Methodology for fine-grained monitoring of the quality perceived by users on 360VR contents

Lara Muñoz[a], César Díaz[a,*], Marta Orduna[a], José Ignacio Ronda[a], Pablo Pérez[b], Ignacio Benito[b], Narciso García[a]

[a] *Grupo de Tratamiento de Imágenes, Information Processing and Telecommunications Center and ETSI Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain*
[b] *Nokia Bell Labs, Madrid, Spain*

Abstract

To properly evaluate the performance of 360VR-specific encoding and transmission schemes, and particularly of solutions based on viewport adaptation, it is necessary to consider not only the bandwidth saved, but also the quality of the portion of the scene actually seen by users over time. With this motivation, we propose a robust, yet flexible methodology for fine-grained monitoring of the quality within the viewport along the visualization session. This novel procedure is based on a complete analysis of the geometric relations involved. Moreover, the designed methodology allows for both offline and online usage by using different approximations. In this way, our methodology can be used regardless of the approach considered to properly evaluate the implemented strategy, obtaining a fairer comparison between them.

Keywords: 360VR, video streaming, video quality, viewport, QoE

1. Introduction

During the last few years, the interest for Virtual Reality (VR) has grown exponentially. Everyday, more and more VR-related applications appear, and the number of VR-ready devices, particularly of head-mounted displays (HMD), is quickly expanding, as they become appealing and affordable to an increasing

*Corresponding author
Email address: cdm@gti.ssr.upm.es (César Díaz)

number of users. One of the most common VR applications is the visualization via streaming of 360 videos in non-interactive environments, covering a wide range of applications such as education [1], medical treatments [2], and simulators [3].

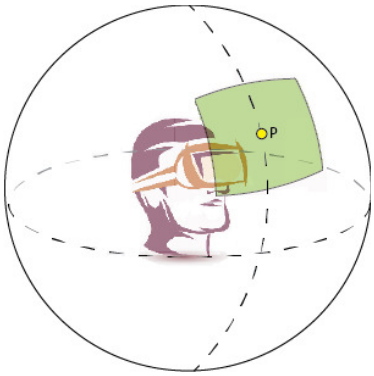


Figure 1: Viewport (presented in green).

The transmission of this type of content is particularly challenging. The main reason is that, due to the nature of the environment and the features of the associated presentation systems, the requirements in terms of image resolution and quality to offer a really immersive experience to the user are especially demanding [4, 5]. This results in sequences that require very high bit rates for their transmission. Thus, it is necessary to incorporate smart coding and transmission management schemes. To provide a smooth playback and good quality service, these schemes should take advantage of the fundamental fact that only a fraction of the image can be seen by the user at a certain moment (Figure 1). This fraction depends on the HMD, whose design sets the field of view (FoV) and, therefore, determines the viewport, the picture area shown to the user. As the FoV is usually a right rectangular pyramid, defined by the two angles between opposing planes (dihedral angles), the viewport is a spherical rectangle that includes the point of gaze (PoG) of the user, that is where the user is really looking at [6].

These so-called viewport-adaptive schemes have the objective of offering high

quality to the users while saving bits. It can be achieved by providing higher quality to the area that will presumably be visible to the user and lower quality to the area with a lower probability of being visible to that person. In this way, they can deliver a good quality of experience (QoE) while saving bandwidth. For this purpose, the raw image is tessellated into rectangular subimages, which are compressed and managed differently in an intelligent way. So, this procedure paves the way for the adaptation of the content presented to the viewer. In this sense, both the encoding and transmission modules need to be ready to enable viewport adaptation and so provide the image or set of images with the distribution of qualities that best suits the user's viewport at a given moment. Regarding encoding, the use of tiles is the technique most commonly employed to process in an independent way different areas of the image. Tiles are a tool included in the H.265/HEVC standard that enables the partition of the picture into independently decodable regions with some shared header information [7]. Before the appearance of tiles, H.264/AVC's Flexible Macroblock Ordering (FMO) [8] could be used instead to distribute heterogeneously the quality in the image. However, this tool was neither efficient nor widely implemented. Regarding transmission, HTTP/TCP-based adaptive bit rate (ABR) streaming techniques [9] are commonly used to deliver omnidirectional video, due to their adaptability. In this scheme, content is encoded at different resolutions and bit rates and divided temporally into self-contained segments of equal duration that invariably start with an Instantaneous Decoding Refresh (IDR) frame. Therefore, the segment that best suits both the system state (channel available bandwidth, terminal capabilities...) and the viewport at a particular moment is delivered to the client to be decoded and presented to the user. The configuration used for certain parameters, such as the number of partitions, the encoding parameter values in each image subdivision or the segment length, and the transmission scheme will influence decisively in several aspects of the system: quality provided and perceived by the user based on his/her behavior, bandwidth used, storage needs, intelligence requirements in different elements of the system, etc.

Indeed, one key problem of viewport-adaptive approaches is the need for a system that adapts quickly to the movements of the user. That is, it is essential that the high quality viewport that corresponds to the new position of the user is presented as soon as possible. Otherwise, the user will perceive low quality areas that will decrease his/her QoE. Therefore, many proposed strategies use short segments and small buffers. However, the drawback is that short segments imply lower coding efficiency [10], so a very low quality or even black areas [11] are necessary to be able to save bandwidth. Instead, there are some authors [12] who propose the use of several streams with shifted IDR frames to allow quick switching between versions prepared for different viewports and representations whereas keeping a long IDR period. However, this proposal has its own disadvantages, such as a greater complexity at the client side and a larger number of versions of the content at the server side. In either case, these changes are still not instantaneous as they require the download and playback of a new IDR, so the user could perceive low quality areas if he/she moves quickly. Furthermore, the size of the high quality areas influences significantly the QoE: the larger the areas are, the lower the probability of seeing low quality areas will be, but also the lower the bandwidth saving for an equivalent quality in the viewport. As a result, the development of a method to reliably measure the quality really perceived by users is fundamental in the design of viewport-adaptive schemes [13]. In this way, we can test the design of the strategy and, when appropriate, improve it by fine-tuning the parameters that characterize it

Despite that, the majority of the proposed methods are evaluated considering only the bandwidth saved [14, 15, 16] and only a limited number of works base the functioning of their strategies on the results of applying an image quality assessment (IQA) method within the viewport. Those strategies use very varied ways to compute the quality. Some use objective IQA metrics, either the original version [17] or a 360VR-aware one [18]), others apply the results of subjective assessments [19] and others resort to completely different measures like the percentage of time that the user looks within the high quality area [20]. However, even if a high-performance IQA method is used, the results might not

be sufficiently reliable. This can occur even if they apply afterwards corrections that consider the characteristics of the projection [21]. The cause is that most often methods are applied on portions of the image that do not exactly match the one really seen by the user. The reason is that most approaches use coarse approximations of the projected viewport, mainly in the shape of rectangles or even squares [22, 23], which are far from representing the actual form of the projected viewport, particularly if the center of the viewport is not near the equator. Even more, many proposals do not detail sufficiently (or at all) the projection methodology they use, which, as mentioned, is crucial to validate a viewport-adaptive VR coding and transmission strategy. Finally, there exist proposals that do present a methodology that result in accurately obtaining the projection of the viewport [24]. However, they can be very time consuming, as the mapping is carried out pixel by pixel.

Hence, we present a detailed methodology, named VAQM (Viewport Adaptive Quality Method), to accurately, yet simply calculate the viewport projection on the equirectangular image and, thus, to enable its use in every scheme looking for an overall quality metric value on the image seen by the user. As any standard metric (e.g. PSNR, SSIM, VMAF, MOS-related...) can be used, a complete solution for the objective quality assessment is provided. Additionally, we also provide a simplified version of the procedure for operation under strict computing time restrictions. So, certain approximations are used for computing the quality, such as using a set of pre-calculated viewport projections.

The rest of the paper is organized as follows. First, in Section 2, the viewport projection is explained in detail. Then, in Section 3, the procedure to obtain a figure of merit reflecting the quality perceived by the user is presented. In Section 4, we describe the full method and the approximated version developed to obtain the quality of the session. The description of the experiments carried out to assess the performance of the method and its results and the corresponding analysis are included in Section 5. Finally, the paper is concluded in Section 6.

2. Viewport projection

Let us consider the coordinate system used for the definition of the FoV, the viewport, and its projection on the flat image. Figure 2 presents the Cartesian and spherical coordinates, where the origins of the two spherical angles follow the usual choices for HMDs .

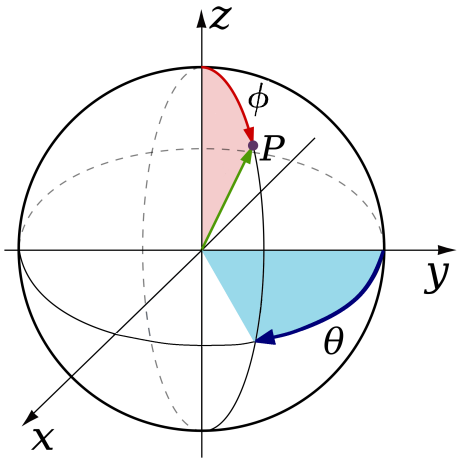


Figure 2: Cartesian and spherical coordinate system on a normalized sphere

Therefore, the transformation between them is described by the sets of equations:

$$\begin{cases} \theta = \arctan\left(x/y\right) \\ \phi = \arctan\left(\sqrt{x^2+y^2} \;/\; z\right) \end{cases} \qquad \begin{cases} x = \sin\phi\sin\theta \\ y = \sin\phi\cos\theta \\ z = \cos\phi \end{cases} \tag{1}$$

Several planar mappings of the sphere (also called projections) have been considered for the representation of 360 video content: equirectangular, cubemap, pyramidal, equiangular... [25, 26, 27]. Since no map of the sphere to the plane can be both conformal and area-preserving, each mapping affects the quality of the different areas of the 360 video content in a different way. Among the different projections that are used, the most common mapping is the equirectangular projection and, therefore, it is the one considered here. Its main advantage

lies in the simple transformation equations between the spherical and the planar coordinates. Assuming that the upper-left corner of the frame is the origin of the planar coordinates, the values of θ and ϕ can be scaled directly to obtain their planar counterparts. Therefore, if the 360 video contents are represented in a $N_H \cdot N_V$ frame, the pixel coordinates of a point on the sphere (θ, ϕ) are $((\theta/360)N_H, (\phi/180)N_V)$ and, thus, we will keep the (θ, ϕ) addressing for the equirectangular plane. Nevertheless, it must be stressed that any other projection can be considered, as the methodology proposed here can be applied on any geometry.

2.1. FoV, viewport, and projected viewport

Let us consider that the FoV is a right rectangular pyramid whose vertex is located in the center of the viewing sphere of the HMD. Thus, the viewport on the sphere is a spherical rectangle and each one of its four sides is a great circle arc. If the FoV is defined by its two dihedral angles $(\theta_{\mathrm{VP}}, \phi_{\mathrm{VP}})$, the solid angle SA subtended at the center of the sphere by the FOV is [28]:

$$SA = 4 \arcsin\left(\sin\frac{\phi_{\mathrm{VP}}}{2} \sin\frac{\theta_{\mathrm{VP}}}{2}\right) \tag{2}$$

Any value of $(\theta_{\mathrm{VP}}, \phi_{\mathrm{VP}})$ is acceptable. However, to help explain the procedure and perform the associated experiments, we have chosen $(\theta_{\mathrm{VP}}, \phi_{\mathrm{VP}}) = (100^o, 85^o)$ since they represent the average value of the FoV parameters found in the most common HMDs. Thus, the solid angle is 2.15 steradians, roughly $1/6$ of the surface of the sphere, which implies a large deformation, where the four spherical angles are clearly larger than 90^o. Therefore, linear approximations, commonly employed in the literature, cannot be used.

