Modular Testbed for KPI Monitoring in Omnidirectional Video Streaming Scenarios

Mario Montagud[1,2], Einar Meyerson[1], Isaac Fraile[1], Sergi Fernández[1]
[1]i2CAT Foundation, Media Internet Unit
[2]Universitat de València, Departament d'Informàtica
[1]Barcelona (Spain); [2]València (Spain)
e-mails: {mario.montagud, einar.meyerson, isaac.fraile, sergi.fernandez}@i2cat.net

***Abstract*—The streaming of high quality and omnidirectional videos in Over-the-top (OTT) environments still faces many challenges. The research community is devoting efforts on devising optimized solutions for a variety of key aspects, such as encoding efficiency, Field-of-View (FoV) based streaming, adaptive quality switching, use of network assisted elements, etc. All these initiatives share a common denominator: it is essential to monitor relevant Quality of Service (QoS) and Quality of Experience (QoE) related metrics to better understand what are the limitations, propose appropriate solutions and corroborate the obtained performance. In this context, this paper presents a modular testbed to monitor and register Key Performance Indicators (KPIs) in (omnidirectional) video streaming scenarios, making use of the increasingly adopted Dynamic Adaptive Streaming over HTTP (DASH) technology. The paper describes the different components of the testbed and provides examples of the KPI metrics that can be collected by using it. Different application contexts where the testbed can provide valuable benefits are also discussed.**

Keywords-360º video; Dynamic Streaming over HTTP (DASH); Quality of Experience (QoE); Quality of Service (QoS).

I. INTRODUCTION

The relevance of media streaming services in the current society is beyond doubt. In the last years, the research community has been devoting efforts on overcoming a variety of existing challenges, especially when considering Over-the-Top (OTT) environments, novel (high quality) media formats and heterogeneous consumption devices. Herein, HTTP Adaptive Streaming (HAS) pull-based solutions have become dominant, due to their multiple advantages compared to traditional push-based solutions [1], such as: ubiquity, scalability and cost-efficiency. Different vendors/companies have devised their own HAS solution. Examples are HTTP Live Streaming (HLS) by Apple, HTTP Dynamic Streaming (HDS) by Adobe, and Microsoft Smooth Streaming. In order to increase the chances of worldwide deployment and maximize inter-operability, an international standardized HAS solution was proposed by Moving Picture Experts Group (MPEG) / International Organization for Standardization / International Electrotechnical Commission (ISO/IEC), called MPEG Dynamic Adaptive Streaming over HTTP (DASH) [2] [3]. Since then, DASH has been adopted by many other related standards, like Hybrid Broadcast Broadband TV (HbbTV) [4], and by many popular video streaming services.

Even though the DASH specification addresses many key aspects to provide successful and efficient streaming services, such as content preparation, delivery and signaling (briefly reviewed in Section II), many others are left open for implementers and/or service providers, but also play a key role. This in particular applies for streaming of omnidirectional (aka 360º) videos, which is more challenging than traditional video streaming in terms of bandwidth and resources consumption [5]. In this context, many open challenges for which the research community is proposing advanced solutions can be highlighted, such as:

- Encoding efficiency (e.g., [5]).
- Bandwidth optimization (e.g., [6]).
- Field-of-View (FoV) based streaming (e.g., [7]).
- Delay minimization (e.g., [8]).
- Hybrid synchronized streaming (e.g., [9]).
- Dynamic quality switching strategies (e.g., [10]).
- Use of network assisted elements (e.g., [11]).

All these research initiatives share a common denominator: it is essential to monitor relevant Quality of Service (QoS) and Quality of Experience (QoE) metrics to better understand what are the limitations, propose appropriate solutions and corroborate the obtained performance (probably in comparison with benchmarking or alternative solutions). With this premise in mind, this paper presents a modular and extensible testbed to monitor and register Key Performance Indicators (KPIs) in (omnidirectional) video streaming scenarios, making use of DASH. The testbed includes server-side components for testing with different encoding, representation and segmentation strategies for 360º videos [5], and for signaling and publishing them. Most interestingly, it includes extensions to an ad-hoc 360º video player, built on top of *dash.js*, to continuously monitor and register KPIs. Dash.js [12] is a reference JavaScript client implementation for DASH, which includes an Application Program Interface (API) to obtain relevant statistics regarding the incoming audio and video streams. The presented testbed periodically collects these statistics and other related ones, and registers them via a (remote) HTTP communication with a lightweight database for their posterior analysis.

