Tile rate selection for 360° Video Streaming to Head-Mounted Displays

Luca Attanasio, Federico Chiariotti, Andrea Zanella
Department of Information Engineering, University of Padova – Via Gradenigo, 6/b, 35131 Padova, Italy
Email: {attanasiol,chiariot,zanella}@dei.unipd.it

***Abstract*—As a fundamental component of virtual reality (VR) technology, 360-degree videos on Head Mounted Displays (HMDs) provide users with an immersive experience as well as a panoramic view, allowing them to freely control their viewing direction and look around during video playback. Nevertheless, only a visible portion of this immersive spherical 360 video is displayed on the HMD. Thus, fetching the entire raw video frame requires intensive bandwidth. In this paper, an analysis of the problem of optimizing 360 video delivery is performed, using a tile rate selection algorithm. We first introduce tiled streaming and then propose a streaming system model based on neural networks that performs tile rate selection after computing the probability of viewing the 360 videos' visible portion based on head movement prediction. Using viewing data collected from a public users dataset, we trained a neural network (NN) that achieves great efficiency, based on trace-driven simulations.**

I. Introduction

Recent breakthroughs in omni-directional capturing systems and interactive display systems as well as in 3D immersion and 360 degree VR video applications have contributed to 3D media popularity over the past few years. Virtual reality (VR) technologies are being developed steadily and their market is estimated to be worth more than $70 billion by 2020 [1]. In current VR video delivery systems, a headset, which is a Head-Mounted Device (HMD), provides 360 degree video content for the HMD wearer, supplying separate images for each eye. Current consumer-grade HDM have a fixed Field of View (FoV). Head movements can be predicted in real time, on the basis of past user's head positions [2] [3]. VR headsets provide the user with an immersive experience, giving a sense of depth in every direction, since spherical videos are mapped into 3D geometry. The most advanced HMDs provide stereo sound and head motion tracking sensors.

Despite the promising nature of 360 VR immersive spherical videos, existing 360 VR video applications suffer from low resolution restrictions compared to their 2D counterparts when streaming content over the Internet. HMDs require at least 4K as a minimum resolution to efficiently deliver videos, thus being intensively bandwidth demanding and involving waste of both processing power and storage space.

When transmitting a high resolution 360 video, the goal is to optimize the viewer *Quality Of Experience (QoE)*. In this paper, we refer to the QoE as composed of three metrics to be perfected: lowering the number of rebuffering events, decreasing the overall quality variations and maximizing the overall quality.

In order to efficiently transmit a 360-degree video to bandwidth-limited devices, some challenges must be tackled. Although rebuffering events are not analyzed in this paper, they are probably the most annoying event likely to happen during a streaming session [4]. Rebuffering events occur when the video gets stuck while reproducing it, because the user's buffer is empty. An additional drawback is accounted for by quality variations, referred to as variations between consecutive segments in the viewport when reproducing a 360° video. Both events bear a negative impact on the viewer's QoE.

In cellular networks, achieving high QoE represents a compelling issue since 360 video streaming requires huge bandwidth demands, metered link, fluctuating throughput, and devices consume high radio energy [3] [5].

To optimize the QoE in 360 video delivery so as to improve the user's viewing experience, it is possible to stream only spatial sub-parts of a video on display. The approach explained is used in this paper. In case of traditional videos, in order to support simultaneous download and playback, a video is temporally segmented into chunks or byte ranges. In addition, to support downloading of a sub-part of a video chunk, the video needs to be spatially segmented. This method of producing spatial sub-parts is called *Spatial Representation Description (SRD)* and it has been incorporated in the MPEG DASH standard [6]. This enables the DASH client to select and retrieve only video streams with resolutions that are relevant, using adaptive 360° video streaming algorithms [3] [2] [7] [8] [9]. The relevant spatial segments can be obtained by predicting the wearer head movements by HMD orientations or by inspecting image saliency maps and motion maps using neural networks [2]. In this paper, we follow the approach of implementing a NN, using only HMD orientation as a feature, since retrieving saliency maps and motion maps are computationally demanding.