Although the shape of the projected viewport in the equirectangular image varies significantly according to the location of the viewport on the sphere, let us remember that the size (subtended solid angle) of the viewport is constant. So, it is useful to obtain this constant value in pixel related units. If the sampling rate at the equator of the equirectangular image is taken as reference, each one of the pixels lying on the equator covers a *unit area*, either in the equirectangular

image or on the sphere. However, pixels located outside the equator cover less area on the sphere as the sampling rate along parallels increases with the latitude.

As said before, the origin of coordinates of the equirectangular image is located in the upper-left corner, as shown in Figure 3. The area covered by each pixel is $a(\theta, \phi) = \sin\phi$ in the above mentioned area units. So, we call the *equivalent number of pixels* the area of the region expressed in these area units. Thus, the equivalent number of pixels of the whole frame is N_{picture}:

$$N_{\text{picture}} = \sum_{i.j} a(\theta_i, \phi_j) = N_H \sum_j \sin\phi_j = \frac{2}{\pi} N_H N_V \tag{3}$$

As the whole frame covers the whole surface of the sphere, the solid angle subtended is 4π steradians. Thus, the equivalent number of pixels of the viewport, N_{viewport}, can be obtained as a proportion of the solid angles subtended:

$$N_{\text{viewport}} = \frac{2}{\pi^2} N_H N_V \arcsin\left(\sin\frac{\phi_{\text{VP}}}{2} \sin\frac{\theta_{\text{VP}}}{2}\right) \tag{4}$$

This result should be rounded if an integer value is required.

Furthermore, this novel expression of the equivalent number of pixels of the viewport sets the maximum effective resolution that can be achieved by the HMD display. As an example, for the usual values considered in this paper, $(\theta_{\text{VP}}, \phi_{\text{VP}}) = (100^o, 85^o)$, $N_H N_V = 3840 \times 1920$, we obtain $N_{\text{viewport}} = 802871$ pixels, clearly lower than 1 Mpixel.

2.2. Procedure for computing the projected viewport

Looking for a simpler set of operations to obtain the shape of the projected viewport, we decompose the computation of the projected viewport into a three-step procedure. First, we consider a base viewport centered on the central point of the equirectangular image and compute its vertices and several points along its four sides that will help define a piecewise linear approximation of those sides. Then, we rotate this set of points to place them around the projected center of the viewport. Finally, we obtain the desired projection by connecting those

points, thus generating a closed region. Pixels within the boundaries marked by these connections belong to the projection and the set of these pixels is called the mask. These steps are explained in detail below.

2.3. Base projected viewport

Every projected viewport is characterized by the location of its four spherical vertices. As stated, we first determine those of the base viewport, which correspond to the initial viewing experience. Throughout the paper and in our experiments, we assume that the user begins looking at the center of the equirectangular image, which corresponds to $O = (180^o, 90^o)$. However, the proposed methodology can be adapted to any other desired initial point located at the equator, due to the special features of the base viewport that are described below.

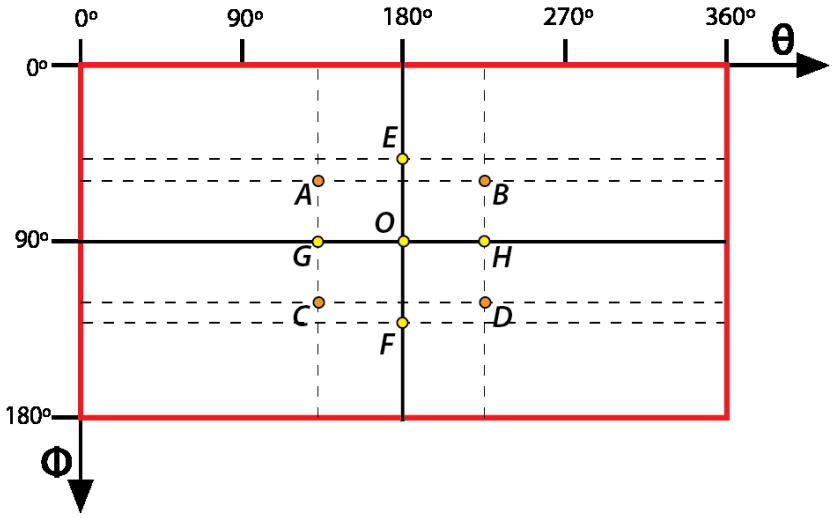


Figure 3: Key points to compute the base viewport

As the base viewport is centered on the equator, the two vertical sides follow two meridians. However, the two horizontal sides do not follow any parallel. The analysis begins determining the coordinates of the middle points of the four sides of the projected viewport (E, F, G and H in Figure 3):

$$\begin{cases} E = (180^o, 90^o - \phi_{\mathrm{VP}}/2) \\ F = (180^o, 90^o + \phi_{\mathrm{VP}}/2) \\ G = (180^o - \theta_{\mathrm{VP}}/2, 90^o) \\ H = (180^o + \theta_{\mathrm{VP}}/2, 90^o) \end{cases} \tag{5}$$

As the viewport sides AC and BD follow two meridians, they are projected as vertical straight lines on the equirectangular image. Therefore their abscissas are equal to those of their midpoints G and H respectively. However, the same does not apply for the other two lines AB and CD, requiring the analysis of the projection of the viewport on the sphere to obtain the values of their ordinates:

$$\begin{cases} \phi_A = \phi_B = 90^o - 2\arctan\left(\tan\frac{\phi_{\mathrm{VP}}}{2}\cos\frac{\theta_{\mathrm{VP}}}{2}\right) \\ \phi_C = \phi_D = 90^o + 2\arctan\left(\tan\frac{\phi_{\mathrm{VP}}}{2}\cos\frac{\theta_{\mathrm{VP}}}{2}\right) \end{cases} \tag{6}$$

Let us now consider the location of the points along the sides. Thus, let AB be a great circle arc, (x_A, y_A, z_A) the Cartesian and (θ_A, ϕ_A) the spherical coordinates of point A, and (x_B, y_B, z_B) the Cartesian and (θ_B, ϕ_B) the spherical coordinates of point B. Additionally, let L be another point in the same great circle arc defined by A and B on the unit sphere and let (x_L, y_L, z_L) be its Cartesian and (θ_L, ϕ_L) its spherical coordinates. Then, since the three points belong to the same great circle arc, and so to the same plane, the determinant of the matrix built with their Cartesian coordinates is zero:

$$\begin{vmatrix} x_L & y_L & z_L \\ x_A & y_A & z_A \\ x_B & y_B & z_B \end{vmatrix} = 0, \tag{7}$$

Solving the equation and taking into account that the point L lies on the surface of the unit sphere, the equation of the line joining the two vertices A and B is:

$$\phi_L = \arctan\left(-\frac{\gamma}{\alpha\sin\theta_L + \beta\cos\theta_L}\right), \tag{8}$$

where

$$\begin{cases} \alpha = & \sin\phi_A \cos\theta_A \cos\phi_B - \sin\phi_B \cos\theta_B \cos\phi_A \\ \beta = & -\sin\phi_A \sin\theta_A \cos\phi_B + \sin\phi_B \sin\theta_B \cos\phi_A \\ \gamma = & \sin\phi_A \sin\theta_A \sin\phi_B \cos\theta_B \\ & -\sin\phi_B \sin\theta_B \sin\phi_A \cos\theta_A \end{cases} . \tag{9}$$

We sample the equation for several θ_L values to obtain their corresponding ϕ_L. The resulting L points will be connected later using a piecewise linear function. Therefore, depending on the number of values used, the line approximation will be coarser (low number of points) or more accurate (high number of points). Figure 4 shows the results obtained in this step.

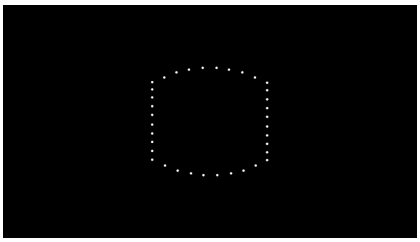


Figure 4: Base projected viewport (first step of the procedure)

2.4. Rotation of central viewport

In this step, the base viewport is moved to its final location. To that end, the points along the edges of the viewport are rotated an angle of $(90^o - \phi)$ vertically and an angle of θ horizontally, where these angles correspond with the coordinates of the center of the viewport on the sphere. The value of these coordinates can be easily obtained frame by frame using the HMD's software. To simplify the operations in the second step, these movements are performed subsequently, and so, separately. First, the viewport is rotated along ϕ and then along θ. Roll movements are not considered, since they are assumed to be negligible.

For the rotation along ϕ, as the central viewport is so far assumed to be centered in $(0, -1, 0)$, it is performed about the $-x$ axis, as can be seen in

Figure 5.

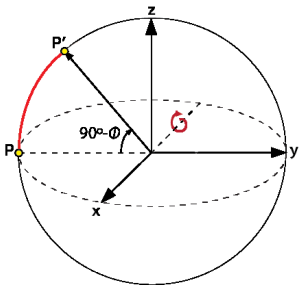


Figure 5: Rotation about $-x$ axis. Point P is rotated an angle of $(90^o - \phi)$ about the $-x$ axis, obtaining point P'.

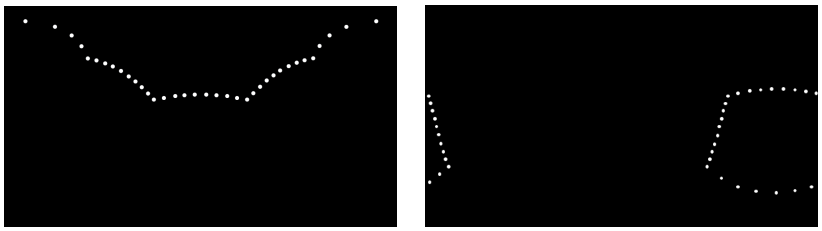
Therefore, the corresponding rotation matrix is:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(90^o - \phi) & \sin(90^o - \phi) \\ 0 & -\sin(90^o - \phi) & \cos(90^o - \phi) \end{bmatrix}. \tag{10}$$

(a) Before horizontal rotation (b) After horizontal rotation

Figure 6: Horizontal rotation of θ degrees of the projected viewport

235 The second part of the rotation is performed as follows. Once the viewport has been rotated vertically, it is moved horizontally on the equirectangular image. In Figure 6, the points of the viewport in (a) are rotated θ degrees horizontally, obtaining the points shown in (b). More examples of the results at the end of this second step are shown in Figure 7.