The rest of the paper is organized as follows. Section II provides some background information and reviews other relevant platforms or testbeds for QoS/QoE evaluation in

 ISBN: 978-1-61208-697-2

video streaming scenarios. Section III describes the presented testbed and details the KPI metrics that can be collected by using it. Different application contexts where the testbed can provide valuable benefits are also discussed. Section IV provides examples of results in a simple evaluation scenario. Finally, Section V concludes the paper, and provides some ideas for future work.

II. BACKGROUND & RELATED WORK

A. Background Information

The basic idea of HAS solutions, including DASH, consists of generating multiple versions (aka representations) of the media content (e.g., in different resolutions or bitrates), and divide each of these versions into a sequence of segments (aka chunks) of a short duration (e.g., from 1 to 10s). Each segment can be decoded and consumed independently of the others. In addition, an index or manifest file, called Media Presentation Description (MPD) in DASH, is created, containing the required metadata to describe the relationships between the segments, representations, and maybe between different available media assets. Both the generated contents and metadata files are stored in a conventional web server. Based on these resources, each client, by means of HTTP requests, will firstly download the manifest file and, after that, will dynamically decide which segment (of which representation) to download at each moment, based on the information contained in the manifest file and on the network (e.g., available bandwidth…) and/or end-system conditions/resources (e.g., buffer level, screen resolution, CPU load…). This process, illustrated in Figure 1, contributes to ensuring the adaptability and continuity of the media playout process.

In addition, when referring to 360º videos services, the streaming of the whole sphere in a high resolution results in an efficient approach in terms of usage of both bandwidth and computational resources. It is due to the fact that, at any moment, users do not watch the whole 360º area, but only a proportion of it, e.g. determined by the FoV of the consumption device in use (around 100º in typical Head Mounted Displays or HMDs). This has prompted the appearance of spatial segmentation strategies, which consist of dividing the 360º video in tiles (i.e., in a matrix of rows and columns), and selectively delivering them based on the current users' viewing direction [5].

The streaming of 360º videos using DASH is currently a hot research topic in order to minimize latency and bandwidth consumption, and to maximize the perceived quality based on the available resources, while enabling freedom to smoothly explore around the 360º area.

B. Related Work

As mentioned, many research efforts are being devoted on overcoming existing challenges for successfully delivering high quality media over non-managed OTT network environments. This is in particular true when streaming 360º video, and when making use of heterogeneous consumption devices, with dissimilar capabilities and/or resources. Due to this, the availability of a proper testbed to be able to collect and analyze relevant QoS and QoE related metrics becomes essential.

Previous works have presented related contributions in this context. The work in [13] presents a toolset for QoS evaluation of video streaming services, when using Real Time Protocol (RTP) and RTP Control Protocol (RTCP) [14] in simulated environments, using Network Simulator 2 (NS-2). That toolset is based on generating (text-based) traces of video files, feeding them into NS-2, and measuring network-level QoS metrics (such as throughput, delay, jitter or loss rate) for specific scenarios. In addition, based on the QoS statistics, reception video traces can be generated, from which an estimation of the received video can be reconstructed and played out. This also allows the measurement of the visual quality of the incoming video streams, by employing the most common objective quality metrics (like Peak Signal-to-Noise Ratio or PSNR) or subjective metrics (like Mean Opinion Score or MOS). However, that toolset is focused for its application for traditional video streaming, when using push-based RTP/RTCP protocols, in simulated environments. The work in [15] presents an emulation platform for evaluating traditional video streaming performance over Long Term Evolution (LTE) environments, by making use of NS-3, providing support for both RTP/RTCP and DASH. The advantage compared to [13] is that the use of NS-3 allows transmitting the actual video files and not traces of them, as in NS-2. However, that work is mainly focused on evaluating throughput and mobility aspects. The work in [16] presents an end-to-end DASH platform, including components for the encoding, segmentation and storage of the media contents at the server side, and for their delivery and adaptive consumption at the client side. Interestingly, that platform includes a newly developed DASH client, which includes a module to monitor KPIs and taking them into account for deciding the most proper quality for each DASH segment to be downloaded in each iteration. These KPIs include the available bandwidth, buffer fullness level, and other specific characteristics and status of the consumption device in use (like the battery level, screen resolution, and CPU load).

