, for a prediction window of 2 s, the linear extrapolation performs better. However, even in this best case, it is only on a par with the probabilistic viewport approach. But since the segment size must be smaller than the prediction window, this can lead to a lot of overhead as concluded in [18]. For larger prediction windows also the performance drops, and no statistically significant difference between the linear extrapolation approach and the neural network approach can be seen based on the 95 % confidence interval.

To sum up, we conclude that the probabilistic viewport prediction performs best if no perfect knowledge is available. Only for a small prediction window of 2 s, the linear extrapolation is on a par with the probabilistic viewport approach, but for larger prediction windows the performance of linear extrapolation and neural network decrease even below the naive baseline (green). Since the playback resolution is only one factor for a QoE analysis, in the following we focus on analyzing quality variations during playback resulting in quality changes.

The average viewport approach (red) and downloading all tiles in the same quality (green) show the smallest variance of all methods. The two user-centric prediction methods, extrapolation (yellow) and RNN (purple), have a high variance which indicates many quality changes.

C. Live Viewport Prediction Performance

For a more detailed analysis of the live viewport prediction performance, different prediction window sizes are compared in this section. For the RNN and the linear extrapolation, we regard a viewed tile as a positive and tile that is not

viewed as a negative. In Figure 4a and Figure 4b, we see their classification functions. In both cases, we get a high accuracy and a high specificity, even for larger prediction windows, because during each segment, only a small subset of all tiles is viewed. However, the sensitivity is very low for large prediction windows. This means that the viewed tiles are not correctly identified. Therefore, we also receive a low F1 score. The fact that our RNN is worse than the extrapolation, indicates that this method should not be applied with such a small data set.

D. Discussion

From the results of our performance evaluation, it can be seen that the probabilistic viewport approach is the most successful. On average, the quality of the video increases by almost one layer while the mean variance of the quality stays the same. However, in the optimal case, we could still improve the average quality by more than one quality layer. Nevertheless, the probabilistic approach has the great advantage that it is not time constrained. This means that we can have a large video buffer, whereas user-centric live prediction only allows for a small buffer since we can only predict a few seconds in advance. A large buffer helps to reduce the frequency of stalling and quality switches. A disadvantage of this approach is that we need to acquire the viewport of many users, in order to apply it. A large enough group of malevolent users may sabotage a video's quality by viewing 'uninteresting' tiles so that other viewers would only download these tiles in high quality.

The extrapolation approach with a prediction window of up to two or three seconds performed better than the current naive approach of YouTube to download all tiles in the same quality. However, it has the disadvantage that the buffer is very low, and a short segment size is required. On the positive side, no additional user information is required. This makes it the preferred method for 360° live streaming scenarios.

While the RNN did not reach good results, we believe that there is potential for applying neural networks in viewport prediction and further research with more samples may allow for better results.

V. Conclusion

While the consumption of virtual reality videos is growing at an explosive rate, the adaptive streaming technologies that are currently employed are still inefficient. While several approaches have recently emerged in literature, an optimal solution to this problem does not yet exist. Thus, in this paper, we presented a linear program to compute the optimal solution for the adaptation in 360° video streaming under given network conditions.

As in a practical application the viewport of the user is not known, we also study optimal adaptation in case of viewport prediction. Therefore, we considered three different viewport prediction methods and evaluated their performance with the help of modified linear programs. Our results showed that there is potential to optimize the quality in 360° by a significant margin compared to a naive approach, which just downloads all segments and tiles in the same quality. We recommend using a statistics-based adaptation when user viewports are available, since this approach leads to the highest average quality and the lowest variance. If live prediction is required, e.g., in live streaming where no historical statistics about the users' viewports are available, linear extrapolation performed best although it only worked well for a prediction window of two seconds. The studied RNN-based approach, on the other hand, did not give a sufficiently good streaming quality, and has to be improved in future work. In future work, it also has to be investigated how the QoE is affected when viewing tiles in different resolutions to better understand the implications of applying adaptive streaming technologies in 360° videos in virtual reality applications.

Acknowledgements

References

[1] X. Corbillon, F. De Simone, and G. Simon. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 199–204. ACM, 2017.

[2] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski. Viewport-adaptive navigable 360-degree video delivery. In *International Conference on Communications (ICC)*, pages 1–7. IEEE, 2017.