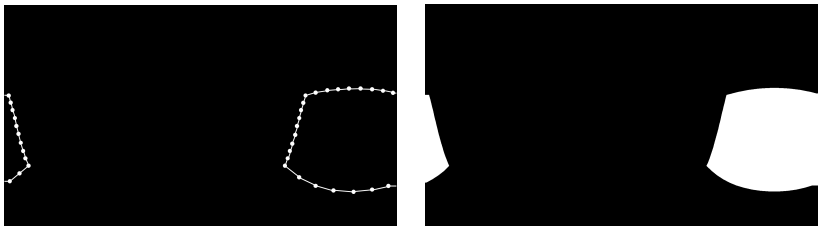


(a) $\theta = 180°, \phi = 30°$ (b) $\theta = 320°, \phi = 110°$

Figure 7: Examples of rotated (vertically plus horizontally) projected viewports

2.5. Mask generation

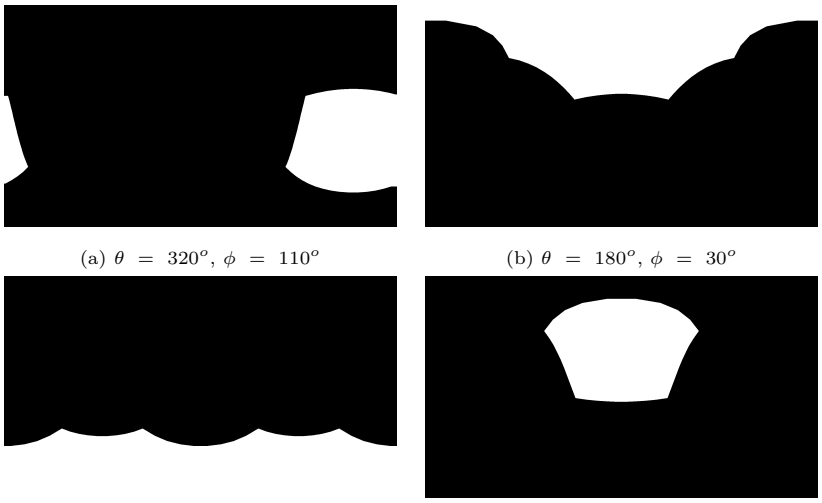
After the first two steps, the obtained points are joined with straight lines, that is, using a piecewise linear function, generating a closed region. This closed region is then filled to obtain the desired mask. Different possible situations must taken into account to correctly identify the region of the image within the mask. For example, Figure 8 presents an special where the mask is not totally connected, but divided in two parts due to circular shifts. Additional examples of the obtained masks are shown in Figure 9.



(a) Perimeter using straight lines (b) Filled closed region

Figure 8: Mask generation procedure (last step of the procedure)

At this point, we have a binary mask, M_i, where the non-zero elements represent the viewport projection on frame i. However, to compensate for the unequal sampling of the sphere by the equirectangular projection, the values of the elements in M_i are weighted according to the area they cover on the sphere. Each of these weights depends exclusively on the latitude and its value is equal to the sine of its corresponding ϕ value ($w_j^a = \sin \phi_j$).



(a) $\theta = 320^o$, $\phi = 110^o$ (b) $\theta = 180^o$, $\phi = 30^o$

(c) $\theta = 180^o$, $\phi = 180^o$ (d) $\theta = 180^o$, $\phi = 60^o$

Figure 9: Examples of masks centered at different locations

$$\tilde{m}_{i,p} = m_{i,p} \cdot w_p^a \qquad \forall p \in M_i \tag{11}$$

Furthermore, the values of the corresponding pixels can be weighted additionally to provide more importance to the more relevant pixels within the viewport, i.e. the central area with respect to the viewport edges. In this case, the pixels in the projected viewport j are weighted considering the distance to the center of that projection j_c as follows: $w_j^c = f\left(d\left(j, j_c\right)\right)$. Afterwards, they are normalized accordingly.

3. Proposed Methodology

3.1. Viewport adaptation

In contrast to traditional methods, viewport adaptation implies that the quality is not uniformly distributed across the image, but it is composed of areas of different qualities. Figure 10 shows the difference between non-viewport-adaptive and viewport-adaptive methods, where MQ represents the quality of

a given image in the traditional scheme, which is the same in the whole image, regardless of whether an area belongs to the viewport, and LQ and HQ respectively correspond to lower and higher qualities to the ones provided in the non-adaptive scheme.

Although the figure presents only two qualities for the non-viewport-adaptive method, more qualities can be used as long as the bitrate is preserved. This non-uniform quality distribution enables that users may observe more than one quality at the same time. The information about the corresponding quality of each of the areas of the image is represented by a grade matrix V, where each entry represents the quality value of one pixel of the image. As mentioned in the introduction, these non-uniform distributions can be implemented thanks to the use of tiles, since each of them may have a different quality.

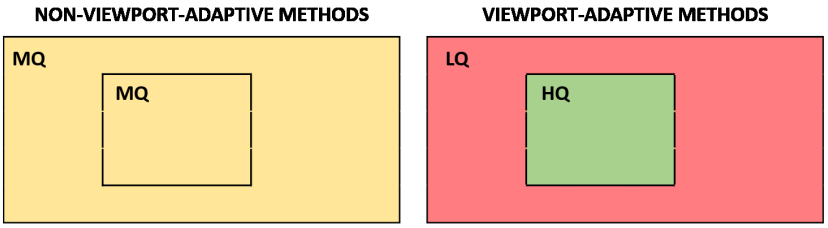


Figure 10: Difference between non-viewport-adaptive and viewport-adaptive methods regarding the quality distribution (LQ: Low Quality, MQ: Medium Quality and HQ: High Quality)

3.2. Methodology

The main idea of the proposed methodology is to provide a figure of merit that reflects the quality really perceived by the user along a temporal window. Therefore, we need a representative value of the quality seen at each frame within this temporal scope. To that end, we define the quality $q_{i,p}$ of each pixel p as the product of a geometric-related component, $m_{i,p}$, and a grade-related one, $v_{i,p}$. The matrix M_i, outcome of the previous section as the mask representing the viewport projection, contains the geometric-related component of each pixel at frame i. A second matrix, V_i, contains the grade-related components of each pixel of the image of the viewport-adaptive content presented to the user at

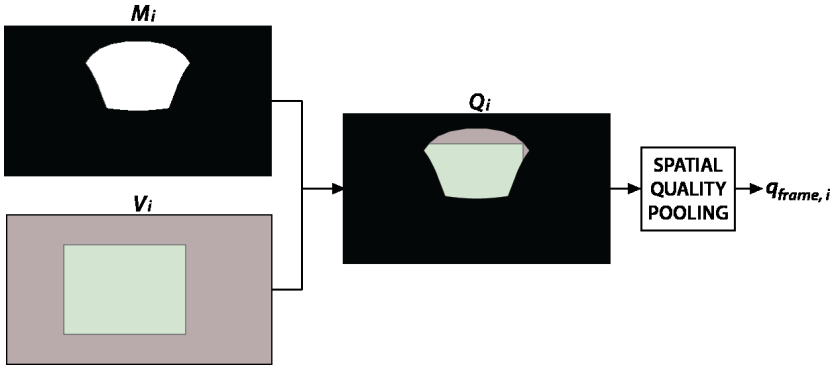


Figure 11: Quality metric procedure to obtain the quality value for frame i

that particular moment. Thus, the resulting matrix Q_i can be formulated as the Hadamard product of the geometric-related and the grade-related matrices:

$$Q_i = M_i \circ V_i. \tag{12}$$

A spatial quality pooling can be obtained for every frame and a temporal quality pooling can be computed to obtain an overall quality figure for the considered temporal window.

Spatial quality pooling

Regarding the spatial pooling, the quality value $q_{\text{frame},i}$ for frame i is computed as the average of all the quality values within Q_i of the pixels in the viewport projection,

$$q_{\text{frame},i} = \frac{1}{N_{\text{viewport}}} \sum_{p \in M_i} q_{i,p} \tag{13}$$

where N_{viewport} is the equivalent number of pixels of the viewport obtained in the previous section, and $q_{i,p}$ is the element in matrix Q_i representing the quality of pixel p. Since all the values outside the viewport projection are null, the summation can be extended to the whole image, delivering the same value:

$$q_{\text{frame},i} = \frac{1}{N_{\text{viewport}}} \sum_{p \in Q_i} q_{i,p} \tag{14}$$

Figure 11 illustrates the proposed methodology up to the output of the spatial quality pooling.

Temporal quality pooling

Regarding the temporal pooling, we have defined two different approaches: the mean and the fraction of time above a threshold. The first one reflects the average quality shown to the user during a temporal window of the streaming session and is obtained by a uniform or weighted average of the spatial quality over time, leading to q_{w},

$$q_{\mathrm{w}} = \frac{1}{N_f} \sum_{i=0}^{N_f-1} q_{\mathrm{frame},i} \tag{15}$$

where N_f is the number of frames in the analyzed temporal window. The second approach gives a figure of the fulfillment of a minimum spatial quality along the analyzed window and is computed as the percentage of frames with a quality value higher than a threshold T_{Q}:

$$f_{\mathrm{w}} = \frac{1}{N_f} \sum_{i=0}^{N_f-1} [q_{\mathrm{frame},i} > T_{\mathrm{Q}}] \tag{16}$$

where $[P]$ is the Iverson bracket, i.e. 1 if P is true and 0 otherwise. The threshold T_{Q} can be set to any specific value between 0 and 1 that designers decide better suited to their goals. The greater T_{Q} is, the more strict the imposed requirements are in terms of maintained quality over time, taking into account the specifics of the session (type of content, user behavior, setup...).

There are two approaches for the quality pooling. On the one hand, if the objective is to evaluate the proportion of high quality area that is presented to the user along the considered temporal window, the entries of matrix V_i must be set either to one, if they belong to the high quality area, or to zero, otherwise. On the other hand, if the objective is to look for an objective assessment, any metric that provides a value per pixel can be used for populating matrix V_i.

Additionally, objective IQA metrics such as MSE, PSNR, SSIM [29], MS-SSIM [30] or VMAF [31], which provide a single value per frame, can be used by

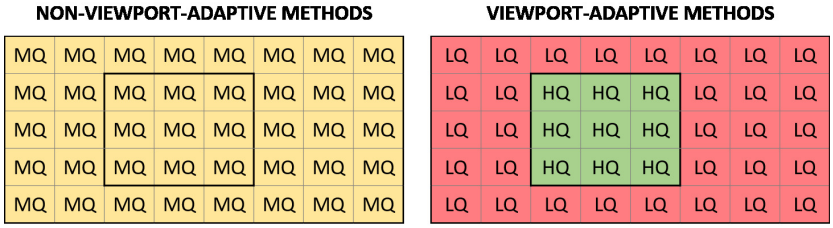


Figure 12: Difference between non-viewport-adaptive and viewport-adaptive methods regarding the quality distribution (LQ: Low Quality, MQ: Medium Quality and HQ: High Quality)

computing V_i on a per area basis. In this case, we can exploit the fact that the equirectangular image can be divided into tiles for encoding and therefore we can apply the desired technique to each of them individually or to sets of them (Figure 12). In this way, all the pixels belonging to the same tile or set of tiles will have the same value in the grade matrix. Nevertheless, although the matrix V_i is computed in a different way, the proposed methodology is maintained.

4. Proposed full and approximated methods

The methodology presented in the previous section requires the computation of matrices M_i and V_i for all the frames in the session. However, computing requirements might be a burden for lightweight real-time applications. Thus, we have defined two methods: the full method, called Viewport Adaptive Quality Method (VAQM), and a lighter version, called Approximated Viewport Adaptive Quality Method (AVAQM), where matrices M_i and V_i are selected from pre-computed sets of masks and grades to speed up the process. Both methods are described next.