To train the NN for HDM orientation prediction, a public dataset is used [10]. In this dataset, to avoid any possible orientation tracking problems such as *gimbal lock*, a rotational system known as a quaternion is used. Furthermore, it is possible to switch between quaternions and Euler angles, characterized by three angles: pitch, way, roll [11], corresponding to rotating along the x,y and z axes.

As for SRD, the technique used consists in splitting the video into tiles, which refer to a spatial partitioning of the video. Tiling allows DASH algorithms to deliver different spatial portions of the video at different quality and bitrate. The tiles are not all watched by the user at the same time but only those which fall into his/her viewport, i.e. the observable region by the user denoted by two angles: α, which is the horizontal margin and β, which is the vertical margin as shown

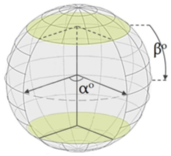

Figure 1: Sphere model of 3D space [9].

in Figure 1. In this paper, the value of $\alpha = 110^\circ$ is set, since a public dataset is used [10] in which the FoV of the HMD (Razer OSVR HDK 2) has an angle of 110°. In addition, $\beta = 45^\circ$ is set. Hence, some tiles can be streamed at lower quality to reduce bandwidth consumption up to 80% [3]. In order to fetch the correct tiles, different projection methods can be used from the 3D space, which can be modeled as a sphere, to the 2D space. The most relevant methods studied in literature are: equirectangular projection and cubic projection.

An undeveloped research field, in order to advance the state of art in adaptive 360 VR video streaming, is to understand how to distribute the predicted bandwidth to each tile.

In this paper, we propose a solution to assign the best quality to each tile by maximizing the user QoE so as to prevent bandwidth waste. This is achieved by calculating the probability of viewing a spatial-segment of the video by the HMD wearer. In addition, in order to distinguish between quality levels of each tile, we resort to the structural similarity (SSIM) index.

II. Related Work

In this section, we present relevant works from the community, focusing on models for adaptive-streaming, projection methods and the use of HTTP/2 to improve streaming flexibility.

A. Dynamic Adaptive streaming

The mainstream approach, also adopted in this paper, relies on a scheme based on head movement prediction to deliver portions of the video [9]. Fetching the entire raw video frame wastes bandwidth (more than 80%). As a result, prediction of the head movements needs to be perfected. Accurate prediction for head movements (more than %90) can be obtained by linear regression or weighted linear regression [3]. Nevertheless, this study involves a limited dataset of users. Another work [2] proposes a *Convolutional Neural Network (CNN)* model to calculate the viewing probability of the next frames, considering both sensor-related features, such as HDM orientation and content-related features such as saliency maps and motion maps as input. This approach is more bandwidth-efficient (22-36%) and reduces the initial buffering time (up to %43). Furthermore, a *tile rate selector* performs rate allocation among video tiles although its behaviour has not been fully explained.

B. Projection methods

Different projection methods can be adopted to store a 360° video in the memory of a device. The main advantages of cube projection and equirectangular projection used respectively in FaceBook and YouTube are studied in [3]. Equirectangular projection, which is the most popular method, is similar to projections employed to display maps of the world. Nevertheless, its main disadvantage is stretching the poles of the sphere, encoding them with more pixels than the equator, potentially wasting space and distorting images. Another type of projection is cubic projection with a cube being built around the sphere. This is more efficient in terms of bandwidth saving and it better represents the 3D space, thus improving the QoE [12] [13]. Another encoding method to better the QoE proposed in literature is *scalable video codec (SVC)* encoding [14].

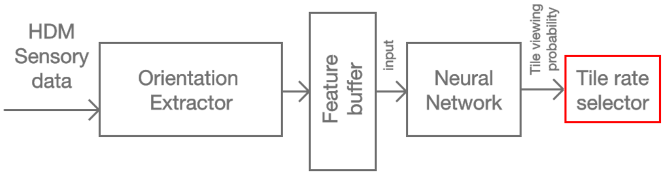

Figure 2: Bandwidth

C. HTTP/2 integration

Adaptive Streaming algorithms for 360 videos available in literature consider replacing traditional HTTP/1.1 with HTTP/2. HTTP/2 *server push* feature delivers multiple video chunks at a time. This characteristic helps reduce the overhead requests due to multiple tiles being requested for each segment from the client. The client, as a matter of fact, makes only one request to the server, specifying the qualities of the video tiles. The server pushes tiles back to back reducing RTT as tiles are sent all together [7]. Other features of HTTP/2 consist in stream termination, achieved by sending a signal from the client or server, and stream priority to efficiently organize tiles delivery [8] by prioritizing the tiles with a higher priority weight.