Acknowledging the relevance of these previous works, they do not provide support for 360º videos, do not include any database for registering and analyzing the collected statistics for multiple sessions, and they require the installation of specific applications for the clients. In the presented testbed, the player is web-based, so only a browser and Internet connectivity are required to make use of it.

III. TESTBED FOR KPI MONITORING IN DASH STREAMING

This section describes the different components of the presented testbed for KPI monitoring in omnidirectional video streaming scenarios, making use of DASH (Figure 1).

 ISBN: 978-1-61208-697-2

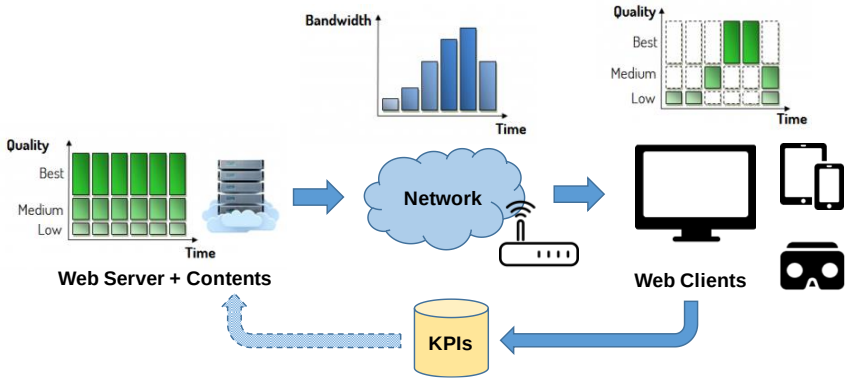

Figure 1. Overview of the testbed for KPI monitoring in (omnidirectional) video streaming services using DASH.

A. *Content Preparation and Publication*

The testbed includes different components and modules to convert input video files, whatever their format, in DASH, and to generate the related metadata. This includes processes for multi-quality encoding, conversion between Equirectangular and CubeMap Projection formats for 360º videos, generation of tiles, and segmentation of contents, with the desired configuration. This also includes the generation of the MPD for each video, and the generation / update of JSON / XML files listing and describing the available videos, respectively. Finally, all generated video and metadata files are stored on a Publication Server, which is basically an HTTP server (e.g., Apache).

More details about such processes can be found in [5][15].

B. *Content Consumption*

The video player has been developed by relying on web-based components, such as *dash.js* and *three.js* [17]. It includes the proper functionalities to select the desired contents from the Publication Server and to parse the JSON files to interpret the required metadata about such contents. If a 360º video with traditional Equirectangular Projection format is selected, the MPD can be directly processed. If a 360º video with traditional CubeMap Projection format and tiling strategies is selected, the player has to additionally process an XML file that describes the available tiles (number, distribution and qualities) [5]. In the latter case, the different tiles are retrieved as independent videos, so the player includes quality switching and inter-media synchronization mechanisms for optimizing the bandwidth consumption and perceiving the multiple meshes/videos as a single one, respectively [5].

C. *Registration of KPIs*

As mentioned, *dash.js* provides an API that allows obtaining statistics about certain KPIs. This API is used during the streaming session to get many of these KPIs, in addition to other related ones that are calculated by using the information from these KPIs, together with newly developed methods.

Although it is possible to register the gathered statistics within the memory of the web browser and/or in local files for each session, a Node.js [18] server with a MongoDB [19] database has been developed in order to register these KPIs, by making use of an HTTP-based communication protocol. This provides a better performance and stability of the player, getting persistence of the gathered statistics, and eliminates the need for having control on the consumption device on which the player has been run to retrieve the statistics. In addition, this allows gathering statistics from distributed and large-scale sessions, and enables higher flexibility for their analysis (e.g., by applying clustering or correlation strategies).