4.1. VAQM (Viewport Adaptive Quality Method)

The scheme followed in the full method is shown in Figure 13. During the session, we constantly collect information about the content that the user is watching and the center of the viewport of the user at each moment. Afterwards, the viewport projection is computed for all the collected samples, obtaining a

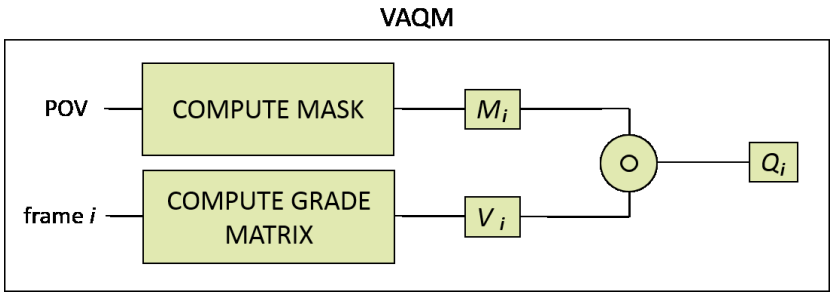


Figure 13: Scheme followed by VAQM

new mask for each instant of time. Moreover, the matrix V_i is generated from applying the desired quality metric (FR or not) to the whole image (e.g. MSE, PSNR, SSIM or VMAF). Thus, this matrix is of the same size as the transmitted video images. In summary, with this approach, we obtain a very high accuracy in exchange for a greater computational cost.

4.2. AVAQM (Approximated Viewport Adaptive Quality Method)

This approach arises from the fact that there are some scenarios where time restrictions may not allow us to apply the previous method directly, as, depending on the resources, it could be computationally costly. Additionally, it is also valid for when accuracy requirements are more flexible. The scheme followed by this approach is shown in Figure 14.

Approximations might be carried out independently on two fronts: in the geometric-related part of the procedure to generate matrix M_i, and in the grade-related part to obtain V_i.

Regarding the geometric part, M_i can be approximated using a finite set of pre-calculated masks with centers uniformly distributed throughout the equirectangular image, as represented by the yellow circles in Figure 15. The selected pre-calculated mask for frame i, M_i', (in green in the same figure) is the one whose center (θ_i', ϕ_i') is the nearest neighbor to the projected viewport center (θ_i, ϕ_i) (in red in the same figure). On the plus side, the use of the geometric approximation hugely decreases the computation load required to perform the

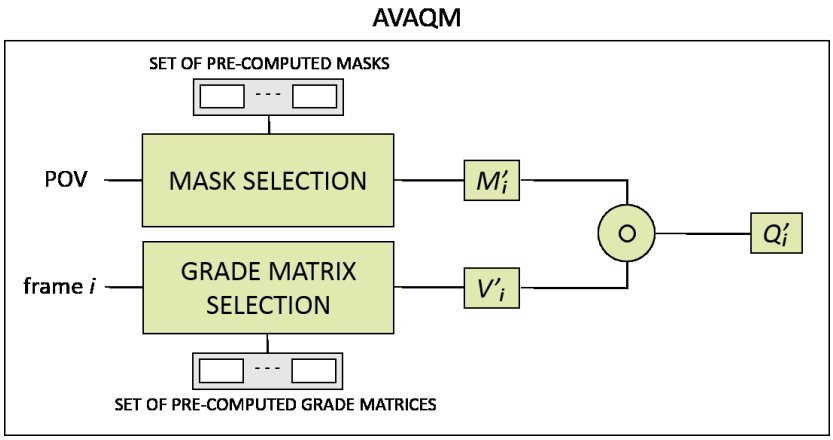


Figure 14: Scheme followed by AVAQM

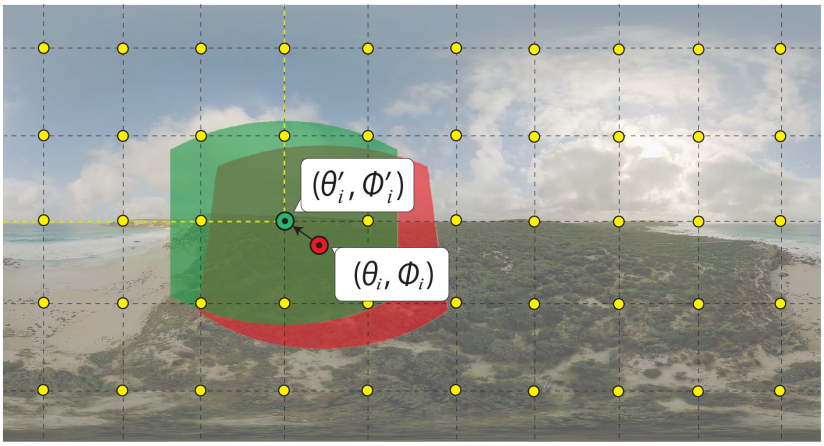


Figure 15: Example of approximation of matrix M_i using 5x10 pre-calculated masks

viewport projection. On the down side, there is a certain loss of accuracy and a slight increase of storage requirements. Both drawbacks heavily depend on the number of pre-calculated masks used.

With respect to matrix V_i, the entries of its approximation V_i' can be computed considering the value of encoding parameters that do not express the resulting image quality but indirectly provide a sufficiently accurate idea of it, like the Quantization Parameter (QP) used to encode the basic processing units (e.g. Coding Tree Units -CTUs- in H.265/HEVC) in the image. In this particular case, lower values correspond to better qualities. The advantage of this approximation relies on the ease and speed when obtaining and mapping such values. The drawback is the gap between these values and any others resulting from the application of a given quality metric in terms of capacity upon representing the quality perceived by users.

Finally, the approximated matrix Q_i' is computed in an analogous way as before as the Hadamard product of matrices M_i' and V_i':

$$Q_i' = M_i' \circ V_i'. \tag{17}$$

Therefore, the actual number of approximation matrices to use is a trade-off between accuracy and storage cost.

5. Experiments and results

This section has two main parts. In the first one, we validate both the full and the approximated methods. In the second one, we test the proposed methodology. For both studies, we set a common scenario where a set of users visualize a number of contents using a generic viewport-adaptive system.

In the first study, we compare the performance of different approaches in terms of geometric accuracy, computation time, and resulting quality. First, we test the full method (VAQM) using different points per side. Second, we check the validity of the approximated method (AVAQM) using different numbers of pre-computed masks: 3x6, 5x10, 10x20 and 20x40. Finally, we use a widely-used

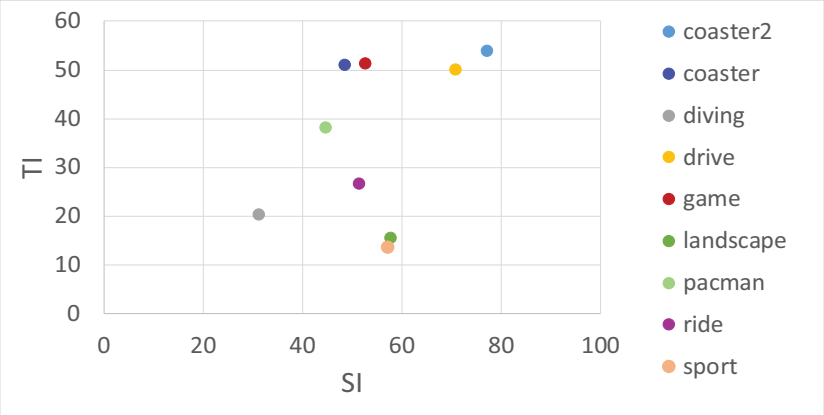


Figure 16: SI and TI of the sequences used in the experiments

coarse method consisting in a non-transformable rectangular viewport. In this approach, the projected viewport is always rectangle-shaped, regardless of its position in the image. a different numbers of pre-calculated masks to evaluate the discrepancy introduced between the quality measured with the AVAQM and the one really perceived by the user, which is provided by the VAQM.

In the latter analysis, we analyze the influence of different parameters in the quality observed by users. It was carried out through two main sets of experiments. The first one considers the effect of the length of the video segments, whereas the second one is focused on the impact of the movements of the user. It is worth mentioning that the scenarios set up for the tests are not intended to be completely realistic or to cover all possible situations, but to be representative and generic enough to be able to properly verify the performance of the proposed methodology. The analysis and conclusions on these tests can be directly extended to other scenarios (including IQA, pooling method, encoding configuration, etc), since the accuracy of the VAQM is independent from their specific characteristics.

We first present the test features that are common to all tests.

Experiment features

For the experiments, we have employed the videos included in the public

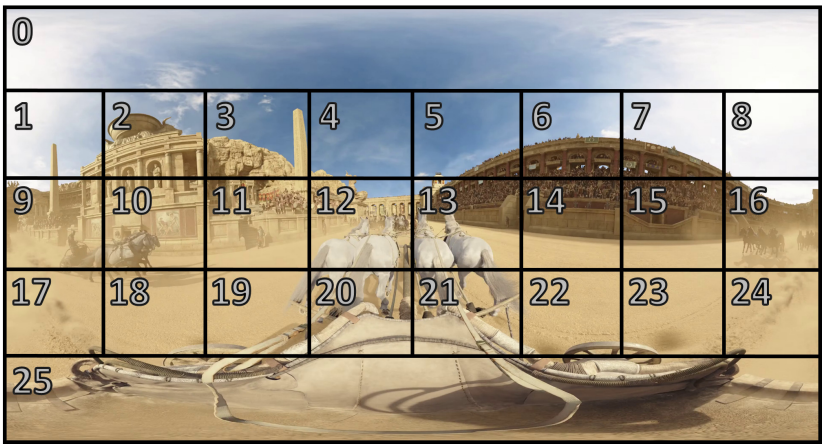


Figure 17: Areas in the 360 image associated to the generated viewport-oriented sequences

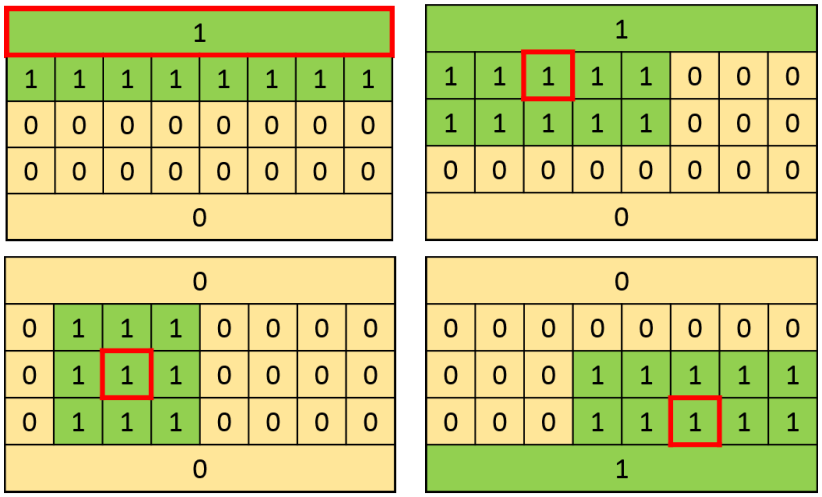


Figure 18: Distribution of qualities corresponding to four different areas. The corresponding area is boxed in red.

dataset 360 Video Viewing Dataset in Head-Mounted Virtual Reality [32]. This dataset is made up of 10 sequences with a resolution of UHD-4K, a frame rate of 30 fps and a duration of one minute. It contains natural and varied content of unequal complexity, as the computed Spatial Information (SI) and Temporal Information (TI) indicators show (see Fig. 16). For each source, it also includes the trajectory followed by 50 different users at one sample per frame.