III. System Model

Figure 2 presents our proposed architecture of a 360 streaming server, whereby we focus on the software components related to the *tile rate selector*. As input of the NN, we consider one sensor-related feature: the orientation from the HMD. We briefly describe the components of the system model as follows.

1) **Orientation extractor**: derives the viewer orientation data, including quaternions from HMD sensors.
2) **Feature buffer**: stores the features later used as input for the NN.
3) **Neural network**: considers as input the orientation from the HDM and outputs future tile viewing probabilities for each tile.
4) **Tile rate selector**: performs rate allocation among video tiles.

We consider the probability prediction on tiles as a multi-label classification problem and we have implemented the NN using Scikit-Learn [15]. In this paper, the best parameters found in [2] are used. In particular, the NN is made up of 512 neurons. To train the NN, the public dataset traces from [10] are used, with a total of 59 users. 5 head orientations are used to compute the future position in a near future of 0.5s, which is the optimal number of seconds for prediction [3]. Among the traces, 80% of them are used for training and 20% for testing. Training and

No. Neu.	Layers	Training Error	Testing Error
512	1	21.9184%	29.5887%

Table I: Neural network performances on a 0.5s prediction in the future.

testing is only performed on the *Diving video* [16], in which the camera is slowly moving and there is no clear horizon. No main focus of attention is expected within the sphere of vision. The NN is trained using Stochastic Gradient Descent with learning rate of 10^{-2}. The probability of viewing a tile is the output of the NN, between 0 and 1 included. If the probability of viewing the tile is high, then the user is expected to view that tile. To increase the QoE of the user, based on the probability of viewing the tile, the best bandwidth is allocated for each tile. The neural network is expected to output a higher probability for viewport tiles. We briefly explain how the viewport tiles are obtained, in order to train the NN in a supervised learning approach. The neural network performances are reported in Table I

To calculate the viewport tiles, given the HDM orientation in quaternions from the *feature buffer*, we follow this approach:

1) Switch from quaternions representation to Euler angles (yaw, pitch, roll) using equations given in [11].
2) Project yaw and pitch angles on a 2D plane to calculate the viewport.
3) Project the viewport on a 2D space to obtain the coordinates of the viewport with given video resolution. This is achieved by using *equirectangular projection* equations.
4) Fetch the viewport tiles and assign them a value of 1. Instead, tiles which are not in the viewport are assigned a value of 0.

With yaw between $-180°$ and $180°$ and pitch between $-90°$ and $90°$, the *equirectangular equations* are described by:

$$x = \frac{yaw + 180°}{360°} w \qquad y = \frac{(90° - pitch)}{180°} h \tag{1}$$

where $h * w$ is the resolution of the video: w is the width and h the height. An example of projection of yaw and pitch on a video is given by Figure 3. The red rectangle is the FoV of the user at a certain time. The colored points indicate the FoV margins, which move while the user is watching the video. Each chunk of the video is split into $m = 16$ tiles. Each tile region is a rectangle delimited by blue lines.

In this paper, we mainly focus on the **tile rate selector**. From the latest stored feature, the probabilities of the next tiles viewed by the user can be predicted. We refer to these probabilities as $p_{i,j}$, where $i \in [1, m]$ refers to the i-th tile the user is watching and $j \in [1, n]$ to the relative segment. Each tile is encoded at different representation levels, each at a different quality $q_{i,j}$ referring to the structural similarity (SSIM) index. The goal of the *tile rate selector* is to maximize the following rule:

$$\arg\max \sum_{j=1}^{n} \sum_{i=1}^{m} q_{i,j} p_{i,j} \leq \sum_{j=1}^{n} \sum_{i=1}^{m} q'_{i,j} p'_{i,j} \tag{2}$$

where $p'_{i,j}$ is the true probability of viewing the i-th tile, n is the number of segments and m the number of tiles in a segment.

