Visual Assessment of Equirectangular images for Virtual Reality Applications In Unity

Adriano Gil, Aasim khurshid, Juliana Postal, Thiago Figueira
SIDIA Instituto de Tecnologia
Manaus, Brazil
Email: {adriano.gil, aasim.khurshid, juliana.postal, thiago.figueira} @sidia.com

***Abstract*— Virtual Reality (VR) applications provide an immersive experience when using panoramic images that contain a 360-degree view of the scene. Currently, the equirectangular image format is the widely used pattern to represent these panoramic images. The development of a virtual reality viewer of panoramic images should consider several parameters that define the quality of the rendered image. Such parameters include resolution configurations, texture-to-objects mappings and deciding from different rendering approach, but to select the optimal value of these parameters, visual quality analysis is required. In this work, we propose a tool integrated within Unity editor to automate this quality assessment using different settings for the visualization of equirectangular images. We compare the texture mapping of a skybox with a procedural sphere and a cubemap using full-reference objective metrics for Image Quality Analysis (IQA). Based on the assessment results, the tool decides how the final image will be rendered at the target device to produce a visually pleasing and high-quality image.**

I. Introduction

Images captured in the 360-degree surroundings of a single point are capable of replicating the entire visual information available from that position. Specific cameras are designed to capture such images as the Samsung Gear 360°, which captures panoramic images and stores them in a suitable format for 360-degree visualization. These images are stored in a specific format to facilitate image processing. The equirectangular format is the most common one. Furthermore, equirectangular images are used in Virtual Reality applications to provide an immersive user experience and allow users to explore the virtual environment from a first-person perspective.

Virtual Reality (VR) devices use a different picture for each eye to simulate depth and increase the sense of presence in the application. Even with the recent technological advancements, VR technology still encounters technical challenges such as offering a high density of pixels per Field of View (FoV) degree. Besides, the human eye has a resolution of 60 pixels per degree, which indicates that a 100-degree capable device should theoretically render the available content at 6k resolution to increase immersion and realism [1].

A 360 image viewer renders its content in a sphere to place visual elements as they would be observed by the user in a real environment. The investigation for the best visual quality involves selecting one in a given set of distinct formats each one with different distortion levels along the existing 360 degrees. To determine the suitable format and resolution for a 360 image it is important to take into consideration the device in which this image will be presented hence the requirement of a tool that can simulate devices and analyze image settings so it can give the most fitting choice.

There are two approaches for image quality assessment: subjective and objective metrics. Subjective Image Quality Analysis (IQA) employs human observers to evaluate and score a sequence of pictures whereas Objective IQA builds mathematical models for automatic image quality assessment. Subjective IQA is accurate, but it is expensive and time-consuming. On the other hand, objective IQA may provide a cost-effective solution for image analysis and can be embedded in VR applications. To build VR applications, various tools have been proposed such as Unity Editor, Source and Panda3D to name a few.

Unity Editor is a game development engine for computers, mobile, console, virtual and augmented reality. It is both used by small development groups as well as big corporations such as Microsoft; it is also the most used development tool for virtual reality. From its GitHub account[1], Unity provides built-in solutions for rendering panoramic images. The developer is also able to create customized solutions through shaders, i.e., code that is executed in the GPU and influences how 3D elements are displayed on the device screen.

In this paper, we propose an equirectangular image quality assessment tool which employs objective metrics integrated into the Unity Editor. To make assessments close to real case scenarios, our tool is capable of simulating visualization with the field of view and resolution values provided by the user. Figure 1 below presents the Unity Editor interface we built. By configuring specific values for the field of view and screenshot (image) resolution, it is possible to simulate the viewport of a given device. For instance, a 101-degree field of view and 1440 x 1480 resolution is close to the user view inside a VR application running on a Samsung S9 device. The screenshot direction field lets the user insert three-dimensional vectors indicating the direction each screenshot will be pointing to. Finally, a report and graphs are generated according to the chosen metrics.