Regarding the content preparation, the sequences were H.265/HEVC encoded with 5x8 tiles, as we have empirically found it to be a good trade-off between coding efficiency (tiles large enough not to significantly restrict motion search range and introduce much overhead) and sufficient granularity to generate efficient viewport-adaptive content (tiles small enough to allow the creation of a sufficient number of versions of the content to allow a closer adaptation to the user's movements). Each viewport-oriented sequence, that is, the version of the content created to be presented to the user when the center of the viewport lies in a specific area of the sphere, is the result of encoding the source content with a given distribution of qualities. Each of these sequences is associated with one of the non-overlapping areas in the equirectangular image depicted in Figure 17, where the upper and lower stripes of tiles have been merged into one area, as the zones towards the poles get heavily distorted in the equirectangular projection. Thus, we have generated 26 viewport-oriented sequences per video source, which are later on segmented to be used in an ABR platform. To help understand these values, please remember that the viewport covered around 1/6 of the surface of the sphere. Additionally, we assume that virtually the whole motion-to-photon latency of the system corresponds to the time that it takes the segment currently in the process of decoding and presentation (which corresponded to the previous viewport) so that the one that is correctly adapted to the head position can start the same process. Thus, we assume that the time required to download segments is negligible. Furthermore, as a consequence, there are no stalls.

5.1. Validity of the methodology: full and approximated methods

5.1.1. Geometry

Performance in terms of accuracy

In this set of experiments, we have used 5 of the sequences of the dataset ('coaster2', 'diving', 'drive', 'game', and 'landscape') and the trajectories of users 1 to 20. Table 1 shows the performance of different methods in terms of geometric accuracy. For every frame, the geometric accuracy of a given method

Table 1: Geometric accuracy of the different approaches: full method (VAQM) using 20, 15, 10, 5 and 3 points per side, approximated method (AVAQM) using different numbers of pre-computed masks (3x6, 5x10, 10x20 and 20x40); and rectangular projected viewport

Content	VAQM 20 pts	VAQM 15 pts	VAQM 10 pts	VAQM 5 pts	VAQM 3 pts	AVAQM 3x6	AVAQM 5x10	AVAQM 10x20	AVAQM 20x40	Rectangular
coaster2	100.00	99.90	99.74	98.53	93.96	65.22	79.35	88.96	95.16	74.32
diving	100.00	99.92	99.82	99.01	96.02	60.94	80.00	87.54	96.04	60.12
drive	100.00	99.91	99.75	98.57	94.16	69.29	80.24	89.14	95.36	73.74
game	100.00	99.91	99.75	98.56	94.10	66.79	80.72	89.18	95.20	75.45
landscape	100.00	99.91	99.76	98.63	94.41	70.73	82.65	88.61	95.24	74.18
Average	100.00	99.91	99.76	98.66	94.53	66.59	80.59	88.69	95.40	71.56

is computed as the portion of pixels belonging to the best possible mask that are also included in the mask obtained using that method. Hence, the more pixels per image belong to both masks simultaneously, the more accurate the method is. The table presents the results per content, as a result of averaging the results of all frames and users, and the global value, which results from averaging the results of all frames, users and contents. The approaches under study are: the full method (VAQM) using 20, 15, 10, 5 and 3 points per side; the approximated method (AVAQM) using 3x6, 5x10, 10x20 and 20x40 pre-computed masks, and a method that uses a rectangle-shaped projected viewport.

We assume that the VAQM using 20 points per side already provides the highest possible geometric accuracy to all intents and purposes. This assumption is substantiated by the results, where, as can be observed in the table, the accuracy converges quickly as the number of points used per side is increased. Thus, using more points per side becomes less and less useful. Indeed, the VAQM with 10 points per side already obtains a figure well above 99%.

Regarding the AVAQM, we study the loss of geometric accuracy that brings the use of M_i' instead of M_i. The accuracy obtained by the method depends completely on the number and distribution of pre-computed masks involved. In these experiments, in all the cases, the pre-computed masks are distributed uniformly in the image, but their number differs. As we can see, this factor impacts very significantly on the geometric accuracy. The more pre-computed

masks are used, the more probable it will be to find a mask that highly overlaps with the real one, and so the more accurate the result will be.

Finally, the method that implements a rectangular projected viewport is used as a means to compare the proposed methods with coarse approximations used by many designers. The main advantage of this approach is its extreme simplicity, which, as will be seen next, translates into a very fast algorithm. However, as can be seen in the table, its lack of accuracy is notable.

Table 2: Computation time per frame in milliseconds of the different approaches: full method (VAQM) using 20, 15, 10, 5 and 3 points per side, approximated method (AVAQM) using different numbers of pre-computed masks (3x6, 5x10, 10x20 and 20x40); and rectangular projected viewport

Content	VAQM 20 pts	VAQM 15 pts	VAQM 10 pts	VAQM 5 pts	VAQM 3 pts	AVAQM 3x6	AVAQM 5x10	AVAQM 10x20	AVAQM 20x40	Rectangular
coaster2	40.21	39.19	39.16	38.76	38.01	0.02	0.02	0.02	0.02	2.96
diving	42.99	42.11	42.10	41.71	40.84	0.02	0.02	0.02	0.02	2.90
drive	40.15	39.21	38.85	38.44	37.71	0.02	0.02	0.02	0.02	2.91
game	38.63	37.97	37.47	36.90	36.26	0.02	0.02	0.02	0.02	2.77
landscape	43.40	41.41	41.13	40.56	39.82	0.02	0.02	0.02	0.02	3.06
Average	41.08	39.98	39.74	39.27	38.53	0.02	0.02	0.02	0.02	2.92

Performance in terms of time

In Table 2, we present the the performance of the same methods introduced above in terms of average computation time per frame in milliseconds (with a 8-core CPU clocked at 1.80 GHz with 15.6 GiB RAM). These figures give a sense of the complexity of the operations involved in obtaining the projected viewport. Naturally, they do not include the time required to apply any IQA methods. We can see that the VAQM methods are rather close to but cannot perform in real time, that is, faster than 33 ms per frame, as the framerate of the videos is 30 fps. However, the similarity between the figures obtained when using different numbers of points per side reveals that most of the elapsed time is not used in computing the base projected viewport (furthermore, it only needs to be computed once) or rotating it, but in joining the points and filling the mask (actually, it takes around 90% of the time). Many operations, like

the rotation of a point, which is independent from that of the rest, or the ones related to the generation of the convex hull and filling the region within, could be nevertheless easily optimized and paralellized. Hence, the full method for computing the mask could be accelerated so as to be able to run in real time in a general-purpose machine with little effort.

On the contrary, the time elapsed to carry out the AVAQM is almost negligible, as it takes no more than using a lookup table. We can also see that it does not vary with the number of pre-computed masks. Therefore, from this perspective, the more pre-computed masks are used, the better, as the accuracy increases and the computing time remains the same. However, in order for this approach to work, all pre-computed masks need to be cached simultaneously or it should be guaranteed somehow that they can be loaded to be used whenever they are required, which might be a problem for devices with limited capabilities. Indeed, in the former approach, assuming that it only takes one bit per pixel to indicate whether it belongs to the projected viewport, it will require approx. 7.4 MB per mask. Thus, several GB could be necessary to cache all the pre-computed masks. These difficulties could be nevertheless partially overcome by using encoding strategies to improve compression and so reduce the storage cost of the masks or by implementing smart caching strategies.

Additionally, as already mentioned, the time required to compute the rectangle-shaped mask is very low. In particular, it is much lower than the time elapsed to apply the VAQM. The main reason behind this significant difference is that the process of joining the points and filling the mask is much more efficient with a rectangular viewport.

5.1.2. Quality performance: geometry plus grade

Finally, we have tested the performance of all the methods in terms of quality accuracy. To that end, we have used five approaches to obtain V_i: as the result of the application of four different IQAs (PSNR, SSIM, MS-SSIM and VMAF) and using the QP employed to encode the basic processing units in the image, as described before.

Table 3: QP value employed to encode the HQ areas to meet the target bitrate for different contents

	coaster2	diving	drive	game	landscape
20 Mbps	18	11	13	15	16
12 Mbps	24	14	19	20	20
7 Mbps	31	22	27	27	25
3 Mbps	42	32	38	37	32

The sequences used in this part where encoded using two QP values: one for the LQ areas and another for the HQ areas. The QP for the LQ areas was always set to 51, whereas that of the HQ areas varied with the content to reach given target bit rates. We used four different target bitrates to cover many possible image qualities: 3, 8, 14, and 20 Mbps. The associated QPs for the HQ areas are depicted in Table 3. A segment length of 500 ms was selected.

Table 4 expresses the global average quality obtained frame by frame for the same contents and users as before using the different methods and applying the five grades just mentioned. In particular, it includes the actual figure obtained for the VAQM using 20 points per side and the absolute difference between this value and the result for the masks obtained applying the rest of the methods. In the same way as before, the values in the table are the outcome of averaging the results for all the frames in the sequence, for all the contents, and for all the users. The lower it is the difference, the more accurate the method is.

It can be seen that the differences between methods observed before also appear in this case. Naturally, the differences are not as big as for the geometric-only case, since the HQ areas include a greater portion of the image than that covered by the masks, and so inaccuracies need to be bigger to lead to the introduction of errors. Nevertheless, the differences in performance in terms of measured quality between methods are significant. The more points per side are used in the full method, the more accurate the measure. However, as can be observed, using a rather low number of points does not impact much on the quality measure. Regarding the approximated method, again, the more masks are used, the more accurate the method is. Nevertheless, a rather low number

Table 4: Quality accuracy of the different approaches: full method (VAQM) using 20, 15, 10, 5 and 3 points per side, approximated method (AVAQM) using different numbers of pre-computed masks (3x6, 5x10, 10x20 and 20x40); and rectangular projected viewport

Metric	Bit rate (Mbps)	VAQM 20 pts Abs.	VAQM 15 pts Diff.	VAQM 10 pts Diff.	VAQM 5 pts Diff.	VAQM 3 pts Diff.	AVAQM 3x6 Diff.	AVAQM 5x10 Diff.	AVAQM 10x20 Diff.	AVAQM 20x40 Diff.	Rectangular Diff.
QP-based	20	20.465	0.007	0.017	0.098	0.409	2.918	1.609	0.869	0.352	2.141
	12	23.955	0.006	0.015	0.087	0.362	2.585	1.425	0.770	0.312	1.896
	7	28.317	0.005	0.013	0.073	0.304	2.168	1.195	0.646	0.261	1.590
	3	34.424	0.004	0.009	0.053	0.222	1.584	0.874	0.472	0.191	1.162
PSNR (dB)	20	46.701	0.004	0.011	0.062	0.259	1.852	1.021	0.551	0.223	1.359
	12	44.546	0.004	0.010	0.055	0.231	1.646	0.908	0.490	0.198	1.207
	7	41.862	0.003	0.008	0.047	0.195	1.390	0.766	0.414	0.168	1.019
	3	37.361	0.002	0.006	0.032	0.134	0.959	0.529	0.286	0.116	0.704
SSIM	20	0.975	0.000	0.000	0.001	0.002	0.016	0.009	0.005	0.002	0.012
	12	0.975	0.000	0.000	0.001	0.002	0.016	0.009	0.005	0.002	0.012
	7	0.974	0.000	0.000	0.001	0.002	0.016	0.009	0.005	0.002	0.012
	3	0.970	0.000	0.000	0.001	0.002	0.016	0.009	0.005	0.002	0.012
MS-SSIM	20	0.971	0.000	0.000	0.001	0.003	0.018	0.010	0.005	0.002	0.013
	12	0.970	0.000	0.000	0.001	0.003	0.018	0.010	0.005	0.002	0.013
	7	0.968	0.000	0.000	0.001	0.003	0.018	0.010	0.005	0.002	0.013
	3	0.957	0.000	0.000	0.001	0.002	0.017	0.009	0.005	0.002	0.012
VMAF	20	87.639	0.017	0.043	0.245	1.026	7.325	4.039	2.181	0.883	5.373
	12	86.636	0.016	0.043	0.242	1.013	7.229	3.986	2.153	0.872	5.303
	7	83.347	0.016	0.041	0.232	0.969	6.915	3.813	2.059	0.834	5.072
	3	72.128	0.013	0.034	0.196	0.818	5.842	3.222	1.740	0.704	4.286

of masks, 10x20, is enough to provide sufficiently accurate results on average. As for the results of the method using a rectangle-shaped projected viewport, they differ significantly from those of the full method. This means that using this rather popular coarse approximation leads to the introduction of errors in the measure of the quality really perceived by the users.