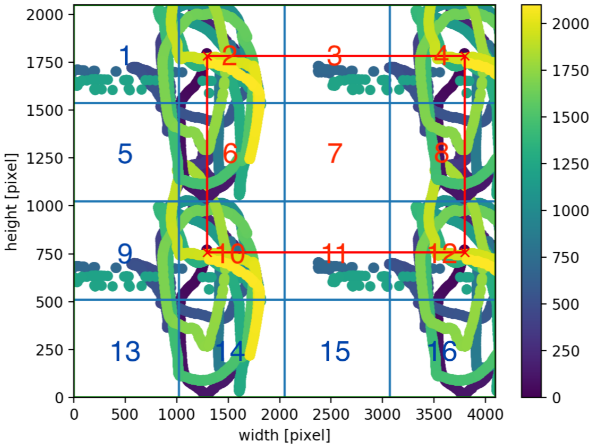

Figure 3: 2D projection of yaw and pitch on a video of 4K video resolution. The red rectangle is the FoV of the user at a certain time. The starting moving index is 0.

Since we only have $p_{i,j}$ available at runtime, we can act on the selection of $q_{i,j}$ to optimize the QoE. If b_j is the estimated bandwidth for the j-th segment, we can distribute it between tiles referring to the same segment j. This task can be achieved by calculating the cumulative probability for each segment:

$$c_j = \sum_{i=1}^{m} p_{i,j} \tag{3}$$

To assign the correct bandwidth to each tile, we use the following rule

$$b_{i,j} = \frac{p_{i,j}}{c_j} b_j \tag{4}$$

Notice that $b_j = \sum_{i=1}^{m} b_{i,j}$ and that $\frac{p_{i,j}}{c_j}$ is the percentage of bandwidth assigned to the i-th tile for the fixed j-th segment.

Finally, to assign the representation level to each tile correctly, $b_{i,j}$ can be mapped to the first smallest bandwidth $b^R_{i,j} = f(b_{i,j})$ corresponding to its representation level. We can now obtain $q_{i,j} = g(b^R_{i,j})$ as a function of the representation level's bandwidth. For example, we assume to have $\vec{b}^R = [5800, 4300, 3750, 3000, 2350, 1750, 110]$ in $kbit/s$. We assume the computed $b_{i,j}$ is $4500 kbit/s$, then the algorithm assigns $b^R_{i,j} = f(b_{i,j}) = 4300 kbit/s$ and the matching $q_{i,j}$.

IV. Results

We assume that a single user has an assigned bandwidth b_j at the j-th segment, given in Figure 4, taken from a DASH streaming session.

The same user moves his head, as shown in Figure 5, taken from the head movements dataset [10].

Since the true head movements are known, we can compute $p'_{i,j}$ correctly and the trained NN can output $p_{i,j}$ based on the previous position of the user. After selecting the corresponding representation level, computing the *tile rate selector* algorithm, we can compare the results of the NN prediction in terms of quality with the effective achievable quality, if the tile