The frequency of measurement and registration of the KPIs (and thus of the communication with the database) can be configured in the player. In particular, the following KPIs are measured and registered in the presented testbed:

1) Objective / QoS-related KPIs:

- *Video Startup Latency*: Delay since the play button is clicked until the video is actually watched.
- *Evolution of Latency*: Periodic measurement of the end-to-end delay during the streaming session. *Jitter* can be also calculated as the variation of the delay.
- *Throughput*: Effective bandwidth during the session.
- Video Quality: The representation or quality index selected for each DASH segment during the session. Based on this KPI, the *Average Video Quality*, as well as the *Number, Frequency and Magnitude of Quality Switches* can be also calculated.
- *Video Stalls*: The evolution of both the playout time and the absolute time (e.g., Network Time Protocol or NTP based timestamps) can be periodically monitored. If they do not advance in accordance, and no playout control commands have been executed, it can be a sign of the occurrence of video stalls or a non-natural evolution of the playout process. This measurement determines how smooth / uninterrupted the playout is during the media session.
- *Buffer Fullness Level*: Occupancy of the playout buffer during the session (e.g., in percentages or in time units). If playout stalls happen, it can be due to buffer underflow / overflow situations. This information can be also used to select the most appropriate quality for the segments to be downloaded in order to avoid, or recover from, the undesirable underflow / overflow situations.

 ISBN: 978-1-61208-697-2

- *Asynchrony*: By comparing the playout and absolute times, the asynchrony (i.e., playout time difference) between media elements played out within the same device (i.e., local inter-media synchronization) and across devices (i.e., inter-device or inter-destination synchronization) can be calculated.
- *Viewing Direction*: When watching 360º videos, the current latitude and longitude angles referred to the center of the FoV can be measured. This information can be more accurate if eye-tracking functionality is available.
- *Duration of the session*: Amount of time since a video is selected and the session for this video is terminated. It can be shorter than the duration of the video if the session is terminated before the end of the video or seek forward commands have been executed, but also can can be longer if pause or seek backwards commands have been executed.

2) Subjective / QoE-related KPIs: In the presented testbed, the received audio and video files can actually be played out. Therefore, having knowledge about the obtained objective KPIs is important to better understand the allowable thresholds and ranges that can be tolerated by users, and result in satisfactory QoE levels, as well as to correlate QoS and QoE metrics. Next, key QoE related aspects are highlighted:

- *Video Startup & End-to-End Latency*: To what extent delays are tolerable by users.
- *Video stalls / pixelation / visual anomalies*: To what extent these effects are tolerable by users.
- *Overall Media Quality*: To what extent the overall media (audio / video) quality is tolerable by users, and the adopted quality switching algorithm can impact the perceived QoE (e.g., abrupt transitions can be noticeable and even annoying to users).
- *Synchronization levels*: To what extent inter-media and inter-device synchronization skews impact the perceived QoE. The impact of playout adjustments to achieve synchronization also apply in this context.
- *Smooth 360º Exploration*: To what extent the transitions between FoVs within the whole 360º area are perceived as smooth / instantaneous to users, and the perceived video quality in such transitions are perceived as acceptable. This KPI metric is relevant when using tiling / FoV-based streaming solutions, which will allow minimizing latency and bandwidth consumption compared to traditional solutions based on streaming the whole 360º area.

All these aspects can be evaluated by conducting subjective tests, making use of questionnaires, including MOS metrics or likert scales as evaluation metrics. Then, the obtained results from the subjective evaluations can be correlated with the measured objective results (e.g., when having used specific configurations and/or having forced specific conditions).

All these metrics can be extended in future release by making use of the *dash.js* API or even extending.

D. Application Contexts

The presented testbed has been developed to obtain valuable statistics and a deeper understanding about the performance in a variety of 360º video streaming scenarios, in which specific technical challenges are being addressed:

1) Minimization of Bandwidth and Delays

Achieving low latency is a big challenge in HAS. It is mainly due to the multi-quality encoding and segmentation processes, to the individual pull-based HTTP requests using the Transmission Control Protocol (TCP), and to the buffering strategies used at the client side.

One possible solution to overcome the delay is downloading the initial video segments in a low quality in order to start the playout as soon as possible, thus minimizing the startup delay. Another potential option is making use of HTTP/2 with *k-push* mechanism in order to get *k* video segments through a simple HTTP GET request. This avoids sending multiple independent requests, which would result in higher delays and traffic overhead. In this context, having an accurate knowledge about the magnitudes of delays and jitter becomes very valuable to set the sizes and thresholds of the playout buffer, in order to minimize latency while guaranteeing continuous playout.