The rest of the paper is organized as follows: Section II reviews the most relevant work in equirectangular images and IQA. Next, the proposed method is detailed in Section III. Furthermore, the preliminary experimental results are presented

[1] https://github.com/Unity-Technologies/SkyboxPanoramicShader

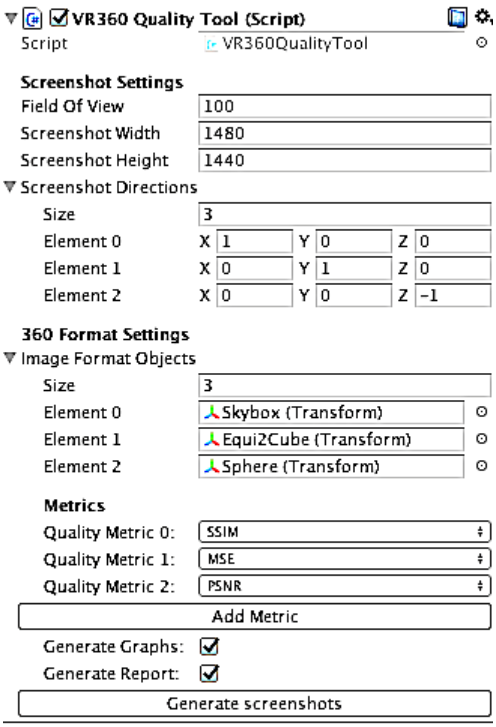

Fig. 1. Proposed tool embedded in Unity interface.

and discussed in Section IV. Finally, conclusions and future work perspectives are given in Section V.

II. Related Work

VR applications differ from other vision applications due to their innate concern to provide content to all possible VR viewpoints. Furthermore, VR headsets can isolate the spectator both visually and acoustically from the real world [2]. Spherical panoramic content may be presented in different projection types: equirectangular (ERP), rectilinear, doughnut, cube-map, and multiview [3]. Moreover, the format of the 360-degree image plays a vital role in the resolution and uniformity of such images [4]. For example, equirectangular images have a high resolution on the poles and high uniformity on the equator line; On the other hand, cube-map images have a high density on the edges and high uniformity in each face of the diagonal [5]. For all these projection types, the quality of the images selected to create a panoramic view is crucial to present the panoramic content.

Image Quality Analysis (IQA) assesses the quality of the images in the creation of panoramic view. IQA is categorized into two types, i.e., subjective analysis and objective analysis [6]. The most reliable strategy to evaluate image quality is through subjective analysis. In this case, human observers assess a set of pictures and score it on a scale from 1 (worst) to 5 (best), this technique is called MOS (Mean Opinion Score) and it calculates the average score given a set of scores for each sample. Pinson et al. compared and analyzed various methodologies for subjective video quality assessment [7]. These test methods include single or double stimulus methods. In single stimulus methods, only one impaired video stream is used for assessment. In double stimulus, however, a reference videos is also provided to the user for comparative analysis [7].

Although subjective IQA methods are precise, they are inconvenient and expensive, especially when a VR environment is evaluated because it is more immersive. Thus, a complementary objective metric would be useful and less expensive. Objective metrics are defined as Full-Reference (FR), No-Reference (NR), and Reduced-Reference (RR) [8]. In FR-IQA methods, a non-distorted reference image is available to evaluate the test image. In RR-IQA methods, a reference image with only selective information is available to compare and measure the quality of the distorted image. Whereas in NR-IQA methods, there is no reference image available to compare, the only image available is the one in which quality is measured.

Prior work on 360-degree IQA is discussed in Quality metric for spherical panoramic video [3]. In this work, we focus on Full-reference IQA. Recent advancements in FR-IQA methods are comprehensively discussed in FR stereo image quality assessment using natural stereo scene statistics [9]. We propose an FR quality assessment tool for the selection of images before using them to render on the target device. Despite our focus on image quality assessment of 360-degree spherical panoramic images, our proposal only assesses screenshots obtained from texture projections inside Unity3D. Therefore, our final target are 2D images as a result of such projections; which is intuitive because still 360-degree images or video-sequences are encoded and transmitted in the 2D format under sphere-to-plane projection.

Similar work can be found at [10] as it proposes a tool for equirectangular assessment as well. The main difference in regards to our current paper is that [10] considers this assessment a problem of Full-Reference (FR) IQA while this work approaches this problem as a No-Reference (NR) IQA. It is debatable if using a skybox as a reference image is better than evaluating different images without reference. At first, it is difficult to claim that skybox rendering is the best format for all cases. However, it is challenging to capture subjective feelings using objective metrics alone.