[3] R. I. T. da Costa Filho, M. C. Luizelli, M. T. Vega, J. van der Hooft, S. Petrangeli, T. Wauters, F. De Turck, and L. P. Gaspary. Predicting the performance of virtual reality video streaming in mobile networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 270–283. ACM, 2018.

[4] eMarketer. Virtual and augmented reality users 2019. https://www.emarketer.com/content/virtual-and-augmented-reality-users-2019, 2019. Accessed: 2019-08-16.

[5] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 67–72. ACM, 2017.

[6] X. Feng, V. Swaminathan, and S. Wei. Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):43, 2019.

[7] M. Graf, C. Timmerer, and C. Mueller. Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP: Design, implementation, and evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 261–271. ACM, 2017.

[8] J. Heyse, M. T. Vega, F. De Backere, and F. De Turck. Contextual bandit learning-based viewport prediction for 360 video. *IEEE Virtual Reality (VR)*, 2019.

[9] M. Hosseini and V. Swaminathan. Adaptive 360 VR video streaming: Divide and conquer. In *International Symposium on Multimedia (ISM)*, pages 107–110. IEEE, 2016.

[10] T. Hossfeld, M. Seufert, C. Sieber, T. Zinner, and P. Tran-Gia. Identifying QoE optimal adaptation of HTTP adaptive streaming based on subjective studies. *Computer Networks*, 81:320–332, 2015.

[11] M. Inc. x265 HEVC encoder / H.265 video codec. http://x265.org/, 2013. Accessed: 2019-04-30.

[12] C. Li, W. Zhang, Y. Liu, and Y. Wang. Very long term field of view prediction for 360-degree video streaming. *arXiv preprint arXiv:1902.01439*, 2019.

[13] K. Miller, N. Corda, S. Argyropoulos, A. Raake, and A. Wolisz. Optimal adaptation trajectories for block-request adaptive video streaming. In *Packet Video Workshop (PV), 20th International*, pages 1–8. IEEE, 2013.

[14] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou. An overview of tiles in hevc. *IEEE journal of selected topics in signal processing*, 7(6):969–977, 2013.

[15] C. Moldovan, L. Skorin-Kapov, P. E. Heegaard, and T. Hoßfeld. Optimal fairness and quality in video streaming with multiple users. In *30th International Teletraffic Congress (ITC 30)*. IEEE, 2018.

[16] S. Petrangeli, G. Simon, and V. Swaminathan. Trajectory-based viewport prediction for 360-degree virtual reality videos. In *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 157–160. IEEE, 2018.

[17] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. An HTTP/2-based adaptive streaming framework for 360 virtual reality videos. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 306–314. ACM, 2017.

[18] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 1–6. ACM, 2016.

[19] O. Research. Global virtual reality market-segmented by product type (hand-held devices, gesture-controlled devices, HMD), VR technology, applications, and region-growth, trends, and forecast (2018-2023). https://www.reuters.com/brandfeatures/venture-capital/article?id=40919, 2018. Accessed: 2019-08-16.

[20] SANDVINE. Mobile internet phenomena report. Tech report, 2019.

[21] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia. A Survey on Quality of Experience of HTTP Adaptive Streaming. *Communications Surveys & Tutorials*, 17(1):469–492, 2015.

[22] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen. A survey of emerging concepts and challenges for QoE management of multimedia services. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(2s):29, 2018.

[23] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):43–57, 2019.

[24] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[25] J. van der Hooft, M. T. Vega, S. Petrangeli, T. Wauters, and F. De Turck. Optimizing adaptive tile-based virtual reality video streaming. In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 381–387. IEEE, 2019.

[26] 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 25th ACM international conference on Multimedia*, pages 315–323. ACM, 2017.

[27] M. B. Yahia, Y. Le Louedec, G. Simon, and L. Nuaymi. Http/2-based streaming solutions for tiled omnidirectional videos. In *International Symposium on Multimedia (ISM)*, pages 89–96. IEEE, 2018.

[28] Q. Yang, J. Zou, K. Tang, C. Li, and H. Xiong. Single and sequential viewports prediction for 360-degree video streaming. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2019.

[29] C. Zhou, Z. Li, and Y. Liu. A measurement study of oculus 360 degree video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pages 27–37. ACM, 2017.