Likewise, the adoption of appropriate quality switching strategies also has an impact on the perceived quality and on the experienced delays (higher quality segments require higher bandwidth and larger delays to be downloaded). Having an accurate knowledge about the available bandwidth, network delays, and other end-system's related metrics (like CPU load, buffer fullness level, number of active clients / services…) can also contribute to a more adequate media quality selection.

2) Smooth Exploration of the whole area in 360º video.

As mentioned, the streaming of the whole sphere in high quality in 360º video results in high processing load and bandwidth consumption. It also becomes inefficient, as the users can only watch a small region of the 360º area, determined by their FoV, at any time. In this context, the design and adoption of advanced encoding techniques, representation formats, spatial segmentation (i.e. tiling) strategies, and adaptive quality switching algorithms, can be proposed to maximize the quality for the current FoV, while reducing latency, bandwidth consumption and processing load [5]. However, these issues involve a key challenge, which is to guarantee a free and smooth exploration of the whole 360º area, without perceiving video stalls and abrupt video quality switches.

3) Hybrid Synchronized Multi-Screen Scenarios

It is possible to enrich conventional TV services by augmenting the traditional contents provided on the main TV with additional 360º videos to be played out on companion devices (e.g., smartphones, tablets, HMD…) [9]. This provides more personalized and immersive experiences. In such a case, it is essential to provide a synchronized playout across all involved devices, regardless of the media modality being consumed and the (broadband or broadcast) technology through which the contents are delivered.

 ISBN: 978-1-61208-697-2

IV. Test Scenario and Results

This section provides examples of results in a simple scenario to demonstrate the capabilities of the testbed.

A. Test Scenario

The test scenario consisted of an HTTP server (Apache) to host the player resources and media assets located at Network A, and a PC, smartphone (Samsung Galaxy S8) and HMDs (Oculus Go and Gear VR with a Samsung Galaxy S8) as consumption devices, located at Network B (same city as Network A) and connected via WiFi. The KPI database was installed on the same PC as the web server at Network A. Different traditional and 360º videos were converted to DASH. In particular, the traditional videos were encoded in 5 qualities and the 360º videos were encoded in 10 qualities, all of them with segments of 3s. This evaluation scenario adheres to the topology represented in Figure 1.

B. Test Results

First, it was assessed whether the type of video and the number of available qualities have an impact on the startup delay. For this purpose, streaming tests with a traditional and a 360º video encoded in both a single high-quality and in multiple qualities were conducted. The mean startup delays for 5 repetitions of all these conditions were: 0.558ms (traditional multi-quality video); 0.571ms (traditional single quality video); 0.781ms (360º multi-quality video); and 1.15s (360º single quality video). The results are as expected, since 360º videos are more heavy than traditional videos, and the availability of multiple qualities allows selecting lower ones at the beginning of the session to reduce startup delays.

Figure 2 shows the duration of 32 sessions for one of the videos, for sessions longer than 4min. Note that the duration of some sessions is higher than the duration of the video, which is 625s. It is due to the fact that some viewers paused the video and/or watched repetitions of some scenes.

Figure 3 represents the evolution of the throughput for three video players in a multi-screen session. In relation to this, Figure 4 represents the evolution of the selected DASH quality for two video players, one having selected a traditional video (5 qualities) and the other one having selected a 360º video (10 qualities). Figure 5 represents the evolution of the playout buffer fullness level in a multi-screen session of two devices. It can be observed that the buffers become progressively filled up to reaching a target threshold (in this case, 60%) at the beginning of the session. Then, the buffer occupancy levels slightly fluctuated during the session, being the fluctuation highly determined by the duration of the DASH segments. Finally, it can be observed that the buffers became progressively empty at the end of the session, as no more DASH segments were downloaded when reaching the end of the video.

Figure 6 represents the evolution of the asynchrony between three video players in a multi-screen session, once having activated the inter-device synchronization solution from [9], and by selecting a synchronization manager as the reference. It can be seen that the asynchrony was kept very low and stable most of the times, and that sporadic asynchrony situations occurred, but they were rapidly corrected thanks to the developed solution.