III. Proposed Method

In this work, we propose a full-reference objective IQA tool to analyze viewport-based patches of panoramic images. Our implementation runs as a customs inspector script in Unity that mimics how the final image is going to be rendered at the target device. Different rendering implementations can be tested using the same equirectangular image. We also describe our implementation of converting an equirectangular image into a cubemap format.

Our solution is a fully-automatic way to compare different rendering configurations of spherical images. After choosing an equirectangular image, a target resolution, a field of view in degrees, a set of viewing directions and a set of rendering solutions, different images patches are generated according

to an approximated viewport inside the panoramic exhibition. Each image patch is evaluated according to objective metrics using a reference patch. In order to compare different rendering implementations, we chose Skybox rendering as the reference image, considering its widespread usage in game and virtual reality applications.

The standard implementation of a spherical image viewer makes use of a sphere with inverted normals. Due to their characteristics, equirectangular images are distorted at poles, while cubemaps are distorted at their corners [4]. That is why it is interesting to test an image in different formats. In this section, we describe a shader-based implementation to fit equirectangular images in a cubemap visual representation.

A. Projecting 360°Images to UV Mapping

Panoramic images comprehend the entire field-of-view of the user. Considering the equirectangular format, some mapping implementations are listed below:

1) Utilize a sphere mesh to render the 360-degree image inside it;
2) Utilize a Skybox to render the 360-degree image on the background;
3) Map the 360-degree image to UV positions of a cubic mesh.

Each mapping possibility has its advantages and disadvantages in terms of resolution offered by angular direction and general distortion of the 360-degree image.

B. Mapping Equirectangular Images to a Sphere

For mapping equirectangular images to sphere, we adopt the standard UV mapping technique for spheres which is based on the latitude/longitude approach. That means we need to find a three-dimension coordinate (x, y, z) for a set of nxm UV coordinates (u,v).

Given n longitude values, the angular size T can be obtained by using:

$$T = \frac{2\pi}{N}. \tag{1}$$

Considering a sphere, an angular position α_i represents the ith longitude value:

$$\alpha_i = i * T, \tag{2}$$

The sine and cosine of the angle T define the X and Z axes positions of the sphere points which belong to the cross section of the sphere. In such manner, assuming a sphere of radius R, the X and Z axes positions can be computed as:

$$x_i = R * \sin(\alpha_i), \tag{3}$$

$$z_i = R * \cos(\alpha_i). \tag{4}$$

In a longitudinal cut, the R-ray of a cross-section varies along the height of the sphere. For this reason, angular size K considering a total of M latitude values can be calculated as:

$$K = \frac{\pi}{M}. \tag{5}$$

The mth latitude value α_{ym} can be obtained by equation:

$$\alpha_{ym} = m * K \tag{6}$$

The Y axis position y_m for each sphere point can be obtained by considering unit radius using:

$$y_m = \cos(\alpha_{ym}), \tag{7}$$

The radius R_{ym} obtained in a cross section at latitude m is defined as:

$$R_{ym} = \sin(\alpha_{ym}) \tag{8}$$

Applying equation 8 in equations 3 and 4 we get positions X and Z of the vertices of the sphere according to a longitude n and latitude m coordinates resulting in equations 9 and 10.

$$x(m,n) = \sin(\alpha_{ym}) * \sin(\alpha_n), \tag{9}$$

$$z(m,n) = \sin(\alpha_{ym}) * \cos(\alpha_n). \tag{10}$$

$$y(m,n) = \cos(\alpha_{ym}), \tag{11}$$

C. Mapping Equirectangular Images to a Skybox

A skybox is rendered when no 3D element is rasterized by the virtual camera. In the rasterization process, it is necessary to identify a UV coordinate for each pixel (or fragment) rendered on screen. Skybox shaders usually utilize 3D textures to store the six faces of a cube through a graphical function called tex3D.