5.2. Test of the methodology

Regarding the configuration related with the proposed methodology, the experiments have been performed using only the approximated version of the methodology, that is, with pre-calculated masks for the viewport projections. As said before, we have assumed a FoV of 100^o horizontally and 85^o vertically. Based on the results shown in Table 4, we have used 10x20 pre-calculated masks. Finally, we have implemented a rather simplistic IQA that reflects the portion

of high-quality image that lies within the viewport of the user, as we believe that it is very illustrative of the functioning of the methodology. To that end, for the set of pre-calculated qualities, we have considered the use of the values 1 and 0 for the high and low quality, respectively, regardless of the actual pixel values. Four examples of the distribution of both values in the areas in as many viewport-oriented sequences is depicted in Figure 18. Thus, this IQA provides a value between 0 and 1 for every frame presented through the HMD, where the higher the value, the greater the portion of the image perceived with high quality. Nevertheless, as mentioned, any other IQA could be used, in accordance with the objectives of the system. Evaluating the performance of different IQAs is out of the scope of this work. However, one can easily find relevant studies addressing the matter and drawing primary conclusions [33, 34].

Finally, the timeline considered for each session is that of the duration of the sequence presented to the user. Therefore, the window for the temporal pooling comprises 1800 frames (one minute at 30 fps). Furthermore, besides the average quality provided within the viewport throughout the considered temporal window (q_{w}), we also assess the quality of the session by means of another metric: the percentage of frames with a quality value over a given threshold. We have arbitrarily selected this value to be 80% of the maximum quality (f_{w} imposing $T_{\mathrm{Q}} = 0.8$).

So, in particular, we have used the described methodology to evaluate the influence of three key elements in the quality observed by users: the segment length, the content and the user. To that end, we have computed the quality during a specific session using the IQA mentioned above. With the aim of presenting an understandable study, we have carried out a one-vs-one analysis of the variables, removing in each case the impact of the third variable under study. Therefore, we present in Table 5 the impact of the content and the segment length without considering the user, plus the isolated influence of the both elements, in Tables 6 and 7 the impact of the user and the segment length, without considering the content, plus the isolated influence of the user, and in Tables 8 and 9 the impact of the user and the content without considering

the segment length. In the first three tables, we include the results of the two temporal quality pooling approaches proposed in Section 3. In the last two ones, we do not include f_{w} due to a lack of space.

Table 5: Average temporal and spatial quality pooling per content and segment length in terms of q_{w} and f_{w} ($T_{\mathrm{Q}} = 0.8$)

Content	Segment length							
	500 ms		2000 ms		6000 ms		Total	
	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}
coaster	0.98	98.48%	0.92	89.81%	0.85	79.55%	0.92	89.28%
coaster2	0.98	98.47%	0.93	89.57%	0.84	76.85%	0.91	88.29%
diving	0.98	98.55%	0.91	86.10%	0.76	63.45%	0.88	82.70%
drive	0.97	97.50%	0.88	81.95%	0.76	64.43%	0.87	81.29%
game	0.98	97.92%	0.90	85.84%	0.82	74.98%	0.90	86.25%
landscape	0.97	97.30%	0.87	78.21%	0.74	60.65%	0.86	78.72%
pacman	0.98	97.80%	0.92	88.37%	0.84	77.61%	0.92	87.93%
panel	0.97	97.33%	0.88	80.59%	0.71	57.58%	0.85	78.50%
ride	0.98	97.87%	0.90	85.81%	0.81	70.37%	0.90	84.68%
sport	0.97	98.28%	0.90	86.04%	0.78	64.49%	0.89	82.94%
Average	0.97	97.95%	0.90	85.23%	0.79	68.99%		

5.3. Impact of the segment length

We have used segments of three different lengths: 500 ms, 2000 ms and 6000 ms. Table 5 includes the spatial and temporal quality pooling per content and segment length averaged for all users. Tables 6 and 7 show the spatial and temporal quality pooling per user and segment length averaged for all contents. As can be observed, the results for a segment length of 500 ms do not vary much with the content or the user, as the system is able to update the content to the new viewport quickly enough to prevent the user from observing significant portions of low-quality areas. However, as the segment length is increased, the average quality and the percentage of time above the threshold are reduced

Table 6: Average temporal and spatial quality pooling per user and segment length in terms of q_{w} and f_{w} ($T_{\mathrm{Q}} = 0.8$) (Part I: users 1 to 25)

User	Segment length							
	500 ms		2000 ms		6000 ms		Total	
	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}
user 1	0.98	99.36%	0.94	92.87%	0.86	80.63%	0.93	90.95%
user 2	0.98	99.28%	0.93	91.69%	0.86	80.28%	0.92	90.42%
user 3	0.98	98.68%	0.92	87.62%	0.85	77.79%	0.92	88.03%
user 4	0.98	98.16%	0.91	86.33%	0.80	69.74%	0.90	84.74%
user 5	0.98	98.72%	0.93	89.31%	0.86	79.48%	0.92	89.17%
user 6	0.97	97.23%	0.86	76.71%	0.73	58.11%	0.85	77.35%
user 7	0.98	99.08%	0.94	93.91%	0.88	85.42%	0.93	92.80%
user 8	0.96	95.87%	0.83	72.74%	0.67	51.46%	0.82	73.36%
user 9	0.97	97.89%	0.88	81.71%	0.74	59.03%	0.86	79.54%
user 10	0.98	98.97%	0.94	91.37%	0.85	79.98%	0.93	90.11%
user 11	0.98	98.54%	0.92	87.60%	0.76	63.80%	0.89	83.31%
user 12	0.97	97.01%	0.91	85.39%	0.79	67.01%	0.89	83.14%
user 13	0.98	98.13%	0.91	86.97%	0.80	71.38%	0.90	85.49%
user 14	0.98	99.02%	0.94	91.33%	0.85	80.60%	0.92	90.32%
user 15	0.96	95.79%	0.83	76.04%	0.66	53.74%	0.82	75.19%
user 16	0.98	98.61%	0.92	88.07%	0.84	73.79%	0.91	86.82%
user 17	0.96	95.08%	0.84	74.11%	0.70	55.33%	0.83	74.84%
user 18	0.98	98.17%	0.92	87.45%	0.80	68.61%	0.90	84.74%
user 19	0.98	98.91%	0.92	88.46%	0.80	70.92%	0.90	86.10%
user 20	0.97	96.98%	0.88	80.80%	0.77	66.29%	0.87	81.36%
user 21	0.98	98.87%	0.93	89.39%	0.81	70.49%	0.90	86.25%
user 22	0.98	97.99%	0.92	86.74%	0.83	72.37%	0.91	85.70%
user 23	0.96	96.64%	0.86	78.56%	0.73	60.07%	0.85	78.42%
user 24	0.98	99.57%	0.94	92.54%	0.86	80.25%	0.93	90.79%
user 25	0.98	98.35%	0.91	86.16%	0.78	68.68%	0.89	84.40%

Table 7: Average temporal and spatial quality pooling per user and segment length in terms of q_{w} and f_{w} ($T_{\mathrm{Q}} = 0.8$) (Part II: users 26 to 50)

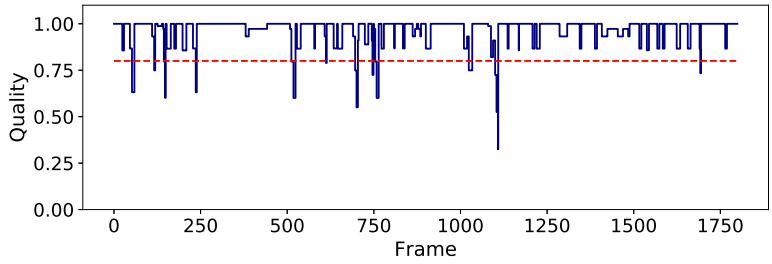
User	Segment length							
	500 ms		2000 ms		6000 ms		Total	
	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}	q_{w}	f_{w}
user 26	0.97	96.24%	0.87	76.68%	0.74	58.29%	0.86	77.07%
user 27	0.98	98.06%	0.92	88.44%	0.82	74.88%	0.91	87.13%
user 28	0.97	96.92%	0.87	78.67%	0.76	61.18%	0.87	78.92%
user 29	0.99	99.94%	0.97	98.17%	0.93	92.05%	0.96	96.72%
user 30	0.97	96.50%	0.85	77.00%	0.69	56.66%	0.84	76.72%
user 31	0.99	99.51%	0.98	97.68%	0.95	92.48%	0.97	96.56%
user 32	0.97	97.99%	0.91	85.86%	0.81	70.25%	0.90	84.70%
user 33	0.98	97.92%	0.92	87.69%	0.81	68.56%	0.90	84.72%
user 34	0.97	97.97%	0.90	82.77%	0.79	64.31%	0.89	81.68%
user 35	0.97	97.96%	0.92	89.32%	0.83	75.03%	0.91	87.44%
user 36	0.97	97.96%	0.90	83.30%	0.78	67.89%	0.88	83.05%
user 37	0.97	97.62%	0.89	83.31%	0.76	65.24%	0.87	82.06%
user 38	0.98	98.46%	0.91	85.12%	0.79	68.57%	0.89	84.05%
user 39	0.97	98.10%	0.90	83.80%	0.75	60.59%	0.87	80.83%
user 40	0.98	99.00%	0.92	89.33%	0.81	73.67%	0.90	87.34%
user 41	0.97	97.14%	0.88	81.62%	0.76	65.12%	0.87	81.30%
user 42	0.97	97.57%	0.88	81.39%	0.74	62.47%	0.87	80.47%
user 43	0.96	96.31%	0.85	78.49%	0.72	59.69%	0.85	78.16%
user 44	0.97	97.85%	0.89	83.79%	0.79	67.52%	0.88	83.05%
user 45	0.97	97.02%	0.85	76.93%	0.68	57.78%	0.83	77.25%
user 46	0.98	99.10%	0.94	92.03%	0.85	78.46%	0.93	89.86%
user 47	0.97	97.79%	0.90	83.92%	0.75	62.84%	0.87	81.52%
user 48	0.97	97.78%	0.90	84.57%	0.80	68.64%	0.89	83.66%
user 49	0.97	97.03%	0.87	79.74%	0.73	58.59%	0.86	78.45%
user 50	0.98	98.91%	0.91	87.97%	0.81	73.75%	0.90	86.88%

Table 8: Average temporal and spatial quality pooling per user and content in terms of q_{w} (Part I: users 1 to 25)