Finally, the instantaneous viewing fields for each 360º video were also measured, by monitoring the latitude and longitude angle values of the center of the FoV, every 2s. As an example, the intensity map for a specific session is represented in Figure 7. This allows representing heatmaps of viewing directions for multiple videos and sessions.

V. Conclusions & Future Work

Having an accurate and detailed knowledge about QoS and QoE related metrics in streaming services becomes essential. This work has presented a modular and extensible testbed to monitor and register KPIs when streaming omnidirectional videos using DASH. The different components of the testbed have been described, and examples of the statistics that can be collected have been provided. As future work, the testbed will be used in the different discussed application contexts, under a variety of conditions. The extension of the gathered KPIs will be also explored. In addition, real-time representation modules and advanced analysis and clustering features will be provided. As an example, the gathered KPIs will be adapted to be able to represent and compare heatmaps of viewing directions, as provided by the tool presented in [20].

Acknowledgment

This work has been funded by European Union's H2020 program, under agreement nº 761974 (ImAc project).

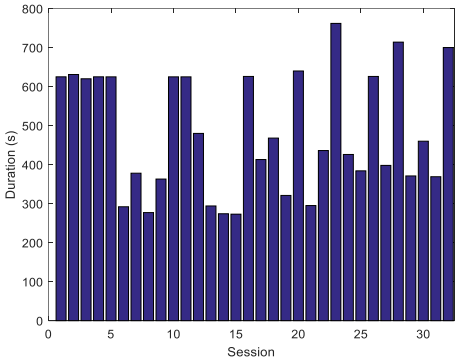

Figure 2. Duration of session.

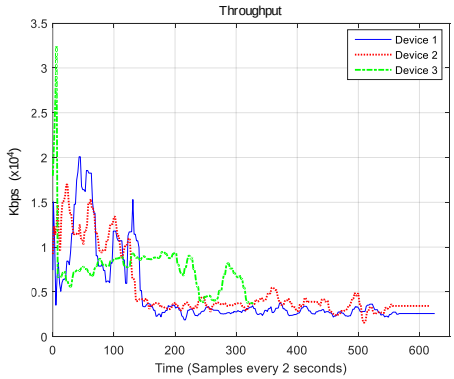

Figure 3. Evolution of throughput during the media session.

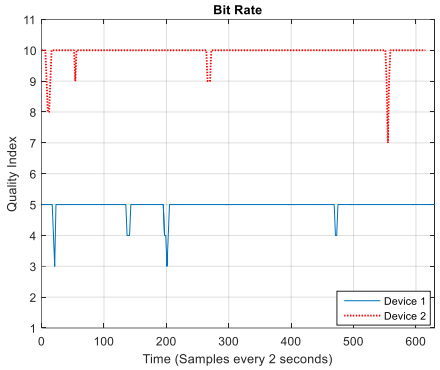

Figure 4. Evolution of the selected DASH quality.

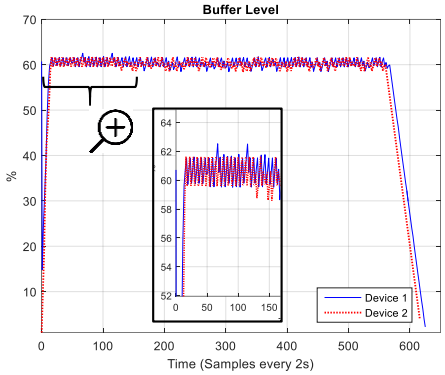

Figure 5. Evolution of the buffer fullness level.

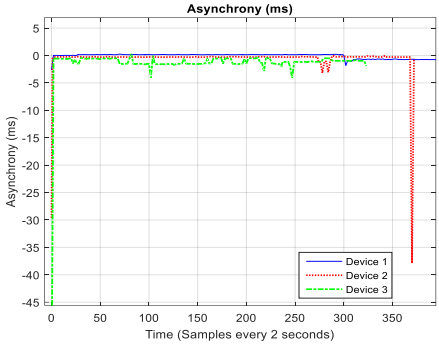

Figure 6. Playout Asynchrony Evolution.