Mapping an equirectangular image to a skybox involves finding the UV vector value given a normalized direction. Considering the vector (x, y, z) as the normalized direction, equation 12 can be used on a vertex shader.

$$uv = (\arctan(\frac{x}{y}), \arccos(y)) \tag{12}$$

Thus, when mapping to a sphere UV coordinates are projected into 3D space and when mapping to a skybox the opposite happens: normalized 3d space positions continuously seek equivalent UV coordinates.

D. Mapping Equirectangular Images to a Cubemap

The first step to use a Cubemap is to generate a cube. The standard cube generated by Unity, however, does not have enough vertices for precise UV mapping. It happens as UV mapping is a sine/cosine function whereas rasterization inside of a triangle obtains UV values through linear interpolation of its vertices, thus causing distortions.

For better results, we divided each triangle into four parts. From a cube of 10 vertices and 12 triangles, we obtained a 4090 vertices/triangles cube.

As we generate each new vertex, it is possible to calculate its respective UV coordinate using equation 11. Noticeably, the cubemap view is equivalent to the discretization of the continuous UV mapping approach in a skybox, i.e., it is calculated per vertex instead of being applied on pixel basis.

E. Image Quality Analysis Metrics

With regards to the metrics, the goal of the objective image quality assessment is to develop a quantitative measure that can determine the quality of any given image. It is difficult, though, to find a single objective and easy-to-calculate measurement that matches the visual inspection and is suitable for a variety of application requirements. To address this problem, we use three different metrics which are: Mean Square Error (MSE), Structural Similarity Index (SSIM) and Peak Signal-to-noise ratio (PSNR). The image which has the smallest MSE, and highest SSIM and PSNR, is assumed to have better quality. MSE can be computed as:

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e(m,n)^2. \quad (13)$$

Similarly, SSIM assumes that neighboring pixels in an image have strong inter-dependencies and these dependencies carry important information about the structure of the objects [11]. SSIM can be calculated as:

$$SSIM(x,y) = \frac{(2 * \mu_x * \mu_y + C_1) * (2 * \sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) * (\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (14)$$

where μ_X and μ_y are the mean intensity value, σ_x^2 and σ_y^2 are the variance of the corresponding images, whereas σ_{xy} is the covariance of image X and Y. Also, $C_1 = (k_1 L)^2$ and $C_1 = (k_2 L)^2$ are stability parameters, where $K_1 = 0.01$ and $k_2 = 0.03$.

Peak Signal-to-noise ratio (PSNR) is the most used metric for image quality assessment and can be computed using MSE as:

$$PSNR = 10 * log_{10} \frac{(2^n - 1)^2}{MSE}. \quad (15)$$

IV. Experimental Results

This section presents the implementation details as well as and qualitative and quantitative evaluation of the proposed method.

A. System Architecture

Figure 2 shows the connected components in the architecture of the proposed method. The proposed architecture contain two layers: a Unity layer and a Python layer. The Unity implementation involves a C# configuration layer in Unity editor to generate images. Moreover, the python layer is used for calculating the objective metrics for each of the Unity-generated images. To make efficient communication, cross-tiered communication among layers takes place through the creation of new processes within the Unity editor.

To make the user experience pleasant, an editor interface was developed in the form of a custom unity inspector, that is,

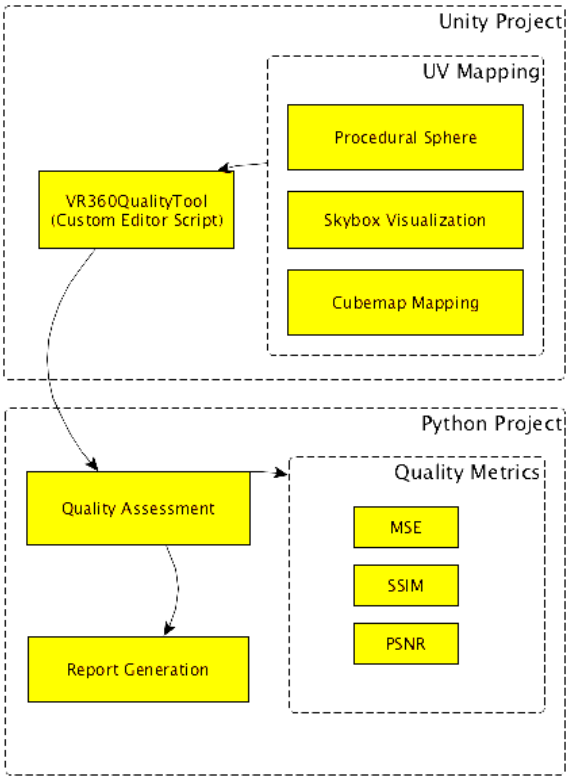

Fig. 2. Architecture of the proposed quality assessment tool.