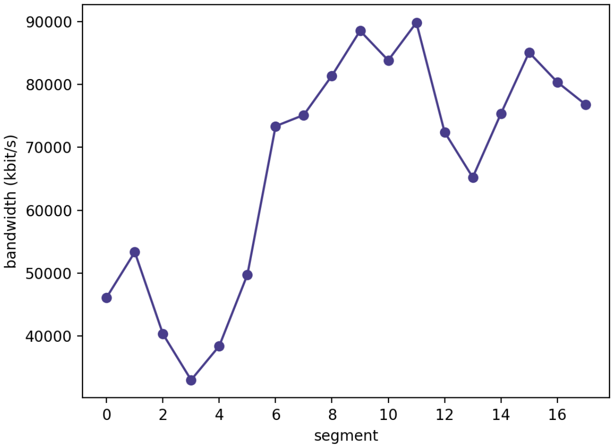

Figure 4: Bandwidth

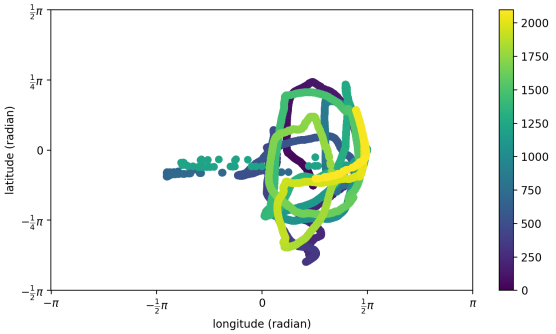

Figure 5: Head movement of a user who starts moving from 0.

prediction is perfect. The quality is referred to as q', in case of perfect achievable quality and as q as maximum achievable quality by the NN. These values are computed as:

$$q = \sum_{j=1}^{n} \sum_{i=1}^{m} q_{i,j} \leq q' = \sum_{j=1}^{n} \sum_{i=1}^{m} q'_{i,j} \tag{5}$$

In this case, $q'_{i,j}$ refers to the quality selected by the *tile rate selector* for the i-th tile at the j-th chunk for the true probability values $p'_{i,j}$. $q_{i,j}$ is the equivalent for the predicted values of $p_{i,j}$.

In Figure 6, we can evaluate how the NN can predict viewport variations, assigning $p_{i,j}$, which would be instantly detected with integer values of 1 or 0 if the future head movements $p'_{i,j}$ were known.

To evaluate the QoE of the user, we now analyze the quality and quality variations delivered in a trace-driven simulation test. In Figure 7, we can notice that the quality achieved by using our rate selector algorithm is, on average, less than the best quality achievable, which is the quality with correct viewport prediction. In addition the achieved quality has much more outliers that lower the overall quality.

Quality variations are defined for each segment, considering the segments have m tiles and the main area of interest is the viewport:

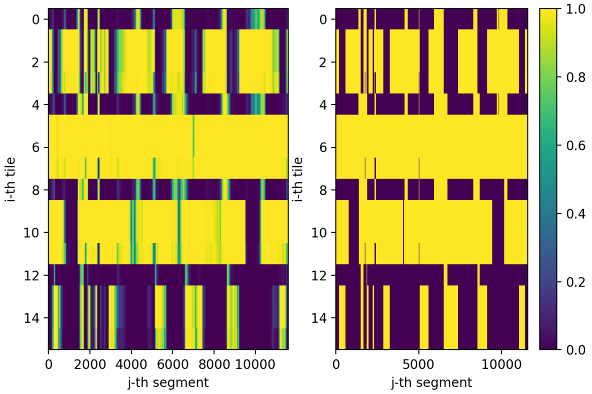

Figure 6: Viewport transition seen from tiles probability prediction $p_{i,j}$ on the left, and from optimal tiles probability $p_{i,j}$ on the right.

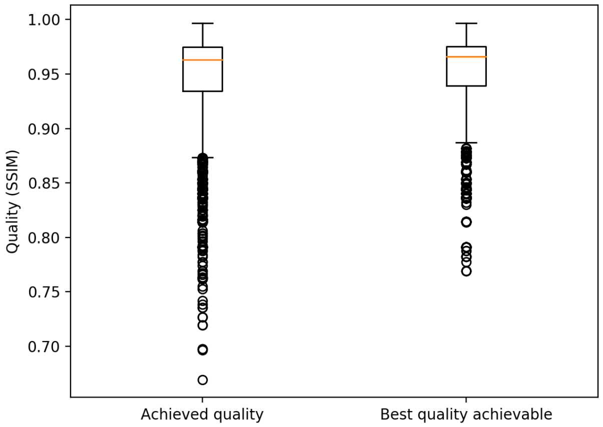

Figure 7: Viewport delivered quality vs best achievable quality with correct tile prediction.

$$\Delta q_j = \sum_{i=1}^{m} [q_{i,j} \times p'_{i,j} - q_{i,j+1} \times p'_{i,j+1}] \tag{6}$$

In Figure 8, the quality variations of the viewport are shown using boxplots. We notice that tiling the video could result in high quality variations if the *rate selector* does not perform well. In particular, if it fails to assign the bandwidth correctly to the tiles, quality variations are emphasized.

