Tile-based 360-Degree Video Streaming

Haotian Guo

Abstract

360○ videos provide an immersive experience to users, but requires requires ultra-high bandwidth to stream compared to regular videos. Tile-based streaming is the most representative method to improve bandwidth efficiency for adaptive 360° video streaming. However, for the tile-based video, unstable viewport prediction performance may leads to content-miss of the user's viewport and damage the viewing experience. Moreover, dividing the video into multiple tiles implies lower encoding efficiency leading to more required bandwidth for video transmission. Based on our observations, we propose and implement an adaptive 360° video streaming scheme that utilizes two video streams: background stream and foreground stream. The background stream downloads low-resolution full-frame video to ensure the coverage for user's viewport and foreground stream downloads high-resolution tiles to ensure high viewing experience. We deploy the two-streams scheme using real-world mobile devices and carefully evaluate the advantage of that.

Keywords: 360° videos, tile-based streaming.

1 Introduction

360° videos, also referred as immersive videos, are always regarded as an important component for virtual reality (VR) application. According to the report [1] by GRAND VIEW RESEARCH about the virtual reality market, the global virtual reality market size reached $15.81 billion in 2020 and is expected to expand at a compound annual growth rate (CAGR) of 18.0% to 2028. However, due to the panoramic nature, 360° videos are much larger (4x to 6x [5]) than conventional videos under the same perceived quality. By leveraging the fact that users have limited viewing regions (referred to as viewports), tile-based streaming [10] is proposed to improve bandwidth efficiency, which spatially divides the panoramic video into independently decodable units (called tiles) and selectively transmits a subset of these non-overlapping tiles in the user viewports.

However, in the tile-based video streaming, the selection of downloaded tiles and their bitrates is sensitive to the prediction accuracy, but the user behaviors are dynamic during the video playback. We check the viewport prediction performance for different video types. We classify the videos based on the shooting environment (indoor or outdoor) and the camera status (static or moving). As shown in Figure 1, for all the four algorithms, the average prediction errors for the videos are at least 9 the camera-static videos. Which shows that the viewport prediction algorithms do not have stable performance.

Moreover, when the video is divided into more tiles, the video encode efficiency will be lower than smaller one. We plot the video size of all video tiles under different tiling settings and plot the results in Figure 2, note

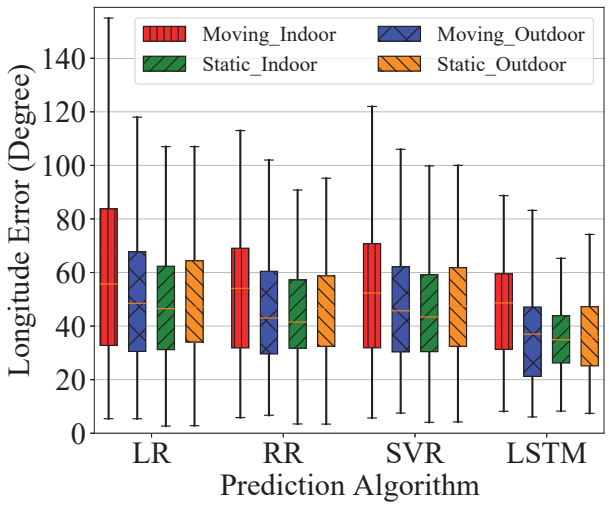

Figure 1. Viewport Prediction Performance with Multiple Algorithms and Video Types

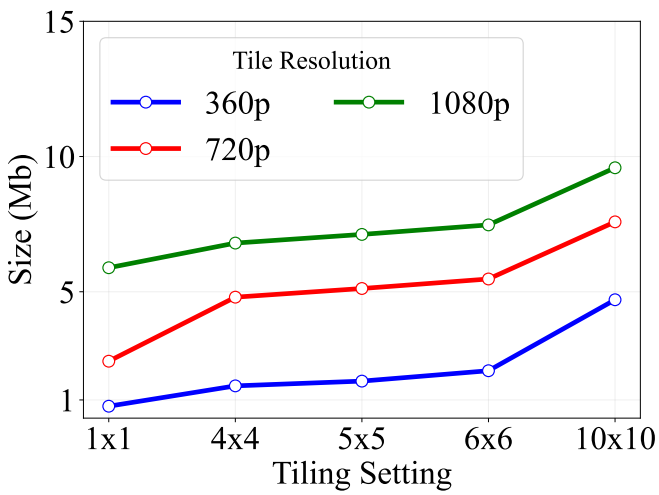

Figure 2. Encoding Efficiency with Different Tiling Settings

that the tiling setting is "1x1" indicates that the video is not divided into tiles. It is obvious that, as the number of tiles increases, the size of video is also increasing, which implies lower encoding efficiency because the reference area (the redundant information that can be compressed) between and within frames is reduced, e.g., motion vectors that reference the best block matches can be cut by tile boundaries.