a custom view of our component in C#. In this component, users can define multiple preferences for the output image. These preferences include field of view angles, width and height as well as directions of the image to be created, comparison metrics to be used, and define whether graphs or a report will be generated at the end of the process.

Furthermore, the python layer is responsible for evaluating the pairs of images generated by the Unity layer. These images are evaluated using the scikit, numpy, and matplotlib libraries and the result of each metric is saved in a report at the end, which summarizes all the results.

B. Qualitative and Quantitative Evaluation

The proposed tool was developed using Unity 2017.3.1F and python 2.7. This custom editor tool can be imported into any Unity project through a unitypackage, a standard format from Unity to distribute resources and tools. When adding VR360QualityTool script to a GameObject, the interface depicted at figure 1 is presented. The field 'Screenshot Directions' allows to define one or more target directions, as explained in section III. For our experiments, we used directions toward right $D_0 = (1.0, 0.0, 0.0)$, upside $D_1 = (0.0, 1.0, 0.0)$ and forward $D_2 = (1.0, 0.0, -1.0)$, respectively. Each direction can be visualized in figures 3, 4 and 5.

Different rendering approaches were used in our experiments: Skybox (used as reference), a sphere-based shader (S), a cubemap shader (C_e) with an interpolation error, and a nor-

(a) Skybox 0; (b) Cubemap 0; (c) Sphere 0; (d) Cubemap (Error Corrected);

Fig. 3. Example of images rendered in D_0 using: a) Skybox image is used as reference; b) result image rendered using cubemap; c) rendered using sphere.

(a) Skybox 1; (b) Cubemap 1; (c) Sphere 1; (d) Cubemap (Error Corrected);

Fig. 4. Example of images rendered in D_1 using: a) Skybox image is used as reference; b) result image rendered using cubemap; c) rendered using sphere.

(a) Skybox 2; (b) Cubemap 2; (c) Sphere 2; (d) Cubemap (Error Corrected);

Fig. 5. Example of images rendered in D_2 using: a) Cubmap image is used as reference; b) result image rendered using cubemap with error; c) rendered using sphere.

TABLE I
METRICS RESULTS EXAMPLE

	Direction 0			Direction 1			Direction 2		
	S	C_e	C_f	S	C_e	C_f	S	C_e	C_f
MSE	50.27	551.27	1.29	39.99	41.89	0.70	27.88	18.60	2.40
SSIM	0.93	0.93	0.99	0.93	0.97	1.00	0.87	0.96	0.99
PSNR	31.12	20.72	47.01	32.11	31.91	49.66	27.88	35.43	44.33

mal cubemap shader (C_f). The distorted cubemap was created intentionally to prove our method is capable of identifying such errors. It occurs in triangles which have UV coordinates between 0.9 to 0.1. Instead of interpolating to the chosen direction, this cubemap interpolates to the opposite direction (0.7, 0.6 ... 0.1, x) and compresses the whole image in a tight

image region. Both cubemap implementations are employing equirectangular conversion as explained at subsection III-D.

To perform the quality assessment each image is compared to its respective direction in the reference image. Figures 3a, 4a and 5a are produced by Skybox rendering which is used as reference for each evaluated direction. The skybox rendered image patch is compared with those obtained with the spherical rendering S in figure 3c, with cubemap rendering C_e which contains interpolation error in figure 3b and with cubemap rendering C_f in figure 4b. Figures 3, 4 and 5 shows all images used as input for our proposed evaluation system and their results can be seen at table I.