V. Conclusions

In this paper, we address the problem of rate selection for 360° video streaming to HMDs using neural networks. The novel rate selection algorithm presented, after training the neural network using sensor-related features, allows to reach great efficiency in terms of deliverable QoE. The use of real HMD orientations publicly available prove fundamental to train the neural network. Through trace-driven simulations, we found that high quality can be delivered for each tile,

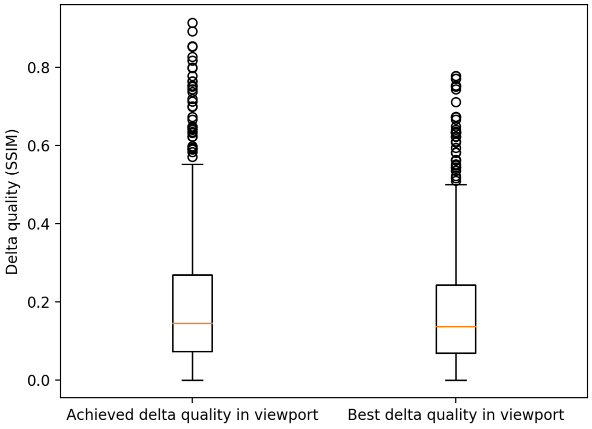

Figure 8: Viewport quality variations.

with few quality variations between segments in the viewport portions of the video, reducing bandwidth waste considerably. We acknowledge that this current work can be extended in several dimensions. For instance, more extensive simulations are needed to understand the behaviour of the neural networks to predict tiles which are further away in the future. Furthermore, training and testing data are performed on a single video. Training the neural network on other types of video could help prove that the rate selection algorithm works well in other scenarios. Last but not least, we only used equirectangular projection to predict tiles; as a result, other projection methods need to be further studied and await future development.

References

[1] “Augmented/virtual reality revenue forecast revised to hit $120 billion by 2020.” http://goo.gl/Lxf4Sy.

[2] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C.-H. Hsu, “Fixation prediction for 360 video streaming to head-mounted displays,” *Proceedings of ACM NOSSDAV 2017*, 2017.

[3] 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*, pp. 1–6, ACM, 2016.

[4] O. Oyman and S. Singh, “Quality of experience for http adaptive streaming services,” *IEEE Communications Magazine*, vol. 50, no. 4, 2015.

[5] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, “A close examination of performance and power characteristics of 4g lte networks,” in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 225–238, ACM, 2012.

[6] O. A. Niamut, E. Thomas, L. D’Acunto, C. Concolato, F. Denoual, and S. Y. Lim, “Mpeg dash srd: spatial relationship description,” in *Proceedings of the 7th International Conference on Multimedia Systems*, p. 5, ACM, 2016.

[7] 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*, pp. 306–314, ACM, 2017.

[8] M. Xiao, C. Zhou, V. Swaminathan, Y. Liu, and S. Chen, “Bas-360: Exploring spatial and temporal adaptability in 360-degree videos over http/2,” in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 953–961, IEEE, 2018.

[9] M. Hosseini and V. Swaminathan, “Adaptive 360 vr video streaming: Divide and conquer,” in *Multimedia (ISM), 2016 IEEE International Symposium on*, pp. 107–110, IEEE, 2016.

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

[11] Wikipedia, “Conversion between quaternions and Euler angles.” http://en.wikipedia.org/w/index.php?title=Conversion%20between%20quaternions%20and%20Euler%20angles&oldid=880101212, 2019.

[12] 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*, pp. 27–37, ACM, 2017.

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

[14] A. TaghaviNasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, “Adaptive 360-degree video streaming using layered video coding,” in *Virtual Reality (VR), 2017 IEEE*, pp. 347–348, IEEE, 2017.

[15] “Tile rate selection for 360 video streaming.” https://github.com/blackwiz4rd/TileRateSelector_360videos.

[16] “Scuba diving short film in 360 green island, taiwan 4k video quality.” https://www.youtube.com/watch?v=2OzlksZBTiA&t=40s.