Based on above two observations, user behavior is very dynamic and rarely predictable, and dividing the video into multiple tiles will reduce encoding efficiency. Therefore, in order to avoid the content-miss of the user's viewport and provide users with a high-quality viewing experience, we proposed and deployed two-streams based 360° video streaming system. First, we adopt the background stream that transmit the low-resolution full-frame video to ensure the coverage for the user's viewport.[1] Second, we adopt the foreground stream that transmit the high-resolution video tiles to update the video quality within the user's viewport.

The rest of the paper is organized as follows. We first introduce the tools and components to deploy the system in Section 2. We next evaluate the performance of the system in Section 3. The Section 4 introduce the related works for tile-based 360° video streaming and and concludes the paper in Section 5.

2 System design and implementation

In this section, we detailed introduce our design and implement of tile-based 360° video streaming system for server side and client side.

2.1 System overview

The video can be regarded as downloading with two streams: background stream that transmitting low-resolution entire video to ensure the coverage of user's viewport, foreground stream that transmitting high-resolution video tiles to ensure the QoE of users.

[1]Note that, low-resolution full-frame video consumes only slightly bandwidth compared high-quality tiles.

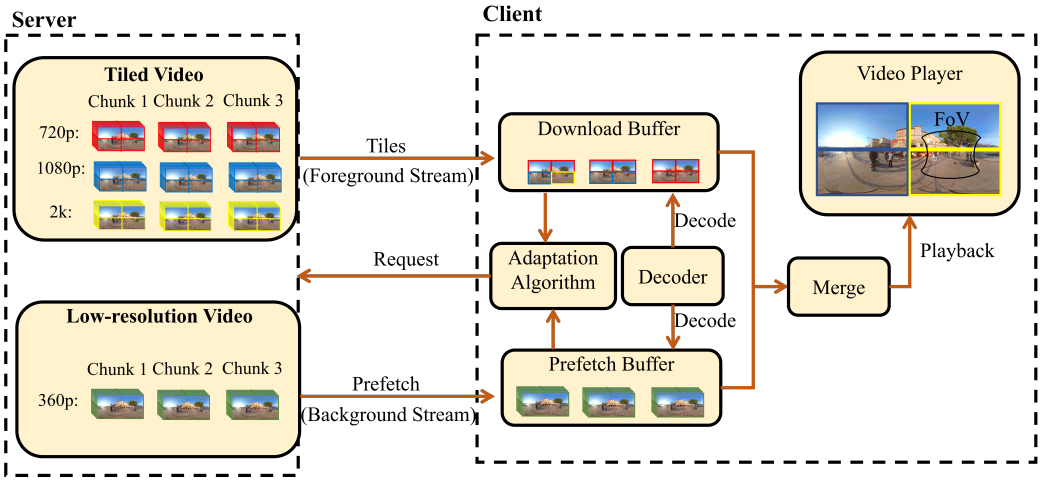

Figure 3. Workflow

We draw the system workflow in Figure 3. The server stores tiled video chunks in various resolutions and entire video chunk in lowest (here is 360p) resolution. The client side includes two buffers that the download buffer contains the video tiles that are request according to the adaptation algorithm, and the prefetch buffer stores the low-resolution background video streams. The videos in these two buffers are decoded and merged (high-resolution tiles are overlaid directly on low-resolution video for better QoE) into the mixed 360° videos. Finally the merged video is played fro the user.

Since the low-resolution video is encoded in low bitrate, the download of background streams only consumes little bandwidth. Thus we set the max occupancy of prefetch buffer to 10 seconds, and empirically set the max occupancy of download buffer to 3 seconds.

2.2 Implement of server side

The main purposes of server side are storing the videos, listing the request of client side, and responding the corresponding sides/tiles.

2.2.1 Preprocessing for videos

We download the 360° videos from open dataset [2]. We then use ffmpeg [3] to downscaling and encoding the source videos with H.264 codec for adaptive streaming. Finally, for videos of each resolution, we use GPAC [4] to divide the videos into multiple tiles and generate the MPD file for supporting tile-based adaptive streaming in DASH. The storage path for the DASH file of each high-resolution video tile is organized as:

$$\text{"}\$videoName/\$tilingSetting/\$videoResolution/\$tileIndex - \$chunkIndex.m4s\text{"}$$

. The storage path for the DASH file of each low-resolution video is organized as specific case for tiled video with *$tilingSetting* = "1x1" and *$tileIndex* = "0".