After processing the images, the results are summarized in the generated report, as demonstrated by the table I. Cubemap C_f rendering approach had the best results in all directions for all metrics. Since the UV mapping of Skybox rendering is similar to the UV mapping of the cubemap approach, they tend to generate more similar images for a same direction. In contrast to this, cubemap C_e presents a very noticeable distortion region for directions D_0 and D_1 as depicted in figures 3b and 4b. As a result the MSE value for both directions are very degraded. Thus we can conclude that MSE, SSIM and PSNR are good enough to identify images patches with shader errors given a reasonable reference image.

Despite the absence of rendering errors in the sphere shader S, its metrics values are surpassed by the cubemap C_f. This is explained by the difference between the distortions of a sphere-based and a Skybox-based rendering. The equirectangular mapping requires a sinusoidal UV mapping. Considering that the sphere has a sinusoidal mapping in its vertices, the UV coordinates are interpolated linearly inside each triangle. On the other hand, skybox mapping employs a sinusoidal mapping inside the pixel shader and consequently affects every pixel produced. However, for the end-user, such differences are not perceptually remarkable.

After every iteration, our tool presents the metrics, for example, the table I. The best rendering candidates are ranked based on their performance in each metric and direction.

V. Conclusion

We proposed the development of an equirectangular image quality assessment tool which uses objective metrics such as MSE, SSIM and PSNR in order to facilitate choosing among different image resolutions and mapping solutions.

Our tool is integrated into the Unity Editor as Unity is the most used development engine for virtual reality applications. Different UV mapping techniques were used to visualize 360° images: latitude/longitude in a inverted sphere; skybox; and cubemap. We also demonstrated how to convert a equirectangular image into a cubemap texturing using a custom shader.

This Unity implemention assess image quality of already implemented rendering approaches for equirectangular image visualization by generating viewport-based image patches. A python layer applies full-reference objective metrics in each image patch using a correspondent patch from Skybox renderization.

One of the limitation of our tool is that the user needs to simulate the target device by emulating a specific resolution and field view. For future work, we plan to obtain viewport patches directly from mobile devices where VR applications can be executed. Thus, we also plan an embedded component in the application that allows to reap results while running the application on the mobile device. We also envisage to conduct subject evaluations in order to pick better metrics for quality assessment of image patches.

Acknowledgment

The authors would like to thank the support provided by SIDIA Instituto de Ciência e Tecnologia and teams.

References

[1] G. Westheimer, "Visual acuity," *Annual Review of Psychology*, vol. 16, pp. 359–380, 1965.

[2] P. Fuchs, *Virtual Reality Headsets-A Theoretical and Pragmatic Approach*. CRC Press, 2017.

[3] V. Zakharchenko, K. P. Choi, and J. H. Park, "Quality metric for spherical panoramic video," in *Proc. of SPIE 9970 Optics and Photonics for Information Processing X*, 2016, pp. 57 – 65. [Online]. Available: https://doi.org/10.1117/12.2235885

[4] C. Dunn and B. Knott, "Resolution-defined projections for virtual reality video compression," in *2017 IEEE Virtual Reality (VR)*, pp. 337–338.

[5] F. Duanmu, Y. He, X. Xiu, P. Hanhart, Y. Ye, and Y. Wang, "Hybrid cubemap projection format for 360-degree video coding," in *2018 Data Compression Conference*. IEEE, 2018, pp. 404–404.

[6] Z. Wang and A. C. Bovik, "Modern image quality assessment," *Synthesis Lectures on Image, Video, and Multimedia Processing*, vol. 2, no. 1, pp. 1–156, 2006.

[7] M. H. Pinson and S. Wolf, "Comparing subjective video quality testing methodologies," in *Visual Communications and Image Processing 2003*, vol. 5150. International Society for Optics and Photonics, 2003, pp. 573–583.

[8] C. Li, M. Xu, S. Zhang, and P. L. Callet, "State-of-the-art in ° video/image processing: Perception, assessment and compression," *arXiv preprint arXiv:1905.00161*, 2019.

[9] S. K. Md, B. Appina, and S. S. Channappayya, "Full-reference stereo image quality assessment using natural stereo scene statistics," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1985–1989, 2015.

[10] A. M. Gil and T. S. Figueira, "Equirectangular image quality assessment tool integrated into the unity editor," in *International Conference on Human-Computer Interaction*. Springer, 2019, pp. 374–381.

[11] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.