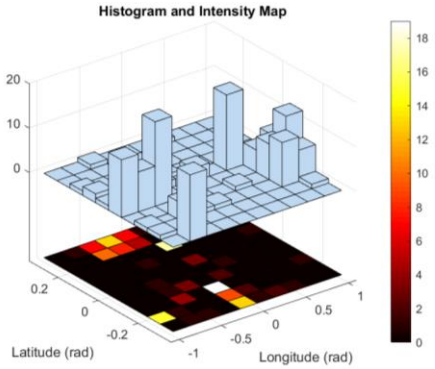

Figure 7. Intensity Map of Viewing Directions for a 360º Video.

REFERENCES

[1] A. Begen, T. Akgul, and M. Baugher, “Watching Video over the Web: Part 1: Streaming Protocols”, IEEE Internet Computing, vol. 15, no. , pp. 54-63, 2010.

[2] T. Stockhammer, “Dynamic adaptive streaming over HTTP: standards and design principles”, ACM MMSYS 2011, pp. 133-144 San Jose, CA (USA), February 2011.

[3] DASH: https://mpeg.chiariglione.org/standards/mpeg-dash Last Access: February 2019

[4] Hybrid Broadcast Broadband TV (HbbTV) standard specifications: https://www.hbbtv.org/resource-library/specifications/ Last Access: February 2019

[5] D. Gómez, J. A. Núñez, I. Fraile, M. Montagud, and S. Fernández, “TiCMP: A lightweight and efficient Tiled Cubemap projection strategy for Immersive Videos in Web-based players”, ACM NOSSDAV’18, pp. 1-6, Amsterdam (The Netherlands), June 2018.

[6] M. Graff, C. Timmerer, and C. Mueller, “Towards Bandwidth Eficient Adaptive Streaming of Omnidirectional Video over HTTP”, MMSYS’17, 261-271, Taipei (Taiwan), June 2017.

[7] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, “Viewport-adaptive navigable 360-degree video delivery”, IEEE ICC 2017, pp. 1-7, Paris (France), 2017.

[8] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. “An HTTP/2-Based Adaptive Streaming Framework for 360° Virtual Reality Videos”, ACM MM’17, Amsterdam (The Netherlands), 2017.

[9] J. A. Núñez, M. Montagud, I. Fraile, D. Gómez, and S. Fernández, “ImmersiaTV: an end-to-end toolset to enable customizable and immersive multi-screen TV experiences”, Workshop on Virtual Reality, co-located with ACM TVX 2018, Seoul (South Korea), June 2018.

[10] A. Bentaleb, B. Taani, A.C. Begen, C. Timmerer, and R. Zimmermann, “A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP”, IEEE Communications Surveys & Tutorials, pp., August 2018.

[11] E. Thomas, M. O. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey, “Enhancing MPEG DASH Performance via Server and Network Assistance”, SMPTE Motion Imaging Journal, 126(1), pp. 22-27, February 2017.

[12] Dash.js: https://github.com/Dash-Industry-Forum/dash.js/wiki Last Access: February 2019

[13] F. Boronat, M. Montagud, and V. Vidal, “A More Realistic RTP/RTCP-Based Simulation Platform for Video Streaming QoS Evaluation”, JMM, 7(1&2), 66-88, 2011.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: a transport protocol for real-time applications”, RFC 3550, July 2003.

[15] A. Fouda, et al., “Real-Time Video Streaming over NS3-based Emulated LTE Networks”, International Journal of Electronics Communication and Computer Technology (IJECCT), 4(3), pp. 659-663, May 2014.

[16] D. Gómez, F. Boronat, M. Montagud, and C. Luzón, "End-to-End DASH Platform including a Network-based and Client-based Adaptive Quality Switching Module", ACM MMSYS 2016, Klagenfurt (Austria), May 2016.

[17] Three.js https://threejs.org/ Last Access: February 2019

[18] Node.js https://nodejs.org/en/ Last Access: February 2019

[19] MongoDB https://docs.mongodb.com/ Last Access: February 2019

[20] S. Rothe, T. Hoellerer, and H. Hussmann, “CVR-Analyzer: A Tool for Analyzing Cinematic Virtual Reality Viewing Patterns”, ACM SUI’18, Berlin (Germany), October 2018

 ISBN: 978-1-61208-697-2