[2] https://vimeo.com/215985064
[3] https://github.com/FFmpeg/FFmpeg
[4] https://github.com/gpac

2.2.2 HTTP-DASH

We employs Flask [5], a lightweight web framework, in combination with Gunicorn [6], a high-performance WSGI server, to run the HTTP-DASH [7] protocol.

This setup efficiently listens to and handles client requests. Flask, provides the necessary tools for quick and effective development. Gunicorn, on the other hand, enhances our server's capability to manage concurrent requests reliably. It listens on a designated port, processes incoming HTTP DASH requests, and delegates these requests to Flask for handling. This streamlined combination of Flask and Gunicorn ensures fast, responsive, and robust management of client interactions in our server environment.

2.3 Implement of client side

We implement the client side using the android-SDK with JAVA.

- **Network protocol module.** The network protocol module primarily handles generating HTTP requests that conform to the format of HTTP-DASH, and sends them to the server to request tiles with different resolution based on the results of the adaptive algorithm. The request specifies the desired download bitrate for each tiles; a bitrate of zero indicates that there is no need to download a high-resolution version of that particular tile. Moreover, this module periodically requests the low-resolution background video to avoid the tile-miss.

- **Viewport Prediction Module.** The viewport prediction module primarily focuses on predicting the user's future viewport position based on user's historical trajectory. Considering that downloading high-resolution tiles consumes significant bandwidth, it is reasonable to download only the tiles within the user's viewport with high-resolution.

- **Rate Adaptation Module.** The rate adaptation module primarily makes the download bitrate for each tile, based on the user's viewport position, network bandwidth, and buffer occupancy. Given the fluctuating nature of network bandwidth, which can sometimes be very low, downloading videos at a fixed bitrate could lead to buffer drain or a lower quality of user experience.

- **Buffer Management Module.** The buffer management module primarily oversees the downloading and playing of video content, ensuring that video playback occurs in a sequential order.

- **Video Decoding Module.** The video decoding module primarily handles the decoding of downloaded videos/tiles and then feeds the decoded video into the corresponding playback/prefetch buffer.

- **Video Rendering Module.** The video rendering module primarily focuses on merging low-resolution video with high-resolution tiles to render the scane to played to the user.

- **Video Playing Module.** The video playing module primarily interacts with the users, capturing and recording their viewport movement and relaying it to the video rendering module to ensure that the played video content is truth for user's viewport.

[5] https://github.com/pallets/flask
[6] https://github.com/benoitc/gunicorn
[7] https://github.com/plotly/dash-docs

3 Results and analysis

3.1 Experiment setup

We consider five resolution levels that the videos can be encoded in : (1) 360p (1Mbps), (2) 720p (5Mbps), (3) 1080p (8Mbps), (4) 2k (16Mbps), (5) 4k (40Mbps); which is recommended of upload encoding bitrates from YouTube[8]. The 360p is set to low-resolution video that is not divided into tiles, and videos with other resolutions are divided into 5x5 tiles.

3.2 Experiment result

We replay the bandwidth trace in dataset [8] to check the performance of our system with Flare [5]. We use the average bitrate of the video and rebuffering ratio while streaming the 360° videos as the performance evaluation metrics.

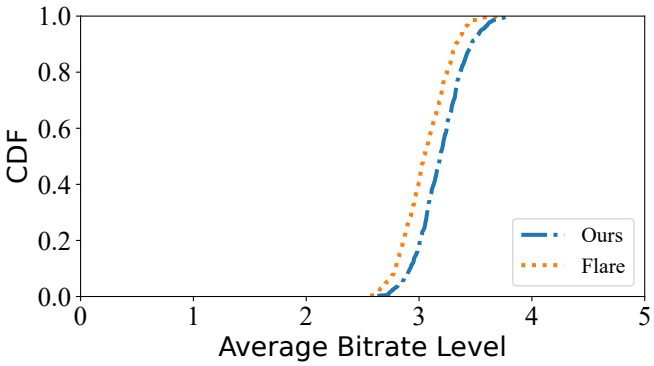

Figure 4. Average Video Bitrate Level

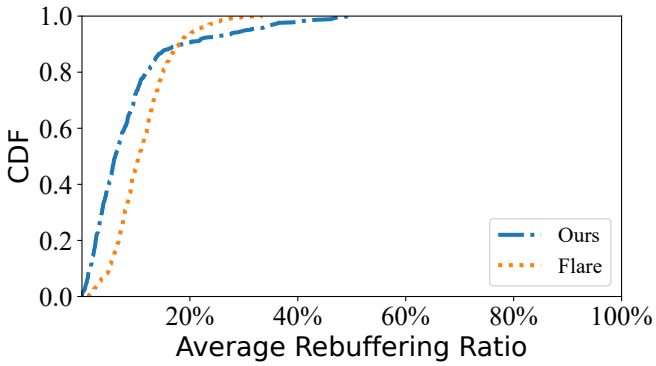

Figure 5. Average Rebuffering Ratio

The result of average video quality is plotted in Figure 4, our method achieved a 5% average bitrate level improvement over Flare. Moreover, the Figure 5 plots the average rebuffering ratio, which shows our proposed method outperforms Flare in most cases (80%). Experiment results illustrate that our proposed method can slightly improve video quality and reduce the frequency of video rebuffering.