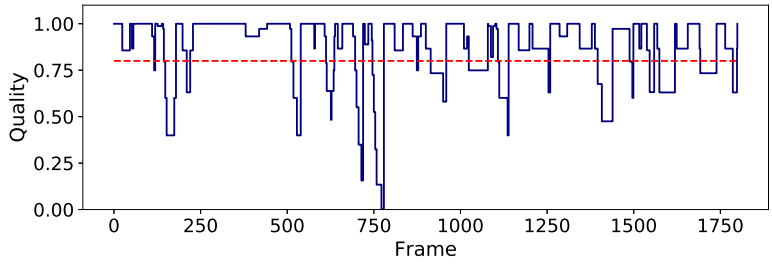
	Content									
User	coaster	coaster2	diving	drive	game	landscape	pacman	panel	ride	sport
user 1	0.97	0.92	0.93	0.92	0.97	0.91	0.98	0.95	0.96	0.94
user 2	0.97	0.97	0.93	0.93	0.96	0.89	0.98	0.87	0.90	0.95
user 3	0.90	0.99	0.92	0.92	0.92	0.91	0.99	0.90	0.91	0.86
user 4	0.94	0.90	0.91	0.91	0.94	0.86	0.95	0.88	0.94	0.91
user 5	0.99	0.99	0.89	0.83	0.98	0.86	0.99	0.84	0.95	0.95
user 6	0.85	0.86	0.88	0.82	0.86	0.85	0.82	0.88	0.90	0.90
user 7	0.91	0.88	0.92	0.97	0.97	0.97	0.93	0.92	0.95	0.95
user 8	0.83	0.80	0.91	0.82	0.83	0.81	0.87	0.80	0.79	0.86
user 9	0.91	0.91	0.89	0.91	0.85	0.87	0.87	0.82	0.93	0.89
user 10	0.98	0.99	0.92	0.91	0.97	0.85	0.98	0.91	0.95	0.95
user 11	0.91	0.97	0.95	0.95	0.89	0.91	0.90	0.91	0.91	0.92
user 12	0.90	0.88	0.97	0.91	0.90	0.93	0.88	0.90	0.93	0.93
user 13	0.93	0.94	0.92	0.94	0.93	0.85	0.98	0.85	0.88	0.91
user 14	0.98	0.96	0.95	0.91	0.93	0.91	0.96	0.89	0.95	0.93
user 15	0.77	0.88	0.86	0.78	0.84	0.84	0.78	0.85	0.88	0.88
user 16	0.96	0.92	0.90	0.89	0.91	0.90	0.95	0.91	0.96	0.93
user 17	0.93	0.90	0.90	0.79	0.78	0.74	0.84	0.74	0.83	0.92
user 18	0.94	0.93	0.95	0.91	0.97	0.91	0.87	0.90	0.92	0.92
user 19	0.95	0.97	0.89	0.88	0.86	0.90	0.96	0.86	0.93	0.96
user 20	0.90	0.93	0.93	0.82	0.91	0.81	0.92	0.90	0.82	0.84
user 21	0.98	0.97	0.91	0.89	0.93	0.93	0.95	0.91	0.92	0.90
user 22	0.92	0.84	0.94	0.94	0.95	0.91	0.95	0.88	0.93	0.95
user 23	0.90	0.87	0.84	0.86	0.85	0.86	0.88	0.83	0.83	0.87
user 24	0.95	0.96	0.96	0.92	0.94	0.92	0.96	0.93	0.95	0.90
user 25	0.96	0.96	0.89	0.88	0.87	0.90	0.93	0.85	0.91	0.91

Table 9: Average temporal and spatial quality pooling per user and content in terms of q_{w} (Part II: users 26 to 50)

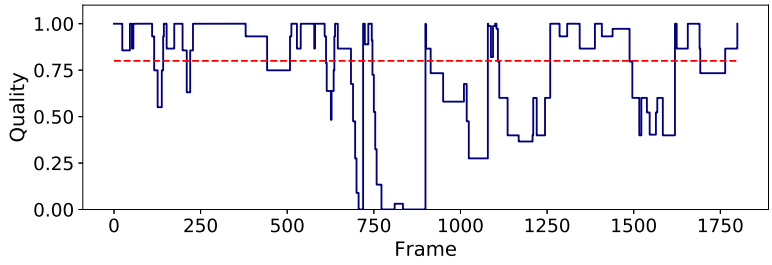
	Content									
User	coaster	coaster2	diving	drive	game	landscape	pacman	panel	ride	sport
user 26	0.94	0.84	0.86	0.88	0.80	0.83	0.91	0.86	0.85	0.89
user 27	0.95	0.99	0.88	0.86	0.91	0.89	0.98	0.86	1.00	0.93
user 28	0.97	0.97	0.89	0.84	0.90	0.74	0.91	0.85	0.86	0.81
user 29	0.97	0.97	0.96	0.99	0.98	0.96	0.98	0.93	0.99	0.95
user 30	0.84	0.74	0.83	0.76	0.85	0.84	0.94	0.89	0.88	0.90
user 31	0.99	0.98	0.97	0.96	0.98	0.99	0.97	0.96	0.99	0.99
user 32	0.90	0.89	0.91	0.88	0.87	0.82	0.97	0.94	0.91	0.95
user 33	0.97	0.98	0.93	0.89	0.90	0.84	0.94	0.89	0.93	0.89
user 34	0.94	0.93	0.88	0.84	0.90	0.85	0.92	0.92	0.89	0.90
user 35	0.96	0.96	0.94	0.86	0.93	0.89	0.93	0.86	0.92	0.91
user 36	0.89	0.93	0.89	0.87	0.93	0.88	0.91	0.88	0.86	0.91
user 37	0.91	0.90	0.88	0.91	0.88	0.86	0.94	0.85	0.87	0.89
user 38	0.94	0.91	0.94	0.92	0.90	0.88	0.96	0.89	0.92	0.86
user 39	0.89	0.88	0.91	0.91	0.90	0.90	0.90	0.90	0.92	0.87
user 40	0.95	0.98	0.96	0.91	0.92	0.88	0.94	0.85	0.94	0.85
user 41	0.88	0.92	0.92	0.89	0.94	0.84	0.81	0.89	0.84	0.90
user 42	0.90	0.87	0.85	0.84	0.90	0.86	0.91	0.93	0.88	0.89
user 43	0.91	0.93	0.84	0.80	0.89	0.85	0.89	0.77	0.80	0.85
user 44	0.89	0.96	0.89	0.88	0.91	0.83	0.97	0.87	0.84	0.86
user 45	0.98	0.99	0.83	0.85	0.80	0.80	0.82	0.77	0.79	0.92
user 46	0.96	0.93	0.94	0.90	0.94	0.92	0.97	0.94	0.95	0.95
user 47	0.89	0.92	0.90	0.89	0.88	0.81	0.93	0.91	0.92	0.90
user 48	0.86	0.97	0.89	0.89	0.91	0.85	0.90	0.91	0.92	0.89
user 49	0.88	0.88	0.90	0.84	0.92	0.86	0.88	0.88	0.87	0.84
user 50	0.96	0.97	0.92	0.97	0.92	0.84	0.90	0.84	0.92	0.89



(a) 500 ms



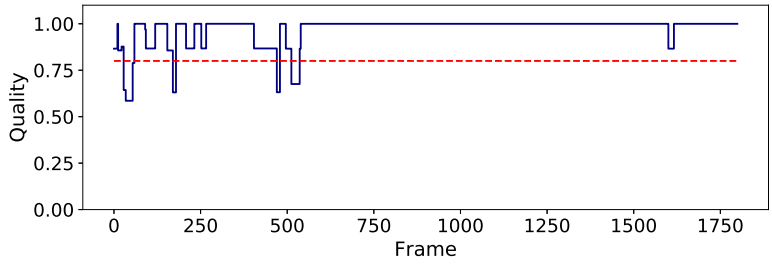
(b) 2000 ms



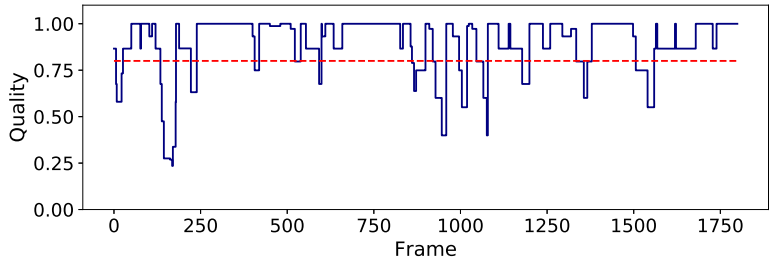
(c) 6000 ms

Figure 19: Evolution of the spatial quality pooling over time for user 32 and content 'game' using three different segment lengths. The dashed red line shows the 80% threshold ($T_{\mathrm{Q}} = 0.8$)

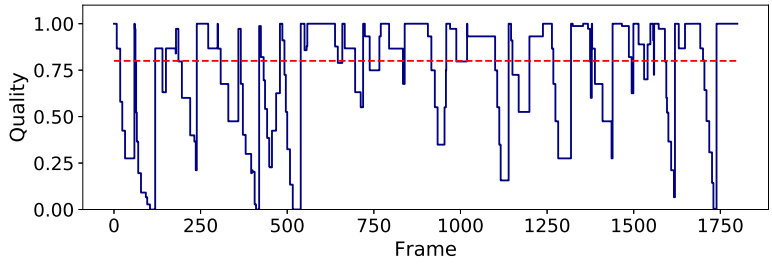
regardless of the user and content. This is due to the fact that the content will more likely be out of date, that is, not centered in the viewport, for a longer period of time. Additionally, there are clear differences between users and contents. More curious users (e.g. user 8) and more exploratory contents (e.g. landscape) get more affected by the use of longer segments, as in these cases, it is more likely for users to observe low-quality portions of the image. We will see these effect more in detail in the following subsections. Moreover, Figure 19



(a) Content 'coaster': driven



(b) Content 'game': neither driven nor exploratory



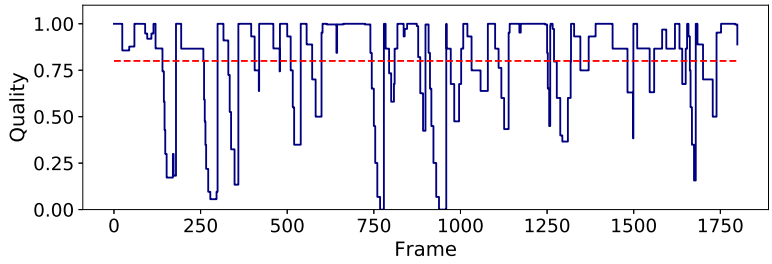
(c) Content 'landscape': exploratory

Figure 20: Evolution of the spatial quality pooling over time for user 28, segment length 2000 ms, and three different contents. The dashed red line shows the 80% threshold ($T_{\mathrm{Q}} = 0.8$)

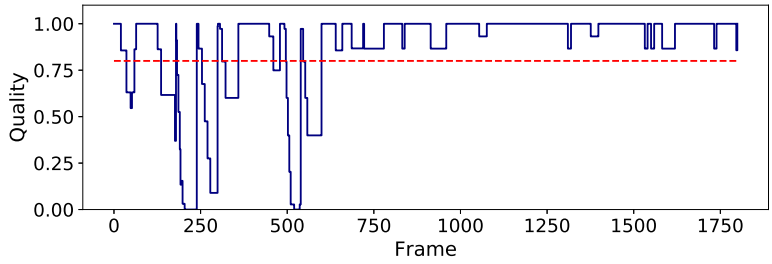
depicts a representative example showing the quality over time observed by user 32 watching content 'game' for the three segment lengths, which evidences the impact of the latter.

5.4. Impact of the content

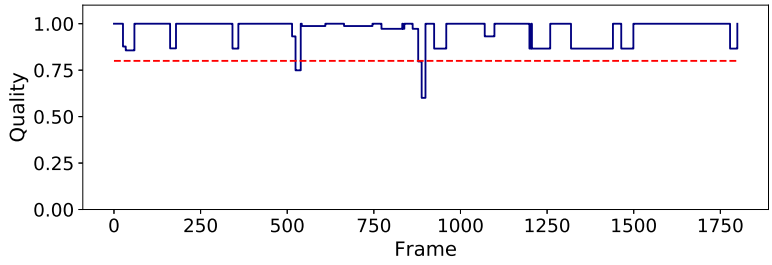
In this subsection, we analyze the degree to which the nature of the content boosts exploration across the image, which can notably impact the quality per-



(a) User 8: curious user



(b) User 42: medium user



(c) User 11: quiet user

Figure 21: Evolution of the spatial quality pooling over time for content 'coaster2', segment length 2000 ms, and three different users. The dashed red line shows the 80% threshold ($T_{\mathrm{Q}} = 0.8$)

ceived by users in viewport-adaptive schemes. As mentioned, Table 5 includes the spatial and temporal quality pooling per content and segment length averaged for all users. Tables 8 and 9 include the spatial and temporal quality pooling per user and content averaged for all segment lengths. The results show that there exist significant differences between contents. The more exploratory it is the content presented to the user, the more it encourages him/her to move,

increasing the likelihood of quality drops. Indeed, more driven contents like 'coaster' end up delivering better overall quality than contents like 'game', which cannot be described as either driven or exploratory. Finally, sessions including the visualization of more exploratory contents like 'landscape' typically end up delivering a significantly lower quality on average. As an illustrative example, we present in Fig. 20 the evolution for user 8 of the observed quality for three sessions with different contents.

5.5. Impact of the user

Finally, we study the influence of the user's behavior in the degree of exploration. As mentioned, Tables 6 and 7 show the spatial and temporal quality pooling per user and segment length averaged for all contents, whereas Tables 8 and 9 include the spatial and temporal quality pooling per user and content for a segment length of 2000 ms. The results show that the more active the user is, the more he/she moves, and so the more quality changes along the session. As before, this type of session correlates with a lower average quality. We can easily find completely opposed examples of user behavior in the tables. For instance, users 8, 17 or 30 are very curious users, whereas users 11, 31, 46 are rather static. As a representative example, Figure 21 depicts the results for three users that could be classified as curious, medium and quiet.

6. Conclusions and future work

The accurate assessment of the quality perceived by users throughout a 360VR video visualization session is key in the design of robust specific encoding and transmission strategies. In particular, the strict requirements to provide 360VR content with good quality have led to the development of many different viewport-adaptation strategies aiming at offering the best possible quality while saving bitrate. To properly evaluate these schemes, not only the saved bitrate, but also the quality of the portion of the scene actually presented through the HMD at all times should be consider. In this paper, we have proposed a

methodology to accurately assess the quality inside the viewport around the user's point of gaze at every moment. This methodology has been made possible thanks to a complete analysis of the geometric relations involved in this particular environment, also detailed in the paper.

The proposed procedure is highly flexible and allows for any trade-off between accuracy and computational load. This is done by selecting the degree of approximations that best suits the specific requirements of the scenario. These options enable the use of the proposed methodology both offline and online, depending on the needs of the system.

Finally, we have shown its operation through a set of descriptive experiments. In particular, we have tested the effect of different essential factors on the observed quality, such as the length of the segments and the amount of movement of the user along the session. The analysis of the results validates the capability of the proposed methods to assess the quality perceived by users from different perspectives.

Going forward, this work could be extended to include an 'accessory' methodology aiming at estimating the location of future viewport locations based on past observations [35, 36], which could lead to the generation of likely future user visualization paths. Those paths, used together with the approximated masks, could lead to huge savings in terms of computation and time. This estimation can help demarcate the area where the new mask will most likely be, saving time that could be used to refine the mask.

Acknowledgments

This work has been partially supported by the Ministerio de Ciencia, Innovación y Universidades (AEI/FEDER) of the Spanish Government under project TEC2016-75981 (IVME).

References

[1] L. Freina, M. Ott . A Literature Review on Immersive Virtual Reality in

Education: State Of The Art and Perspectives. In: eLearning and Software for Education (eLSE). 2015,.

[2] J. H. Seo, B. M. Smith, M. Cook, E. Malone, M. Pine, S. Leal, Z. Bai, J. Suh . Anatomy builder VR: applying a constructive learning method in the virtual reality canine skeletal system. In: Springer International Conference on Applied Human Factors and Ergonomics. 2017, p. 245–52.

[3] K. Brunnström, M. Sjöström, M. Imran, M. Pettersson, M. Johanson . Quality Of Experience For A Virtual Reality Simulator. In: Human Vision and Electronic Imaging (HVEI). 2018,.

[4] R.S. Allison, K. Brunnström, D.M. Chandler, H. Colett, P. Corriveau, S. Daly, J. Goel, J.Y. Long, L.M. Wilcox, Y. Yaacob, S.-N. Yang, Y. Zhang . Perspectives on the definition of visually lossless quality for mobile and large format displays. Journal of Electronic Imaging 2018;27(5):1–23.

[5] R. G. de A. Azevedo, N. Birkbeck, F. De Simone, I. Janatra, B. Adsumilli, P. Frossard . Visual Distortions in 360-degree Videos. IEEE Transactions on Circuits and Systems for Video Technology 2019;Early Access:1–.

[6] Y. Rai, J. Gutiérrez, P. Le Callet . A Dataset of Head and Eye Movements for 360 Degree Images. In: ACM on Multimedia Systems Conference (MMSys). 2017, p. 205–10.

[7] G. J. Sullivan, J.-R. Ohm,, W.-J. Han, T. Wiegand . Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology 2012;22(12):1649–68.

[8] P. Lambert, W. De Neve, Y. Dhondt, R. Van de Walle . Flexible macroblock ordering in H.264/AVC. Elsevier Journal of Visual Communication and Image Representation 2006;17(2):358–75.

[9] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, R. Zimmermann . A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. IEEE Communications Surveys Tutorials 2019;21(1):562–85.

[10] L. Bedogni, M. Di Felice, L. Bononi . Dynamic segment size selection in HTTP based adaptive video streaming. In: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2017, p. 665–70.

[11] C. Díaz, J. Cabrera, M. Orduna, L. Muñoz, P. Pérez, J. J. Ruiz, N. García . Viability analysis of content preparation configurations to deliver 360VR video via MPEG-DASH technology. In: IEEE International Conference on Consumer Electronics (ICCE). 2018, p. 1–2.

[12] Y. Sanchez, D. Podborski, C. Hellge, T. Schierl . Shifted IDR representations for low delay live DASH streaming using HEVC tiles. In: IEEE International Symposium on Multimedia (ISM). 2016, p. 87–92.

[13] G. Zhai, K. Gu, J. Wang, W. Samek . Quality perception of advanced multimedia systems. Digital Signal Processing 2019;91:1 – 2.

[14] R. Ghaznavi-Youvalari, M. M. Hannuksela, A. Aminlou, M. Gabbouj . Viewport-dependent delivery schemes for stereoscopic panoramic video. In: 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON). 2017, p. 1–4.

[15] M. Hosseini . View-aware tile-based adaptations in 360 virtual reality video streaming. In: IEEE Virtual Reality (VR). 2017, p. 423–4.

[16] A. Zare,, A. Aminlou, M. M. Hannuksela, M. Gabbouj . HEVC-compliant tile-based streaming of panoramic video for virtual reality applications. In: ACM on Multimedia Conference. 2016, p. 601–5.

[17] C. Ozcinar, A. De Abreu, A. Smolic . Viewport-aware adaptive 360 video streaming using tiles for virtual reality. In: IEEE International Conference on Image Processing (ICIP). 2017, p. 2174–8.

[18] C. Timmerer, M. Graf, C. Mueller . Adaptive Streaming of VR/360-Degree Immersive Media Services with High QoE. In: NAB Broadcast Engineering and IT Conf. 2017,.

[19] S. Xie, Y. Xu, Q. Qian, Q. Shen, Z. Ma, W. Zhang . Modeling the perceptual impact of viewport adaptation for immersive video. In: IEEE International Symposium on Circuits and Systems (ISCAS). 2018, p. 1–5.

[20] S. Petrangeli, V. Swaminathan, M. Hosseini, F. De Turck . An HTTP/2-based adaptive streaming framework for 360 virtual reality videos. In: ACM on Multimedia Conference. 2017, p. 306–14.

[21] Z. Chen, Y. Li, Y. Zhang . Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation. Signal Processing 2018;146:66 – 78.

[22] D. Liu, P. An, R. Ma, W. Zhan, L. Ai . Scalable Omnidirectional Video Coding for Real-Time Virtual Reality Applications. IEEE Access 2018;6:56323–32.

[23] Y. Zhu, G. Zhai, X. Min . The prediction of head and eye movement for 360 degree images. Signal Processing: Image Communication 2018;69:15 – 25.

[24] G. He, J. Hu, H. Jiang, Y. Li . Scalable Video Coding Based on User's View for Real-Time Virtual Reality Applications. IEEE Communications Letters 2018;22(1):25–8.

[25] D. Podborski, E. Thomas, M. M. Hannuksela, S. Oh, T. Stockhammer, S. Pham . 360-Degree Video Streaming with MPEG-DASH. SMPTE Motion Imaging Journal 2018;127(7):20–7.

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

[27] C. Brown . Bringing pixels front and center in VR video. https://bloggoogle/products/google-ar-vr/bringing-pixels-front-and-center-vr-video/ Accessed: 2018-12-28;.

[28] I. Todhunter . Spherical trigonometry (5th ed.). Macmillan; 1886.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli . Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004;13(4):600–12.

[30] Z. Wang, E. P. Simoncelli, A. C. Bovik . Multiscale structural similarity for image quality assessment. In: IEEE Asilomar Conference on Signals, Systems & Computers; vol. 2. 2003, p. 1398–402.

[31] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, M. Manohara . Toward A Practical Perceptual Video Quality Metric. https://mediumcom/netflix-techblog/toward-a-practical-perceptual-video-quality-metric-653f208b9652 2016;.

[32] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, C.-H. Hsu . 360 video viewing dataset in head-mounted virtual reality. In: ACM on Multimedia Systems Conference. 2017, p. 211–6.

[33] E. Upenik, M. Rerabek, T. Ebrahimi . On the performance of objective metrics for omnidirectional visual content. In: International Conference on Quality of Multimedia Experience (QoMEX). 2017, p. 1–6.

[34] M. Orduna, C. Díaz, L. Muñoz, P. Pérez, I. Benito, N. García . Video Multimethod Assessment Fusion (VMAF) on 360VR contents. IEEE Transactions on Consumer Electronics 2020;66(1):22–31.

[35] Y. Zhu and G. Zhai and X. Min and J. Zhou . The Prediction of Saliency Map for Head and Eye Movements in 360 Degree Images. IEEE Transactions on Multimedia 2019;.

[36] E. J. David, J. Gutiérrez, A. Coutrot, M. P. Da Silva, P. Le Callet . A dataset of head and eye movements for 360 videos. In: Proceedings of the 9th ACM Multimedia Systems Conference. 2018, p. 432–7.