4 Related works for tile-based streaming

To reduce the excessive bandwidth consumption for streaming 360° videos, tile-based steaming is proposed and widely adopted [12] [4] [6] [9]. By identifying the tiles in the user viewport, the client can assign higher bitrates to the tiles that matter the most for user experience, which implies that tile-based streaming requires certain knowledge about user viewing behaviors. The existing works design and study different schemes to work with or on top of tile-based streaming to achieve better performance and efficiency. PARIMA [2] adopts a fast and efficient online viewport prediction method, and allocates bitrates for tiles using a pyramid-based scheme to improve user experience. DRL360 [14] leverages a deep reinforcement learning (RL) framework to predict bandwidth and viewport, and implements an LSTM-based actor-critic model to allocates the tile bitrates. Pano [3] builds a perceptive quality model considering three specific influencing factors for 360° videos,

[8]https://support.google.com/youtube/answer/1722171

based on which a variable sized tiling scheme is proposed to balance the perceived quality and the video encoding efficiency. TBRA [13] proposes an adaptive tiling scheme to adjust the tiling setting based on viewport prediction performance to strike the best trade-off between user experience and transmission efficiency. Therefore, tile-based streaming has been demonstrated to have great flexibility to be customized for comprehensive streaming frameworks. Muster [11] proposed a multi-source video streaming framework that jointly consider the server selection and rate adaptation for tile-based video streaming in cloud native 5G networks. Tang [7] et al. propose an online adaptive bitrate algorithm that learns the heterogeneity of user FoV and bandwidth to enhance the user's QoE for tile-based video streaming.

5 Conclusion

In this work, we proposed a two-streams based 360° video streaming method to ensure the integrity of user viewport content and improve video quality of the user's viewport.

References

[1] Virtual reality & size, share and trends analysis report by technology (semi & fully immersive, non-immersive), by device (hmd, gtd), by component (hardware, software), by application, and segment forecasts, 2021 - 2028. https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market, Accessed: 2021.

[2] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. Parima: Viewport adaptive 360-degree video streaming. In *Proceedings of the Web Conference 2021*, pages 2379–2391, 2021.

[3] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 394–407. 2019.

[4] Duc V Nguyen, Huyen TT Tran, Anh T Pham, and Truong Cong Thang. An optimal tile-based approach for viewport-adaptive 360-degree video streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):29–42, 2019.

[5] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 99–114, 2018.

[6] Jangwoo Son and Eun-Seok Ryu. Tile-based 360-degree video streaming for mobile virtual reality in cyber physical system. *Computers & Electrical Engineering*, 72:361–368, 2018.

[7] Ming Tang and Vincent W.S. Wong. Online bitrate selection for viewport adaptive 360-degree video streaming. *IEEE Transactions on Mobile Computing*, 21(7):2506–2517, 2022.

[8] Jeroen Van Der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. Http/2-based adaptive streaming of hevc video over 4g/lte networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[9] Xuekai Wei, Mingliang Zhou, Sam Kwong, Hui Yuan, and Weijia Jia. A hybrid control scheme for 360-degree dynamic adaptive video streaming over mobile devices. *IEEE Transactions on Mobile Computing*, 21(10):3428–3442, 2022.

[10] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming 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, 2017.

[11] Xinjing Yuan, Lingjun Pu, Jianxin Shi, Qianyun Gong, and Jingdong Xu. Muster: Multi-source streaming for tile-based 360° videos within cloud native 5g networks. *IEEE Transactions on Mobile Computing*, 22(11):6616–6632, 2023.

[12] Alireza Zare, Alireza Aminlou, Miska M Hannuksela, and Moncef Gabbouj. Hevc-compliant tile-based streaming of panoramic video for virtual reality applications. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 601–605, 2016.

[13] Lei Zhang, Yanyan Suo, Ximing Wu, Feng Wang, Yuchi Chen, Laizhong Cui, Jiangchuan Liu, and Zhong Ming. Tbra: Tiling and bitrate adaptation for mobile 360-degree video streaming. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4007–4015, 2021.

[14] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. Drl360: 360-degree video streaming with deep reinforcement learning. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1252–1260. IEEE, 2019.